Package ‘phylosim’

February 20, 2015

Version 2.1.1
Date 2011-06-28
Title R package for simulating biological sequence evolution
Author Botond Sipos <sbotond@ebi.ac.uk>, Gregory Jordan <greg@ebi.ac.uk>
Maintainer Botond Sipos <sbotond@ebi.ac.uk>
Depends R (>= 2.15.2), R.oo (>= 1.13.0), ape (>= 2.3), compoisson (>= 0.3), ggplot2 (>= 0.9.3)

Suggests
Imports R.methodsS3

Description
PhyloSim is an extensible object-oriented framework for the Monte Carlo simulation of sequence evolution written in 100 percent R. It is built on the top of the R.oo and ape packages and uses Gillespie's direct method to simulate substitutions, insertions and deletions.

License GPL (>= 3)

URL http://www.ebi.ac.uk/goldman-srv/phylosim
LazyLoad TRUE
NeedsCompilation no
Repository CRAN
Date/Publication 2014-01-10 14:42:31

R topics documented:

==.Alphabet ................................................................. 12
==.Process ................................................................. 13
Alphabet ................................................................. 14
AminoAcidAlphabet ....................................................... 16
AminoAcidSequence ....................................................... 18
AminoAcidSubst ............................................................ 19
AnyAlphabet ................................................................. 22
areSynonymous.CodonAlphabet ............................................ 23
### Topics Documented

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>as.character.Alphabet</td>
<td>24</td>
</tr>
<tr>
<td>as.character.Event</td>
<td>25</td>
</tr>
<tr>
<td>as.character.GeneralSubstitution</td>
<td>26</td>
</tr>
<tr>
<td>as.character.PhyloSim</td>
<td>27</td>
</tr>
<tr>
<td>as.character.Process</td>
<td>28</td>
</tr>
<tr>
<td>as.character.QMatrix</td>
<td>29</td>
</tr>
<tr>
<td>as.character.Sequence</td>
<td>30</td>
</tr>
<tr>
<td>as.character.Site</td>
<td>31</td>
</tr>
<tr>
<td>attachHookToNode.PhyloSim</td>
<td>32</td>
</tr>
<tr>
<td>attachProcess.Sequence</td>
<td>33</td>
</tr>
<tr>
<td>attachProcess.Site</td>
<td>34</td>
</tr>
<tr>
<td>attachSeqToNode.PhyloSim</td>
<td>35</td>
</tr>
<tr>
<td>BinaryAlphabet</td>
<td>36</td>
</tr>
<tr>
<td>BinarySequence</td>
<td>37</td>
</tr>
<tr>
<td>BinarySubst</td>
<td>39</td>
</tr>
<tr>
<td>BrownianInserter</td>
<td>41</td>
</tr>
<tr>
<td>BrownianPath.BrownianInserter</td>
<td>44</td>
</tr>
<tr>
<td>buildFromPAML.AminoAcidSubst</td>
<td>45</td>
</tr>
<tr>
<td>buildFromPAML.CodonSubst</td>
<td>46</td>
</tr>
<tr>
<td>checkConsistency.Alphabet</td>
<td>47</td>
</tr>
<tr>
<td>checkConsistency.AminoAcidSubst</td>
<td>48</td>
</tr>
<tr>
<td>checkConsistency.BinarySubst</td>
<td>48</td>
</tr>
<tr>
<td>checkConsistency.BrownianInserter</td>
<td>49</td>
</tr>
<tr>
<td>checkConsistency.CodonAlphabet</td>
<td>50</td>
</tr>
<tr>
<td>checkConsistency.CodonSequence</td>
<td>50</td>
</tr>
<tr>
<td>checkConsistency.CodonSubst</td>
<td>51</td>
</tr>
<tr>
<td>checkConsistency.CodonUNREST</td>
<td>52</td>
</tr>
<tr>
<td>checkConsistency.ContinuousDeletor</td>
<td>52</td>
</tr>
<tr>
<td>checkConsistency.ContinuousInsertor</td>
<td>53</td>
</tr>
<tr>
<td>checkConsistency.DiscreteDeletor</td>
<td>54</td>
</tr>
<tr>
<td>checkConsistency.DiscreteInsertor</td>
<td>54</td>
</tr>
<tr>
<td>checkConsistency.Event</td>
<td>55</td>
</tr>
<tr>
<td>checkConsistency.F81</td>
<td>56</td>
</tr>
<tr>
<td>checkConsistency.F84</td>
<td>56</td>
</tr>
<tr>
<td>checkConsistency.FastFieldDeletor</td>
<td>57</td>
</tr>
<tr>
<td>checkConsistency.GeneralDeleter</td>
<td>58</td>
</tr>
<tr>
<td>checkConsistency.GeneralInDel</td>
<td>58</td>
</tr>
<tr>
<td>checkConsistency.GeneralInserter</td>
<td>59</td>
</tr>
<tr>
<td>checkConsistency.GeneralSubstitution</td>
<td>60</td>
</tr>
<tr>
<td>checkConsistency.GTR</td>
<td>60</td>
</tr>
<tr>
<td>checkConsistency.GY94</td>
<td>61</td>
</tr>
<tr>
<td>checkConsistency.HKY</td>
<td>62</td>
</tr>
<tr>
<td>checkConsistency.JC69</td>
<td>62</td>
</tr>
<tr>
<td>checkConsistency.K80</td>
<td>63</td>
</tr>
<tr>
<td>checkConsistency.K81</td>
<td>64</td>
</tr>
<tr>
<td>checkConsistency.PhyloSim</td>
<td>64</td>
</tr>
<tr>
<td>checkConsistency.Process</td>
<td>65</td>
</tr>
<tr>
<td>checkConsistency.PSRoot</td>
<td>66</td>
</tr>
</tbody>
</table>
R topics documented:

checkConsistency.PSRootSummary ........................................ 66
checkConsistency.QMatrix .................................................. 67
checkConsistency.Sequence ................................................. 68
checkConsistency.Site ....................................................... 68
checkConsistency.T92 ....................................................... 69
checkConsistency.TN93 ....................................................... 70
checkConsistency.UNREST ................................................... 70
clearStates.Sequence ....................................................... 71
clon:GeneralSubstitution ................................................... 72
clon:Process ................................................................. 73
clon:Sequence ............................................................... 74
CodonAlphabet ............................................................... 75
CodonSequence .............................................................. 77
CodonSubst ................................................................. 79
CodonUNREST ............................................................... 81
ContinuousDeleter ............................................................ 83
ContinuousInsertor .......................................................... 85
copySubSequence.Sequence ................................................ 88
cpREV ......................................................................... 89
Debug.PhyloSim .............................................................. 91
deleteSubSequence.Sequence ............................................. 92
detachHookFromNode.PhyloSim .......................................... 93
detachProcess.Sequence ................................................... 94
detachProcess.Site ......................................................... 95
detachSeqFromNode.PhyloSim ............................................ 96
DiscreteDeleter .............................................................. 97
DiscreteInsertor ............................................................. 99
ECMrest ................................................................. 102
ECMunrest .............................................................. 104
enableVirtual.PSRoot ..................................................... 106
Event ................................................................. 107
exportStatTree.PhyloSim ................................................... 109
F81 ................................................................. 111
F84 ................................................................. 113
FastFieldDeleter .......................................................... 116
flagTotalRate.Site .......................................................... 119
GeneralDeleter ............................................................. 120
GeneralInDel .............................................................. 122
GeneralInsertor ........................................................... 125
GeneralSubstitution ....................................................... 128
generateInsert:GeneralInsertor ........................................ 131
getAcceptBy:GeneralInDel ............................................... 133
getAcceptWin:GeneralInsertor ......................................... 134
getAlignment.PhyloSim .................................................. 135
getAlignmentLength.PhyloSim ......................................... 136
getAlphabet:GeneralSubstitution ....................................... 137
getAlphabet:Process ...................................................... 138
getAlphabet:QMatrix ....................................................... 139
getAlphabet.Site .................................................. 140
getAlphabets.Sequence ........................................... 141
getAncestral.Sequence ........................................... 142
getAncestral.Site .................................................. 143
getBaseFreqs.F81 .................................................. 144
getBaseFreqs.F84 .................................................. 145
getBaseFreqs.GTR .................................................. 146
getBaseFreqs.HKY .................................................. 147
getBaseFreqs.K80 .................................................. 148
getBaseFreqs.K81 .................................................. 149
getBaseFreqs.TN93 .................................................. 150
getBigRate.Sequence ............................................. 151
getBranchEvents.PhyloSim ....................................... 152
getCodonFreqs.GY94 ............................................. 153
getComments.PSRoot ............................................. 154
getCumulativeRates.Sequence ................................... 155
getCumulativeRatesFromRange.Sequence ....................... 156
getDeletionTolerance.Sequence ................................ 157
getDist.ContinuousDeleter ...................................... 158
getDist.ContinuousInsertor .................................... 159
getEdge.PhyloSim .................................................. 160
getEdges.PhyloSim .................................................. 161
getEquDist.GeneralSubstitution ................................ 162
getEventRate.GeneralSubstitution ............................... 163
getEventRate.QMatrix ............................................. 164
getEventRateAtSite.GeneralSubstitution ....................... 165
getEvents.Sequence ............................................... 166
getEvents.Site ...................................................... 167
getEventsAtSite.FastFieldDeleter ............................... 168
getEventsAtSite.GeneralDeleter ................................. 169
getEventsAtSite.GeneralInsertor ............................... 170
getEventsAtSite.GeneralSubstitution ......................... 171
getEventsAtSite.GY94 ............................................. 172
getEventsAtSite.Process ........................................ 173
getGenerateBy.GeneralInsertor ................................ 174
getHandler.Event .................................................... 175
getId.PhyloSim ..................................................... 176
getId.Process ....................................................... 177
getId.QMatrix ....................................................... 178
getId.Sequence ...................................................... 179
getInsertHook.GeneralInsertor ................................ 180
getInsertionTolerance.Sequence ................................. 181
getKappa.F84 ....................................................... 182
getKappa.GY94 ....................................................... 183
getLength.Sequence ................................................ 184
getLengthParam1.FastFieldDeleter ............................. 185
getLengthParam2.FastFieldDeleter ............................. 186
getLogFile.PhyloSim ............................................. 187
getLogLevel.PhyloSim ......................................................... 188
getMatrix.QMatrix ............................................................. 189
getMaxLength.ContinuousDeleter ........................................... 190
getMaxLength.ContinuousInsertor ......................................... 191
getMethodsList.PSRoot ....................................................... 192
getName.Event ................................................................. 193
getName.PhyloSim ............................................................. 194
getName.Process ............................................................... 195
getName.QMatrix ............................................................... 196
getName.Sequence ............................................................. 197
getNedges.PhyloSim ........................................................... 198
getNodes.PhyloSim ............................................................ 199
getNtips.PhyloSim ............................................................. 200
getOmegas.CodonSequence ................................................. 201
getOmegaScalingFactor.GY94 .............................................. 202
getParameterAtSite.Process ............................................... 203
getParameterAtSites.Sequence ............................................ 204
getPhylo.PhyloSim ............................................................ 205
getPosition.Event ............................................................. 206
getProbs.DiscreteDeleter .................................................. 207
getProbs.DiscreteInsertor ................................................ 208
getProcess.Event ............................................................. 209
getProcess.QMatrix ........................................................... 210
getProcesses.Sequence ..................................................... 211
getProcesses.Site .............................................................. 212
getProposeBy.GeneralInDel ............................................... 213
gtQMatrix.GeneralSubstitution .......................................... 214
gtRate.Event ................................................................. 215
gtRate.GeneralInDel ......................................................... 216
gtRate.GeneralSubstitution ............................................... 217
gtRate.QMatrix ............................................................... 218
gtRateList.GeneralSubstitution ......................................... 219
gtRateList.QMatrix .......................................................... 220
gtRateMultipliers.Sequence .............................................. 221
gtRateParam.F81 .............................................................. 222
gtRateParam.F84 .............................................................. 222
gtRateParam.GTR .............................................................. 223
gtRateParam.HKY .............................................................. 224
gtRateParam.K80 .............................................................. 225
gtRateParam.K81 .............................................................. 226
gtRateParam.T92 .............................................................. 227
gtRateParam.TN93 ............................................................. 228
gtRateParamList.F81 ......................................................... 229
gtRateParamList.F84 ......................................................... 230
gtRateParamList.GTR ........................................................ 231
gtRateParamList.HKY ......................................................... 232
gtRateParamList.K80 ........................................................ 233
gtRateParamList.K81 ........................................................ 234
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>getRateParamList.T92</td>
<td>235</td>
</tr>
<tr>
<td>getRateParamList.TN93</td>
<td>236</td>
</tr>
<tr>
<td>getRootNode.PhyloSim</td>
<td>237</td>
</tr>
<tr>
<td>getRootSeq.PhyloSim</td>
<td>238</td>
</tr>
<tr>
<td>getScale.BrownianInsertor</td>
<td>239</td>
</tr>
<tr>
<td>getScaledMatrix.QMatrix</td>
<td>240</td>
</tr>
<tr>
<td>getSeqFromNode.PhyloSim</td>
<td>241</td>
</tr>
<tr>
<td>getSequence.Site</td>
<td>242</td>
</tr>
<tr>
<td>getSequences.PhyloSim</td>
<td>243</td>
</tr>
<tr>
<td>getSite.Event</td>
<td>244</td>
</tr>
<tr>
<td>getSites.Sequence</td>
<td>245</td>
</tr>
<tr>
<td>getSiteSpecificParamIds.Process</td>
<td>246</td>
</tr>
<tr>
<td>getSiteSpecificParamList.Process</td>
<td>247</td>
</tr>
<tr>
<td>getSize.Alphabet</td>
<td>248</td>
</tr>
<tr>
<td>getSizes.DiscreteDeleter</td>
<td>249</td>
</tr>
<tr>
<td>getSizes.DiscreteInsertor</td>
<td>250</td>
</tr>
<tr>
<td>getState.Site</td>
<td>251</td>
</tr>
<tr>
<td>getStates.Sequence</td>
<td>252</td>
</tr>
<tr>
<td>getString.Sequence</td>
<td>253</td>
</tr>
<tr>
<td>getSymbolFreqs.Sequence</td>
<td>254</td>
</tr>
<tr>
<td>getSymbolLength.Alphabet</td>
<td>255</td>
</tr>
<tr>
<td>getSymbols.Alphabet</td>
<td>256</td>
</tr>
<tr>
<td>getTableId.CodonAlphabet</td>
<td>257</td>
</tr>
<tr>
<td>getTemplateSeq.GeneralInsertor</td>
<td>258</td>
</tr>
<tr>
<td>getTheta.T92</td>
<td>259</td>
</tr>
<tr>
<td>getTipLabels.PhyloSim</td>
<td>260</td>
</tr>
<tr>
<td>getTips.PhyloSim</td>
<td>261</td>
</tr>
<tr>
<td>getToleranceMargin.FastFieldDeleter</td>
<td>262</td>
</tr>
<tr>
<td>getTotalRate.Site</td>
<td>263</td>
</tr>
<tr>
<td>getTotalRates.Sequence</td>
<td>264</td>
</tr>
<tr>
<td>getTotalRatesFromRange.Sequence</td>
<td>264</td>
</tr>
<tr>
<td>getTransTable.CodonAlphabet</td>
<td>265</td>
</tr>
<tr>
<td>getTreeLength.PhyloSim</td>
<td>266</td>
</tr>
<tr>
<td>getType.Alphabet</td>
<td>267</td>
</tr>
<tr>
<td>getType.BrownianInsertor</td>
<td>268</td>
</tr>
<tr>
<td>getType.FastFieldDeleter</td>
<td>269</td>
</tr>
<tr>
<td>getUniqueAlphabets.Sequence</td>
<td>270</td>
</tr>
<tr>
<td>getUniqueProcesses.Sequence</td>
<td>271</td>
</tr>
<tr>
<td>getWriteProtected.Alphabet</td>
<td>272</td>
</tr>
<tr>
<td>getWriteProtected.Event</td>
<td>273</td>
</tr>
<tr>
<td>getWriteProtected.Process</td>
<td>274</td>
</tr>
<tr>
<td>getWriteProtected.QMatrix</td>
<td>275</td>
</tr>
<tr>
<td>getWriteProtected.Sequence</td>
<td>276</td>
</tr>
<tr>
<td>globalConsistencyCheck.PSRoot</td>
<td>277</td>
</tr>
<tr>
<td>GTR</td>
<td>278</td>
</tr>
<tr>
<td>GY94</td>
<td>280</td>
</tr>
<tr>
<td>hasSiteSpecificParameter.Process</td>
<td>283</td>
</tr>
<tr>
<td>hasSymbols.Alphabet</td>
<td>284</td>
</tr>
</tbody>
</table>
R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>hasUndefinedRate.GeneralInDel</td>
<td>285</td>
</tr>
<tr>
<td>hasUndefinedRate.GeneralSubstitution</td>
<td>286</td>
</tr>
<tr>
<td>hasUndefinedRate.Process</td>
<td>287</td>
</tr>
<tr>
<td>HKY</td>
<td>288</td>
</tr>
<tr>
<td>insertSequence.Sequence</td>
<td>289</td>
</tr>
<tr>
<td>intersect.list.PSRoot</td>
<td>290</td>
</tr>
<tr>
<td>is.Alphabet</td>
<td>291</td>
</tr>
<tr>
<td>is.CodonAlphabet</td>
<td>292</td>
</tr>
<tr>
<td>is.CodonUNREST</td>
<td>293</td>
</tr>
<tr>
<td>is.Event</td>
<td>294</td>
</tr>
<tr>
<td>is.Process</td>
<td>295</td>
</tr>
<tr>
<td>is.GeneralDeletor</td>
<td>296</td>
</tr>
<tr>
<td>is.GeneralInDel</td>
<td>297</td>
</tr>
<tr>
<td>is.GeneralInsertor</td>
<td>298</td>
</tr>
<tr>
<td>is.GeneralSubstitution</td>
<td>299</td>
</tr>
<tr>
<td>is.GY94</td>
<td>300</td>
</tr>
<tr>
<td>is.na.PSRoot</td>
<td>301</td>
</tr>
<tr>
<td>is.phylo</td>
<td>302</td>
</tr>
<tr>
<td>is.PSRoot</td>
<td>303</td>
</tr>
<tr>
<td>is.QMatrix</td>
<td>304</td>
</tr>
<tr>
<td>is.Sequence</td>
<td>305</td>
</tr>
<tr>
<td>is.Site</td>
<td>306</td>
</tr>
<tr>
<td>is.tip.PhyloSim</td>
<td>307</td>
</tr>
<tr>
<td>isAttached.Site</td>
<td>308</td>
</tr>
<tr>
<td>isEmpty.Alphabet</td>
<td>309</td>
</tr>
<tr>
<td>isStartCodon.CodonAlphabet</td>
<td>310</td>
</tr>
<tr>
<td>isStopCodon.CodonAlphabet</td>
<td>311</td>
</tr>
<tr>
<td>JC69</td>
<td>312</td>
</tr>
<tr>
<td>JTT</td>
<td>313</td>
</tr>
<tr>
<td>JTT.dcmut</td>
<td>314</td>
</tr>
<tr>
<td>K80</td>
<td>315</td>
</tr>
<tr>
<td>K81</td>
<td>316</td>
</tr>
<tr>
<td>LG</td>
<td>317</td>
</tr>
<tr>
<td>ll.PSRoot</td>
<td>318</td>
</tr>
<tr>
<td>Log.PhyloSim</td>
<td>319</td>
</tr>
<tr>
<td>mtArt</td>
<td>320</td>
</tr>
<tr>
<td>mtMam</td>
<td>321</td>
</tr>
<tr>
<td>mtREV24</td>
<td>322</td>
</tr>
<tr>
<td>MtZoa</td>
<td>323</td>
</tr>
<tr>
<td>NucleotideAlphabet</td>
<td>324</td>
</tr>
<tr>
<td>NucleotideSequence</td>
<td>325</td>
</tr>
<tr>
<td>omegaHist.CodonSequence</td>
<td>326</td>
</tr>
<tr>
<td>omegaVarM0.CodonSequence</td>
<td>327</td>
</tr>
<tr>
<td>omegaVarM1.CodonSequence</td>
<td>328</td>
</tr>
<tr>
<td>omegaVarM2.CodonSequence</td>
<td>329</td>
</tr>
<tr>
<td>newAAMatrix.AminoAcidSubst</td>
<td>330</td>
</tr>
<tr>
<td>newMatrix.CodonSubst</td>
<td>331</td>
</tr>
<tr>
<td>my.all.equal.PSRoot</td>
<td>332</td>
</tr>
<tr>
<td>omegaHist.CodonSequence</td>
<td>333</td>
</tr>
<tr>
<td>omegaVarM0.CodonSequence</td>
<td>334</td>
</tr>
<tr>
<td>omegaVarM1.CodonSequence</td>
<td>335</td>
</tr>
<tr>
<td>omegaVarM2.CodonSequence</td>
<td>336</td>
</tr>
</tbody>
</table>
omegaVarM3.CodonSequence ................................................. 347
omegaVarM4.CodonSequence ................................................. 348
PAM ................................................................. 350
PAM.dcmut ................................................................. 352
Perform.Event ................................................................. 354
PhyloSim ................................................................. 355
plot.ContinuousDeleter ....................................................... 361
plot.ContinuousInsertor ...................................................... 362
plot.DiscreteDeleter ......................................................... 363
plot.DiscreteInsertor ......................................................... 364
plot.GeneralSubstitution ..................................................... 365
plot.PhyloSim ............................................................... 366
plot.Sequence ............................................................... 368
plotParametersAtSites.Sequence ........................................... 369
plusGamma.Sequence ......................................................... 370
plusInvGamma.Sequence ....................................................... 372
print.PSRootSummary ......................................................... 373
print.QMatrix .............................................................. 374
Process ................................................................. 375
proposeLength.GeneralInDel .................................................. 377
PSRoot ............................................................... 378
PSRootSummary ............................................................ 380
QMatrix ................................................................. 381
readAlignment.PhyloSim ....................................................... 383
readTree.PhyloSim .......................................................... 384
rescaleQMatrix.GeneralSubstitution ......................................... 386
revComp.NucleotideSequence ................................................ 387
sampleState.GeneralSubstitution ........................................... 388
sampleStates.Sequence ....................................................... 389
saveAlignment.PhyloSim ....................................................... 390
Scale.QMatrix ............................................................. 391
scaleTree.PhyloSim .......................................................... 392
Sequence ................................................................. 393
setAcceptBy.GeneralInDel ................................................... 397
setAcceptWin.GeneralInsertor ............................................. 399
setAlignment.PhyloSim ....................................................... 400
setAlphabet.GeneralSubstitution ........................................... 400
setAlphabet.Process ........................................................ 401
setAlphabet.QMatrix ......................................................... 402
setAlphabet.Site ............................................................ 403
setAlphabets.Sequence ....................................................... 404
setAncestral.Sequence ....................................................... 405
setAncestral.Site ............................................................ 406
setBaseFreqs.F81 ............................................................ 407
setBaseFreqs.F84 ............................................................ 408
setBaseFreqs.GTR ............................................................ 409
setBaseFreqs.HKY ............................................................ 410
setBaseFreqs.K80 ............................................................. 411
setBaseFreqs.K81 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 412
setBaseFreqs.TN93 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 412
setBigRate.Sequence . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 413
setBranchEvents.PhyloSim . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 414
setCodonFreqs.GY94 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 415
setComments.PSRoot . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 416
setCumulativeRates.Sequence . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 417
setDeletionTolerance.Sequence . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 417
setDist.ContinuousDeleter . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 418
setDist.ContinuousInsulator . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 419
setEdges.PhyloSim . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 420
setEquidist.GeneralSubstitution . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 421
setEvents.Site . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 422
setGenerateBy.GeneralInsulator . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 423
setHandler.Event . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 424
setId.PhyloSim . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 425
setId.Process . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 425
setId.QMatrix . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 426
setId.Sequence . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 427
setInsertHook.GeneralInsulator . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 427
setInsertionTolerance.Sequence . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 429
setKappa.F84 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 430
setKappa.GY94 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 431
setLength.Sequence . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 432
setLengthParam1.FastFieldDeleter . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 432
setLengthParam2.FastFieldDeleter . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 433
setLogFile.PhyloSim . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 434
setLogLevel.PhyloSim . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 435
setMatrix.QMatrix . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 436
setMaxLength.ContinuousDeleter . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 437
setMaxLength.ContinuousInsulator . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 438
setMethodsList.PSRoot . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 439
setName.Event . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 439
setName.PhyloSim . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 440
setName.Process . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 441
setName.QMatrix . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 442
setName.Sequence . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 443
setNedges.PhyloSim . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 444
setNodes.PhyloSim . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 445
setNtips.PhyloSim . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 445
setOmegas.CodonSequence . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 446
setParameterAtSite.Process . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 447
setParameterAtSites.Sequence . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 448
setPhylo.PhyloSim . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 449
setPosition.Event . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 450
setProbs.DiscreteDeleter . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 451
setProbs.DiscreteInsulator . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 452
setProcess.Event . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 453
<table>
<thead>
<tr>
<th>Function/Attribute</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>setProcess.QMatrix</td>
<td>454</td>
</tr>
<tr>
<td>setProcesses.Sequence</td>
<td>455</td>
</tr>
<tr>
<td>setProcesses.Site</td>
<td>456</td>
</tr>
<tr>
<td>setProposeBy.GeneralInDel</td>
<td>457</td>
</tr>
<tr>
<td>setQMatrix.GeneralSubstitution</td>
<td>458</td>
</tr>
<tr>
<td>setRate.Event</td>
<td>459</td>
</tr>
<tr>
<td>setRate.GeneralInDel</td>
<td>460</td>
</tr>
<tr>
<td>setRate.GY94</td>
<td>461</td>
</tr>
<tr>
<td>setRate.QMatrix</td>
<td>462</td>
</tr>
<tr>
<td>setRateList.GeneralSubstitution</td>
<td>463</td>
</tr>
<tr>
<td>setRateList.QMatrix</td>
<td>464</td>
</tr>
<tr>
<td>setRateMultipliers.Sequence</td>
<td>465</td>
</tr>
<tr>
<td>setRateParam.F81</td>
<td>466</td>
</tr>
<tr>
<td>setRateParam.F84</td>
<td>467</td>
</tr>
<tr>
<td>setRateParam.GTR</td>
<td>468</td>
</tr>
<tr>
<td>setRateParam.HKY</td>
<td>469</td>
</tr>
<tr>
<td>setRateParam.K80</td>
<td>470</td>
</tr>
<tr>
<td>setRateParam.K81</td>
<td>471</td>
</tr>
<tr>
<td>setRateParam.T92</td>
<td>472</td>
</tr>
<tr>
<td>setRateParam.TN93</td>
<td>473</td>
</tr>
<tr>
<td>setRateParamList.F81</td>
<td>474</td>
</tr>
<tr>
<td>setRateParamList.F84</td>
<td>475</td>
</tr>
<tr>
<td>setRateParamList.GTR</td>
<td>476</td>
</tr>
<tr>
<td>setRateParamList.HKY</td>
<td>477</td>
</tr>
<tr>
<td>setRateParamList.K80</td>
<td>478</td>
</tr>
<tr>
<td>setRateParamList.K81</td>
<td>479</td>
</tr>
<tr>
<td>setRateParamList.T92</td>
<td>480</td>
</tr>
<tr>
<td>setRateParamList.TN93</td>
<td>481</td>
</tr>
<tr>
<td>setRootNode.PhyloSim</td>
<td>482</td>
</tr>
<tr>
<td>setRootSeq.PhyloSim</td>
<td>483</td>
</tr>
<tr>
<td>setScale.BrownianInsertor</td>
<td>484</td>
</tr>
<tr>
<td>setScaledMatrix.QMatrix</td>
<td>485</td>
</tr>
<tr>
<td>setSequence.Site</td>
<td>486</td>
</tr>
<tr>
<td>setSequences.PhyloSim</td>
<td>487</td>
</tr>
<tr>
<td>setState.Event</td>
<td>488</td>
</tr>
<tr>
<td>setSiteSpecificParamIds.Process</td>
<td>489</td>
</tr>
<tr>
<td>setSiteSpecificParamList.Process</td>
<td>490</td>
</tr>
<tr>
<td>setSize.Alphabet</td>
<td>490</td>
</tr>
<tr>
<td>setSize.DiscreteDeleter</td>
<td>491</td>
</tr>
<tr>
<td>setSize.DiscreteInsertor</td>
<td>492</td>
</tr>
<tr>
<td>setState.Site</td>
<td>493</td>
</tr>
<tr>
<td>setState.Sequence</td>
<td>494</td>
</tr>
<tr>
<td>setString.Sequence</td>
<td>495</td>
</tr>
<tr>
<td>setSymbolLength.Alphabet</td>
<td>496</td>
</tr>
<tr>
<td>setSymbols.Alphabet</td>
<td>496</td>
</tr>
<tr>
<td>setTableId.CodonAlphabet</td>
<td>497</td>
</tr>
<tr>
<td>setTemplateSeq.GeneralInsertor</td>
<td>498</td>
</tr>
</tbody>
</table>
R topics documented:

setTheta.T92 ................................................................. 499
setTipLabels.PhyloSim ................................................. 500
setTips.PhyloSim ......................................................... 501
setToleranceMargin.FastFieldDeletor .............................. 502
setTotalRate.Site .......................................................... 503
setTotalRates.Sequence ............................................... 503
setTransTable.CodonAlphabet ........................................ 504
setTreeLength.PhyloSim ............................................... 505
setType.Alphabet ....................................................... 506
setType.BrownianInsertor ............................................. 507
setType.FastFieldDeletor .............................................. 507
setUniqueAlphabets.Sequence ....................................... 508
setUniqueProcesses.Sequence ...................................... 509
setWriteProtected.Alphabet .......................................... 510
setWriteProtected.Event ............................................. 511
setWriteProtected.Process ........................................... 512
setWriteProtected.QMatrix ........................................... 513
setWriteProtected.Sequence ......................................... 513
Simulate.PhyloSim ....................................................... 514
Site ................................................................. 515
stringLength ............................................................ 518
stringLengthVector .................................................... 519
summary.Alphabet ...................................................... 520
summary.AminoAcidSubst ............................................. 521
summary.BinarySubst .................................................. 522
summary.BrownianInsertor ......................................... 523
summary.CodonAlphabet .............................................. 524
summary.CodonSubst ................................................... 525
summary.ContinuousDeletor ......................................... 526
summary.ContinuousInsertor ....................................... 527
summary.DiscreteDeletor ............................................. 528
summary.DiscreteInsertor ........................................... 529
summary.Event ........................................................ 530
summary.F81 .......................................................... 531
summary.F84 ........................................................ 532
summary.FastFieldDeletor .......................................... 533
summary.GeneralDeletor .............................................. 534
summary.GeneralInDel ................................................. 535
summary.GeneralInsertor .......................................... 536
summary.GeneralSubstitution ....................................... 537
summary.GTR .......................................................... 538
summary.GY94 ........................................................ 539
summary.HKY .......................................................... 540
summary.JC69 .......................................................... 541
summary.K80 .......................................................... 541
summary.K81 .......................................................... 542
summary.PhyloSim ..................................................... 543
summary.Process ....................................................... 544
Check if two alphabet objects have the same symbol set.

Usage

```r
## S3 method for class 'Alphabet'
e1 == e2, ...
```

Arguments

- **e1**: An Alphabet object.
- **e2**: An Alphabet object.
- **...**: Not used.

Value

TRUE or FALSE

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Alphabet`. 
Examples

# create an alphabet object
a <- Alphabet(symbols = c(0, 1));
# clone object
b <- clone(a)
# compare the two objects
print(a == b)
# modify symbol set in b
b$symbols <- c("AT", "GC");
print(a == b)

==.Process

Check whether the two supplied Process objects are identical

Description

Check whether the two supplied Process objects are identical.

Usage

## S3 method for class 'Process'
e1 == e2, ...

Arguments

e1 A Process object.
e2 A Process object.
... Not used.

Value

TRUE or FALSE;

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Process.
Examples

```r
# create some Process objects
p1 <- Process()
p2 <- clone(p1)
# check object equality
p1 == p1
p1 == p2
```

The Alphabet class

**Description**

Class representing an alphabet (a set of symbols). Package: alphabet

**Class Alphabet**

```
Object
|~~|
|~~+---PSRoot
|~~~~~~|
|~~~~~~~~Alphabet
```

**Directly known subclasses:**

AminoAcidAlphabet, AnyAlphabet, BinaryAlphabet, CodonAlphabet, NucleotideAlphabet

public static class **Alphabet**

extends **PSRoot**

**Usage**

`Alphabet(symbols=NA, type="Generic", ...)`

**Arguments**

- **symbols**
  
  A character vector containing the symbols for the alphabet. All elements must have the same length and no duplicates are allowed. No element of the vector may contain the dash symbol, which is reserved for gaps.

- **type**

  An identifier for the Alphabet object.

- **...**

  Not used.

**Fields and Methods**

**Methods:**
Alphabet

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is.na, is, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
[, , [, [, $, $<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)
Botond Sipos, Gregory Jordan

See Also
Site Process Event BinaryAlphabet NucleotideAlphabet AminoAcidAlphabet

Examples
# create an alphabet object
a<-Alphabet(type="Binary",symbols=c("0","1");
# print summary
summary(a);
# change the identifier
a$type<-"Nucleotide";
# change the symbol set
a$symbols<-c("A","T","G","C");
# print summary again
summary(a);
# clone the alphabet object
b <- clone(a);
# test the equality of the symbol sets
a == b;

## AminoAcidAlphabet

### Class of Alphabet objects representing amino acids, using the one-letter IUPAC amino acid codes as symbol set:

### IUPAC code Amino acid

- A Alanine
- C Cysteine
- D Aspartic Acid
- E Glutamic Acid
- F Phenylalanine
- G Glycine
- H Histidine
- I Isoleucine
- K Lysine
- L Leucine
- M Methionine
- N Asparagine
- P Proline
- Q Glutamine
- R Arginine
- S Serine
- T Threonine
- V Valine
- W Tryptophan
- Y Tyrosine

### Package:

**Class AminoAcidAlphabet**
AminoAcidAlphabet

~~~~~~~~~~~~~~~+-AminoAcidAlphabet

**Directly known subclasses:**

public static class **AminoAcidAlphabet**
extends **Alphabet**

**Usage**

AminoAcidAlphabet(...)

**Arguments**

... Not used.

**Fields and Methods**

**Methods:**

*No methods defined.*

**Methods inherited from Alphabet:**

!=, ==, as.character, checkConsistency, getSize, getSymbolLength, getSymbols, getType, getWriteProtected, hasSymbols, is, isEmpty, setSize, setSymbolLength, setSymbols, setType, summary

**Methods inherited from PSRoot:**

checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, na, is, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

**Methods inherited from Object:**

[, [<- , $ , $<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

Alphabet

**Examples**

a<-AminoAcidAlphabet();
# get object summary
summary(a)
The AminoAcidSequence class

Description

Sequence objects aggregating Site objects having an Amino Acid Alphabet attached by default.

Package:
Class AminoAcidSequence

Object
```
| ~~~| PSRoot
| ~~~~~~~~~| Sequence
| ~~~~~~~~~~~| AminoAcidSequence
```

Directly known subclasses:

public static class AminoAcidSequence
extends Sequence

Usage

AminoAcidSequence(name=NA, string=NA, length=NA, processes=NA, ancestral.obj=NA, ...)

Arguments

| name | Name of the Sequence object. |
| string | A string specifying the length and the states of the Sequence object. |
| length | The length of the sequence. Mutually exclusive with "string". |
| processes | A list of lists of Process objects, to be attached to the aggregated Site objects. Recycled if shorter than the length of the sequence. |
| ancestral.obj | The ancestral object of the Sequence object (a valid Sequence or Process object). |
| ... | Not used. |

Fields and Methods

Methods:
No methods defined.
Methods inherited from Sequence:
as.character, attachProcess, checkConsistency, clearStates, clone, copySubSequence, deleteSubSequence, detachProcess, getAlphabets, getAncestral, getBigRate, getCumulativeRates, getCumulativeRatesFromRange, getDeletionTolerance, getEvents, getId, getInsertionTolerance, getLength, getName, getOmegas, getParameterAtSites, getProcesses, getRateMultipliers, getStates, getString, getSymbolFreqs, getTotalRates, getTotalRatesFromRange, getUniqueAlphabets, getUniqueProcesses, getWriteProtected, insertSequence, is, plot, plotParametersAtSites, plusGamma, plusInvGamma, sampleStates, setAlphabets, setAncestral, setBigRate, setCumulativeRates, setDeletionTolerance, setName, setInsertionTolerance, setLength, setOmegas, setParameterAtSites, setProcesses, setRateMultipliers, setStates, setState, setString, setTotalRates, setUniqueAlphabets, setUniqueProcesses, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is.na, is, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
[[, [<-, $, $<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstanceationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)
Botond Sipos, Gregory Jordan

See Also
Sequence AminoAcidAlphabet

Examples

# create an empty AminoAcidSequence object
s<-AminoAcidSequence(length=58)

# set states
s$states<-c("C","C","G","Y")

# create a sequence object by specifying a string
s<-AminoAcidSequence(string="CNNGYCCNGYYYY")

---

AminoAcidSubst        The AminoAcidSubst class
Description

This is a class implementing a continuous-time Markov process acting on the state-space defined by the AminoAcidAlphabet class. The rate matrix of this model is completely unrestricted.

The rate matrix can be built from PAML files specified by the pamlNfile argument. Alternatively the rates can be specified as a list through the rateNlist parameter.

Package:
Class AminoAcidSubst

Usage
AminoAcidSubst(name="Anonymous", pamlNfile=NA, rateNlist=NA, equNdist=NA, ...)

Arguments
name
The name of the object.
pamlNfile
The name of the PAML file used to construct the rate matrix.
rateNlist
A list of unscaled substitution rates (see setRateList.GeneralSubstitution).
equNdist
Equilibrium distribution.
...
Additional arguments.

Fields and Methods

Methods:

buildFromPAML -
checkConsistency -
newAAMatrix -
setEquDist -
Methods inherited from GeneralSubstitution:
  as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

Methods inherited from Process:
  !=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
  checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is.na, is, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
  [, [, , $, $<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstanceationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `AminoAcidSubst`.

Examples

```r
# create an object
p <- AminoAcidSubst()
# build rate matrix from paml file
# buildFromPAML(p, "path_to_paml_file") # do not run this
# set a rate
setRate(p, "A->D", 2)
# get object summary
summary(p)
```
The AnyAlphabet class

Description

This is a special Alphabet class which matches any alphabet. The ‘==’.Alphabet method always returns TRUE when one of the compared objects inherits from AnyAlphabet. This behaviour is handy when creating processes that have no alphabet preference (like a deletion process).

Package:

Class AnyAlphabet

Usage

AnyAlphabet(...)

Arguments

... Not used.

Fields and Methods

Methods: No methods defined.

Methods inherited from Alphabet: !=, ==, as.character, checkConsistency, getSize, getSymbolLength, getSymbols, getType, getWriteProtected, hasSymbols, is, isEmpty, setSize, setSymbolLength, setSymbols, setType, summary

Methods inherited from PSRoot: checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, is, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden
areSynonymous.CodonAlphabet

Methods inherited from Object:
[[<-, $, $<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach,
equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)
Botond Sipos, Gregory Jordan

See Also
Alphabet

Examples

# create some alphabet objects
a<-BinaryAlphabet()
b<-NucleotideAlphabet()
any<-AnyAlphabet()
# compare objects
a == b
any == a
any == b

---

areSynonymous.CodonAlphabet

*Check whether two codons are synonymous*

Description

Check whether two codons are synonymous.

Usage

## S3 method for class 'CodonAlphabet'
areSynonymous(this, codons, ...)

Arguments

- **this**
  A CodonAlphabet object.
- **codons**
  A vector containing two codons.
- **...**
  Not used.

Value

TRUE or FALSE.
as.character.Alphabet

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see CodonAlphabet.

Examples

```r
C create a CodonAlphabet object
a<-CodonAlphabet()
areSynonymous(a,c("TCC","TCT"))
areSynonymous(a,c("TCC","CCT"))
```

Description

Get the character representation of an Alphabet object.

Usage

```r
## S3 method for class 'Alphabet'
as.character(x, ...)
```

Arguments

- `x` An Alphabet object
- `...` Not used.

Value

Returns the character vector containing the symbol set.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Alphabet.
Examples

```r
# create alphabet object
a <- Alphabet(symbols = c("A", "T", "G", "C", "N"))
# get character representation
as.character(a)
```

Description

Get the character representation of an Event object.

The character representation of an Event object has the following format: "event name" ("event rate")
`<-- "generator process id", like "A->T (0.3333333333333333) <-- JC69:Anonymous:44780832".`

Usage

```r
## S3 method for class 'Event'
as.character(x, ...)
```

Arguments

- `x`: An Event object.
- `...`: Not used.

Value

A character vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Event`.

Examples

```r
# create a sequence and attach a process
s <- NucleotideSequence(string = "ATGC", processes = list(list(JC69())))
# get the first active event from the first site
e <- s$sites[[1]]$events[[1]]
# get the character representation of e
as.character(e)
# or more simply
e
```
Description

Return the character representation of a GeneralSubstitution object. The character representation is
the object id as returned by the getId.Process method defined in the parent class.

Usage

```r
## S3 method for class 'GeneralSubstitution'
as.character(x, ...)
```

Arguments

- `x` : A GeneralSubstitution object.
- `...` : Not used.

Value

A character vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `GeneralSubstitution`.

Examples

```r
# create a GeneralSubstitution object
p<-GeneralSubstitution(name="MySubst")
# get character representation
as.character(p)
# the same implicitly
p
```
as.character.PhyloSim

Return the character representation of a PhyloSim object

Description

Return the character representation of a PhyloSim object.

The character representation is the identifier of the PhyloSim object as returned by the getId method.

Usage

## S3 method for class 'PhyloSim'
as.character(x, ...)

Arguments

x
A PhyloSim object.

... Not used.

Value

A character vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see PhyloSim.

Examples

# create a PhyloSim object
o<-PhyloSim(name="MySim");
# get character representation
as.character(o)
Description

Get the character representation of a Process object. The string returned is the unique Process object identifier (class name + process name + object hash).

Usage

```r
## S3 method for class 'Process'
as.character(x, ...)
```

Arguments

- `x`: A Process object
- `...`: Not used.

Value

A character vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Process`.

Examples

```r
# create a Process object
p <- Process()
# get character representation
x <- as.character(p)
print(x)
```
as.character.QMatrix

Return the character representation of a QMatrix object

Description

Return the character representation of a QMatrix object.

Usage

```r
## S3 method for class 'QMatrix'
as.character(x, ...)
```

Arguments

- `x`: A QMatrix object.
- `...`: Not used.

Value

A character vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `qmatrix`.

Examples

```r
# create a QMatrix object
m <- QMatrix()
# get the character representation
as.character(m)
# the same, but implicitly
m
```
as.character.Sequence  *Get the string representation of a Sequence object*

Description

Get the string representation of a Sequence object. The string representation is the concatenation of the states of the aggregated Site object. Undefined states (NA-s) are represented by question marks.

Usage

```r
## S3 method for class 'Sequence'
as.character(x, ...)
```

Arguments

- `x` A Sequence object.
- `...` Not used.

Value

A character vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Sequence`.

Examples

```r
# create object
s <- Sequence(length=10)
# get character representation
as.character(s)
```
as.character.Site

Get the character representation of a Site object

Description

Get the character representation of a Site object.

Usage

## S3 method for class 'Site'
as.character(x, ...)

Arguments

x    A Site object.
...
Not used.

Value

A character vector of length one containing the current state.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Site.

Examples

# create site object
s<-Site(alphabet=NucleotideAlphabet(),state="A")
# get character representation
x<-as.character(s)
x
attachHookToNode.PhyloSim

Attach a callback function to a given node of a phylo object aggregated by a PhyloSim object.

Description

Attach a callback function to a given node of a phylo object aggregated by a PhyloSim object.

A "node hook" is a function which accepts a Sequence object through the named argument "seq" and returns a Sequence object. The node hook function must accept any object which inherits from the Sequence class!

After simulating the branch leading to the node, the resulting Sequence object is passed to the node hook and the returned object is used to simulate the downstream branches.

By using node hooks the attached processes can be replaced during simulation, hence enabling the simulation of non-homogeneous sequence evolution.

Usage

```r
C3 sS method for class 'PhyloSim'
attachHookToNode(this, node=NA, fun=NA, ...)
```

Arguments

- `this`: A PhyloSim object.
- `node`: Node identifier.
- `fun`: A function (see above).
- `...`: Not used.

Value

The PhyloSim object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see PhyloSim.
### Examples

```r
# Create a PhyloSim object.
# Provide the phylo object
# and the root sequence.
sim <- PhyloSim(
  name = "TinySim",
  phylo = rcoal(3),
  root.seq = NucleotideSequence(string = "ATGC", processes = list(JC69())))

# create a node hook function
hook <- function(seq = NA){
  # replace the substitution process with F84
  if(inherits(seq, "NucleotideSequence")){
    cat("Replacing JC69 with F84 \n");
    seq$processes <- list(F84(rate.params = list("Kappa" = 2))));
  }
  return(seq);
}

# attach hook function to node 5
attachHookToNode(sim, 5, hook);

# Run the simulation
simulate(sim);

# Check if the processes have been truly replaced
lapply(sim$sequences, getUniqueProcesses.Sequence)
# Print the resulting alignment
sim$alignment
```

### Description

Attach a Process object to a set of Site objects aggregated by a Sequence object.

### Usage

```r
## S3 method for class 'Sequence'
attachProcess(this, process, index, ...)
```

### Arguments

- **this**: A Sequence object.
- **process**: A Process object.
- **index**: An integer vector specifying a set of positions. It is set to 1:seq$length if omitted.
- **...**: Not used.
attachProcess.Site

**Value**

The Sequence object (invisible).

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `Sequence`.

**Examples**

```r
# create a Sequence object of length 6
s <- Sequence(length=10, alphabets=list(NucleotideAlphabet()))
# attach a JC69 substitution process
attachProcess(s, JC69())
# get the list of attached processes
s$processes
# attach the GTR substitution process to range 3:6
attachProcess(s, GTR(3:6))
# get the list of attached processes
s$processes
```

---

**attachProcess.Site**  
*Attach a Process object to a Site object*

**Description**

Attach a Process object to a Site object.

The Alphabet objects associated with the Site and Process objects must have the same symbol set, or at least one of them should inherit from the class `AnyAlphabet`.

During the attachment, the site-process specific parameter templates are copied from the Process object and stored in the Site object. The Process objects are marked as write protected if the attachment was successful.

**Usage**

```r
## S3 method for class 'Site'
attachProcess(this, process, ...)
```

**Arguments**

- `this`  
  A Site object.
- `process`  
  A Process object.
- `...`  
  Not used.
Value

The Site object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

Site Process detachProcess setProcesses getProcesses isAttached

Examples

# create a Site object and the associated NucleotideAlphabet object
s<-Site(alphabet=NucleotideAlphabet())
# create a K80 substitution process
p<-K80()
# attach p to s
attachProcess(s,p)
# get the list of attached processes
s$processes
# check write protection for p
p$writeProtected

attachSeqToNode.PhyloSim

Associate a Sequence object with a given node of a phylo object aggregated by a PhyloSim object

Description

Associate a Sequence object with a given node of a phylo object aggregated by a PhyloSim object.
This method is mainly used internally.

Usage

## S3 method for class 'PhyloSim'
attachSeqToNode(this, node=NA, seq=NA, ...)

Arguments

this A PhyloSim object.
node Node identifier.
seq A Sequence object.
... Not used.
Value

The PhyloSim object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see PhyloSim.

---

**BinaryAlphabet**

*The BinaryAlphabet class*

---

Description

Class of Alphabet objects with the c("0","1") symbol set.

Package:

**Class BinaryAlphabet**

```
Object
~~|  
~~---PSRoot
~~~~~~|  
~~~~~~~~--Alphabet
~~~~~~~~~~|  
~~~~~~~~~~~binaryalphabet
```

Directly known subclasses:

public static class **BinaryAlphabet**
extends **Alphabet**

Usage

BinaryAlphabet(...)

Arguments

... Not used.
Fields and Methods

Methods:
No methods defined.

Methods inherited from Alphabet:
!=, ==, as.character, checkConsistency, getSize, getSymbolLength, getSymbols, getType, getWriteProtected, hasSymbols, is, isEmpty, setSize, setSymbolLength, setSymbols, setType, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, na, is, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
[, [[ <-, $, $<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)
Botond Sipos, Gregory Jordan

See Also
Alphabet

Examples

# create a binary alphabet
b<-BinaryAlphabet()
# get alphabet summary
summary(b)

---

The BinarySequence class

Description

Sequence objects aggregating Site objects having a BinaryAlphabet attached by default.

Package:

Class BinarySequence

Object
~~|
~~+--PSRoot
~~~~~~|
~~~~~~~~+--Sequence
~~~~~~~~~~~~|
BinarySequence

Directly known subclasses:

public static class BinarySequence
extends Sequence

Usage

BinarySequence(name=NA, string=NA, length=NA, processes=NA, ancestral.obj=NA, ...)

Arguments

name       Name of the Sequence object.
string     A string specifying the length and the states of the Sequence object.
length     The length of the sequence. Mutually exclusive with "string".
processes  A list of lists of Process objects, to be attached to the aggregated Site objects. Recycled if shorter than the length of the sequence.
ancestral.obj The ancestral object of the Sequence object (a valid Sequence or Process object). Not used.

Fields and Methods

Methods:
No methods defined.

Methods inherited from Sequence:
as.character, attachProcess, checkConsistency, clearStates, clone, copySubSequence, deleteSubSequence, detachProcess, getAlphabets, getAncestral, getBigRate, getCumulativeRates, getCumulativeRatesFromRange, getDeletionTolerance, getEvents, getId, getInsertionTolerance, getLength, getName, getOmegas, getParameterAtSites, getProcesses, getRateMultipliers, getStates, getString, getSymbolFreqs, getTotalRates, getTotalRatesFromRange, getUniqueAlphabets, getUniqueProcesses, getWriteProtected, insertSequence, is, plot, plotParametersAtSites, plusGamma, plusInvGamma, sampleStates, setAlphabets, setAncestral, setBigRate, setCumulativeRates, setDeletionTolerance, setId, setInsertionTolerance, setLength, setName, setOmegas, setParameterAtSites, setProcesses, setRateMultipliers, setStates, setUsername, setTotalRates, setUniqueAlphabets, setUniqueProcesses, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is.na, is, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
[. , [[<-, $, $<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldName, getFieldName, getFields, getInstantiationTime, getStaticInstance, hasField, hashCache, ll, load, objectSize, print, registerFinalizer, save
Author(s)
Botond Sipos, Gregory Jordan

See Also
Sequence BinaryAlphabet

Examples

```c
# create an empty BinarySequence object
s<-BinarySequence(length=50)

# set states
s$states<-c(0,0,1,0,1,1)

# create a sequence object by specifying a string
s<-BinarySequence(string="0000110010001111")
```

---

**BinarySubst**

The *BinarySubst* class

---

**Description**

This is a class implementing a continuous-time Markov process acting on the state space defined by the BinaryAlphabet class.

Package: **Class BinarySubst**

```
Object
~~|  
~~++--PSRoot
~~~~~~|  
~~~~~~~~--Process
~~~~~~~~~~|  
~~~~~~~~~~~~--GeneralSubstitution
~~~~~~~~~~~~~~|  
~~~~~~~~~~~~~~~~~~~~~~--BinarySubst
```

**Directly known subclasses:**

public static class **BinarySubst**
extends **GeneralSubstitution**
Usage

BinarySubst(name = "Anonymous", rate.list = NA, equ.dist = NA, ...)

Arguments

name The name of the object.
rate.list A list of substitution rates (see setRateList-GeneralSubstitution).
equ.dist Equilibrium distribution.
... Additional arguments.

Fields and Methods

Methods:

checkConsistency -
summary -

Methods inherited from GeneralSubstitution:
as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is.na, is, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
[.], [[<-, $, $<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)

Botond Sipos, Gregory Jordan

See Also

GeneralSubstitution GTR WAG
**Examples**

```r
# The following code demonstrates
# the use of the BinarySubst process
# during a simulation.
p <- BinarySubst(rate=0.25, name="Binary", rate.list=list("0->1"=2,"1->0"=1))
# create a sequence object, attach process p
s <- BinarySequence(string="0000000000000000", processes=list(list(p)));
# make the first five positions invariable
setRateMultipliers(s,p,0,1:5)
# get rate multipliers
getRateMultipliers(s,p)
# simulate
sim <- PhyloSim(root.seq=s, phylo=coal(3))
Simulate(sim)
# print alignment
sim$alignment
```

---

**BrownianInsertor**

The BrownianInsertor class inherits from the DiscreteInsertor or ContinuousInsertor class depending on the type constructor argument ("discrete" or "continuous").

This process generates the insert sequence based on the sites flanking the insertions as follows:

- An insert length is sampled by calling the function stored in the proposeBy virtual field.
- A sequence object is constructed.
- The processes attached to both flanking sites are attached to the insert sequence. If there are no common processes, the processes from a randomly chosen site will be attached to the insert.
- The site-process specific parameters are sampled from Brownian paths with linear trends having the values from the flanking sites as endpoints.

The "noisiness" of the Brownian path can be controlled through the scale virtual field/constructor parameter.

**Package:**

**Class BrownianInsertor**

```
Object
|   |
|   | PSRoot
|   | ~----------Process
|   | ~---------------GeneralInDel
```
BrownianInsertor

Directly known subclasses:

public static class BrownianInsertor
extends DiscreteInsertor

Usage

BrownianInsertor(name="Anonymous", type="discrete", scale=0.001, ...)

Arguments

name Object name.
type Process type (see above).
scale Brownian path scale parameter.
... Additional arguments.

Fields and Methods

Methods:

BrownianPath -
checkConsistency -
getScale -
getType -
setScale -
setType -
summary -

Methods inherited from DiscreteInsertor:
checkConsistency, getProbs, getSizes, plot, setProbs, setSize, summary

Methods inherited from GeneralInsertor:
checkConsistency, generateInsert, getAcceptWin, getEventsAtSite, getGenerateBy, getInsertHook, getTemplateSeq, is, setAcceptWin, setGenerateBy, setInsertHook, setTemplateSeq, summary

Methods inherited from GeneralInDel:
checkConsistency, getAcceptBy, getProposeBy, getRate, hasUndefinedRate, is, proposeLength, setAcceptBy, setProposeBy, setRate, summary
Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is.na, is, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
[, [, $, $<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)
Botond Sipos, Gregory Jordan

See Also
DiscreteInsertor ContinuousInsertor GeneralInsertor GeneralInDel

Examples

# create a BrownianInsertor process, discrete type
p <- BrownianInsertor(
  type="discrete",
  scale=0.05,
  sizes=1:4,
  probs=c(3/6,1/6,1/6,1/6),
  rate=0.05
)

# get object summary
summary(p)
# plot insert length distribution
plot(p)
# create a nucleotide sequence, attach processes
s <- NucleotideSequence(string="AAAAAAAAAA", processes=list(list(p,JC69())))
# create simulation object
sim <- PhyloSim(root.seq=s, phylo=rcoal(2))
# simulate and show alignment
Simulate(sim)
sim$alignment
# check the rate multipliers and insertion tolerances in one of the sequences
res <- sim$sequences[[2]]
getRateMultipliers(res,p)
geInsertionTolerance(res,p)
Generate a Brownian path.

This method generates a Brownian path given the scale parameter \( a \) (determining "noisiness") and the vector \( p \) describing the trends. More useful as a static method.

**Usage**

```r
## S3 method for class 'BrownianInsertor'
BrownianPath(this, p=NA, a=NA, ...)
```

**Arguments**

- `this` A BrownianInsertor object.
- `p` Path parameter (a numeric vector).
- `a` Scale parameter (a numeric vector of length one).
- `...` Not used.

**Value**

A numeric vector.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `BrownianInsertor`.

**Examples**

```r
path<-BrownianInsertor$BrownianPath(a=2,p=1:10); path plot(path)
```
Build rate matrix from PAML file

Description

Build rate matrix from PAML file.

Usage

```r
## S3 method for class 'AminoAcidSubst'
buildFromPAML(this, paml.file, ...)
```

Arguments

- `this` An AminoAcidSubst object.
- `paml.file` Path to the PAML file.
- `...` Not used.

Value

The AminoAcidSubst object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `AminoAcidSubst`.

Examples

```r
# create an object
p<-AminoAcidSubst()
# build rate matrix from paml file
# buildFromPAML(p,"path_to_paml_file") # do not run this
# get object summary
summary(p)
```
### buildFromPAML.CodonSubst

**Build rate matrix from PAML file**

#### Description

Build rate matrix from PAML file.

#### Usage

```r
## S3 method for class 'CodonSubst'
buildFromPAML(this, paml.file, ...)
```

#### Arguments

- `this`: A CodonSubst object.
- `paml.file`: Path to the PAML file.
- `...`: Not used.

#### Value

The CodonSubst object (invisible).

#### Author(s)

Botond Sipos, Gregory Jordan

#### See Also

For more information see `CodonSubst`.

#### Examples

```r
# create an object
p <- CodonSubst()
# build rate matrix from paml file
# buildFromPAML(p,"path_to_paml_file") # do not run this
# get object summary
summary(p)
```
Description

Check object consistency.

Usage

```r
## S3 method for class 'Alphabet'
checkConsistency(this, ...)
```

Arguments

- `this` An object.
- `...` Not used.

Value

Returns an invisible TRUE if no inconsistencies found, throws an error otherwise.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `alphabet`.

Examples

```r
# create an alphabet object
a <- Alphabet(symbols = c(0, 1));
# check consistency
print(checkConsistency(a));
# mess up with the internals
a$symbols[1] <- "BAD";
# NOT run: consistency check now will throw an error
## Not run: print(checkConsistency(a));
```
checkConsistency.Amino Acid Subst

Check object consistency

Description

Check object consistency.

Usage

```r
# S3 method for class 'Amino Acid Subst'
checkConsistency(this, ...)
```

Arguments

- `this`: An object.
- `...`: Not used.

Value

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Amino Acid Subst`.

checkConsistency.Binary Subst

Check object consistency

Description

Check object consistency.

Usage

```r
# S3 method for class 'Binary Subst'
checkConsistency(this, ...)
```
Arguments

this An object.
...

Value

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see BrownianInsertor.
checkConsistency.CodonAlphabet

_Check object consistency_

**Description**

Check object consistency.

**Usage**

```r
## S3 method for class 'CodonAlphabet'
checkConsistency(this, ...)
```

**Arguments**

- `this`: An object.
- `...`: Not used.

**Value**

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `CodonAlphabet`.

---

checkConsistency.CodonSequence

_Check object consistency_

**Description**

Check object consistency.

**Usage**

```r
## S3 method for class 'CodonSequence'
checkConsistency(this, ...)
```
checkConsistency.CodonSubst

Arguments

  this    An object.
  ...    Not used.

Value

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see CodonSequence.

---

checkConsistency.CodonSubst

*Check object consistency*

**Description**

Check object consistency.

**Usage**

```r
## S3 method for class 'CodonSubst'
checkConsistency(this, ...)```

**Arguments**

  this    An object.
  ...    Not used.

**Value**

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see CodonSubst.
checkConsistency.CodonUNREST

Check object consistency

Description

Check object consistency.

Usage

```r
## S3 method for class 'CodonUNREST'
checkConsistency(this, ...)
```

Arguments

- `this`: An object.
- `...`: Not used.

Value

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `CodonUNREST`.

checkConsistency.ContinuousDeleter

Check object consistency

Description

Check object consistency.

Usage

```r
## S3 method for class 'ContinuousDeleter'
checkConsistency(this, ...)
```
Arguments

this  An object.
...  Not used.

Value

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see ContinuousDeletor.

checkConsistency. ContinuousInsertor

Check object consistency

Description

Check object consistency.

Usage

```r
## S3 method for class 'ContinuousInsertor'
checkConsistency(this, ...)  
```

Arguments

this  An object.
...  Not used.

Value

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see ContinuousInsertor.
checkConsistency.DiscreteDeletor

Check object consistency

Description
Check object consistency.

Usage
```r
## S3 method for class 'DiscreteDeletor'
checkConsistency(this, ...)
```

Arguments
- `this` An object.
- `...` Not used.

Value
Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see DiscreteDeletor.

checkConsistency.DiscreteInsertor

Check object consistency

Description
Check object consistency.

Usage
```r
## S3 method for class 'DiscreteInsertor'
checkConsistency(this, ...)
```
checkConsistency.Event

Arguments

this An object.
... Not used.

Value

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see DiscreteInsertor.

---

Description

Check object consistency.

Usage

```r
# S3 method for class 'Event'
checkConsistency(this, ...)
```

Arguments

this An object.
... Not used.

Value

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Event.
checkConsistency.F81  Check object consistency

Description

Check object consistency.

Usage

```r
# S3 method for class 'F81'
checkConsistency(this, ...)
```

Arguments

- `this`: An object.
- `...`: Not used.

Value

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `F81`.

checkConsistency.F84  Check object consistency

Description

Check object consistency.

Usage

```r
# S3 method for class 'F84'
checkConsistency(this, ...)
```

Arguments

- `this`: An object.
- `...`: Not used.
Value

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see f84.
checkConsistency.\textit{GeneralDeleter}

\textit{Check object consistency}

\subsection*{Description}
Check object consistency.

\subsection*{Usage}

\begin{verbatim}
## S3 method for class 'GeneralDeleter'
checkConsistency(this, ...)
\end{verbatim}

\subsection*{Arguments}
\begin{itemize}
  \item \texttt{this} \hspace{1cm} An object.
  \item \texttt{...} \hspace{1cm} Not used.
\end{itemize}

\subsection*{Value}
Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

\subsection*{Author(s)}
Botond Sipos, Gregory Jordan

\subsection*{See Also}
For more information see \texttt{GeneralDeleter}.

---

checkConsistency.\textit{GeneralInDel}

\textit{Check object consistency}

\subsection*{Description}
Check object consistency.

\subsection*{Usage}

\begin{verbatim}
## S3 method for class 'GeneralInDel'
checkConsistency(this, ...)
\end{verbatim}
checkConsistency. GeneralInsertor

**Arguments**

- `this`: An object.
- `...`: Not used.

**Value**

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `GeneralInDel`.

---

checkConsistency. GeneralInsertor

*Check object consistency*

---

**Description**

Check object consistency.

**Usage**

```r
# S3 method for class 'GeneralInsertor'
checkConsistency(this, ...)
```

**Arguments**

- `this`: An object.
- `...`: Not used.

**Value**

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `GeneralInsertor`.
Description

Check object consistency.

Usage

\[
\text{checkConsistency}(\text{this}, \ldots)
\]

Arguments

- **this**: An object.
- **...**: Not used.

Value

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see \texttt{GeneralSubstitution}.
checkConsistency.GY94

Value

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see GTR.

checkConsistency.GY94  *Check object consistency*

Description

Check object consistency.

Usage

```r
## S3 method for class 'GY94'
checkConsistency(this, ...)
```

Arguments

- `this`: An object.
- `...`: Not used.

Value

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see GY94.
checkConsistency.HKY  Check object consistency

Description
Check object consistency.

Usage

```r
## S3 method for class 'HKY'
checkConsistency(this, ...)
```

Arguments
- `this` An object.
- `...` Not used.

Value
Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see HKY.

checkConsistency.JC69  Check object consistency

Description
Check object consistency.

Usage

```r
## S3 method for class 'JC69'
checkConsistency(this, ...)
```

Arguments
- `this` An object.
- `...` Not used.
Value

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see JC69.
checkConsistency.K81  

Description  
Check object consistency.

Usage  
```r  
## S3 method for class 'K81'
checkConsistency(this, ...)  
```

Arguments  
- `this`  An object.
- `...`  Not used.

Value  
Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)  
Botond Sipos, Gregory Jordan

See Also  
For more information see K81.

checkConsistency.PhylSim  

Description  
Check object consistency.

Usage  
```r  
## S3 method for class 'PhylSim'
checkConsistency(this, ...)  
```

Arguments  
- `this`  An object.
- `...`  Not used.
checkConsistency.Process

Value

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see PhyloSim.

---

checkConsistency.Process

*Check object consistency*

---

Description

Check object consistency.

Usage

```r
## S3 method for class 'Process'
checkConsistency(this, ...)
```

Arguments

- `this` An object.
- `...` Not used.

Value

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Process.
checkConsistency.PSRoot

*Check object consistency*

**Description**

Check object consistency. The consistency check is not implemented in plain PSRoot objects, the method prints out a warning about that.

**Usage**

```r
## S3 method for class 'PSRoot'
checkConsistency(this, ...)
```

**Arguments**

- `this`: An object.
- `...`: Not used.

**Value**

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `PSRoot`.

---

checkConsistency.PSRootSummary

*Check object consistency*

**Description**

Check object consistency. The consistency check is not implemented for PSRootSummary objects, the method prints out a warning about that.

**Usage**

```r
## S3 method for class 'PSRootSummary'
checkConsistency(this, ...)
```
Arguments

this An object.
... Not used.

Value

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `PSRootSummary`.
checkConsistency.Sequence

Check object consistency

Description

Check object consistency.

Usage

```r
## S3 method for class 'Sequence'
checkConsistency(this, omit.sites=FALSE, ...)
```

Arguments

- `this`: An object.
- `omit.sites`: Do not check aggregated site objects.
- `...`: Not used.

Value

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Sequence`.

checkConsistency.Site

Check object consistency

Description

Check object consistency.

Usage

```r
## S3 method for class 'Site'
checkConsistency(this, ...)
```
checkConsistency.T92

Arguments

this An object.
...

Value

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Site.

checkConsistency.T92  Check object consistency

Description

Check object consistency.

Usage

## S3 method for class 'T92'
checkConsistency(this, ...)

Arguments

this An object.
...

Value

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see T92.
checkConsistency.TN93  
*Check object consistency*

**Description**

Check object consistency.

**Usage**

```r
## S3 method for class 'TN93'
checkConsistency(this, ...)
```

**Arguments**

- `this`  
  An object.

- `...`  
  Not used.

**Value**

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see [TN93](#).

---

checkConsistency.UNREST  
*Check object consistency*

**Description**

Check object consistency.

**Usage**

```r
## S3 method for class 'UNREST'
checkConsistency(this, ...)
```

**Arguments**

- `this`  
  An object.

- `...`  
  Not used.
clearStates.Sequence

Value

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see UNREST.

clearStates.Sequence  Set the states of a collection of Site objects aggregated by a Sequence object to undefined (NA)

Description

Set the states of a collection of Site objects aggregated by a Sequence object to undefined (NA).

Usage

## S3 method for class 'Sequence'
clearStates(this, index, ...)

Arguments

this  
A Sequence object.

index  
An integer vector specifying a set of positions. It is set to 1:seq$length if omitted.

...  
Not used.

Value

The Sequence object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Sequence.
Examples

```r
# create a nucleotide sequence
s <- NucleotideSequence(string = "ATGC")

# set states to NA in the range 2:3
clearStates(s, 2:3)

# set all states to NA
clearStates(s)
```

Description

Clone a GeneralSubstitution object.

This method also clones the aggregated QMatrix object, but not the aggregated Alphabet object, as that is a good target for recycling.

Usage

```r
## S3 method for class 'GeneralSubstitution'
clone(this, ...)
```

Arguments

- `this`: A GeneralSubstitution object.
- `...`: Not used.

Value

A GeneralSubstitution object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `GeneralSubstitution`.
Examples

```r
# create a GeneralSubstitution object
p <- GeneralSubstitution(  
  alphabet=BinaryAlphabet(),  
  rate.list=list("0->1"=1,"1->0"=2),  
  name="MyBinary"
)

# clone p
pp <- clone(p)
# do some checks
p; pp
p == p
p == pp
equals(p$qMatrix, pp$qMatrix)
```

---

**Description**

Clone a process object. Write protection is set to FALSE for the new Process object.

**Usage**

```r
## S3 method for class 'Process'
clone(this, ...)
```

**Arguments**

- `this` A Process object.
- `...` Not used.

**Value**

A Process object.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `Process`. 
Examples

```r
# create a process object
p <- Process()
# clone process object
pp <- clone(p)
# check identity
p == pp
```

Description

Clone a Sequence object.

The cloning of Sequence objects involves the cloning of all aggregated Site objects. Because of that the cloning of long sequences is quite expensive. The cloned Site objects have the original Site objects as ancestral. The new Sequence objects has the original object as ancestral.

Usage

```r
## S3 method for class 'Sequence'
clone(this, ...)
```

Arguments

- `this` A Sequence object.
- `...` Not used.

Value

A Sequence object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

Sequence clone.Object
Examples

# create a nucleotide sequence
s<-NucleotideSequence(string="ATG")
# clone the sequence
cs<-clone(s)
# get some properties
equals(s,s)
equals(s,cs)
cs$ancestral
cs$sites[[1]]$ancestral

---

CodonAlphabet The CodonAlphabet class

Description

This class implements codon alphabets and handles the translation of codons to their corresponding amino acids. Stop codons are excluded from the symbol set, thus the symbol set depends on the genetic code table. The genetic code table can be specified through the `table.id` constructor parameter.

The available genetic code tables:

1 Standard  
2 Vertebrate Mitochondrial  
3 Yeast Mitochondrial  
4 Mold, Protozoan, and Coelenterate Mitochondrial and Mycoplasma/Spiroplasma  
5 Invertebrate Mitochondrial  
6 Ciliate, Dasycladacean and Hexamita Nuclear  
9 Echinoderm Mitochondrial  
10 Euplotid Nuclear  
11 Bacterial  
12 Alternative Yeast Nuclear  
13 Ascidian Mitochondrial  
14 Flatworm Mitochondrial  
15 Blepharisma Nuclear  
16 Chlorophycean Mitochondrial  
21 Trematode Mitochondrial  
22 Scenedesmus obliquus Mitochondrial  
23 Thraustochytrium Mitochondrial

Package:  
Class CodonAlphabet

Object

~~
Directly known subclasses:

public static class CodonAlphabet
extends Alphabet

Usage

CodonAlphabet(table.id=1, ...)

Arguments

table.id  The identifier of the genetic code table.
...  Not used.

Methods:

areSynonymous  -
checkConsistency  -
getTableId  -
getTransTable  -
is  -
isStartCodon  -
isStopCodon  -
setTableId  -
setTransTable  -
summary  -
translateCodon  -

Methods inherited from Alphabet:
!=, ==, as.character, checkConsistency, getSize, getSymbolLength, getSymbols, getType, getWriteProtected, hasSymbols, is, isEmpty, setSize, setSymbolLength, setSymbols, setType, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, is, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden
Methods inherited from Object:
[[, [[<-, $, $<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see CodonAlphabet.

Examples

```r
# create a CodonAlphabet object
a <- CodonAlphabet(table.id=2)
# get object summary
summary(a)
```

---

The CodonSequence class

Description

Sequence objects aggregating Site objects having a CodonAlphabet attached by default.

Package:

Class CodonSequence

```
Object
|--|
|---PSRoot
|   |
|   +---Sequence
|   |   |
|   |   +---CodonSequence

Directly known subclasses:

public static class CodonSequence
extends Sequence
CodonSequence

Usage

CodonSequence(
  name=NA, string=NA, length=NA,
  table.id=1, processes=NA, ancestral.obj=NA, ...
)

Arguments

name
  Name of the Sequence object.
string
  A string specifying the length and the states of the Sequence object.
length
  The length of the sequence. Mutually exclusive with "string".
table.id
  The genetic code table to use in the attached CodonAlphabet object ("Standard" by default).
processes
  A list of lists of Process objects, to be attached to the aggregated Site objects. Recycled if shorter than the length of the sequence.
ancestral.obj
  The ancestral object of the Sequence object (a valid Sequence or Process object).
...
  Not used.

Fields and Methods

Methods:

  checkConsistency -
  getOmegas -
  omegaHist -
  omegaVarM0 -
  omegaVarM1 -
  omegaVarM2 -
  omegaVarM3 -
  omegaVarM4 -
  setOmegas -
  Translate -

Methods inherited from Sequence:

as.character, attachProcess, checkConsistency, clearStates, clone, copySubSequence, deleteSubSequence, detachProcess, getAlphabets, getAncestral, getBigRate, getCumulativeRates, getCumulativeRatesFromRange, getDeletionTolerance, getEvents, getId, getInsertionTolerance, getLength, getName, getOmegas, getParameterAtSites, getProcesses, getRateMultipliers, getStates, getString, getSymbolFreqs, getTotalRates, getTotalRatesFromRange, getUniqueAlphabets, getUniqueProcesses, getWriteProtected, insertSequence, is, plot, plotParametersAtSites, plusGamma, plusInvGamma, sampleStates, setAlphabets, setAncestral, setBigRate, setCumulativeRates, setDeletionTolerance, setId, setInsertionTolerance, setLength, setName, setOmegas, setParameterAtSites, setProcesses, setRateMultipliers, setStates, setString, setTotalRates, setUniqueAlphabets, setUniqueProcesses, setWriteProtected, summary
**CodonSubst**

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.logical, is, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
[.[, ., .<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see CodonSequence.

Examples
```r
# create an empty CodonSequence object
s <- CodonSequence(length=50)
s
# set states
s$states <- c("ATG", "CGA", "TTT", "CTA")
s
# create a sequence object by specifying a string
s <- CodonSequence(string="ATCTTTCGAATG")
s
```

---

**CodonSubst**

The CodonSubst class

Description
This is a class implementing a continuous-time Markov process acting on the state-space defined by the CodonAlphabet class.

The rate matrix can be built from PAML files specified by the paml.file argument. Alternatively the rates can be specified as a list through the rate.list parameter.

Class CodonSubst

```
Object
~|  
~~++PSRoot
~~~~~~~|  
~~~~~~~~~~++Process
~~~~~~~~~~~~~~|  
```
Directly known subclasses:
ECMrest, ECMunrest

public static class CodonSubst
extends GeneralSubstitution

Usage
CodonSubst(name="Anonymous", paml.file=NA, rate.list=NA, equ.dist=NA, ...)

Arguments
name The name of the object.
paml.file The name of the PAML file used to construct the rate matrix.
rate.list A list of unscaled substitution rates (see setRateList.GeneralSubstitution).
equ.dist Equilibrium distribution.
... Additional arguments.

Fields and Methods

Methods:

buildFromPAML -
checkConsistency -
newMatrix -
setEquDist -
summary -

Methods inherited from GeneralSubstitution:
as.character, checkConsistency, clone, getAlphabet, getEquDist, getRate, getRateAtSite, getEventsAtSite, getQMatrix, getEventRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, inter-
sect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
$, $<-, [, [[, [<-., as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, save

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see CodonSubst.

Examples

# create an object
p<-CodonSubst()
# build rate matrix from paml file
# buildFromPAML(p,"path_to_paml_file") # do not run this
# get object summary
summary(p)

---
CodonUNREST The CodonUNREST class
---

Description
This class implements a time-continuous Markov process acting on a state space defined by the symbol set of a CodonAlphabet object. The rate matrix of this model is unrestricted, so it can be used to implement empirical codon models or more restricted mechanistic models.

Package:
Class CodonUNREST

Object
~~|
~~++--PSRoot
~~~~~~|
~~~~~~~~~~~~++--Process
~~~~~~~~~~~~|
~~~~~~~~~~~~~~++--GeneralSubstitution
~~~~~~~~~~~~~~~~~~|
~~~~~~~~~~~~~~~~~~++--CodonUNREST
**Directly known subclasses:**

GY94

class CodonUNREST extends GeneralSubstitution

**Usage**

CodonUNREST(name="Anonymous", table.id=1, rate.list=NA, equ.dist=NA, ...)

**Arguments**

- **name**
  - The name of the object.

- **table.id**
  - The identifier of the genetic code table (see CodonAlphabet).

- **rate.list**
  - A list of unscaled substitution rates (see setRateList.GeneralSubstitution).

- **equ.dist**
  - Equilibrium distribution.

- **...**
  - Additional arguments.

**Fields and Methods**

**Methods:**

- checkConsistency
- is

**Methods inherited from GeneralSubstitution:**

- as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

**Methods inherited from Process:**

- !=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, hasWriteProtected, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

**Methods inherited from PSRoot:**

- checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is.na, is, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

**Methods inherited from Object:**

- [, [<-., $, $<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save
ContinuousDeletor

Author(s)
Botond Sipos, Gregory Jordan

See Also
GeneralSubstitution GY94

Examples
# create a CodonUNREST object
p<-=CodonUNREST(table.id=2)
# get object summary
summary(p)

ContinuousDeletor  The ContinuousDeletor class

Description
This class implements a process which performs deletions with lengths sampled from a user-
specified R expression returning a numeric value. See GeneralDeletor for the how the deletion
processes works.

Package:
Class ContinuousDeletor

Object
~~|
~~=PSRoot
~~~~~|
~~~~~~~~~Process
~~~~~~~~~~~~| ~~~~~~~~~~~GeneralInDel
~~~~~~~~~~~~~~~~| ~~~~~~~~~~~~~~~~~| ~~~~~~~~~~~GeneralDeletor
~~~~~~~~~~~~~~~~| ~~~~~~~~~~~~~~~~~| ~~~~~~~~~~~ContinuousDeletor

Directly known subclasses:

public static class ContinuousDeletor
extends GeneralDeletor
ContinuousDeletor

Usage

ContinuousDeletor(name="Anonymous", rate=NA, dist=NA, max.length=NA, ...)

Arguments

name The name of the object.
rate The general rate.
dist The length sampling expression.
max.length Maximum event length.
... Additional arguments.

Fields and Methods

Methods:  

checkConsistency
getDist
getMaxLength
plot
setDist
setMaxLength
summary

Methods inherited from GeneralDeletor:
checkConsistency, getEventsAtSite, is, summary

Methods inherited from GeneralInDel:
checkConsistency, getAcceptBy, getProposeBy, getRate, hasUndefinedRate, is, proposeLength, setAcceptBy, setProposeBy, setRate, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is.na, is, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
[, [<- , $, $<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save
Author(s)

Botond Sipos, Gregory Jordan

See Also

GeneralDeletor DiscreteDeletor GeneralInDel

Examples

```r
# create a ContinuousDeletor process
o <- ContinuousDeletor(
  name = "Conty",
  rate = 0.25,
  dist = expression(1),
  max.length = 2
)

# get object summary
summary(o)

# set/get length sampling expression
o$dist <- expression(rnorm(1, mean=3, sd=3))

# set/get maximum event length
o$maxLength <- 4

# plot length density
plot(o)

# The following code illustrates how to use
# a ContinuousDeletor process in a simulation

# create a sequence object, attach process o
s <- NucleotideSequence(string = "AAAAAAAAAGGGGAAAAAAAAA", processes = list(list(o))

# set the deletion tolerance to zero in range 11:15
setDeletionTolerance(s, o, 0, 11:15)

# creating a region rejecting all deletions
getDeletionTolerance(s, o)

# create a simulation object
sim <- PhyloSim(root.seq = s, phylo = rcoal(2))

# simulate
Simulate(sim)

# print resulting alignment
sim$alignment
```
Description

This class implements a process which performs insertions with lengths sampled from a user-specified R expression returning a numeric value. See GeneralInsertor for the how the insertion process works.

Package:
Class ContinuousInsertor

Usage
ContinuousInsertor(name="Anonymous", rate=NA, dist=NA, max.length=NA, ...)

Arguments
name The name of the object.
rate The general rate.
dist The length sampling expression.
max.length Maximum event length.
...

Fields and Methods

Methods:

  checkConsistency -
  getDist -
  getMaxLength -
Methods inherited from GeneralInsertor:
checkConsistency, generateInsert, getAcceptWin, getEventsAtSite, getGenerateBy, getInsertHook, 
getTemplateSeq, is, setAcceptWin, setGenerateBy, setInsertHook, setTemplateSeq, summary

Methods inherited from GeneralInDel:
checkConsistency, getAcceptBy, getProposeBy, getRate, hasUndefinedRate, is, proposeLength, set-
AcceptBy, setProposeBy, setRate, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, get-
ParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSite-
SpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSite-
SpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, inter-
sect.list, is.na, is, il, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssign-
mentForbidden

Methods inherited from Object:
[[], $[-, $<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, 
equals, extend, finalize, gc, getEnvironment, getFieldModifier, getModifiers, getFields, getInstance-
tiationTime, getStaticInstance, hasField, hashCode, il, load, objectSize, print, registerFinalizer, save

Author(s)
Botond Sipos, Gregory Jordan

See Also
GeneralInsertor DiscreteInsertor GeneralInDel

Examples
# create a ContinuousInsertor process
o<-ContinuousInsertor(
    name="Conty",
    rate=0.1,
    dist=expression(1),
    max.length=2
)

# set template sequence
o$templateSeq<-NucleotideSequence(string="CAC")

# get object summary
summary(o)

# set/get length sampling expression
copySubSequence.Sequence

Copy a collection of Site objects aggregated by a Sequence object into a new Sequence object

Description

Copy a collection of Site objects aggregated by a Sequence object into a new Sequence object.

Usage

```r
## S3 method for class 'Sequence'
copySubSequence(this, index, process=NA, ...)
```

Arguments

- **this**: A Sequence object.
- **index**: An index vector specifying a collection of sites to be copied. It is set to 1:seq$length if omitted.
- **process**: The Process object performing the copy (optional).
- **...**: Not used.
Value

A new Sequence object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Sequence.

Examples

```r
# create a nucleotide sequence
s <- NucleotideSequence(string = "ATATATATATATA")
# copy sites in the range 3:8 in a new object
s2 <- copySubSequence(s, 3:8)
s2
# copy sites 1,3 and 5 from s2
s3 <- copySubSequence(s2, c(1, 3, 5))
s3
```

---

### cpREV

*The cpREV empirical amino acid substitution model*

---

Description

Package: cpREV

Class cpREV

```
Object
| ~~~ |
| ~~~+-PSRoot
| ~~~~~~~~~~| ~~~+-Process
| ~~~~~~~~~~| ~~~~~~~~~~+-GeneralSubstitution
| ~~~~~~~~~~| ~~~~~~~~~~| ~~~~~~~~~~+-AminoAcidSubst
| ~~~~~~~~~~| ~~~~~~~~~~| ~~~~~~~~~~| ~~~~~~~~~~+-cpREV
```

Directly known subclasses:
public static class cpREV
extends AminoAcidSubst

Usage

cpREV(equ.dist=NA, ...)

Arguments

equ.dist            Equilibrium distribution.
...                  Not used.

Fields and Methods

Methods:
No methods defined.

Methods inherited from AminoAcidSubst:
buildFromPAML, checkConsistency, newAAMatrix, setEquDist, summary

Methods inherited from GeneralSubstitution:
as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, has UndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, has UndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is.na, is, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
[, [[]<-, $, $<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)

Botond Sipos, Gregory Jordan

References

See Also

AminoAcidSubst GeneralSubstitution UNREST

Examples

```r
# create substitution model object
p <- cpREV()
# get object summary
summary(p)
# display a bubble plot
plot(p)

  # The following code demonstrates how to use
  # the process in a simulation.

# create a sequence, attach process p
s <- AminoAcidSequence(length=10, processes=list(list(p)))
# sample states
sampleStates(s)
  # make the first three positions invariable
setRateMultipliers(s, p, 0, 1:3)
  # get rate multipliers
getRateMultipliers(s, p)
  # create a simulation object
sim <- PhyloSim(root.seq = s, phylo = rcoal(2))
  # run simulation
Simulate(sim)
  # print alignment
sim$alignment
```

Description

Save a debug message in the PhyloSim log file.

The debug message is written to the log file only if the log level is non-negative. You can use this method for logging debug messages in the case you write classes for PhyloSim.

Usage

```r
## S3 method for class 'PhyloSim'
Debug(this, message, ...)
```

Arguments

- `this`: A PhyloSim object.
- `message`: A character vector of length one.
- `...`: Not used.
Value
The message (invisible).

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see PhyloSim.

Examples
```
# create a PhyloSim object,
# with logLevel set to zero
sim<-PhyloSim(log.level=0);
# log a debug message
Debug(sim,"Some useful detail...");
# close log connection
close(sim$.log.connection)
# print out the log file
cat(paste(scan(file=sim$LogFile,what=character(),sep="\n"),collapse="\n"));cat("\n");
# clean up
unlink(sim$logfile)
```
Value
Invisible TRUE or error.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see Sequence.

Examples
# create a nucleotide sequence
s <- NucleotideSequence(string = "ATATATATATATAT")
# delete sites 2, 4 and 6
deleteSubSequence(s, c(2, 4, 6))
s
# delete sites in the range 3:6
deleteSubSequence(s, 3:6)
s

detachHookFromNode.PhyloSim

Detach a node hook function from a given node of a phylo object aggregated by a PhyloSim object

Description
Detach a node hook function from a given node of a phylo object aggregated by a PhyloSim object.

Usage
## S3 method for class 'PhyloSim'
detachHookFromNode(this, node = NA, ...)

Arguments
this A PhyloSim object.
node Node identifier.
... Not used.

Value
The PhyloSim object (invisible).
Author(s)
Botond Sipos, Gregory Jordan

See Also
attachHookToNode PhyloSim Simulate.PhyloSim

Examples

# Create a PhyloSim object.
# Provide the phylo object
# and the root sequence.
sim<-PhyloSim(
  name="TinySim",
  phylo=rcoal(3),
  root.seq=NucleotideSequence(string="ATGC",processes=list(JC69)))

# create a node hook function
hook<-function(seq=NA){
  # replace the substitution process with F84
  if(inherits(seq,"NucleotideSequence")){
  cat("Replacing JC69 with F84.
  seq$processes<-list(list(F84(rate.params=list("Kappa" = 2))));
  } return(seq);
}

# attach hook function to node 5
attachHookToNode(sim,5,hook);
# detach hook from node 5
detachHookFromNode(sim,5);
# Run the simulation again
Simulate(sim); # You should not see the message printed out by the "hook" function.

---

detachProcess.Sequence

Detach a Process object from a set of Site objects aggregated by a Sequence object

Description
Detach a Process object from a set of Site objects aggregated by a Sequence object.

Usage

## S3 method for class 'Sequence'
detachProcess(this, process, index, ...)


detachProcess.Site

Arguments

this
A Sequence object.

process
A Process object.

index
An integer vector specifying a set of positions. It is set to 1:seq$length if omitted.

... Not used.

Value

The Sequence object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Sequence.

Examples

# create a sequence object with two processes attached
p<-JC69()
s<-Sequence(length=4,alphabets=list(NucleotideAlphabet()),processes=list(list(p,K80())))
# get the list of attached processes
s$processes
# detach JC69 from range c(1,4)
detachProcess(s,p,c(1,4))
s$processes

detachProcess.Site Site

Description

Site. The site-process specific parameters stored in the Site object and belonging to the detached Process objects will be destroyed.

Usage

## S3 method for class 'Site'
detachProcess(this, process, ...)
**Arguments**

- **this**: A Site object
- **process**: A Process object
- **...**: Not used.

**Value**

The Site object (invisible).

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `Site`.

**Examples**

```r
# create a Site objects and attach some processes
s <- Site()
s$alphabet <- NucleotideAlphabet()
p1 <- JC69(); p2 <- K80()
attachProcess(s, p1)
attachProcess(s, p2)
# get the list of attached processes
s$processes
# detach p1
detachProcess(s, p1)
s$processes
```

---

```r
detachSeqFromNode.PhyloSim

Detach a Sequence object from a given node of a phylo object aggregated by a PhyloSim object
```

**Description**

Detach a Sequence object from a given node of a phylo object aggregated by a PhyloSim object. This method is mainly used internally.

**Usage**

```r
## S3 method for class 'PhyloSim'
detachSeqFromNode(this, node=NA, ...)
```
DiscreteDeletor

Arguments

this A PhyloSim object.
node Node identifier.
... Not used.

Value

The PhyloSim object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see PhyloSim.

The DiscreteDeletor class

Description

This class implements a process which performs deletions with lengths sampled from a user-specified discrete distribution. See GeneralDeletor for how the deletion process works.

Package:

Class DiscreteDeletor

Object
~~|  
~~|PSRoot
~~~~~~~|  
~~~~~~~~~Process
~~~~~~~~~~~|  
~~~~~~~~~~~~~GeneralInDel
~~~~~~~~~~~~~~~~GeneralDeletor
~~~~~~~~~~~~~~~~~~~~~~~~DiscreteDeletor

Directly known subclasses:

public static class DiscreteDeletor
extends GeneralDeletor
Usage

DiscreteDeletor(name=\"Anonymous\", rate=NA, sizes=NA, probs=NA, ...) 

Arguments

name The name of the object.
rate The general rate.
sizes The deletion sizes to propose.
probs A vector with the probabilities of the deletion sizes.
... Additional arguments.

Fields and Methods

Methods:

checkConsistency -
getProbs -
getSizes -
plot -
setProbs -
setSizes -
summary -

Methods inherited from GeneralDeletor:
checkConsistency, getEventsAtSite, is, summary

Methods inherited from GeneralInDel:
checkConsistency, getAcceptBy, getProposeBy, getRate, hasUndefinedRate, is, proposeLength, setAcceptBy, setProposeBy, setRate, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is.na, is, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
[., [<<-, $<<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save
Author(s)
Botond Sipos, Gregory Jordan

See Also
GeneralDeletor ContinuousDeletor GeneralInDel

Examples

```r
# create a DiscreteDeleter process
d <- DiscreteDeleter(
  name = "M.D.",
  rate = 0.25,
  sizes = c(1, 2),
  probs = c(1/2, 1/2)
)
# get object summary
summary(d)
# set/get deletions sizes
d$sizes <- c(1:3)
d$sizes
# set/get length probabilities
d$probs <- c(3, 2, 1)/6
d$probs
# plot length distribution
plot(d)

# The following code illustrates how to use
# a DiscreteDeleter process in a simulation

# create a sequence object, attach process d
s <- NucleotideSequence(string = "AAAAAAAAAGGGGGGGGGGGGGGG", processes = list(list(d)))
# set the deletion tolerance to zero in the range 11:15
setDeletionTolerance(s, d, 0, 11:15)
# creating a region rejecting all deletions
setDeletionTolerance(s, d, 0, 11:15)
# get deletion tolerances
getDeletionTolerance(s, d)
# create a simulation object
sim <- PhyloSim(root.seq = s, phylo = rcoal(2))
# simulate
Simulate(sim)
# print resulting alignment
sim$alignment
```
**Description**

This class implements a process which performs insertions with lengths sampled from a user-specified discrete distribution. See `GeneralInsertor` for how the insertion process works.

**Package:**

**Class DiscreteInsertor**

```java
public static class DiscreteInsertor
extends GeneralInsertor
```

**Usage**

```java
DiscreteInsertor(name="Anonymous", rate=NA, sizes=NA, probs=NA, ...)
```

**Arguments**

- **name** The name of the object.
- **rate** The general rate.
- **sizes** The insertion sizes to propose.
- **probs** A vector with the probabilities of the insertion sizes.
- **...** Additional arguments.

**Fields and Methods**

**Methods:**

```java
checkConsistency -
getProbs -
getSizes -
plot -
```
Methods inherited from GeneralInsertor:
checkConsistency, generateInsert, getAcceptWin, getEventsAtSite, getGenerateBy, getInsertHook, getTemplateSeq, is, setAcceptWin, setGenerateBy, setInsertHook, setTemplateSeq, summary

Methods inherited from GeneralInDel:
checkConsistency, getAcceptBy, getProposeBy, getRate, hasUndefinedRate, is, proposeLength, setAcceptBy, setProposeBy, setRate, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is.na, is, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
[. [, , <-, $, $<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)
Botond Sipos, Gregory Jordan

See Also
GeneralInsertor ContinuousInsertor GeneralInDel

Examples

# create a DiscreteInsertor process
i<-$DiscreteInsertor(
  name=\"M11\",
  rate=0.25,
  sizes=c(1,2),
  probs=c(1/2,1/2)
)

# set template sequence
i$templateSeq<-NucleotideSequence(string=\"C\")

# get object summary
summary(i)

# set/get insertion sizes
i$sizes<-1:3
```r
i$sizes
  # set/get length probabilities
  i$probs <- c(3, 2, 1)/6
  i$probs

  # plot length distribution
  plot(i)

  # The following code illustrates how to use
  # a DiscreteInsertor process in a simulation

  # create a sequence object and attach process i to it
  s <- NucleotideSequence(string="AAAAAAAAAGGGGAAAAAAA", processes=list(list(i)))
  # set the insertion tolerance to zero in range 11:15
  # creating a region rejecting all insertions
  setInsertionTolerance(s, i, 0, 11:15)
  # get insertion tolerances
  getInsertionTolerance(s, i)

  # create a simulation object
  sim <- PhyloSim(root.seq=s, phylo=rcoal(2))
  # simulate
  Simulate(sim)
  # print resulting alignment
  sim$alignment
```

---

**ECMrest**

*The ECMrest empirical codon substitution model*

---

**Description**

**Class ECMrest**

- **Object ECMrest**
  - ~~~|  
  - ~~~+-PSRoot  
  - ~~~~~|  
  - ~~~~~+-Process  
  - ~~~~~~~~~|  
  - ~~~~~~~~~+-GeneralSubstitution  
  - ~~~~~~~~~|  
  - ~~~~~~~~~+-CodonSubst  
  - ~~~~~~~~~|  
  - ~~~~~~~~~+--ECMrest

**Directly known subclasses:**
public static class ECMrest
extends CodonSubst

Usage

ECMrest(equ.dist=NA, ...)

Arguments

equ.dist       Equilibrium distribution.
...            Not used.

Fields and Methods

Methods:
No methods defined.

Methods inherited from CodonSubst:
buildFromPAML, checkConsistency, newMatrix, setEquDist, summary

Methods inherited from GeneralSubstitution:
as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
$, $<-, [ ], [[]<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, save

Author(s)

Botond Sipos, Gregory Jordan

References

See Also

CodonSubst GeneralSubstitution CodonUNREST

Examples

```r
# create substitution model object
p<-ECMunrest()
# get object summary
summary(p)
# display a bubble plot
plot(p)

# The following code demonstrates how to use the process in a simulation.

# create a sequence, attach process p
s<-CodonSequence(length=10,processes=list(list(p)))
# sample states
sampleStates(s)
# make the first three positions invariable
setRateMultipliers(s,p,0:1:3)
# get rate multipliers
getRateMultipliers(s,p)
# create a simulation object
sim<-PhyloSim(root.seq=s,phylo=rcoul(2))
# run simulation
Simulate(sim)
# print alignment
sim$alignment
```

---

**ECMunrest**

*The ECMunrest empirical codon substitution model*

---

**Description**

**Class ECMunrest**

```
Object
~~| ~~~---PSRoot
~~| ~~~| ~~~~---Process
~~| ~~~| ~~~~~~~~~| ~~~~~~~~~---GeneralSubstitution
~~| ~~~~~~~~~| ~~~~~~~~~---CodonSubst
```

---

**ECMunrest**

*The ECMunrest empirical codon substitution model*

---

**Description**

**Class ECMunrest**

```
Object
~~| ~~~---PSRoot
~~| ~~~| ~~~~---Process
~~| ~~~| ~~~~~~~~~| ~~~~~~~~~---GeneralSubstitution
~~| ~~~~~~~~~| ~~~~~~~~~---CodonSubst
```
Directly known subclasses:

public static class ECMunrest
extends CodonSubst

Usage
ECMunrest(equ.dist=NA, ...)

Arguments
- equ.dist: Equilibrium distribution.
- ...: Not used.

Fields and Methods

Methods:
No methods defined.

Methods inherited from CodonSubst:
buildFromPAML, checkConsistency, newMatrix, setEquDist, summary

Methods inherited from GeneralSubstitution:
as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
$, $<-, |[], |<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, save

Author(s)
Botond Sipos, Gregory Jordan
enableVirtual.PSRoot

Enable the use of virtual fields for a given object

Description
Enable the use of virtual fields for a given object. R.oo disables the virtual field feature inside get/set methods. This method can be used to re-enable virtual fields.

Usage
```r
## S3 method for class 'PSRoot'
enableVirtual(this, ...)```

References

See Also
CodonSubst GeneralSubstitution CodonUNREST

Examples
```r
# create substitution model object
p<-ECMunrest()
# get object summary
summary(p)
# display a bubble plot
plot(p)

# The following code demonstrates how to use
# the process in a simulation.

# create a sequence, attach process p
s<-CodonSequence(length=10,processes=list(list(p)))
# sample states
sampleStates(s)
# make the first three positions invariable
setRateMultipliers(s,p,0,1:3)
# get rate multipliers
getRateMultipliers(s,p)
# create a simulation object
sim<-PhyloSim(root.seq=s,phylo=rcal(2))
# run simulation
Simulate(sim)
# print alignment
sim$alignment
```
Event

Arguments

- **this**: A PSRoot object.
- **...**: Not used.

Value

The PSRoot object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `PSRoot`.

---

**Event**

*The Event class*

**Description**

This is the class representing an event. Event objects usually are generated by the `getEventsAtSite.*` methods based on the state of attached Site object and the properties of the generating Process object.

The Perform method will refuse to modify target objects if the position field is not set. The Perform method can be called only once for any Event object.

Package:  

**Class Event**

```
Object
~|   
~~=PSRoot
~~~~~|
~~~~~+=Event
```

**Directly known subclasses:**

```
public static class Event
  extends PSRoot
```

**Usage**

```
Event(name=NA, rate=NA, site=NA, position=NA, process=NA, handler=NA, ...)
```
Arguments

- **name**: The name of the Event object. Often stores useful information.
- **rate**: The rate of the event.
- **site**: The associated Site object.
- **position**: The position of associated Site object in the enclosing Sequence object (if any).
- **process**: The generator Process object.
- **handler**: The handler function for the Event object. It will be called by `Perform.Event` with the Event object as an argument to make the changes corresponding to the event.

... Not used.

Fields and Methods

Methods:

- `as.character`
- `checkConsistency`
- `getHandler`
- `getName`
- `getPosition`
- `getProcess`
- `getRate`
- `getSite`
- `getWriteProtected`
- `is`
- `Perform`
- `setHandler`
- `setName`
- `setPosition`
- `setProcess`
- `setRate`
- `setSite`
- `setWriteProtected`
- `summary`

Methods inherited from `PSRoot`:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is.na, is, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from `Object`:
[., `[<-. S, S<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstanceTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save
Author(s)
Botond Sipos, Gregory Jordan

See Also
Site Process Sequence getEventsAtSite.GeneralSubstitution

Examples

```r
# Note: the following examples are not very useful
# unless you want to implement your own process.

# create a Sequence object and a Process object
seq<-NucleotideSequence(string="A");
p<-Process(alphabet=NucleotideAlphabet())
# get the Site object from the Sequence object
s<-seq$sites[[1]]
# attach p to s
attachProcess(s,p)
# create an Event object
e<-Event(name="A->G",rate=0.1,site=s,process=p,position=1)
# get object summary
summary(e)
# get event name
e$name
# set/get event rate
e$rate<-0.2
e$rate
# get site
e$site
# set/get event handler
e$.handler<-function(this)(this$.site$state<-"G")
e$handler
# perform the event
Perform(e)
# check the state of the target site
s$state
```

Export the per-branch counts of an event as a phylo object

Description
Export the per-branch counts of an event as a phylo object. During simulation the number of events performed on every branch is recorded. The recorded events can be "basic" events, like "insertion", "deletion" and "A->T" or events which are sums of basic events, like "substitution". The getBranchEvents method returns a character vector with the
names of the events having per-branch statistics recorded. The method should be called after the simulation is finished.

The per-branch statistics can be exported as phylo objects by using the `exportStatTree` method. The branch lengths of the exported phylo objects are set to the value of the respective per-branch event count.

Usage

```r
## S3 method for class 'PhyloSim'
exportStatTree(this, event, ...)
```

Arguments

- `this`: A PhyloSim object.
- `event`: The name of the event as returned by the `getBranchEvents` method.
- `...`: Not used.

Value

A phylo object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `PhyloSim`.

Examples

```r
# Create a PhyloSim object.
# Provide the phylo object
# and the root sequence.

# NOTE: this will be a little bit slow
sim <- PhyloSim(
  phylo=rcoral(3),
  root.seq=CodonSequence(
    string="ATGATTATT",
    processes=list(list(GY94(kappa=2, omega.default=0.5))))
)

# make the tree longer to have more events
scaleTree(sim,5)
# plot the tree
plot(sim)
# run simulation
Simulate(sim)
# get the list of recorded per-branch event counts
getBranchEvents(sim)
```
The F81 class

Description

This class implements the F81 GTR-submodel.

Usage

F81(name="Anonymous", base.freqs=c(0.25, 0.25, 0.25, 0.25), ...)

```R
# export the number of substitutions as a phylo object
subst<-exportStatTree(sim,"substitution")
# plot the exported phylo object
plot(subst)
#export the number of synonymous substitutions as a phylo object
subst<-exportStatTree(sim,"nr.syn.subst")
# plot the exported phylo object
plot(subst)
```
Arguments

name  Object name.
base.freqs  Base frequency parameters.
...  Not used.

Fields and Methods

Methods:

- checkConsistency
- getBaseFreqs
- getRateParam
- getRateParamList
- setBaseFreqs
- setRateParam
- setRateParamList
- summary

Methods inherited from GTR:
checkConsistency, getBaseFreqs, getRateParam, getRateParamList, setBaseFreqs, setRateParam, setRateParamList, summary

Methods inherited from UNREST:
checkConsistency, summary

Methods inherited from GeneralSubstitution:
as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is.na, is, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
[[, [[<-, $, $<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstanceationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)

Botond Sipos, Gregory Jordan
**Reference**


**See Also**

GTR UNREST GeneralSubstitution HKY

**Examples**

```r
# create substitution process object
p <- F84(base.freqs = c(1, 2, 3, 4)/10)
# get a summary
summary(p)
# display a bubble plot
plot(p)

# The following code demonstrates how to use
# the process in a simulation.

# create a sequence, attach process p
s <- NucleotideSequence(length = 20, processes = list(list(p)))
# sample states
sampleStates(s)
# make the first five positions invariable
setRateMultipliers(s, p, 0, 1:5)
# get rate multipliers
getRateMultipliers(s, p)
# create a simulation object
sim <- PhyloSim(root.seq = s, phylo = rcoal(2))
# run simulation
Simulate(sim)
# print alignment
sim$alignment
```

---

**F84**

The **F84 class**

**Description**

This class implements the F84 GTR-submodel.

The rate parameters are the following: Kappa. Package: **Class F84**

```r
Object
~ ~
~ ~
```
Directly known subclasses:

public static class F84
    extends GTR

Usage

F84(name="Anonymous", rate.params=list(Kappa = 0), base.freqs=c(0.25, 0.25, 0.25, 0.25))

Arguments

name          Object name.
rate.params   Rate parameters.
base.freqs    Base frequency parameters.
...            Not used.

Fields and Methods

Methods:

checkConsistency -
getBaseFreqs -
getKappa -
getRateParam -
getRateParamList -
setBaseFreqs -
setKappa -
setRateParam -
setRateParamList -
summary -

Methods inherited from GTR:
checkConsistency, getBaseFreqs, getRateParam, getRateParamList, setBaseFreqs, setRateParam, setRateParamList, summary

**Methods inherited from UNREST:**
checkConsistency, summary

**Methods inherited from GeneralSubstitution:**
as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

**Methods inherited from Process:**
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

**Methods inherited from PSRoot:**
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is.na, is, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

**Methods inherited from Object:**
[.[, $, $<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

**Author(s)**
Botond Sipos, Gregory Jordan

**See Also**
GTR UNREST GeneralSubstitution HKY

**Examples**

```r
# create substitution process object
p <- F84(rate.params=list("Kappa"=2), base.freqs=c(1,2,3,4))
# get a summary
summary(p)
# display a bubble plot
plot(p)

# The following code demonstrates how to use
# the process in a simulation.

# create a sequence, attach process p
s <- NucleotideSequence(length=20, processes=list(list(p)))
# sample states
sampleStates(s)
# make the first five positions invariable
setRateMultipliers(s,p,0,1:5)
# get rate multipliers
```

getRateMultipliers(s,p)  # create a simulation object
sim<-PhyloSim(root.seq=s,phylo=rcoal(2))  # run simulation
Simulate(sim)  # print alignment
sim$alignment

---

FastFieldDeletor  The FastFieldDeletor class

Description

The DiscreteDeletor and ContinuousDeletor processes generate deletion event objects with rates determined by the general rate of the process and the “rate.multiplier” parameter. The probability of rejecting an event is determined by the product of the “deletion.tolerance” parameters from the affected sites. If the majority of the sites have low deletion tolerance most of the events are rejected, which slows down the simulation without performing much events.

The FastFieldDeletor process scales down the rate and length distribution of the proposed events based on the highest insertion tolerance value observed in the root sequence (see the package vignette for details), thus making the simulation more efficient.

The available length distributions are (see also the package vignette):

- Geometric (default) - lengthParam1 is Lambda.
- Poisson+1 - lengthParam1 is Lambda.
- Conway-Maxwell Poisson+1 - lengthParam1 is Lambda, lengthParam2 is nu
- Negative Binomial+1 - lengthParam1 is Lambda, lengthParam2 is r

Insertion processes can insert sites with deletion tolerance higher than the largest deletion tolerance observed in the root sequence. The user can specify the largest expected tolerance value through the toleranceMargin virtual field. The process is then scaled by max(initial_highest_tolerance, toleranceMargin).

Package:

Class FastFieldDeletor

Object
~~|  
~~++-PSRoot
~~~~~~~~|  
~~~~~~~~+++--Process
~~~~~~~~~~~|  
~~~~~~~~~~~~+++--GeneralInDel
~~~~~~~~~~~~~~|  
~~~~~~~~~~~~~~~~~~++--GeneralDeletor
~~~~~~~~~~~~~~~~~~~|
Directly known subclasses:

public static class FastFieldDeletor
extends GeneralDeletor

Usage

FastFieldDeletor(
    name="Anonymous", type="geometric",
    length.param.1=NA, length.param.2=NA, tolerance.margin=0, ...
)

Arguments

name Object name.
type Type of the process (see above).
length.param.1 Object name.
length.param.2 Object name.
tolerance.margin Object name.
...
Additional arguments.

Fields and Methods

Methods:

checkConsistency -
getEventsAtSite -
getLengthParam1 -
getLengthParam2 -
getToleranceMargin -
getType -
setLengthParam1 -
setLengthParam2 -
setParameterAtSite -
setToleranceMargin -
setType -
summary -

Methods inherited from GeneralDeletor:
checkConsistency, getEventsAtSite, is, summary
Methods inherited from GeneralInDel:
checkConsistency, getAcceptBy, getProposeBy, getRate, hasUndefinedRate, is, proposeLength, setAcceptBy, setProposeBy, setRate, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is.na, is, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
[, [, [<-, $<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldName, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see FastFieldDeletor.

Examples

# create a FastFieldDeletor object, default (geometric) type
# length.param.1 is "lambda"
p<-FastFieldDeletor(rate=1,length.param.1=0.9, tolerance.margin=0.8)
# get type
p$type
# get object summary
summary(p)
# set/get lambda
p$lengthParam1<-0.8
p$lengthParam1
# set/get tolerance margin
p$toleranceMargin<-0.5
p$toleranceMargin

# create a nucleotide sequence, attach process
s<-NucleotideSequence(length=30,processes=list(list(p)))
# set state pattern
s$states<-c("A","A","T","T","G","G","C","C")
# sample deletion tolerances
setDeletionTolerance(s,p,sample(seq(from=0,to=0.8,by=0.01),30,replace=TRUE))
# plot deletion tolerance
plotParametersAtSites(s,p,"deletion.tolerance")
# simulate
sim<-PhyloSim(root.seq=s, phylo=rcoal(2))
flagTotalRate.Site

**Description**

Flag the total event rate. This method sets the cached total active event rate to NA, which will trigger its recalculation when next accessed via getTotalRate.

**Usage**

```r
## S3 method for class 'Site'
flagTotalRate(this, ...)
```

**Arguments**

- `this` A Site object.
- `...` Not used.

**Value**

Invisible NA.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `Site`.

**Examples**

```r
# create a site object:
p<-K80(rate.params=list("Alpha"=2,"Beta"=0.5))
s<-Site(alphabet=NucleotideAlphabet(), state="G", processes=list(p))
# get site rate
s$totalRate
# modifying site object in a dangerous way (do not do this under any circumstances!)
s$.processes = list() # site object is now inconsistent!
# get cached rate
s$totalRate # incorrect value
# flag total rate
flagTotalRate(s)
# get site rate
```
GeneralDeletor

The GeneralDeletor class

Description

This is the class implementing a process generating deletion events. The rates of the deletion events are calculated as the product of the general rate of the process and the "rate.multiplier" site-process-specific parameter.

The simulation code calls the `perform` method on the selected deletion event objects, which call their `handler` function to perform the deletion.

The deletion lengths are proposed by the function stored in the `proposeBy` virtual field. The function must have the following arguments: process (the process object), sequence (the target sequence), position (the position of the site which generated the event).

The deletion randomly affects the sites from the left or from the right of the target position (but never both). Positions which are out of range are discarded.

The proposed deletions are accepted or rejected by the function stored in the `acceptBy` virtual field. The function must have the following arguments: process (the deletion process), sequence (the target sequence), range (a vector of positions affected by the deletion).

The probability of accepting a deletion is calculated as the product of the "deletion.tolerance" site-process-specific parameters from the sites affected by the deletion event.

Package:
Class GeneralDeletor

Object
---
---PSRoot
--------|---Process
--------------|---GeneralInDel
------------------|---GeneralDeletor

Directly known subclasses:
ContinuousDeletor, DiscreteDeletor, FastFieldDeletor

public static class GeneralDeletor
extends GeneralInDel
Usage

GeneralDeletor(name="Anonymous", rate=NA, propose.by=NA, accept.by=NA, ...)

Arguments

name         The name of the object.
rate         The general rate of the object.
propose.by   A function used to propose events.
accept.by    A function used to accept/reject events.
...          Additional arguments.

Fields and Methods

Methods:

checkConsistency -
getEventsAtSite  -
is             -
summary         -

Methods inherited from GeneralInDel:
checkConsistency, getAcceptBy, getProposeBy, getRate, hasUndefinedRate, is, proposeLength, setAcceptBy, setProposeBy, setRate, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is.na, is, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
[, [[<-, $, $<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)

Botond Sipos, Gregory Jordan

See Also

GeneralInDel DiscreteDeletor ContinuousDeletor FastFieldDeletor
Examples

```r
# create a GeneralDeleter object
# proposed deletion length: 4, always accept
d <- GeneralDeleter(
  name = "DEL",
  rate = 1,
  propose.by = function(process, sequence, position){ 4 },
  accept.by = function(process, sequence, range){ TRUE }
)

# check if object inherits from GeneralDeleter
is.GeneralDeleter(d)
# get object summary
summary(d)
# set/get name
d$name <- "Del Bosque"
d$name
# set/get rate
d$rate <- 0.5
d$rate
# set/get proposeBy
# propose deletion lengths between 3 and 6
d$proposeBy <- function(process, sequence, position){ sample(3:6,1) }
d$proposeBy
# set/get acceptBy
# reject half of the events
d$acceptBy <- function(process, sequence, range){ sample(c(TRUE, FALSE), 1) }
d$acceptBy
# create a sequence object, attach process
s <- NucleotideSequence(string = "AATGGCCCGGTTAA", processes = list(list(d)))
# set the rate multiplier
setRateMultipliers(s, d, 2)
# get the list of active events at site 6
events <- getEventsAtSite(d, s$sites[[6]])
events;
# print sequence
s
# set the position for the event object
e <- events[[1]]; e$.position <- -6;
# perform the deletion event
Perform(e)
# check the results
s
```

GeneralInDel  The GeneralInDel class
**Description**

This is a class implementing the methods which are needed by both the GeneralInsertor and GeneralDeletor process.

**Package:**

**Class GeneralInDel**

```
Object
|~~|
|~~--PSRoot
|~~~~~~~~|
|~~~~~~~~|--Process
|~~~~~~~~~|
|~~~~~~~~~~|---GeneralInDel
```

**Directly known subclasses:**

`BrownianInsertor`, `ContinuousDeletor`, `ContinuousInsertor`, `DiscreteDeletor`, `DiscreteInsertor`, `FastFieldDeletor`, `GeneralDeletor`, `GeneralInsertor`

```
public static class GeneralInDel
extends Process
```

**Usage**

`GeneralInDel(name="Anonymous", rate=NA, propose.by=NA, accept.by=NA, ...)`

**Arguments**

- **name**  The name of the object.
- **rate**  The general rate of the object.
- **propose.by**  A function used to propose events.
- **accept.by**  A function used to accept/reject events.
- **...**  Additional arguments.

**Fields and Methods**

**Methods:**

- `checkConsistency`
- `getAcceptBy`
- `getProposeBy`
- `getRate`
- `hasUndefinedRate`
- `is`
- `proposeLength`
setAcceptBy -
setProposeBy -
setRate -
summary -

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParams, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParams, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is.na, is, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
[ , [[<-, $, $<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)
Botond Sipos, Gregory Jordan

See Also
Process GeneralInsertor GeneralDeletor GeneralSubstitution

Examples
# create a GeneralInDel object
# rejecting half of the events
# and proposing sizes in the range 1:10
o<-GeneralInDel(
  rate=1,
  propose.by=function(process){sample(1:10,1)},
  accept.by=function(){sample(c(TRUE,FALSE),1)}
);

# check if inherits from GeneralInDel
is.GeneralInDel(o)
# check if it has undefined rates
hasUndefinedRate(o)
# get object summary
summary(o)
# set/get proposeBy function via virtual field
o$proposeBy<-function(process){return(3)} # fixed event length
o$proposeBy
# set/get acceptBy function via virtual field
o$acceptBy<-function(){return(TRUE)} # accept all events
o$acceptBy
GeneralInsertor

The GeneralInsertor class

Description

This is a class implementing a process generating insertion events. The rate of each event is calculated as the product of the general rate of the process and the "rate.multiplier" site-process specific parameter. The simulation code calls the perform method on the selected insertion event objects, which call their insertion event handler to perform the insertion.

The insert lengths are proposed by the function stored in the proposeBy virtual field. The function must have the following arguments: process (the insertion process object). The insertion events are accepted or rejected by the function stored in the acceptBy virtual field. The function must have the following arguments: process (the insertion process object), sequence (the target sequence object), window (a vector of positions affecting acceptance). The probability of accepting an insertion is calculated as the product of the site-process-specific "insertion.tolerance" parameters of the sites neighboring the insertion. The number of sites considered is determined by the acceptWin virtual field.

The insert is generated by the generateInsert method by calling the function stored in the generateBy virtual field. The default generator function truncates/duplicates the sequence object stored in the templateSeq virtual field to get a sequence having the sampled length. After constructing the Sequence object, it calls the sampleStatesSequence method on the resulting object. That means that if we start with a template sequence which has NA states, but has a substitution process attached, then the resulting sequence will be different every time.

Before inserting the sequence returned by generateInsert, the handler function will pass the object through the function stored in the insertHook virtual field. This allows to perform arbitrary modifications on the inserted Sequence object.

The sequence is inserted randomly on the left or the right of the target position.

Package:

Class GeneralInsertor

Object

---|---
---|---PSRoot
---|---|
---|---Process
---|---|
---|---GeneralInDel
---|---|
Directly known subclasses:
BrownianInsertor, ContinuousInsertor, DiscreteInsertor

Usage

GeneralInsertor(
    name="Anonymous", rate=NA, propose.by=NA,
    accept.by=NA, template.seq=NA, insert.hook=NA, accept.win=NA, ...
)

Arguments

name The name of the object.
rate The general rate of the object (no default).
propose.by A function used to propose events (no default).
accept.by A function used to accept/reject events (no default).
template.seq A Sequence object used as a template for generating insertions (no default).
insert.hook A function object, see setInsertHook (no default).
accept.win A window of sites affecting the acceptance of insert events.
... Additional arguments.

Fields and Methods

Methods:

checkConsistency -
generateInsert -
getAcceptWin -
getEventsAtSite -
getGenerateBy -
getInsertHook -
getTemplateSeq -
is -
setAcceptWin -
setGenerateBy -
setInsertHook -
setTemplateSeq -
summary -
Methods inherited from GeneralInDel:
checkConsistency, getAcceptBy, getProposeBy, getRate, hasUndefinedRate, is, proposeLength, setAcceptBy, setProposeBy, setRate, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, na, is, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
[, [<-, $, $<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)
Botond Sipos, Gregory Jordan

See Also
GeneralInDel DiscreteInsertor ContinuousInsertor BrownianInsertor

Examples
# create a GeneralInsertor object
i<-GeneralInsertor(
    name="GIN",
    rate=1,
    propose.by=function(process){4}, # fixed insert length
    acceptBy=function(process,sequence,window){TRUE}, # always accept insertions
    template.seq=NucleotideSequence(string="A"), # a boring template sequence
    insert.hook=function(seq){return(seq)}, # a boring insert hook
    accept.win=2 #4 sites affecting acceptance
)
i
# check if object inherits from GeneralInsertor
is.GeneralInsertor(i)
# get object summary
summary(i)
# set/get general rate
i$rate<-0.5
i$rate
# set/get name
i$name<-"Ins"
i$name
# set/get proposeBy
# sample insertion length between 1 and 10
The GeneralSubstitution class

Description

This a class representing a continuous-time Markov process acting on the state space defined by the symbols stored in the Alphabet object passed to the object constructor.
The `GeneralSubstitution` objects generate Event objects corresponding to substitution events based on the state of the attached Site objects.

The `GeneralSubstitution` objects aggregate a QMatrix object, which stores the unscaled and scaled rate matrices. The scaled rate matrices, along with the site-process specific rate multiplier parameters define the rates of the generated Event objects.

**Package:**

**Class GeneralSubstitution**

```
Object
~~|  
~~+++PSRoot
~~~~~~|  
~~~~~~~~+++Process
~~~~~~~~~~~~|  
~~~~~~~~~~~~~~~~+++GeneralSubstitution
```

**Directly known subclasses:**

`AminoAcidSubst`, `BinarySubst`, `CodonUNREST`, `cpREV`, `F81`, `F84`, `GTR`, `GY94`, `HKY`, `JC69`, `JTT`, `JTT.dcmut`, `K80`, `K81`, `LG`, `mtArt`, `mtMam`, `mtREV24`, `MtZoa`, `PAM`, `PAM.dcmut`, `T92`, `TN93`, `UNREST`, `WAG`

public static class `GeneralSubstitution` extends `Process`

**Usage**

`GeneralSubstitution(name="Anonymous", alphabet=NA, rate.list=NA, equ.dist=NA, ...)`

**Arguments**

- `name` The name of the object.
- `alphabet` The alphabet on which the process acts (Alphabet object).
- `rate.list` A list with the substitution rates. It will be passed to `setRateList` method.
- `equ.dist` The equilibrium distribution (see `setEquDist`. `GeneralSubstitution`).
- `...` Not used.

**Fields and Methods**

**Methods:**

- `as.character` -
- `checkConsistency` -
- `clone` -
- `getAlphabet` -
- `getEquDist` -
Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is.na, is, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
[, [[<-, $, $<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)
Botond Sipos, Gregory Jordan

See Also
Process QMatrix Event Site GeneralIndel GTR WAG

Examples
# Create a GeneralSubstitution object
# also provide an Alphabet objects
# and the list of unscaled rates
a<-BinaryAlphabet()
gs<-GeneralSubstitution(
name="MyBinarySubst",
alphabet=a,
rate.list=list("0->1"=2,"1->0"=1)
# check if inherits from GeneralSubstitution
is.GeneralSubstitution(gs)
# get an object summary
summary(gs)
# get a bubble plot
plot(gs)
# get unscaled rate for "0->1" by event name
getRate(gs,"0->1")
# get unscaled rate for "0->1" by states
getRate(gs,from="0", to="1")
# get scaled rate for "0->1" by name
getEventRate(gs,"0->1")
# get the list of unscaled event rates
gs$rateList
# set the \textit{unscaled} rates
gs$rateList<-list("0->1"=1,"1->0"=1)
# reset equilibrium distribution
gs$equiDist<- 5 * gs$equiDist
# get the equilibrium distribution
gs$equiDist
# sample a state form the equilibrium distribution
sampleState(gs)
# get the associated QMatrix object
gs$qMatrix
s<-Site(alphabet=a, state="0")
# attach gs to s
s$processes<-list(gs)
# set rate multiplier for s and gs
setParameterAtSite(gs,s,id="rate.multiplier",value=2)
# get site specific rate for "0->1"
getEventsAtSite(gs,s,"0->1")
# get the list of active event objects given the state of s
getEventsAtSite(gs,s)
# get the associated Alphabet object
gs$alphabet
# clone the object
gsc<-clone(gs)
# modify the alphabet of gsc
gsc$alphabet<-NucleotideAlphabet()
summary(gsc)
# check if gsc has undefined rates
hasUndefinedRate(gsc)

generateInsert.GeneralInsertor

Generate an insert
Description

Generate an insert.

This method uses the function stored in the proposeBy virtual field to sample the insert length and then calls the function stored in the generateBy field to generate the insert.

The default generateBy function set by the GeneralInsertor constructor truncates/repeats the template sequence stored in the templateSeq field to have the sequence with the right size and then calls the sampleStates method on the resulting object. That means that if we start with a template sequence which has NA states, but has a substitution process attached, then the resulting sequence will be different every time the generateInsert method is called.

The generateBy, proposeBy and templateSeq fields must be set in order to use this method.

Usage

```r
## S3 method for class 'GeneralInsertor'
generateInsert(this, length=NA, target.seq=NA, event.pos=NA, insert.pos=NA, ...)
```

Arguments

- **this**: A GeneralInsertor object.
- **length**: Generate an insert with the specified length if this argument is present.
- **target.seq**: The Sequence object targeted by the insertion (optional). This argument is passed to the generateBy method.
- **event.pos**: The position of the site proposing the insertion (optional). This argument is passed to the generateBy method.
- **insert.pos**: The position of the insertion in the target sequence (optional). This argument is passed to the generateBy method.
- **...**: Not used.

Value

A Sequence object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `GeneralInsertor`.

Examples

```r
# build the template sequence
ts<-NucleotideSequence(length = 10, processes=list(list(JC69())));
# fix some site states
setStates(ts,"A",c(1,2));
setStates(ts,"T",c(5,6));
```
getAcceptBy.GeneralInDel

getAcceptBy.GeneralInDel

Get the function used for accepting/rejecting indel events

Description

Get the function used for accepting/rejecting indel events.

Usage

```r
## S3 method for class 'GeneralInDel'
getAcceptBy(this, ...)
```

Arguments

- `this` A GeneralInDel object.
- `...` Not used.

Value

A function object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `GeneralInDel`.
Examples

```r
# create a GeneralIndel object
# rejecting half of the events
o <- GeneralIndel(
  rate = 1,
  propose.by = function(process) { return(5) },
  accept.by = function() { sample(c(TRUE, FALSE), 1) }
);

# set/get the acceptBy function
setAcceptBy(o, value = function() { return(FALSE) }) # reject all events
getAcceptBy(o)

# set/get acceptBy function via virtual field
o$acceptBy <- function() { return(TRUE) } # accept all events
o$acceptBy
```

---

**getAcceptWin.** GeneralInsertor

*Get the size of the acceptance window*

---

**Description**

Get the size of the acceptance window

This parameter determines the number of sites neighbouring the position (from left and right) of the insertion considered when accepting/rejecting a proposed insertion. The "insertion.tolerance" parameter is retrieved from sites falling in the window specified by this parameter. The default value is 1, so the two neighbouring sites are considered by default.

**Usage**

```r
## S3 method for class 'GeneralInsertor'
getAcceptWin(this, ...)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>this</td>
<td>A GeneralInsertor object.</td>
</tr>
<tr>
<td>...</td>
<td>Not used.</td>
</tr>
</tbody>
</table>

**Value**

A numeric vector of length one.

**Author(s)**

Botond Sipos, Gregory Jordan
Description

Get the alignment stored in a PhyloSim object.

Usage

```r
## S3 method for class 'PhyloSim'
getAlignment(this, ...)
```

Arguments

- `this` A PhyloSim object.
- `...` Not used.

Value

The alignment as a matrix. Gap are represented by strings composed of dashes.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `PhyloSim`. 
Examples

# Create a PhyloSim object.
# Provide the phylo object
# and the root sequence.
sim <- PhyloSim(
  name = "TinySim",
  phylo = rcoal(3),
  root.seq = NucleotideSequence(string = "ATG", processes = list(list(JC69())))
);
# run the simulation
Simulate(sim)
# get the resulting alignment
getAlignment(sim)
# via virtual fileld:
sim$alignment

getAlignmentLength.PhyloSim

Get the alignment length from a PhyloSim object

Description

Get the alignment length from a PhyloSim object.
This method returns the number of columns in the alignment stored in the PhyloSim object.

Usage

## S3 method for class 'PhyloSim'
getAlignmentLength(this, ...)

Arguments

this A PhyloSim object.
...
Not used.

Value

A numeric vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see PhyloSim.
Examples

```r
# create a PhyloSim object and run a simulation:
sim<-Simulate(
    PhyloSim(phy=rcoal(3),
             root=NucleotideSequence(string="ATGC", proc=list(list(JC69()))))
)

# get the alignment length
getAlignmentLength(sim)
```

getAlphabet. GeneralSubstitution

*Get the Alphabet object aggregated by a GeneralSubstitution object*

Description

Get the Alphabet object aggregated by a GeneralSubstitution object.

This method also sets the alphabet for the associated QMatrix object, which will set all rates to NA.

Usage

```r
## S3 method for class 'GeneralSubstitution'
getAlphabet(this, ...)
```

Arguments

- **this**: A GeneralSubstitution object.
- **...**: Not used.

Value

An Alphabet object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see GeneralSubstitution.
Examples

```r
# create a GeneralSubstitution object with an attached BinaryAlphabet object
p <- GeneralSubstitution(alphabet=BinaryAlphabet())
# get object summary
summary(p)
# get alphabet
getAlphabet(p)
# get alphabet via virtual field
p$alphabet
# set a new alphabet
setAlphabet(p,NucleotideAlphabet())
summary(p)
# set alphabet via virtual field
p$alphabet<-BinaryAlphabet()
p$alphabet
```

---

**getAlphabet.Process**  
*Get the Alphabet object associated with a given Process object*

**Description**

Get the Alphabet object associated with a given Process object.

**Usage**

```r
## S3 method for class 'Process'
getAlphabet(this, ...)
```

**Arguments**

- `this`  
  A Process object.
- `...`  
  Not used.

**Value**

An Alphabet object.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `Process`. 
Examples

```r
# create a process object
p <- Process(alphabet = NucleotideAlphabet())
# get associated Alphabet object
getAlphabet(p)
# via virtual field
p$alphabet
```

getAlphabet.QMatrix  Get the Alphabet object associated with a QMatrix object

Description

Get the Alphabet object associated with a QMatrix object.

Usage

```r
## S3 method for class 'QMatrix'
getAlphabet(this, ...)
```

Arguments

- `this`: A QMatrix object.
- `...`: Not used.

Value

An Alphabet object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see QMatrix.

Examples

```r
# create a QMatrix object
m <- QMatrix()
# set the alphabet
setAlphabet(m, NucleotideAlphabet())
# get the alphabet
getAlphabet(m)
# set alphabet via virtual field
m$alphabet <- BinaryAlphabet()
summary(m)
```
Description

Get the Alphabet object attached to a Site object.

Usage

```r
## S3 method for class 'Site'
getAlphabet(this, ...)
```

Arguments

- `this` A Site object.
- `...` Not used.

Value

An Alphabet object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Site`.

Examples

```r
# create a site object
s <- Site()
# create an Alphabet object
a <- Alphabet(c("A", "T", "G"))
# attach alphabet to site object
setAlphabet(s, a)
# get attached alphabet
getAlphabet(s)
# get attached alphabet via virtual field
s$alphabet
```
**getAlphabets.Sequence**

*Get the list of the Alphabet objects attached to the Site objects aggregated by a Sequence object*

**Description**

Get the list of the Alphabet objects attached to the Site objects aggregated by a Sequence object.

**Usage**

```r
## S3 method for class 'Sequence'
getAlphabets(this, index, ...)
```

**Arguments**

- `this` A Sequence object.
- `index` An integer vector specifying a set of positions. It is set to 1:seq$length if omitted.
- `...` Not used.

**Value**

A list of Alphabet objects.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `Sequence`.

**Examples**

```r
# create a Sequence object with NucleotideAlphabet and BinaryAlphabet objects attached
s <- Sequence(alphabets = list(NucleotideAlphabet(), BinaryAlphabet()), length = 5)
setStates(s, c("A", "0"))
# get the list of attached Alphabet objects
getAlphabets(s)
# get Alphabets from a range
getAlphabets(s, c(2:3, 5))
# get alphabets via virtual field
s$alphabets
```
getAncestral.Sequence  Get the ancestral object of a Sequence object

Description

Get the ancestral object of a Sequence object.

Usage

```r
## S3 method for class 'Sequence'
getAncestral(this, ...)
```

Arguments

- `this` A Sequence object.
- `...` Not used.

Value

A Sequence object or a Process object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Sequence`.

Examples

```r
# create a sequence object
s <- Sequence(length=4)
# get ancestral object
goAncestral(s) # newly created sequences have the "Root insertion process" as ancestral
# clone sequence
cs <- clone(s)
# get ancestral object id via virtual field
cs$ancestral$id
```
getAncestral.Site

Get the ancestral object of a Site object

Description

Get the ancestral object of a Site object.

Usage

```r
## S3 method for class 'Site'
getAncestral(this, ...)
```

Arguments

- `this` A Site object.
- `...` Not used.

Value

A Site object, a Process object or NA.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Site`.

Examples

```r
a <- Alphabet();
# create some site object
s1 <- Site(alphabet=a)
s2 <- Site(ancestral=s1, alphabet=a)
# get ancestral objects
getAncestral(s1)
s2$ancestral
```
getBaseFreqs.F81

Get the base frequency parameters

Description

Get the base frequency parameters.

Usage

```r
## S3 method for class 'F81'
getBaseFreqs(this, ...)
```

Arguments

- `this` A F81 object.
- `...` Not used.

Value

A matrix containing the base frequency parameters.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `F81`.

Examples

```r
# construct a F81 object
p<-F81()
# set/get base frequency parameters
setBaseFreqs(p,c(2,1,2,1)/6)
getBaseFreqs(p)
# set/get base frequency parameters via virtual field
p$baseFreqs<-c(4,4,1,1)/10
p$baseFreqs
# get object summary
summary(p)
```
**Description**

Get the base frequency parameters.

The order of the frequency parameters must match with the order of symbols in the NucleotideAlphabet objects.

**Usage**

```r
## S3 method for class 'F84'
getBaseFreqs(this, ...)
```

**Arguments**

- `this` A F84 object.
- `...` Not used.

**Value**

A matrix containing the base frequency parameters.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `F84`.

**Examples**

```r
# construct object
p<-F84()
# set/get base frequency parameters
setBaseFreqs(p, c(2,1,2,1)/6)
geBaseFreqs(p)
# set/get base frequency parameters via virtual field
p$baseFreqs<-c(4,4,1,1)/10
p$baseFreqs
# get object summary
summary(p)
```
**getBaseFreqs.GTR**  
*Get the base frequency parameters*

**Description**

Get the base frequency parameters.

**Usage**

```r
## S3 method for class 'GTR'
getBaseFreqs(this, ...)```

**Arguments**

- **this**  
  A GTR object.
- **...**  
  Not used.

**Value**

A matrix containing the base frequencies.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see **GTR**.

**Examples**

```r
# construct a GTR object
p<-GTR()
# set/get base frequency parameters
setBaseFreqs(p,c(2,1,2,1)/6)
getBaseFreqs(p)
# set/get base frequency parameters via virtual field
p$baseFreqs<-c(4,4,1,1)/10
p$baseFreqs
# get object summary
summary(p)```
**getBaseFreqs.HKY**

*Get the base frequency parameters*

---

**Description**

Get the base frequency parameters.

The order of the frequency parameters must match with the order of symbols in the NucleotideAlphabet objects.

**Usage**

```r
## S3 method for class 'HKY'
getBaseFreqs(this, ...)  
```

**Arguments**

- `this` An HKY object.
- `...` Not used.

**Value**

A matrix containing the base frequency parameters.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `HKY`.

**Examples**

```r
# construct object
p<-HKY()
# set/get base frequency parameters
setBaseFreqs(p, c(2,1,2,1)/6)
getBaseFreqs(p)
# set/get base frequency parameters via virtual field
p$baseFreqs<-c(4,4,1,1)/10
p$baseFreqs
# get object summary
summary(p)
```
Description

Get the base frequency parameters.

The order of the frequency parameters must match with the order of symbols in the NucleotideAlphabet objects. The K80 model has equal base frequencies.

Usage

```r
## S3 method for class 'K80'
getBaseFreqs(this, ...)
```

Arguments

- `this`: A K80 object.
- `...`: Not used.

Value

A matrix containing the base frequency parameters.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `K80`.

Examples

```r
# construct object
p <- K80()
# get base frequency parameters
getBaseFreqs(p) # uniform
# set/get base frequency parameters via virtual field
p$baseFreqs # uniform
```
**getBaseFreqs.K81**

*Get the base frequency parameters*

**Description**

Get the base frequency parameters.

The order of the frequency parameters must match with the order of symbols in the NucleotideAlphabet objects. The K81 model has equal base frequencies.

**Usage**

```r
## S3 method for class 'K81'
getBaseFreqs(this, ...)
```

**Arguments**

- `this` A K81 object.
- `...` Not used.

**Value**

A matrix containing the base frequency parameters.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see **K81**.

**Examples**

```r
# construct object
p<-K81()
# get base frequency parameters
getBaseFreqs(p) # uniform
# set/get base frequency parameters via virtual field
p$baseFreqs # uniform
```
Get the base frequency parameters

Description

Get the base frequency parameters.

Usage

```r
## S3 method for class 'TN93'
getBaseFreqs(this, ...)
```

Arguments

- `this`  
  A TN93 object.
- `...`  
  Not used.

Value

A matrix containing the base frequencies.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `TN93`.

Examples

```r
# construct a TN93 object
p<-TN93()
# set/get base frequency parameters
setBaseFreqs(p,c(2,1,2,1)/6)
getBaseFreqs(p)
# set/get base frequency parameters via virtual field
p$baseFreqs<-c(4,4,1,1)/10
p$baseFreqs
# get object summary
summary(p)
```
getBigRate.Sequence

Get the sum of all active event rates from a Sequence object

Description
Get the sum of all active event rates from a Sequence object. The sum of active event rates depends on all Site object states and on the attached Process objects. It basically returns the last element of the cumulative site rates vector.

Usage
```r
## S3 method for class 'Sequence'
getBigRate(this, ...)
```

Arguments
- `this` A Sequence object.
- `...` Not used.

Value
A numeric vector of length one.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see `Sequence`.

Examples
```r
# create a nucleotide sequence attach a process
s<-NucleotideSequence(length=5);
s$processes<-list(list(JC69()))
# get the sum of active event rates
getBigRate(s) # returns NA because site states are undefined
# set site states
s$states<-c("A","T")
# get big rate via virtual field
s$bigRate
```
getBranchEvents.PhyloSim

Description

Get the list of events having per-branch statistics recorded.

During simulation the number of events performed on every branch is recorded. The recorded
events can be "basic" events, like "insertion", "deletion" and "A->T" or events which are sums of
basic events, like "substitution". The getBranchEvents method returns a character vector with the
names of the events having per-branch statistics recorded. The method should be called after the
simulation is finished.

The per-branch statistics can be exported as phylo objects by using the exportStatTree method.
The branch lengths of the exported phylo objects are set to the value of the respective per-branch
event count.

Usage

## S3 method for class 'PhyloSim'
getBranchEvents(this, ...)

Arguments

- **this**: A PhyloSim object.
- **...**: Not used.

Value

A character vector.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see PhyloSim.

Examples

# Create a PhyloSim object.
# Provide the phylo object
# and the root sequence.

# NOTE: this will be a little bit slow
sim<--PhyloSim(
  phylo=rcoal(3),


0

1
getCodonFreqs.GY94

root.seq=CodonSequence(
  string="ATGATTATT",
  processes=list(list(GY94(kappa=2,omega.default=0.5))))
)

# make the tree longer to have more events
scaleTree(sim,5)
# plot the tree
plot(sim)
# run simulation
Simulate(sim)
# get the list of recorded per-branch event counts
getBranchEvents(sim)
# export the number of substitions as a phylo object
subst<exportStatTree(sim,"substitution")
# plot the exported phylo object
plot(subst)
#export the number of synonymous substitutions as a phylo object
subst<exportStatTree(sim,"nr.syn.subst")
# plot the exported phylo object
plot(subst)

getCodonFreqs.GY94   Get codon frequencies

Description
Get codon frequencies.

Usage
## S3 method for class 'GY94'
getCodonFreqs(this, ...)

Arguments

  this             A GY94 object.
  ...               Not used.

Value
A matrix containing the codon frequencies.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see GY94.
Examples

```r
# create a GY94 object
p <- GY94()
# get codon frequencies
getCodeFrequents(p)
p$codonFrequents
# set codon frequencies
p$codonFrequents <- rep(c(1,2,3,4), length.out=61)
p$codonFrequents
```

getComments.PSRoot  
Get the comments associated with an object

Description

Get the comments associated with an object.

The comment field can contain any type of object.

Usage

```r
## S3 method for class 'PSRoot'
getComments(this, ...)
```

Arguments

- `this`: A PSRoot object.
- `...`: Not used.

Value

The value of the comment field.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `PSRoot`.  

"getComments.PSRoot"
**getCumulativeRates.Sequence**

*Get the total site rates from a Sequence object*

**Description**

Get the total site rates from a Sequence object. This method simply calls `getCumulativeRatesFromRange(this)`. See `getCumulativeRates.Sequence` for more details.

**Usage**

```r
## S3 method for class 'Sequence'
getCumulativeRates(this, ...)
```

**Arguments**

- `this` A Sequence object.
- `...` Not used.

**Value**

A numeric vector containing the total site rates.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

`getCumulativeRatesFromRange.Sequence`
getCumulativeRatesFromRange.Sequence

Get the cumulative site rates for a collection of Site objects aggregated by a Sequence object

Description
Get the cumulative site rates for a collection of Site objects aggregated by a Sequence object.

Usage
## S3 method for class 'Sequence'
getCumulativeRatesFromRange(this, index, ...)

Arguments
this A Sequence object.
index An integer vector specifying a set of positions. It is set to 1:seq$length if omitted.
... Not used.

Value
A numeric vector.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see Sequence.

Examples
# create a sequence with some processes attached
s<-Sequence(
  string="ATGC",
  alphabets=list(NucleotideAlphabet()),
  processes=list(list(JC69()),list(JC69()),GTR())
)
# get cumulative rates for positions 1 and 3
gCumulativeRatesFromRange(s,c(1,3))
# get all cumulative rates via virtual field
s$cumulativeRates # via the "getCumulativeRates.Sequence" method
getDeletionTolerance.Sequence

Get the deletion tolerance site-process specific parameter for a collection of Site objects aggregated by a Sequence object

Description

Get the deletion tolerance site-process specific parameter for a collection of Site objects aggregated by a Sequence object. This method does some error checking and the calls getParameterAtSites(this=this, process=process, index=index). See getParameterAtSites.Sequence for more details.

Usage

```r
## S3 method for class 'Sequence'
getDeletionTolerance(this, process, index, ...)
```

Arguments

- **this**: A Sequence object.
- **process**: A Process object.
- **index**: An integer vector specifying a set of positions. It is set to 1:seq$length if omitted.
- **...**: Not used.

Value

A numeric vector.

Author(s)

Botond Sipos, Gregory Jordan

See Also

getParameterAtSites.Sequence
getDist.ContinuousDeletor

Get the length sampling expression

Description
Get the length sampling expression.

The length sampling expression can be any valid R expression returning a numeric vector of length one. The value returned by the expression will be rounded.

Usage

## S3 method for class 'ContinuousDeletor'
getDist(this, ...)

Arguments

this	A ContinuousDeletor object.

... Not used.

Value

An R expression object.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see ContinuousDeletor.

Examples

# create object
o<-ContinuousDeletor(rate=1)
# set/get length sampling expression
setDist(o, expression(rnorm(1,mean=3, sd=2)))
getDist(o)
# set/get length sampling expression via virtual field
o$dist<-expression(rnorm(1,mean=6, sd=3))
o$dist
# set maxLength
o$maxLength<-10
# propose a length
proposeLength(o)
getDist.ContinuousInsertor

Description

Get the length sampling expression.

The length sampling expression can be any valid R expression returning a numeric vector of length one. The value returned by the expression will be rounded.

Usage

```r
## S3 method for class 'ContinuousInsertor'
getDist(this, ...)
```

Arguments

this

A ContinuousInsertor object.

... Not used.

Value

An R expression object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see ContinuousInsertor.

Examples

```r
# create object
o <- ContinuousInsertor(rate=1)
# set/get length sampling expression
setDist(o, expression(rnorm(1, mean=3, sd=2)))
getDist(o)
# set/get length sampling expression via virtual field
o$dist <- expression(rnorm(1, mean=6, sd=3))
o$dist
# set maxLength
o$maxLength <- 10
# propose a length
proposeLength(o)
```
**getEdge.PhyloSim**

*Get and edge from the edge matrix*

Description

Get and edge from the edge matrix.

Usage

```r
## S3 method for class 'PhyloSim'
getEdge(this, number=NA, ...)
```

Arguments

- **this**: A PhyloSim object.
- **number**: The edge number.
- **...**: Not used.

Value

The edge as a matrix with a single row.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `PhyloSim`

Examples

```r
# create a PhyloSim object
sim <- PhyloSim(phylo=coal(5));
# get edge number 3
getEdge(sim, 3)
```
getEdges.PhyloSim

Get the edge matrix from a phylo object aggregated by a PhyloSim object.

Description

Get the edge matrix from a phylo object aggregated by a PhyloSim object.
The rows of the edge matrix contain the nodes connected by the edge and the edge length.

Usage

## S3 method for class 'PhyloSim'
getEdges(this, ...)

Arguments

this
A PhyloSim object.

... Not used.

Value

A matrix.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see PhyloSim.

Examples

# create a PhyloSim object
sim<-PhyloSim(phylo=coal(5));
# get the edge matrix
getEdges(sim)
# get the edge matrix via virtual field
sim$edges
getEquDist.GeneralSubstitution

Get the equilibrium distribution from a GeneralSubstitution object

Description

Get the equilibrium distribution from a GeneralSubstitution object.

Usage

```r
## S3 method for class 'GeneralSubstitution'
getEquDist(this, dummy=NA, ...)
```

Arguments

- `this`: A GeneralSubstitution object.
- `dummy`: Not used.
- `...`: Not used.

Value

The new equilibrium distribution (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `GeneralSubstitution`.

Examples

```r
# create a GeneralSubstitution object
# provide an Alphabet object and the rates
p<-GeneralSubstitution(
    alphabet=BinaryAlphabet(),
    rate.list=list("1->0"=1,"0->1"=1)
)

# get equilibrium distribution
getEquDist(p)

# get equilibrium distribution via virtual field
p$equDist

# re-set the equilibrium distribution
dist<-p$equDist * 3
dist
setEquDist(p,dist)
p$equDist
```
getEventRate. GeneralSubstitution

Get the scaled rate of an event from a GeneralSubstitution object

Description

Get the scaled rate of an event from a GeneralSubstitution object.

This method return the element from the scaled rate matrix stored in the associated QMatrix object corresponding to a given event. The event can be specified by the initial and target states ("from" and "to" arguments), or by the event name ("from->to"). The event name takes precedence over the "from" and "to" arguments.

This method doesn’t take into account the site specific rate multipliers in any way.

Usage

```r
## S3 method for class 'GeneralSubstitution'
getEventRate(this, name=NA, from=NA, to=NA, ...)
```

Arguments

- `this` A GeneralSubstitution object.
- `name` The name of the event.
- `from` The initial state.
- `to` Target state.
- `...` Not used.

Value

A Numeric vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `GeneralSubstitution`.

# re-set equilibrium distribution via virtual field
p$equDist<-p$equDist * 2
p$equDist
Examples

```r
# create a GeneralSubstitution object
# provide an Alphabet object and the rates
p <- GeneralSubstitution(alphabet=BinaryAlphabet(), rate.list=list("1->0"=1,"0->1"=1))
# get the scaled rate of "0->1" by name
getEventRate(p,"0->1")
# get the scaled rate of "0->1" by states
getEventRate(p, from="0", to="1")
```

### getEventRate.QMatrix

*Get the unscaled rate of an event from a QMatrix object*

**Description**

Get the unscaled rate of an event from a QMatrix object.

This method returns the element corresponding to a given event from the scaled rate matrix stored in a QMatrix object. The event can be specified by the initial and target states ("from" and "to" arguments), or by the event name ("from->to"). The event name takes precedence over the "from" and "to" arguments.

This method returns NA if the rescaling of the rates was not performed. The scaling is performed by the `rescaleQMatrix` method.

**Usage**

```r
## S3 method for class 'QMatrix'
getEventRate(this, name=NA, from=NA, to=NA, ...)
```

**Arguments**

- `this` 
  A QMatrix object.
- `name` 
  The name of the event.
- `from` 
  The initial state.
- `to` 
  Target state.
- `...` 
  Not used.

**Value**

A Numeric vector of length one.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see QMatrix.
getEventRateAtSite.Generalsubstitution

Get the site specific rate of an event from a GeneralSubstitution object

Description

Get the site specific rate of an event from a GeneralSubstitution object.

This method returns the element from the associated QMatrix object corresponding to a given event multiplied by the "rate.multiplier" site-process specific parameter stored in the specified site object. The event can be specified by the initial and target states ("from" and "to" arguments), or by the event name ("from->to"). The event name takes precedence over the "from" and "to" arguments.

Usage

```r
## S3 method for class 'GeneralSubstitution'
getEventRateAtSite(this, site, name=NA, from=NA, to=NA, ...)
```

Arguments

- **this**: A GeneralSubstitution object. It must be attached to the provided Site object.
- **site**: A Site object.
- **name**: The name of the event.
- **from**: The initial state.
- **to**: Target state.
- **...**: Not used.

Value

A Numeric vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `GeneralSubstitution`.

Examples

```r
# create a QMatrix object
# provide an Alphabet object and the rates
m<-QMatrix(alphabet=BinaryAlphabet(), rate=list(0->1=1,1->0=1))
# get the unscaled rate of "0->1" by name
getEventRate(m,0->1) # returns NA
# get the unscaled rate of "0->1" by states
getEventRate(m,from="0",to="1") # returns NA
```
Examples

# create a GeneralSubstitution object
# provide an Alphabet object and the rates
p<-GeneralSubstitution(alphabet=BinaryAlphabet(), rate.list=list("1->0"=1,"0->1"=1))
# create a Site object
s<-Site(alphabet=BinaryAlphabet())
# attach process p to site s
s$processes<-list(p)
# set the rate multiplier for s and p
  setParameterAtSite(p,s,id="rate.multiplier",value=2)
# get the site specific rate of "0->1" by name
getEventRateAtSite(p,s,"0->1")
# get the site specific rate of "0->1" by states
getEventRateAtSite(p,s,from="0",to="1")

getEvents.Sequence  Get the list of active Event objects for a set of Site objects aggregated by a Sequence object

Description

Get the list of active Event objects for a set of Site objects aggregated by a Sequence object.

Usage

## S3 method for class 'Sequence'
getEvents(this, index, ...)

Arguments

this A Sequence object.
index An integer vector specifying a set of positions. It is set to 1:seq$length if omitted.
... Not used.

Value

A list of Event objects.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Sequence.
Examples

# create a sequence with a process attached
s<-Sequence(
  string="ATGC",
  alphabets=list(NucleotideAlphabet()),
  processes=list(list(JC69()))
)

# get the active events from range 1:3
getEvents(s,1:3)

# get all active events via virtual field
s$events

---

getEvents.Site  Get the list of active event objects given the current state of the Site object

Description

Get the list of active event objects given the current state of the Site object. The list of active event object might change according to the state of the Site object.

Usage

```r
## S3 method for class 'Site'
getEvents(this, ...)
```

Arguments

- **this**: A Site object.

- **...**: Not used.

Value

A list of event objects.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Site`.
Examples

# create a site object with a JC69 substitution process attached
s<-Site(alphabet=NucleotideAlphabet(), state="A", processes=list(JC69()))
# get the list of active event objects
getEvents(s)
# modify site state
s$state<-"T"
# get the list of active event objects via virtual field
s$events

getEventsAtSite.FastFieldDeletor

Generate a deletion event object given the state of the target site

Description

Generate a deletion event object given the state of the target site.

Usage

## S3 method for class 'FastFieldDeletor'
getEventsAtSite(this, target.site, ...)

Arguments

this A FastFieldDeletor object.
target.site A Site object.
... Not used.

Value

A list of Event objects.

Author(s)

Botond Sipos, Gregory Jordan

See Also

GeneralDeletor getEventsAtSite.GeneralDeletor
Description

Title.

This method generates a list containing a single deletion event object. The rate of the event is calculated as the product of the general rate of the process and the "rate.multiplier" site-process specific parameter. An empty list is returned if the rate is zero or NA.

Usage

```r
## S3 method for class 'GeneralDeletor'
getEventsAtSite(this, target.site, ...)
```

Arguments

- `this`: A GeneralDeletor object.
- `target.site`: The target Site object.
- `...`: Not used.

Value

A list of event objects.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `GeneralDeletor`.

Examples

```r
# create the Sequence object
s <- NucleotideSequence(string = "ATGCCCCGGATTTATA");
# create a GeneralDeletor object
# proposed deletion length: 4, always accept
d <- GeneralDeletor(
  name = "Del Bosque",
  rate = 0.5,
  propose.by = function(process, sequence, position) { 4 },
  accept.by = function(process, sequence, range) { TRUE }
)
# attach process to site
attachProcess(s, d);
getEventsAtSite.GeneralInsertor

Description

Generate insertion event object given the state of the target site.

This method generates a list with one insertion event. The rate of the event is calculated as the product of the general rate of the process and the "rate.multiplier" site-process specific parameter. An empty list is returned if the rate is zero or NA.

Usage

## S3 method for class 'GeneralInsertor'
getEventsAtSite(this, target.site, ...)

Arguments

- `this`: A GeneralInsertor object.
- `target.site`: A Site object.
- `...`: Not used.

Value

A list of Event objects.

Author(s)

Botond Sipos, Gregory Jordan

See Also

GeneralInsertor GeneralInDel Process Event
Examples

```r
# create a sequence object
s <- NucleotideSequence(string = "AAAA")
# create a GeneralInsertor process, provide template sequence.
# proposed insert lengths: 3, always accept.
i <- GeneralInsertor(
  rate = 0.5,
  template.seq = NucleotideSequence(string = "GGG"),
  propose.by = function(process) {3},
  accept.by = function(process, sequence, window) {TRUE}
)
# attach process to site
s$processes <- list(list(i));
# set rate multiplier
setRateMultipliers(s, i, 2)
# get the list of active events from site 2
events <- getEventsAtSite(i, s$sites[[2]])
events
# set position for event
e <- events[[1]]
e$.position <- 2
# print sequence
s
# perform event
Perform(e)
# check sequence again
s
```

---

`getEventsAtSite.Generalsubstitution`  
*Generate the list of active Event objects for a given attached Site object*

**Description**

Generate the list of active Event objects for a given attached Site object.

This is the single most important method in the `GeneralSubstitution` class. It generates a list of the active Event objects given the transition rate matrix (Q matrix) and the "rate.multiplier" Site-Process specific parameter. It returns an empty list if the state of the site is "NA".

**Usage**

```r
## S3 method for class 'GeneralSubstitution'
getEventsAtSite(this, target.site, ...)
```
getEventsAtSite.GY94

Arguments

- `this`: A GeneralSubstitution object.
- `target.site`: A Site object. The GeneralSubstitution object must be attached to the Site object.
- `...`: Not used.

Value

A list of the active Event objects.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `GeneralSubstitution`.

Examples

```r
# create an Alphabet object
a<-BinaryAlphabet()
# create a Site object
s<-Site(alphabet=a);
# create a GeneralSubstitution object
p<-GeneralSubstitution(alphabet=a,rate_list=list("0->1"=1,"1->0"=1))
# attach process p to site object s
attachProcess(s,p)
# get the rate of active events
getEventsAtSite(p,s) # empty list
# set the state of s
s$state<-1;
# get the rate of active events
getEventsAtSite(p,s)
```

getEventsAtSite.GY94  Generate the list of active Event objects for a given attached Site object

Description

Generate the list of active Event objects for a given attached Site object.

This method is almost an exact duplicate of the `getEventsAtSite.GeneraSubstitution`, with the exception of the portions dealing with the omega site-process specific parameter.
**Usage**

```r
## S3 method for class 'GY94'
getEventsAtSite(this, target.site, ...)
```

**Arguments**

- `this` A GY94 object.
- `target.site` A Site object. The GY94 object must be attached to the Site object.
- `...` Not used.

**Value**

A list of the active Event objects.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

- `getEventsAtSite.GetGeneralSubstitution GeneralSubstitution`

---

**generate the list of active Event objects given a Site object**

**Description**

Generate the list of active Event objects given a Site object. The Process object must be attached to the specified Site object.

This method is crucial for the simulations. For the Process class it returns an empty list. Descendant classes should implement meaningful getEventsAtSite methods.

**Usage**

```r
## S3 method for class 'Process'
getEventsAtSite(this, site, position, ...)
```

**Arguments**

- `this` A Process object.
- `site` A valid Site object.
- `position` The position of the site in the enclosing Sequence object (if any).
- `...` Not used.
getGenerateBy GENERALINSERTOR

Value
An empty list.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see Process.

Examples

# create objects
a<-NucleotideAlphabet()
p<-Process(alphabet=a)
s<-Site(alphabet=a)
# attach Process p to Site s
s$processes<-list(p)
# get active Event objects (empty list)
getEventsAtSite(p,s)

getGenerateBy GENERALINSERTOR

Get the function object used for generating inserts

Description
Get the function object used for generating inserts.

Usage

## S3 method for class 'GENERALINSERTOR'
getGenerateBy(this, ...)

Arguments

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>this</td>
<td>A GENERALINSERTOR object.</td>
</tr>
<tr>
<td>...</td>
<td>Not used.</td>
</tr>
</tbody>
</table>

Value
A function object.

Author(s)
Botond Sipos, Gregory Jordan
See Also

For more information see GeneralInsertor.

Examples

# create a GeneralInsertor object
i <- GeneralInsertor(
  rate = 0.5,
  propose.by = function(process) { sample(c(5:10), 1) }, # inserts between 5 and 10
  template.seq = NucleotideSequence(string = "AAAAAA")
)

# save insert generator
old.gen <- getGenerateBy(i)
# set a new insert generator
i$generateBy <- function(
  process,
  length,
  target.seq,
  event.pos,
  insert.pos
)
{
  return(NucleotideSequence(string = "AATGGCC"))
}

# get the generator function
i$generateBy
# generate insert
generateInsert(i)
# restore old generator
i$generateBy <- old.gen
# generate insert
generateInsert(i)

getHandler.Event

Get the handler function of an Event object

Description

Get the handler function of an Event object.

Usage

## S3 method for class 'Event'
getHandler(this, ...)

Arguments

this An Event object.
... Not used.
Value
  A function object.

Author(s)
  Botond Sipos, Gregory Jordan

See Also
  For more information see Event.

Examples
  # create a sequence and attach a process
  s <- NucleotideSequence(string = "ATGC", processes = list(list(JC69())))
  # get the first active event from the first site
  # only Sequence methods set .position,
  # so s$sites[[1]]$events[[1]] wouldn't work.
  e <- getEvents(s, 1)[[1]]
  # get the handler of e
  getHandler(e)

getid.PhyloSim

  Get the unique identifier of a PhyloSim object

Description
  Get the unique identifier of a PhyloSim object. The unique identifier is the concatenation of
  the class, the object name as returned by getName() and the object hash as returned by hashCode().

Usage
  ## S3 method for class 'PhyloSim'
  getId(this, ...)

Arguments
  this
    A PhyloSim object.
  ...  
    Not used.

Value
  A character vector of length one.

Author(s)
  Botond Sipos, Gregory Jordan
See Also

For more information see PhyloSim.

Examples

```r
# create a PhyloSim object
o <- PhyloSim(name="MySim");
# get id
getId(o);
# get id via virtual field
o$id;
```

---

**getId.Process**  
*Get the unique identifier of a Process object*

### Description

Get the unique identifier of a Process object. The unique identifier is the concatenation of the class, the object name as returned by getName() and the object hash as returned by hashCode().

### Usage

```r
## S3 method for class 'Process'
getId(this, ...)
```

### Arguments

- `this`  
  A Process object.

- `...`  
  Not used.

### Value

A character vector of length one.

### Author(s)

Botond Sipos, Gregory Jordan

### See Also

For more information see Process.
Examples

# create a Process object
p <- Process()
# get unique id
getid(p)
# get unique id via virtual field
p$id

---

getId.QMatrix

Get the unique identifier of a QMatrix object

Description

Get the unique identifier of a QMatrix object.

Usage

## S3 method for class 'QMatrix'
getId(this, ...)

Arguments

this A QMatrix object.
... Not used.

Value

A character vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see QMatrix.

Examples

# create a QMatrix object
m <- QMatrix()
# get object id
getId(m)
# get object id via virtual field
m$id
Description

Get the unique identifier of a Sequence object. The identifier is the concatenation of the object name and the object hash code as returned by hashCode().

Usage

```r
## S3 method for class 'Sequence'
getId(this, ...)
```

Arguments

- `this`: A Sequence object.
- `...`: Not used.

Value

A character vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Sequence`.

Examples

```r
# create a Sequence object.
s <- Sequence(length=5)
# get id
getId(s)
# get id via virtual field
s$id
```
getInsertHook. GeneralInsertor

Get the insert hook function

Description

Get the insert hook function.

The insert hook allows to make various modifications on the insert before performing the insertion. The insert hook function is called by the insertion event handler function. The insert hook takes the sequence generated by the generateInsert method through the "seq" argument. The function must return a Sequence object, which will be inserted in the target sequence.

Usage

```r
## S3 method for class 'GeneralInsertor'
getInsertHook(this, ...)
```

Arguments

- `this` A GeneralInsertor object.
- `...` Not used.

Value

A function object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `GeneralInsertor`.

Examples

```r
# create a GeneralInsertor object
i<-GeneralInsertor(
  rate=0.5,
  propose.by=function(process){sample(c(5:10),1)}, # inserts between 5 and 10
  template.seq=NucleotideSequence(length=5,processes=list(list(JC69())))
)

# set a dummy insert hook
setInsertHook(i,function(seq){return(seq)})

# set a new insert hook via virtual field
i$insertHook<-function(seq){
  seq$processes<-list(list(GTR())) # replace the subsitution process
  return(seq)
```
getInsertionTolerance.Sequence

Get the insertion tolerance site-process specific parameter for a collection of Site objects aggregated by a Sequence object.

Description

Get the insertion tolerance site-process specific parameter for a collection of Site objects aggregated by a Sequence object. This method does some error checking and the calls getParameterAtSites(this=this, process=process, index=index). See getParameterAtSites.Sequence for more details.

Usage

```r
## S3 method for class 'Sequence'
getInsertionTolerance(this, process, index, ...)
```

Arguments

- `this` A Sequence object.
- `process` A Process object.
- `index` An integer vector specifying a set of positions. It is set to 1:seq$length if omitted.
- `...` Not used.

Value

A numeric vector.

Author(s)

Botond Sipos, Gregory Jordan

See Also

getParameterAtSites.Sequence
getKappa.F84

Get the transition transversion rate ratio

Description

Get the transition transversion rate ratio.

Usage

```r
## S3 method for class 'F84'
getKappa(this, ...)
```

Arguments

- `this` A F84 object.
- `...` Not used.

Value

A numeric vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `fXT`.

Examples

```r
# construct a F84 object
p<-F84();
# set/get Kappa
setKappa(p,2)
getKappa(p)
# set/get Kappa via virtual field
p$kappa<-4
p$kappa
# get object summary
summary(p)
```
Description

Get the transition/transversion rate ratio.

Usage

```r
## S3 method for class 'GY94'
getKappa(this, ...)
```

Arguments

- `this`: A GY94 object.
- `...`: Not used.

Value

A numeric vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `GY94`.

Examples

```r
# create a GY94 object
p <- GY94()
# set/get kappa
setKappa(p, 2)
getKappa(p)
# set/get kappa via virtual field
p$kappa <- 3
p$kappa
```
getLength.Sequence

Get the number of Site objects aggregated in a Sequence object

Description

Get the number of Site objects aggregated in a Sequence object.

Usage

```r
## S3 method for class 'Sequence'
getLength(this, ...)
```

Arguments

- `this`: A Sequence object.
- `...`: Not used.

Value

An integer vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Sequence`.

Examples

```r
# create a Sequence object
s <- Sequence(length=5)
# get sequence length
getLength(s)
# get length via virtual field
s$length
```
getLengthParam1.FastFieldDeletor

Get the first length parameter

**Description**

Get the first length parameter.

**Usage**

```r
## S3 method for class 'FastFieldDeletor'
getAddressParam1(this, ...)
```

**Arguments**

- `this` A FastFieldDeletor object.
- `...` Not used.

**Value**

A numeric vector of length one.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `FastFieldDeletor`.

**Examples**

```r
# create a geometric FastFieldDeletor
p<-FastFieldDeletor()
# set/get the first length parameter
setLengthParam1(p,0.5)
getAddressParam1(p)
# set/get the first length parameter via virtual field
p$lengthParam1<-0.2
p$lengthParam1
```
getLengthParam2.FastFieldDeletor

Get the second length parameter

Description
Get the second length parameter.

Usage

```r
## S3 method for class 'FastFieldDeletor'
getLengthParam2(this, ...)
```

Arguments

- `this`: A FastFieldDeletor object.
- `...`: Not used.

Value

A numeric vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `FastFieldDeletor`.

Examples

```r
# create a geometric FastFieldDeletor
p<-FastFieldDeletor()
# set/get the second length parameter
setLengthParam2(p, 0.5)
getLengthParam2(p)
# set/get the second length parameter via virtual field
p$lengthParam2<-0.2
p$lengthParam2
```
getLogFile.PhyloSim

Get the name of the file used for logging

Description

Get the name of the file used for logging.

Usage

## S3 method for class 'PhyloSim'

getLogFile(this, ...)

Arguments

- `this` A PhyloSim object.
- `...` Not used.

Value

A character vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see PhyloSim.

Examples

```
# Create a PhyloSim object
sim<-PhyloSim();
# get the name of the log file
getLogFile(sim)
# modify log file name
setLogFile(sim, "OldLog.txt")
# get/set log file name via virtual field
sim$logFile
sim$logFile<="NewLog"
sim$logFile
```
getLogLevel.PhyloSim  

Get the log level from a PhyloSim object

Description

Get the log level from a PhyloSim object.

Usage

```r
## S3 method for class 'PhyloSim'
getLogLevel(this, ...)
```

Arguments

- `this`: A PhyloSim object.
- `...`: Not used.

Value

The log level as an integer vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

- `setLogLevel.PhyloSim`

Examples

```r
# Create a PhyloSim object
sim <- PhyloSim()
# get/set log level
getLogLevel(sim)
setLogLevel(sim, 0)
# set/get log level via virtual field
sim$logLevel <- -1
sim$logLevel
# clean up
unlink(sim$logFile)
```
getMatrix.QMatrix

Get the unscaled rate matrix from a QMatrix object

Description

Get the unscaled rate matrix from a QMatrix object.

Usage

## S3 method for class 'QMatrix'
getMatrix(this, ...)

Arguments

this        A QMatrix object.
...
    Not used.

Value

A matrix object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see QMatrix.

Examples

# create a GeneralSubstitution object
# provide an Alphabet object and the rates
p<-GeneralSubstitution(alphabet=BinaryAlphabet(), rate.list=list("1->0"=1,"0->1"=3))
# get the QMatrix object from p
m<-p$QMatrix
# get the unscaled rate matrix from m
m$matrix
getMaxLength.ContinuousDeletor

Get the maximum length

Description

Get the maximum length.

Usage

```r
## S3 method for class 'ContinuousDeletor'
getMaxLength(this, ...)
```

Arguments

- `this` A ContinuousDeletor object.
- `...` Not used.

Value

A numeric vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `ContinuousDeletor`.

Examples

```r
# create object
o <- ContinuousDeletor(rate=1)
# set length sampling expression via virtual field
o$dist <- expression(rnorm(1, mean=6, sd=3))
# set/get maxLength
setMaxLength(o, 3)
getMaxLength(o)
# set/get maxLength via virtual field
o$maxLength <- 10
o$maxLength
# propose a length
proposeLength(o)
```
getMaxLength.ContinuousInsertor

Get the maximum length

Description
Get the maximum length.

Usage
```r
## S3 method for class 'ContinuousInsertor'
getMaxLength(this, ...)
```

Arguments
- `this` A ContinuousInsertor object.
- `...` Not used.

Value
A numeric vector of length one.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see `ContinuousInsertor`.

Examples
```r
# create object
do <- ContinuousInsertor(rate=1)
# set length sampling expression via virtual field
do$dist <- expression(rnorm(1, mean=6, sd=3))
# set/get maxLength
setMaxLength(do, 3)
getMaxLength(do)
# set/get maxLength via virtual field
do$maxLength <- 10
do$maxLength
# propose a length
proposeLength(do)
```
getMethodsList.PSRoot  Get the list of applicable methods for an object

Description

Get the list of applicable methods for an object.

Usage

```r
## S3 method for class 'PSRoot'
getMethodsList(this, ...)
```

Arguments

- **this**: A PSRoot object.
- **...**: Not used.

Value

The list of applicable methods.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `PSRoot`.

Examples

```r
# create an object
o<-PSRoot()
# get the applicable methods
getMethodsList(o)
# get methods via virtual field
o$methodsList
```
Description

Get the name of an Event object.

Usage

## S3 method for class 'Event'
getName(this, ...)

Arguments

this  An event object.
...  Not used.

Value

A character vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Event.

Examples

# create an Event object
e<-Event(name="MyEvent")
# get event name
getName(e)
# get name via virtual field
e$name
### Description

Get the name of a PhyloSim object.

### Usage

```r
## S3 method for class 'PhyloSim'
getName(this, ...)```

### Arguments

- **this**: A PhyloSim object.
- **...**: Not used.

### Value

A character vector of length one.

### Author(s)

Botond Sipos, Gregory Jordan

### See Also

For more information see `PhyloSim`.

### Examples

```r
# create a PhyloSim object
o <- PhyloSim();
# set/get name
setName(o, "MySim");
getName(o, "MySim");
# set/get name via virtual field
o$name <- "George";
o$name
```
getName.Process

Get the name of a Process object

Description
Get the name of a Process object.

Usage

```r
## S3 method for class 'Process'
getName(this, ...)
```

Arguments

- `this`: A Process object.
- `...`: Not used.

Value
A character vector of length one.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see `Process`.

Examples

```r
# create a Process object
p <- Process()
# get object name
getName(p)
# get name via virtual field
p$name
```
 Description
Get the name of a QMatrix object.

Usage
```r
## S3 method for class 'QMatrix'
getName(this, ...)
```

Arguments
- `this` A QMatrix object.
- `...` Not used.

Value
A character vector of length one.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see `QMatrix`.

Examples
```
# create a QMatrix object
m<-QMatrix()
# set/get name
setName(m,"Susie Q")
getName(m)
# set/get name via virtual field
m$name<"Q"
m$name
```
Description

Get the name of a Sequence object.

Usage

```r
## S3 method for class 'Sequence'
getName(this, ...)
```

Arguments

- `this` A Sequence object.
- `...` Not used.

Value

A character vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Sequence`.

Examples

```r
# create object
s <- Sequence(length=10);
# get object name
getName(s)
# get name via virtual field
s$name
```
getNedges.PhyloSim

Get the number of edges from phylo object aggregated by a PhyloSim object

Description

Get the number of edges from phylo object aggregated by a PhyloSim object.

Usage

## S3 method for class 'PhyloSim'
getNedges(this, ...)

Arguments

this A PhyloSim object.
...
Not used.

Value

A numeric vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see PhyloSim.

Examples

# create a PhyloSim object
sim<-PhyloSim(phylo=rcol(5));
# get the number of the edges
getNedges(sim)
# get the number of the edges via virtual field
sim$nedges
**getNodes.PhyloSim**

Get the node identifiers from a PhyloSim object

**Description**

Get the node identifiers from a PhyloSim object.

**Usage**

```r
## S3 method for class 'PhyloSim'
getNodes(this, ...)
```

**Arguments**

- `this` A PhyloSim object.
- `...` Not used.

**Value**

A numeric vector.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `PhyloSim`.

**Examples**

```r
# create a PhyloSim object
sim <- PhyloSim(phyl = rcoal(5));
# get the node IDs
getNodes(sim)
# get the node IDs via virtual field
sim$nodes
```
**getNtips.PhyloSim**

Get the number of the tips form a phylo object aggregated by a PhyloSim object

---

**Description**

Get the number of the tips form a phylo object aggregated by a PhyloSim object.

**Usage**

```r
## S3 method for class 'PhyloSim'
getNtips(this, ...) 
```

**Arguments**

- `this` A PhyloSim object
- `...` Not used.

**Value**

A numeric vector of length one.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `PhyloSim`.

**Examples**

```r
# create a PhyloSim object
sim<-PhyloSim(phylo=rcoral(5));
# get the number of tips
getNtips(sim)
# get the number of tips via virtual field
sim$n tips
```
getOmegas.CodonSequence

Get the omegas from a collection of sites

Description

Get the omegas from a collection of sites.

Usage

```r
## S3 method for class 'CodonSequence'
getOmegas(this, process, index, ...)
```

Arguments

- **this**: A CodonSequence object.
- **process**: A process object inheriting from GY94.
- **index**: A vector of positions.
- **...**: Not used.

Value

A numeric vector.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `CodonSequence`.

Examples

```r
# create a GY94 process
p<-GY94()
# create a CodonSequence object,
# attach a process p
s<-CodonSequence(length=20,processes=list(list(p)))
# set omega values in range 1:5
setOmegas(s,p,c(0.5,1.5,1:5))
# get omega values from siutes 1,2,3,10, and 20
getOmegas(s,p,c(1:3,10,20))
```
getOmegaScalingFactor.GY94

Get the omega scaling factor

Description

Get the omega scaling factor.

The rate matrix of the GY94 model is scaled in a way that the expected number of potential substitutions per site is equal to one at equilibrium. The codeml program from the PAML package scales the rate matrix in order to have the expected number of accepted substitutions per site equal to one.

This method calculates the branch length multiplier needed for switching to PAML-style scaling given a fixed omega.

Usage

```r
## S3 method for class 'GY94'
getOmegaScalingFactor(this, omega, ...)
```

Arguments

- `this`: A GY94 object.
- `omega`: The value of omega.
- `...`: Not used.

Value

A numeric vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see GY94.

Examples

```r
# construct a GY94 process object
p <- GY94(kappa=4)
# Calculate scaling factor for omega=2
getOmegaScalingFactor(p, omega=2)
```
**getParameterAtSite.Process**

*Get the value of a site-process specific parameter from a Site object attached to a Process object*

---

**Description**

Get the value of a site-process specific parameter from a Site object attached to a Process object. The Process object must be attached to the Site object.

**Usage**

```r
## S3 method for class 'Process'
getParameterAtSite(this, site, id, ...)
```

**Arguments**

- `this` A Process object.
- `site` A Site object.
- `id` The identifier of the site-process specific parameter.
- `...` Not used.

**Value**

The site-process specific parameter, which is a list containing the following keys: id, name, value, type.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `Process`.

**Examples**

```r
# create a Site and a Process object
a<-Alphabet()
S<-Site(alphabet=a)
P<-Process(alphabet=a)
# attach the process
attachProcess(s,p)
# get the value of the rate multiplier
getParameterAtSite(p,S,"rate.multiplier")
```
getParameterAtSites.Sequence

Get the values of a site-process specific parameter for a process and a collection of Site objects aggregated by a Sequence object.

Description

Get the values of a site-process specific parameter for a process and a collection of Site objects aggregated by a Sequence object.

Usage

```r
## S3 method for class 'Sequence'
getParameterAtSites(this, process, id, index, ...)
```

Arguments

- `this`: A Sequence object.
- `id`: The identifier of the site-process specific parameter.
- `index`: An integer vector specifying a set of positions. It is set to `1:seq$length` if omitted.
- `...`: Not used.

Value

A list of site specific parameters. A site specific parameter is a list storing the id, the name, the value and the type of the parameter.

Author(s)

Botond Sipos, Gregory Jordan

See Also

Site Process For more information see `Sequence`.

Examples

```r
# create a sequence, attach a process
p <- K88()
s <- Sequence(length=6, alphabets=list(NucleotideAlphabet()), processes=list(list(p)))
# set a new pattern of rate multipliers in the
# range 1:3, the default value is 1.0 by the way
setParameterAtSites(s,p,"rate.multiplier",c(2,3),1:3)
# get rate multipliers
getParameterAtSites(s,p,"rate.multiplier")
```
# set a new value for the whole sequence
setParameterAtSites(s,p,"rate.multiplier",0.5)
# get rate multipliers
getParameterAtSites(s,p,"rate.multiplier")

getPhylo.PhyloSim

Get the phylo object aggregated in a PhyloSim object

Description
Get the phylo object aggregated in a PhyloSim object.

Usage

## S3 method for class 'PhyloSim'
getPhylo(this, ...)

Arguments

this
A PhyloSim object.

... Not used.

Value

A phylo object or NA.

Author(s)
Botond Sipos, Gregory Jordan

See Also
The PhyloSim class, the ape package.

Examples

# create a PhyloSim object
sim<-PhyloSim();
# create a phylo object
tree<-rcoal(3);
# get/set phylo object
setPhylo(sim,tree);
getPhylo(sim,tree);
# get/set phylo object via virtual field
sim$tree<-rcoal(5);
sim$tree;
getPosition.Event

Get the position of the Site object associated to an Event object in the enclosing Sequence object.

Description

Get the position of the Site object associated to an Event object in the enclosing Sequence object.

Usage

```r
## S3 method for class 'Event'
getPosition(this, ...)
```

Arguments

- `this`: An Event object.
- `...`: Not used.

Value

A numeric vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Event`.

Examples

```r
# create a sequence and attach a process
s <- NucleotideSequence(string = "ATGC", processes = list(list(JC69())))
# get the first active event from the first site
e <- getEvents(s, 1)[[1]]
# get the position of the site associated with e
ggetPosition(e)
# get position via virtual field
e$position
```
getProbs.DiscreteDeleter

Get the deletion length probabilities

Description

Get the deletion length probabilities.

Usage

## S3 method for class 'DiscreteDeleter'
getProbs(this, ...)

Arguments

this A DiscreteDeletr object.
...
Not used.

Value

A numeric vector with the deletion length probabilities.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see DiscreteDeleter.

Examples

# create a DiscreteDeleter object
d <- DiscreteDeleter(rate=1, sizes=1:3)
# set/get length probabilities
setProbs(d, c(1/3, 1/3, 1/3)) # equal probabilities
getProbs(d)
# set/get length probabilities via virtual field
x <- c(2, 2, 1)
# normalize x
x <- x/sum(x)
d$sprobs <- x
d$sprobs
getProbs.DiscreteInsertor

*Get the insertion length probabilities*

**Description**

Get the insertion length probabilities.

**Usage**

```r
## S3 method for class 'DiscreteInsertor'
getProbs(this, ...)
```

**Arguments**

- `this` A DiscreteInsertor object.
- `...` Not used.

**Value**

A numeric vector with the insertion length probabilities.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `DiscreteInsertor`.

**Examples**

```r
# create a DiscreteInsertor object
i <- DiscreteInsertor(rate = 1, sizes = 1:3)
# set/get length probabilities
setProbs(i, c(1/3, 1/3, 1/3)) # equal probabilities
getProbs(i)
# set/get length probabilities via virtual field
x <- c(2, 2, 1)
# normalize x
x <- x / sum(x)
i$probs <- x
i$probs
```
**getProcess.Event**

*Get the Process object which generated an Event object*

**Description**

Get the Process object which generated an Event object.

**Usage**

```r
## S3 method for class 'Event'
getProcess(this, ...)
```

**Arguments**

- `this`: An Event object.
- `...`: Not used.

**Value**

A Process object.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `Event`.

**Examples**

```r
# create a sequence and attach a process
s<-NucleotideSequence(string="ATGC",processes=list(list(JC69())))
# get the first active event from the first site
e<-s$sites[[1]]$events[[1]]
# get the generator process for e
e$process
```
getProcess.QMatrix

Get the process object associated with a QMatrix object

Description

Get the process object associated with a QMatrix object.

Usage

```r
## S3 method for class 'QMatrix'
getProcess(this, ...)
```

Arguments

- `this`: A QMatrix object.
- `...`: Not used.

Value

A process object, most likely one which inherits from GeneralSubstitution.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see QMatrix.

Examples

```r
# Create a GeneralSubstitution object
p <- GeneralSubstitution(alphabet=BinaryAlphabet())
p
# get the associated QMatrix object from p
m <- p$qMatrix
summary(m)
# get the associated process from m
m$process
# clone p
pp <- clone(p)
# associate m with pp
pp$qMatrix <- m
# associate pp with m
m$process <- pp
m$process
```
getProcesses.Sequence

Get the Process objects attached to the Site objects aggregated by a Sequence object

Description

Get the Process objects attached to the Site objects aggregated by a Sequence object.

Usage

```r
## S3 method for class 'Sequence'
getProcesses(this, index, ...)
```

Arguments

- `this`: A Sequence object.
- `index`: An integer vector specifying a set of positions. It is set to 1:seq$length if omitted.
- `...`: Not used.

Value

A list of lists of Process objects.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Sequence`.

Examples

```r
# create a sequence object with some processes attached
s <- Sequence(
  length = 4,
  alphabets = list(NucleotideAlphabet()),
  processes = list(list(JC69(), K80()), list(GTR()))
)

# get the list of lists of attached processes from positions 1 and 3
getProcesses(s, c(1, 3))

# get processes via virtual field
s$processes
```
getProcesses.Site  Get the list of Process objects attached to a Site object

Description

Get the list of Process objects attached to a Site object.

Usage

```r
## S3 method for class 'Site'
getProcesses(this, ...)
```

Arguments

- `this` A Site object.
- `...` Not used.

Value

A list of Process objects.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Site`.

Examples

```r
# create a Site object with some processes attached
s <- Site(alphabet = NucleotideAlphabet(), processes = list(K80(), JC69()))
# get list of attached Process objects
getProcesses(s)
# get list of attached Process objects via virtual field
s$processes
```
Description

Get the function used for proposing indel lengths.

Usage

```r
## S3 method for class 'GeneralInDel'
getProposeBy(this, ...)
```

Arguments

- `this` A `GeneralInDel` object.
- `...` Not used.

Value

A function object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `GeneralInDel`.

Examples

```r
# create a GeneralInDel object
# proposing events with a constant length of 5
o <- GeneralInDel(rate = 1, propose.by = function(process) {return(5)});
# set/get the proposeBy function
setProposeBy(o, value = function(process) {return(6)})
getProposeBy(o)
# set/get proposeBy function via virtual field
o$proposeBy <- function(process) {return(3)}
o$proposeBy
```
getQMatrix, GeneralSubstitution

_get the QMatrix object aggregated by a GeneralSubstitution object_

**Description**

Get the QMatrix object aggregated by a GeneralSubstitution object.
This method is mostly used internally.

**Usage**

```r
## S3 method for class 'GeneralSubstitution'
getQMatrix(this, ...)
```

**Arguments**

- `this`: A GeneralSubstitution object.
- `...`: Not used.

**Value**

A QMatrix object.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `GeneralSubstitution`.

**Examples**

```r
# create a GeneralSubstitution object
# provide an Alphabet object and the rates
p <- GeneralSubstitution(alphabet = BinaryAlphabet(), rate.list = list("1->0" = 1, "0->1" = 1))
# get the QMatrix object
gQM <- getQMatrix(p)
# get the QMatrix object via virtual field
gQM[, , 1] <- p$qMatrix
# tweak with the QMatrix
setRate(q, "0->1", 2)
# set a new QMatrix for p
setQMatrix(p, q)
summary(p)
# set new QMatrix via virtual field
setRate(q, "1->0", 2)
p$qMatrix <- q
```
getRate.Event

Get the rate of an Event object

Description

Get the rate of an Event object.

Usage

```r
## S3 method for class 'Event'
getRate(this, ...)
```

Arguments

- `this`: An Event object.
- `...`: Not used.

Value

A numeric vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Event`.

Examples

```r
# create an Event object
e <- Event(rate = 0.1)
# get rate
getRate(e)
# get rate via virtual field
e$rate
```
getRate.GeneraInDel

Get the general rate

Description

Get the general rate.

Usage

### S3 method for class 'GeneralInDel'

getRate(this, ...)

Arguments

- this: A GeneralInDel object.
- ...: Not used.

Value

A numeric vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see GeneralInDel.

Examples

```r
# create a GeneralInDel object
go<-'GeneralInDel(rate=0.5)
# get/set general rate
gRate(o)
sRate(o, 1.5)
# get/set rate via virtual field
o$rate
o$rate<-0.3
o$rate
```
getRate.Generalsubstitution

Get an unscaled rate of an event from a GeneralSubstitution object

Description

Get an unscaled rate of an event from a GeneralSubstitution object.

This method gets the element corresponding to a given event form the unscaled Q matrix. A given event. The event can be specified by the initial and target states ("from" and "to" arguments), or by the event name ("from->to"). The event name takes precedence over the "from" and "to" arguments. The rescaled rates (used during simulations) are returned by the getEventRate method. This method doesn’t take into account the site specific rate multipliers in any way.

Usage

```
## S3 method for class 'GeneralSubstitution'
getRate(this, name=NA, from=NA, to=NA, ...)
```

Arguments

- `this` A GeneralSubstitution object.
- `name` The name of the event.
- `from` The initial state.
- `to` Target state.
- `...` Not used.

Value

A Numeric vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see GeneralSubstitution.

Examples

```
# create a GeneralSubstitution object
# provide an Alphabet object and the rates
p<-GeneralSubstitution(alphabet=BinaryAlphabet(), rate.list=list("1->0"=1,"0->1"=1))
# get the unscaled rate of "0->1" by name
getRate(p,"0->1")
# get the unscaled rate of "0->1" by states
```
getRate(p, from="0", to="1")

---

getRate.QMatrix  Get an unscaled rate of an event from a QMatrix object

**Description**

Get an unscaled rate of an event from a QMatrix object.

This method gets the element corresponding to a given event form the *unscaled* rate matrix. A given event. The event can be specified by the initial and target states ("from" and "to" arguments), or by the event name ("from->to"). The event name takes precedence over the "from" and "to" arguments.

**Usage**

```r
## S3 method for class 'QMatrix'
getRate(this, name=NA, from=NA, to=NA, ...)
```

**Arguments**

- **this**: A QMatrix object.
- **name**: The name of the event.
- **from**: The initial state.
- **to**: Target state.
- **...**: Not used.

**Value**

A Numeric vector of length one.

**See Also**

For more information see `QMatrix`.

**Examples**

```r
# create a QMatrix object
# provide an Alphabet object and the rates
m<-QMatrix(alphabet=BinaryAlphabet(), rate=list("1->0"=1,"0->1"=1))
# get the unscaled rate of "0->1" by name
getRate(m,"0->1")
# get the unscaled rate of "0->1" by states
getRate(m,from="0",to="1")
```
getRateList GENERALSUBSTITUTION

Description

Get a list of events and their unscaled rates from a GeneralSubstitution object.

This method returns the list of event rates from the unscaled Q matrix (as returned by the getEventRate method). The returned list contains the rates associated with the corresponding event names.

Usage

## S3 method for class 'GeneralSubstitution'
getRateList(this, ...)

Arguments

this A GeneralSubstitution object.

... Not used.

Value

A list of event rates.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see GeneralSubstitution.

Examples

# create a GeneralSubstitution object
# provide an Alphabet object and the rates
p <- GeneralSubstitution(alphabet = BinaryAlphabet(), rate.list = list("1->0"=1, "0->1"=3))
# get the event rates from the unscaled Q matrix
getRateList(p)
# get rates from the unscaled rate matrix via virtual field p$rateList
# set rates in the unscaled rate matrix
setRateList(p, list("0->1"=1, "1->0"=1))
p$rateList
# set rates in the unscaled rate matrix via virtual field p$rateList <- list("0->1"=3, "1->0"=1);
# check the content of the associated QMatrix object
getRateList.QMatrix

Get a list of events and their unscaled rates from a QMatrix object

Description
Get a list of events and their unscaled rates from a QMatrix object.
This method returns the list of event rates from the unscaled rate matrix. The returned list contains the rates associated with the corresponding event names.

Usage
## S3 method for class 'QMatrix'
getRateList(this, ...)

Arguments
this A GeneralSubstitution object.
... Not used.

Value
A list of event rates.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see QMatrix.

Examples
# create a GeneralSubstitution object
# provide an Alphabet object and the rates
p <- GeneralSubstitution(alphabet=BinaryAlphabet(), rate.list=list("1->0"=1,"0->1"=3))
# get the QMatrix object from p
m <- p$QMatrix
# get the event rates from the unscaled Q matrix
getRateList(m)
# get rates from the unscaled rate matrix via virtual field
m$rateList
# set rates in the unscaled rate matrix
setRateList(m, list("0->1"=1,"1->0"=1))
m$rateList
# set rates in the unscaled rate matrix via virtual field
getRateMultipliers.Sequence

Get the values of the rate multiplier parameters for a given Process object and a collection of Site object aggregated by a Sequence object.

Description

Get the values of the rate multiplier parameters for a given Process object and a collection of Site object aggregated by a Sequence object. This method just calls getParameterAtSites(this=this, process=process, id="rateMultipliers", ...).

See getParameterAtSites.Sequence for details.

Usage

## S3 method for class 'Sequence'
getRateMultipliers(this, process, index, ...)

Arguments

- **this**: A Sequence object.
- **process**: A valid Process object.
- **index**: An integer vector specifying a set of positions. It is set to `1:seq$length` if omitted.
- **...**: Not used.

Value

A numeric vector with the current values of the rate multiplier in the specified range.

Author(s)

Botond Sipos, Gregory Jordan

See Also

setParameterAtSites.Sequence
getRateParam.F81  Forbidden action: the value of a rate parameters

Description
Forbidden action: the value of a rate parameters.
This model has no rate parameters.

Usage

## S3 method for class 'F81'
getRateParam(this, name, ...)

Arguments
- `this`: An object.
- `name`: The name of the rate parameter.
- `...`: Not used.

Value
The object.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see F81.

getRateParam.F84  Get the value of a rate parameter

Description
Get the value of a rate parameter.
The rate parameters are: Kappa.

Usage

## S3 method for class 'F84'
getRateParam(this, name, ...)

getRateParam.GTR

Arguments

this
A F84 object.

name
The name of the rate parameter.

... Not used.

Value

A numeric vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see F84.

Examples

# create F84 object
p<-F84()
# set/get rate parameters
setRateParamList(p,list("Kappa"=3))
getRateParamList(p)
# set/get rate parameters via virtual field
p$rateParamList<-list("Kappa"=2.5)
p$rateParamList
# get object summary
summary(p)

getRateParam.GTR Get the value of a rate parameter

Description

Get the value of a rate parameter.

The rate parameters are named as in PAML (see PAML documentation: http://bit.ly/9SQKZf).

Usage

## S3 method for class 'GTR'
getRateParam(this, name, ...)


Arguments

this A GTR object.
name The name of the rate parameter.
...

Value

A numeric vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see GTR.

Examples

# construct a GTR object
p<-GTR();
# set/get a rate parameter
setRateParam(p,"a",4)
getRateParam(p,"a")
# get object summary
summary(p)

getRateParam.HKY  Get the value of a rate parameter

Description

Get the value of a rate parameter.
The rate parameters are: Alpha, Beta.

Usage

## S3 method for class 'HKY'
getRateParam(this, name, ...)

Arguments

this An HKY object.
name The name of the rate parameter.
...

Not used.
Value
The list of rate parameters.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see HKY.

Examples
```r
# construct HKY object
p <- HKY();
# set/get a rate parameter
setRateParam(p,"Alpha",4)
getRateParam(p,"Beta")
# get object summary
summary(p)
```

getRateParam.K80  
*Get the value of a rate parameter*

Description
Get the value of a rate parameter.
The rate parameters are: Alpha, Beta.

Usage
```r
## S3 method for class 'K80'
getRateParam(this, name, ...)
```

Arguments
- `this`: A K80 object.
- `name`: The name of the rate parameter.
- `...`: Not used.

Value
A numeric vector of length one.

Author(s)
Botond Sipos, Gregory Jordan
getRateParam.K81

Get the value of a rate parameter

Description

Get the value of a rate parameter.

The rate parameters are: Alpha, Beta, Gamma.

Usage

```r
## S3 method for class 'K81'
getRateParam(this, name, ...)
```

Arguments

- `this` A K81 object.
- `name` The name of the rate parameter.
- `...` Not used.

Value

The value of the rate parameter.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see K81.
getRateParam.T92

Examples

```r
# construct a K81 object
p <- K81();
# set/get a rate parameter
setRateParam(p, "Alpha", 4)
getRateParam(p, "Gamma")
# get object summary
summary(p)
```

getRateParam.T92  Get the value of a rate parameter

Description

Get the value of a rate parameter.

The rate parameters are: Alpha, Beta.

Usage

```r
## S3 method for class 'T92'
getRateParam(this, name, ...)
```

Arguments

- `this`  A T92 object.
- `name`  The name of the rate parameter.
- `...`  Not used.

Value

A numeric vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `T92`.  

Examples

```r
# construct a T92 object
p<-T92();
# set/get a rate parameter
setRateParam(p,"Alpha",4)
geRateParam(p,"Beta")
# get object summary
summary(p)
```

---

**getRateParam.TN93**  
*Get the value of a rate parameter*

Description

Get the value of a rate parameter.

The rate parameters are: Alpha1, Alpha2, Beta.

Usage

```r
## S3 method for class 'TN93'
geRateParam(this, name, ...)
```

Arguments

- `this`: A TN93 object.
- `name`: The name of the rate parameter.
- `...`: Not used.

Value

A numeric vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `TN93`. 
Examples

```r
# construct a TN93 object
p <- TN93();
# set/get a rate parameter
setRateParam(p, "Beta", 4)
getRateParam(p, "Beta")
# get object summary
summary(p)
```

Description

Forbidden action: getting the list of rate parameters.

This model has no rate parameters.

Usage

```r
## S3 method for class 'F81'
getRateParamList(this, ...)
```

Arguments

- `this`: An object.
- `...`: Not used.

Value

The object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see F81.
getRateParamList.F84  Get the rate parameters

Description

Get the rate parameters.

The rate parameters are: Kappa.

Usage

## S3 method for class 'F84'
getRateParamList(this, ...)

Arguments

this  A F84 object.

...  Not used.

Value

The list of rate parameters.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see F84.

Examples

```r
# create F84 object
p<-F84()
# set/get rate parameters
setRateParamList(p,list("Kappa"=3))
getRateParamList(p)
# set/get rate parameters via virtual field
p$rateParamList<-list("Kappa"=2.5)
p$rateParamList
# get object summary
summary(p)
```
Get the rate parameters.

The rate parameters are named as in PAML (see PAML documentation: http://bit.ly/9SQK2f).

## Usage

```r
## S3 method for class 'GTR'
getRateParamList(this, ...)
```

## Arguments

- `this`: A GTR object.
- `...`: Not used.

## Value

A list of rate parameters.

## Author(s)

Botond Sipos, Gregory Jordan

## See Also

For more information see `GTR`.

## Examples

```r
# create GTR object
p <- GTR()
# set/get rate parameters
setRateParamList(p, list(
  "a"=1, "b"=2, "c"=3,
  "d"=1, "e"=2, "f"=3
))
getRateParamList(p)
# set/get rate parameters via virtual field
p$rateParamList <- list(
  "a"=4, "b"=1, "c"=4,
  "d"=1, "e"=4, "f"=1
)
p$rateParamList
# get object summary
summary(p)
```
getRateParamList.HKY  

Get the rate parameters

Description

Get the rate parameters.
The rate parameters are: Alpha, Beta.

Usage

```r
## S3 method for class 'HKY'
getRateParamList(this, ...)
```

Arguments

- `this`  
  An HKY object.
- `...`  
  Not used.

Value

The list of rate parameters.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `HKY`.

Examples

```r
# create HKY object
p<-HKY()
# set/get rate parameters
setRateParamList(p,list(
  "Alpha"=1,
  "Beta"=0.5
))
getRateParamList(p)
# set/get rate parameters via virtual field
p$rateParamList<-list(
  "Alpha"=1,
  "Beta"=3
)
p$rateParamList
# get object summary
summary(p)
```
Description

Get the rate parameters.
The rate parameters are: Alpha, Beta.

Usage

```R
## S3 method for class 'K80'
getRateParamList(this, ...)
```

Arguments

- `this`: A K80 object.
- `...`: Not used.

Value

The list of rate parameters.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `K80`.

Examples

```R
# create K80 object
p<-K80()
# set/get rate parameters
setRateParamList(p,list(
  "Alpha"=1,
  "Beta"=0.5
))
getRateParamList(p)
# set/get rate parameters via virtual field
p$rateParamList<-list(
  "Alpha"=1,
  "Beta"=3
)
p$rateParamList
# get object summary
summary(p)
```
**getRateParamList.K81**  
*Get the rate parameters*

**Description**
Get the rate parameters.

The rate parameters are: Alpha, Beta, Gamma.

**Usage**

```r
## S3 method for class 'K81'
generateRateList(this, ...)  
```

**Arguments**

- `this`: A K81 object.
- `...`: Not used.

**Value**

The list of rate parameters.

**Author(s)**
Botond Sipos, Gregory Jordan

**See Also**
For more information see K81.

**Examples**

```r
# create K81 object
p<-K81()
# set/get rate parameters
setRateParamList(p,list(
  "Alpha"=1,
  "Beta"=0.5,
  "Gamma"=2
))
generateRateList(p)
# set/get rate parameters via virtual field
p$rateParamList<-list(
  "Alpha"=1,
  "Beta"=3,
  "Gamma"=2
)
p$rateParamList
```
getRateParamList.T92

# get object summary
summary(p)

---

**getRateParamList.T92  Get the rate parameters**

**Description**

Get the rate parameters.

The rate parameters are: Alpha, Beta.

**Usage**

```r
## S3 method for class 'T92'
getRateParamList(this, ...)
```

**Arguments**

- **this**: A T92 object.
- **...**: Not used.

**Value**

The list of rate parameters.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see T92.

**Examples**

```r
# create a T92 object
p<-T92()
# set/get rate parameters
setRateParamList(p,list(
   "Alpha"=1,
   "Beta"=0.5
 ))
getRateParamList(p)
# set/get rate parameters via virtual field
p$rateParamList<-list(
   "Alpha"=1,
   "Beta"=3
 )
```
Description

Get the rate parameters.

The rate parameters are: Alpha1, Alpha2, Beta.

Usage

```r
## S3 method for class 'TN93'
getRateParamList(this, ...)  
```

Arguments

- `this`: A TN93 object.
- `...`: Not used.

Value

The list of rate parameters.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `TN93`.

Examples

```r
# create TN93 object
p<-'TN93()
# set/get rate parameters
setRateParamList(p,list(
  "Alpha1"=1,
  "Alpha2"=2,
  "Beta"=0.5
))
getRateParamList(p)
# set/get rate parameters via virtual field
p$rateParamList<-'list(
  "Alpha1"=1,
```
**getRootNode.PhyloSim**

Get the identifier of the root node from a PhyloSim object.

### Description

Get the identifier of the root node from a PhyloSim object.

### Usage

```r
## S3 method for class 'PhyloSim'
getRootNode(this, ...) 
```

### Arguments

- `this`: A PhyloSim object.
- `...`: Not used.

### Value

A numeric vector of length one.

### Author(s)

Botond Sipos, Gregory Jordan

### See Also

For more information see `PhyloSim`.

### Examples

```r
# create a PhyloSim object
sim<-PhyloSim(phylo=rcoal(5));
# get the root node ID
getRootNode(sim)
# get the root node ID via virtual field
sim$rootNode
```
getRootSeq.PhyloSim  

Get the root sequence aggregated by a PhyloSim object

Description

Get the root sequence aggregated by a PhyloSim object.

Usage

## S3 method for class 'PhyloSim'
getRootSeq(this, ...)

Arguments

this       A PhyloSim object.
...        Not used.

Value

The root Sequence object or NA.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see PhyloSim. Sequence Process

Examples

# create some objects
sim<-PhyloSim(phylo=rcalc(3));
seq<-NucleotideSequence(string="ATGCC");
# set/get root sequence
setRootSeq(sim, seq);
getRootSeq(sim, seq);
# set/get root sequence via virtual field
sim$rootSeq<-BinarySequence(string="1110011000");
sim$rootSeq;
Description

Get scale parameter.

Usage

## S3 method for class 'BrownianInsertor'
getScale(this, ...)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>this</td>
<td>A BrownianInsertor object.</td>
</tr>
<tr>
<td>...</td>
<td>Not used.</td>
</tr>
</tbody>
</table>

Value

A numeric vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see BrownianInsertor.

Examples

```r
# create a BrownianInsertor object
p <- BrownianInsertor(scale=0.002)
# set/get scale parameter
setScale(p, 0.1)
getScale(p)
# set/get scale parameter via virtual field
p$scale <- 0.1
p$scale
```
getScaledMatrix.QMatrix

Get the scaled rate matrix form a QMatrix object

Description

Get the scaled rate matrix form a QMatrix object.

Usage

```r
## S3 method for class 'QMatrix'
getScaledMatrix(this, ...)
```

Arguments

- `this`: A QMatrix object.
- `...`: Not used.

Value

A matrix object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `QMatrix`.

Examples

```r
# create a GeneralSubstitution object
# provide an Alphabet object and the rates
p<-GeneralSubstitution(alphabet=BinaryAlphabet(), rate.list=list("1->0"=1,"0->1"=3))
# get the QMatrix object from p
m<-p$QMatrix
# get the scaled rate matrix from m
m$scaledMatrix
```
getSeqFromNode.PhyloSim

Get the Sequence object associated with a given node of a phylo object aggregated by a PhyloSim object

Description

Get the Sequence object associated with a given node of a phylo object aggregated by a PhyloSim object.

Usage

```r
## S3 method for class 'PhyloSim'
getSeqFromNode(this, node=NA, ...)
```

Arguments

- **this**: A PhyloSim object.
- **node**: Node identifier.
- **...**: Not used.

Value

A Sequence object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `PhyloSim`.

Examples

```r
# Create a PhyloSim object.
# Provide the phylo object
# and the root sequence.
sim<-PhyloSim(
  name="TinySim",
  phylo=rcoral(3),
  root.seq=NucleotideSequence(string="ATG",processes=list(list(JC69()))))
# get the sequence associated with node 5
getSeqFromNode(sim,5) # Should be NA
# Run the simulation
Simulate(sim)
# try again
```
getSeqFromNode(sim, 5)

---

### getSequence.Site

Get the Sequence object associated with a given Site object.

**Description**

Get the Sequence object associated with a given Site object.

**Usage**

```r
## S3 method for class 'Site'
getSequence(this, 
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>this</td>
<td>A Site object.</td>
</tr>
<tr>
<td>...</td>
<td>Not used.</td>
</tr>
</tbody>
</table>

**Value**

A Sequence object or NA.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `Site`.

**Examples**

```r
# create a site object
s <- Site(sequence=Sequence())
# get the associated Sequence object
getSequence(s)
# get the associated Sequence object via virtual field
s$sequence
```
getSequences.PhyloSim

*Gets all the Sequence objects associated with the nodes of a phylo object aggregated by a PhyloSim object*

**Description**

Gets all the Sequence objects associated with the nodes of a phylo object aggregated by a PhyloSim object.

The order of the Sequence objects in the returned list reflects the identifiers of the associated nodes.

**Usage**

```r
## S3 method for class 'PhyloSim'
getSequences(this, ...)
```

**Arguments**

- `this` A PhyloSim object.
- `...` Not used.

**Value**

A list of sequence objects.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `PhyloSim`.

**Examples**

```r
# Create a PhyloSim object.
# Provide the phylo object
# and the root sequence.
sim<-PhyloSim(
  name="TinySim",
  phylo=rcoal(3),
  root.seq=NucleotideSequence(string="ATG",processes=list(list(JC69()))));
# run the simulation
Simulate(sim)
# get all the associated sequence objects
getSequences(sim)
# get the sequence associated with node 3
# via virtual field
```
**getDescription.**

Get the Site object associated with an Event object.

**Usage**

```r
## S3 method for class 'Event'
getSite(this, ...)
```

**Arguments**

- `this`: An Event object.
- `...`: Not used.

**Value**

A Site object.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `Event`.

**Examples**

```r
# create a sequence and attach a process
s <- NucleotideSequence(string = "ATGC", processes = list(list(JC69())))
# get the first active event from the first site
e <- s$sites[[1]]$events[[1]]
# get the site associated with e
gSite(e)
# get site via virtual field
e$site
```
## getSites.Sequence

**Get the list of the Site object aggregated in a Sequence object**

### Description

Get the list of the Site object aggregated in a Sequence object. Warning: there is no setSites method!

### Usage

```r
## S3 method for class 'Sequence'
getSites(this, ...)
```

### Arguments

- **this**: A Sequence object.
- **...**: Not used.

### Value

A list of Site objects.

### Author(s)

Botond Sipos, Gregory Jordan

### See Also

For more information see `Sequence`.

### Examples

```r
# create a sequence object
s <- Sequence(alphabets=list(NucleotideAlphabet()), string="AATGCCC")
# get the list of aggregated Site objects
getSites(s)
# get Site objects via virtual field
s$sites
```
getSiteSpecificParamIds.Process

*Get the site specific parameter identifiers from a Process object*

---

**Description**

Get the site specific parameter identifiers from a Process object.

**Usage**

```r
## S3 method for class 'Process'
getSiteSpecificParamIds(this, ...)
```

**Arguments**

- `this`: A Process object.
- `...`: Not used.

**Value**

A character vector.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `Process`.

**Examples**

```r
# create process object
p <- Process()
# get site specific parameter identifiers
getSiteSpecificParamIds(p)
# via virtual field
p$sitespecificparamids
```
getSiteSpecificParamList.Process

Get the list of site specific parameters of a Process object

Description
Get the list of site specific parameters of a Process object. Every site specific parameter is a list storing the name, the identifier and the type of the given parameter.

Usage

```r
## S3 method for class 'Process'
getSiteSpecificParamList(this, ...)
```

Arguments

- `this` A Process object.
- `...` Not used.

Value

A list of site specific parameters.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see `Process`.

Examples

```r
# create a process object
p <- Process()
# get the list of site specific parameters
getSiteSpecificParamList(p)
# get it via virtual field
p$siteSpecificParamList
```
GetSize.Alphabet  
*Get the number of symbols in an Alphabet object*

Description

Get the number of symbols in an Alphabet object.

Usage

```
## S3 method for class 'Alphabet'
getSize(this, ...)
```

Arguments

- `this`: An Alphabet object.
- `...`: Not used.

Value

An integer vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Alphabet`.

Examples

```
# create an alphabet object
a<-Alphabet(symbols=c(0,1,2,3,4,5))
a
# get alphabet size
getSize(a)
```
**getSizes.DiscreteDeletor**

*Get the sizes of the proposed deletions*

---

**Description**

Get the sizes of the proposed deletions.

**Usage**

```r
## S3 method for class 'DiscreteDeletor'
getSizes(this, ...)
```

**Arguments**

- `this` A DiscreteDeletor object.
- `...` Not used.

**Value**

A vector of integers.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `DiscreteDeletor`.

**Examples**

```r
# create a DiscreteDeletor object
d <- DiscreteDeletor(rate=1)
# set deletion sizes
setSizes(d, c(1, 2, 3))
# get deletion sizes
getSizes(d)
# set/get sizes via virtual field
d$sizes <- 1:10
d$sizes
```
getSizes.DiscreteInsertor

*Get the sizes of the proposed insertions*

**Description**

Get the sizes of the proposed insertions.

**Usage**

```r
## S3 method for class 'DiscreteInsertor'
getSizes(this, ...)
```

**Arguments**

- `this`: A DiscreteInsertor object.
- `...`: Not used.

**Value**

A vector of integers.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `DiscreteInsertor`.

**Examples**

```r
# create a DiscreteInsertor object
i<-DiscreteInsertor(rate=1)
# set insertion sizes
sizes(i,c(1,2,3))
# get insertion sizes
getSizes(i)
# set/get sizes via virtual field
i$sizes<-1:10
i$sizes
```
Description

Get the current state of a Site object.

Usage

```r
## S3 method for class 'Site'
getState(this, ...)
```

Arguments

- `this`: A Site object.
- `...`: Not used.

Value

A character vector of length one containing the state (a symbol belonging to the attached Alphabet object).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Site`. 

Examples

```r
# create a Site object with an Alphabet object attached
s <- Site(alphabet=Alphabet(symbols=c(0,1)), state=1);
# get current state
getState(s)
# get state via virtual field
s$state
```
**getStates.Sequence**

Get the states of a set of Site objects aggregated by a Sequence object.

### Description

Get the states of a set of Site objects aggregated by a Sequence object.

### Usage

```r
## S3 method for class 'Sequence'
getStates(this, index, ...)
```

### Arguments

- `this` A Sequence object.
- `index` An integer vector specifying a set of positions. It is set to 1:seq$length if omitted.
- `...` Not used.

### Value

A character vector.

### Author(s)

Botond Sipos, Gregory Jordan

### See Also

For more information see `Sequence`.

### Examples

```r
# create a sequence object
s <- Sequence(alphabets = list(NucleotideAlphabet()), string = "AATGCCCCCTTGG")
# get all Site states
getStates(s)
# get states for a collection of sites
getStates(s, c(1:3, 5, 8))
# get states via virtual field
s$states
```
getString.Sequence  

Get the string representation of a Sequence object

Description

Get the string representation of a Sequence object. The string representation is the concatenation of the states of the aggregated Site object. Undefined states (NA-s) are represented by question marks.

Usage

```r
## S3 method for class 'Sequence'
getString(this, ...)
```

Arguments

- **this**: A Sequence object.
- **...**: Not used.

Value

A character vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Sequence`.

Examples

```r
# create object
s <- Sequence(length=10)
# get character representation
getString(s) # a bunch of '?'-s
# get string representation via virtual field
s$string
```
getSymbolFreqs.Sequence

Get a table with the frequencies of the states of a collection of Site objects aggregated by a Sequence object

Description

Get a table with the frequencies of the states of a collection of Site objects aggregated by a Sequence object.

Usage

## S3 method for class 'Sequence'
getSymbolFreqs(this, index, ...)

Arguments

- **this**: A Sequence object.
- **index**: An integer vector specifying a set of positions. It is set to 1:seq$length if omitted.
- **...**: Not used.

Value

A table.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Sequence.

Examples

```r
# create a nucleotide sequence
s <- NucleotideSequence(length = 30, processes = list(list(JC69())))
# sample states
sampleStates(s)
# get state frequencies from ranges 1:10 and 20:30
getSymbolFreqs(s, c(1:10, 20:30))
# get symbol frequencies for the full sequence
getSymbolFreqs(s)
```
Description

Get the length of the symbols in a given alphabet object.

Usage

```r
## S3 method for class 'Alphabet'
getSymbolLength(this, ...)
```

Arguments

- `this`: An Alphabet object.
- `...`: Not used.

Value

A numeric vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `alphabet`.

Examples

```r
# create an alphabet object
a <- Alphabet(symbols=c("AAA","AAC"));
# get symbol length
getSymbolLength(a);
# get symbol length via virtual field
a$symbolLength
```
getSymbols.Alphabet  
*Get the symbol set from an Alphabet object*

**Description**

Get the symbol set from an Alphabet object.

**Usage**

```r
## S3 method for class 'Alphabet'
getSymbols(this, ...)
```

**Arguments**

- `this`: An Alphabet object.
- `...`: Not used.

**Value**

A character vector containing the symbol set of the Alphabet object.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `Alphabet`.

**Examples**

```r
# create a new alphabet object
a <- Alphabet(symbols = c("AC","GT"));
# get the symbols
getSymbols(a)
# get the symbols by using the virtual field
a$symbols
```
getTableId.CodonAlphabet

Get the genetic code id

Description

Get the genetic code id.

Usage

```r
## S3 method for class 'CodonAlphabet'
getTableId(this, ...)
```

Arguments

- `this` A CodonAlphabet object.
- `...` Not used.

Value

A numeric vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `CodonAlphabet`.

Examples

```r
# create CodonAlphabet object
a<-CodonAlphabet()
# get genetic code id
globalId(a)
# get genetic code id via virtual field
a$tableId
```
getTemplateSeq.GeneralInsertor

Get the template sequence object

Description

Get the template sequence object. The template sequence object is used by the default `generateBy` function to generate insert sequences.

Usage

```r
## S3 method for class 'GeneralInsertor'
getTemplateSeq(this, ...)
```

Arguments

- `this`: A `GeneralInsertor` object.
- `...`: Not used.

Value

A `Sequence` object or NA.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `GeneralInsertor`.

Examples

```r
# create a GeneralInsertor object
i <- GeneralInsertor(
  rate = 0.5,
  propose.by = function(process)(sample(c(5:10),1)), # inserts between 5 and 10
  template.seq = NucleotideSequence(string = "AAAAAA")
)
# get template sequence
getTemplateSeq(i)
# get template sequence via virtual field
i$templateSeq
# set template sequence
setTemplateSeq(i, NucleotideSequence(string = "C"));
# generate insert
generateInsert(i)
# set template sequence via virtual field
i$templateSeq <- NucleotideSequence(string = "G")
```
get Theta.T92

```r
# generate insert
generateInsert(i)
```

---

**getTheta.T92**  
*Get the GC content*

---

**Description**

Get the GC content.

**Usage**

```r
## S3 method for class 'T92'
getTheta(this, ...)
```

**Arguments**

- `this`  
  A T92 object.

- `...`  
  Not used.

**Value**

A numeric vector of length one.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `T92`.

**Examples**

```r
# construct a T92 object
p<-T92()
# set/get GC content
setTheta(p, 0.6)
gamma(p)
# set/get GC content via virtual field
p$theta<-0.3
p$theta
# get object summary
summary(p)
```
getTipLabels.PhyloSim  *Get the tip labels from a phylo object aggregated by a PhyloSim object*

**Description**

Get the tip labels from a phylo object aggregated by a PhyloSim object.

**Usage**

```r
## S3 method for class 'PhyloSim'
getTipLabels(this, ...)  
```

**Arguments**

- `this`: A PhyloSim object.
- `...`: Not used.

**Value**

A matrix containing the tip labels.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `PhyloSim`.

**Examples**

```r
# create a PhyloSim object
sim <- PhyloSim(phylo = rcoal(5));
# get the tip labels
getTipLabels(sim)
# get the tip labels via virtual field
sim$tipLabels
```
getTips.PhyloSim

Get the node identifiers of the tip nodes from a PhyloSim object

Description

Get the node identifiers of the tip nodes from a PhyloSim object.

Usage

```r
## S3 method for class 'PhyloSim'
getTips(this, ...)
```

Arguments

- `this`: A PhyloSim object
- `...`: Not used.

Value

A numeric vector.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `PhyloSim`.

Examples

```r
# create a PhyloSim object
sim <- PhyloSim(phylo=rcola(5));
# get the tip IDs
getTips(sim)
# get the tip IDs via virtual field
sim$tips
```
getToleranceMargin.FastFieldDeletor

*Get the tolerance margin*

**Description**

Get the tolerance margin.

**Usage**

```r
# S3 method for class 'FastFieldDeletor'
getToleranceMargin(this, ...)  
```

**Arguments**

- `this` A `FastFieldDeletor` object.
- `...` Not used.

**Value**

A numeric vector of length one.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `FastFieldDeletor`.

**Examples**

```r
# create a geometric `FastFieldDeletor`
p <- FastFieldDeletor()
# set/get tolerance margin
setToleranceMargin(p, 0.8)
getToleranceMargin(p)
# set/get tolerance margin via virtual field
p$toleranceMargin <- 0.75
p$toleranceMargin
```
getTotalRate.Site

Get the total active event rate

Description

Get the total active event rate. The total rate is the sum of the rates of all active events given the current state of the Site object.

Usage

```r
## S3 method for class 'Site'
getTotalRate(this, ...)
```

Arguments

- `this`: A Site object.
- `...`: Not used.

Value

A numeric vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Site`.

Examples

```r
# create a nucleotide site with a JC69 substitution process attached
s <- Site(state="A", alphabet=NucleotideAlphabet(), processes=list(JC69()))
# get the total rate
g getTotalRate(s)
# add a new process
attachProcess(s, K80(rate.params=list("Alpha"=1, "Beta"=0.5)))
# get the total rate via virtual field
s$totalRate
```
getTotalRatesFromRange.Sequence

Get the total site rates from a Sequence object

Description

Get the total site rates from a Sequence object. This method simply calls getTotalRatesFromRange(this). See getTotalRatesFromRange.Sequence for more details.

Usage

```r
## S3 method for class 'Sequence'
getTotalRates(this, ...)
```

Arguments

- `this` A Sequence object.
- `...` Not used.

Value

A numeric vector containing the total site rates.

Author(s)

Botond Sipos, Gregory Jordan

See Also

g getTotalRatesFromRange.Sequence

g getTotalRatesFromRange.Sequence

Get the vector of total site rates for a collection of Site objects aggregated by a Sequence object

Description

Get the vector of total site rates for a collection of Site objects aggregated by a Sequence object.

Usage

```r
## S3 method for class 'Sequence'
getTotalRatesFromRange(this, index, ...)```
getTransTable.CodonAlphabet

Arguments

this A Sequence object.

index An integer vector specifying a set of positions. It is set to 1:seq$length if omitted.

... Not used.

Value

A numeric vector.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Sequence.

Examples

# create a sequence with some processes attached
s <- Sequence(
  string = "ATGC",
  alphabets = list(NucleotideAlphabet()),
  processes = list(list(JC69()), list(JC69(), GTR))
)

# get total rates for positions 1 and 3
getTotalRatesFromRange(s, c(1, 3))
# get all total rates via virtual field
s$totalRates # via the "getTotalRates.Sequence" method

getTransTable.CodonAlphabet

Get the list storing the genetic code table

Description

Get the list storing the genetic code table.

Usage

## S3 method for class 'CodonAlphabet'
getTransTable(this, ...)

getTreeLength.PhyloSim

**Arguments**

- `this` A CodonAlphabet object.
- `...` Not used.

**Value**

A list.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `CodonAlphabet`.

**Examples**

```r
# create object
da <- CodonAlphabet()
# get genetic code table
getTransTable(da)
# get genetic code table via virtual field
da$transTable
```

---

**Description**

Get the tree length from a PhyloSim object.

This method returns the sum of the edge lengths stored in the aggregated phylo object.

**Usage**

```r
## S3 method for class 'PhyloSim'
getTreeLength(this, ...)
```

**Arguments**

- `this` A PhyloSim object.
- `...` Not used.

**Value**

A numeric vector of length one.
**getAlpha.Alphabet**

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see *PhyloSim*.

**Examples**

```r
# create a PhyloSim object
sim <- PhyloSim(phylo = rcoal(5));
# get the tree length
getTreeLength(sim)
# get tree length via virtual field
sim$treeLength
```

---

**getDescription**

Get Alphabet he object type.

**Usage**

```r
## S3 method for class 'Alphabet'
getMethod(this, ...)  
```

**Arguments**

- `this` An Alphabet object.
- `...` Not used.

**Value**

A character vector of length one.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see *Alphabet*.  

Examples

```r
# create alphabet object
a <- Alphabet(symbols = c(0, 1), type = "Binary");
# get alphabet type
getType(a)
a$type
```

---

**getDescription.**

*Get the type of the BrownianInsertor process*

**Description**

Get the type of the BrownianInsertor process.

If type is discrete, than the process inherits from `DiscreteInsertor`. If the type is continuous, then the object inherits from `ContinuousInsertor`.

**Usage**

```r
## S3 method for class 'BrownianInsertor'
getType(this, ...)
```

**Arguments**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>this</td>
<td>A BrownianInsertor object.</td>
</tr>
<tr>
<td>...</td>
<td>Not used.</td>
</tr>
</tbody>
</table>

**Value**

A character vector of length one.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `BrownianInsertor`.

Examples

```r
p <- BrownianInsertor(type = "discrete")
# get type
getType(p)
# get upstream classes
class(p)
p <- BrownianInsertor(type = "continuous")
# get type via virtual field
p$type
# get upstream classes
class(p)
```

---

**getDescription.FastFieldDeletor**

*Get the type of a FastFieldDeletor object*

**Description**

Get the type of a FastFieldDeletor object.

**Usage**

```r
## S3 method for class 'FastFieldDeletor'
getType(this, ...)
```

**Arguments**

- `this` A FastFieldDeletor object.
- `...` Not used.

**Value**

A character vector of length one.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `FastFieldDeletor`.
Examples

```r
# create a FastFieldDeleter, default type (geometric)
p <- FastFieldDeleter()
# get type
getType(p)
# create a FastFieldDeleter, poisson type
p <- FastFieldDeleter(type = "poisson")
p$type
```

**Description**

Get the list of unique Alphabet objects associated to Site objects aggregated by a Sequence object. The returned list contains unique instances of the Alphabet class. The symbol sets are not compared, so two instances of the same class are considered to be different.

**Usage**

```r
## S3 method for class 'Sequence'
getUniqueAlphabets(this, ...)
```

**Arguments**

- `this` A Sequence object.
- `...` Not used.

**Value**

A list of Alphabet objects.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `Sequence`. 
getUniqueProcesses.Sequence

Examples

# create a Sequence object with some Alphabet objects attached
s <- Sequence(
    alphabets = list(NucleotideAlphabet(),
                    BinaryAlphabet(),
                    NucleotideAlphabet(),
                    length = 10
    )
)
# get the list of attached alphabets
s$alphabets
# get the unique list of attached Alphabet objects
getUniqueAlphabets(s)

getUniqueProcesses.Sequence

Get the list of unique Process instances attached to the Site objects
aggregated by a Sequence object

Description

Get the list of unique Process instances attached to the Site objects aggregated by a Sequence object.

Usage

## S3 method for class 'Sequence'
getUniqueProcesses(this, ...)

Arguments

this A Sequence object.
...
Not used.

Value

A list of Process objects.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Sequence.
Examples

```r
# create a sequence object and attach processes
p <- JC69()
s <- Sequence(
  length=4,
  alphabets = list(NucleotideAlphabet()),
  processes = list(p, K80(), list(p))
)

# get the unique list of attached Process instances
getUniqueProcesses(s)
# via virtual field
s$uniqueProcesses
```

getWriteProtected.Alphabet

*Check if the object is write protected*

Description

Check if the object is write protected. Write protected objects cannot be modified through get/set methods and virtual fields.

Usage

```r
## S3 method for class 'Alphabet'
getWriteProtected(this, ...)
```

Arguments

- `this`: An object.
- `...`: Not used.

Value

TRUE or FALSE.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see *Alphabet*. 
getWriteProtected.Event

*Check if the object is write protected*

**Examples**

```r
# create an object
o<-Alphabet()
# toggle write protection
o$writeProtected<-TRUE
# check if it's write protected
gWriteProtected(o)
# check write protection via virtual field
o$writeProtected
```

**Description**

Check if the object is write protected. Write protected objects cannot be modified through get/set methods and virtual fields.

**Usage**

```r
## S3 method for class 'Event'
gWriteProtected(this, ...)
```

**Arguments**

- `this` An object.
- `...` Not used.

**Value**

TRUE or FALSE

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see Event.
Examples

```r
# create an object
o<-Event()
# toggle write protection
o$writeProtected<-TRUE
# check if it's write protected
getWriteProtected(o)
# check write protection via virtual field
o$writeProtected
```

getWriteProtected.Process

Check if the object is write protected

Description

Check if the object is write protected. Write protected objects cannot be modified through get/set methods and virtual fields.

Usage

```r
## S3 method for class 'Process'
getWriteProtected(this, ...)
```

Arguments

- `this` A Process object.
- `...` Not used.

Value

TRUE or FALSE

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Process.
Examples

```r
# create an object
o <- Process()
# toggle write protection
o$writeProtected <- TRUE
# check if it's write protected
getWriteProtected(o)
# check write protection via virtual field
o$writeProtected
```

getWriteProtected.QMatrix

**Check if the object is write protected**

Description

Check if the object is write protected.

QMatrix object do not have a write protection flag of their own, but they use the one from the associated Process object. Write protected objects cannot be modified through get/set methods and virtual fields.

Usage

```r
## S3 method for class 'QMatrix'
getWriteProtected(this, ...)
```

Arguments

- `this` A QMatrix object.
- `...` Not used.

Value

TRUE or FALSE

Author(s)

Botond Sipos, Gregory Jordan

See Also

getWriteProtected.Process
**getWriteProtected.Sequence**

*Check if the object is write protected*

**Description**

Check if the object is write protected. Write protected objects cannot be modified through get/set methods and virtual fields.

**Usage**

```r
## S3 method for class 'Sequence'
getWriteProtected(this, ...)
```

**Arguments**

- `this`: An object.
- `...`: Not used.

**Value**

`TRUE` or `FALSE`

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see [Sequence](#).

**Examples**

```r
# create an object
o <- Sequence()
# toggle write protection
o$writeProtected <- TRUE
# check if it's write protected
getWriteProtected(o)
# check write protection via virtual field
o$writeProtected
```
globalConsistencyCheck.PSRoot

Check the consistency of all objects inheriting form PSRoot in the current environment

Description

Check the consistency of all objects inheriting form PSRoot in the current environment.
This method searches for objects which inherit from PSRoot and calls checkConsistency() for all of them, which can take a lots of time. Prints the results of the checks as text.

Usage

## S3 method for class 'PSRoot'
globalConsistencyCheck(...)  

Arguments

...  

Not used.

Value

Returns invisible TRUE if all checks are successful.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see PSRoot.

Examples

# create some objects
a<-NucleotideAlphabet()
s<-Site()
p<-Process()
# ask for a global consistency check
PSRoot$globalConsistencyCheck();
The GTR class

Description

This class implements the general time-reversible nucleotide substitution model (GTR, REV). The rate parameters are named as in PAML (see PAML documentation: http://bit.ly/9SQK2f).

The default value for the rate parameters is 1 and the default value for the base frequencies is 0.25. So the GTR objects are equivalent to JC69 objects by default.

Package:
Class GTR

Usage

```r
GTR(
  name = "Anonymous", rate.params = list(a = 1, b = 1, c = 1, d = 1, e = 1, f = 1),
  base.freqs = rep(0.25, times = 4), ...
)
```

Arguments

- **name**: Object name.
- **rate.params**: A list of unscaled rates (see setRateList.GeneralSubstitution).
- **base.freqs**: Equilibrium distribution.
- **...**: Additional arguments.
Fields and Methods

Methods:

checkConsistency -
getBaseFreqs -
getRateParam -
getRateParamList -
setBaseFreqs -
setRateParam -
setRateParamList -
summary -

Methods inherited from UNREST:
checkConsistency, summary

Methods inherited from GeneralSubstitution:
as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is.na, is, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
[1, , , <-, $, $<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)

Botond Sipos, Gregory Jordan

References


See Also

GeneralSubstitution UNREST HKY
Examples

```r
# create substitution process object
p<-GTR(
  rate.params=list(
    "a"=1, "b"=2, "c"=3,
    "d"=1, "e"=2, "f"=3
  ),
  base.freqs=c(2,2,1,1)/6
)
  
  # get a summary
  summary(p)
  
  # display a bubble plot
  plot(p)
  
  # The following code demonstrates how to use
  # the process in a simulation.
  
  # create a sequence, attach process p
  s<-NucleotideSequence(length=20,processes=list(list(p))))
  
  # sample states
  sampleStates(s)
  
  # make the first five positions invariable
  setRateMultipliers(s,p,0,1:5)
  
  # get rate multipliers
  getRateMultipliers(s,p)
  
  # create a simulation object and run simulation
  sim<-PhyloSim(root.seq=s,phylo=rcoal(2))
  
  Simulate(sim)
  
  # print alignment
  sim$alignment
```

GY94

The GY94 class

Description

This class implements the codon substitution model of Goldman and Yang (1994). The transition/transversion rate ratio is stored in the kappa virtual field. The nonsynonymous/synonymous substitution rate ratio (omega) is a site-process specific parameter with a default value of one. Hence, after the attachment of the process the variation of omega ratios among sites follows the M0 model (see Yang et al. 2000).

The rate matrix of the GY94 model is scaled in a way that the expected number of potential substitutions per site is equal to one at equilibrium. The codeml program from the PAML package scales the rate matrix in order to have the expected number of accepted substitutions per site equal to one. Use the getOmegaScalingFactor.GY94 method to calculate a branch length scaling factor which allows to switch to a PAML-style scaling given an average omega.

If the scale.nuc constructor argument is TRUE, the rates of the returned Event objects will be multiplied by 3 to obtain a process which has the expected number of nucleotide substitutions (not
codon substitutions) equal to one at equilibrium. This is useful when simulating mixed sequences. This option doesn’t affect the rate matrix in any way.

The M1-M4 models are implemented in the omegaVar\textsuperscript{M1-4}.CodonSequence methods. Simulations under more complex models (M5-M13) can be achieved by first discretizing them using the M5-13 tool from the INDELible software package (http://abacus.gene.ucl.ac.uk/software/indelible/). After discretization, the M5-M13 models can be simulated through the M3 (discrete) model.

Package:
Class GY94

Object
~~|  
~~|---PSRoot  
~~~~~~|  
~~~~~~|---Process  
~~~~~~~~~~|  
~~~~~~~~~~|---GeneralSubstitution  
~~~~~~~~~~|  
~~~~~~~~~~|---CodonUNREST  
~~~~~~~~~~|  
~~~~~~~~~~|---GY94

Directly known subclasses:

public static class GY94
extends CodonUNREST

Usage

GY94(
    name="Anonymous", table.id=1, kappa=1,
    omega.default=1, codon.freqs=NA, scale.nuc=FALSE, ...  
)

Arguments

name Name of the object.
table.id The identifier of the genetic code table to use (1 by default).
kappa The transition/transversion rate ratio (1 by default).
omega.default The default value of the omega site-process specific parameter (1 by default).
codon.freqs A vector of codon frequencies.
scale.nuc Scale to nucleotide substitutions if TRUE (see above).
... Additional arguments.
Fields and Methods

Methods:

checkConsistency, getCodonFreqs, getEventsAtSite, getKappa, getOmegaScalingFactor, is, setCodonFreqs, setKappa, setRate, summary

Methods inherited from CodonUNREST:
checkConsistency, is

Methods inherited from GeneralSubstitution:
as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, na, is, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
[[], [<-, $, $<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldType, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)
Botond Sipos, Gregory Jordan

References
See Also

CodonUNREST GeneralSubstitution CodonSequence GTR WAG

Examples

```r
# create a GY94 object
p <- GY94(kappa=2)
# check if inherits from GY94
is.GY94(p)
# get object summary
summary(p)
# display a bubble plot
plot(p)
# create a codon sequence, attach process
s <- CodonSequence(length=5, processes=list(list(p)))
# sample states
sampleStates(s)
# make first three positions invariable
setRateMultipliers(s, p, 0, 1:3)
# sample omega values from the M3 (discrete) model.
omegaVarM3(s, p, omegas=c(0, 1, 2, 3), probs=c(2/5, 1/5, 1/5, 1/5))
# get a histogram of omega values in s
omegahist(s, p, breaks=50)
    sim <- PhyloSim(root.seq=s, phylo=rcoal(2))
    # run simulation
    Simulate(sim)
# get the list of recorded per-branch event counts
getBranchEvents(sim)
# export the number of synonymous substitutions as a phylo object
syn.subst <- exportStatTree(sim, "nr.syn.subst")
syn.subst
# plot the exported phylo object
plot(syn.subst)
# print alignment
sim$alignment
```

### hasSiteSpecificParameter.Process

Check if a Process object has the site-process specific parameter with the given id.

#### Description

Check if a Process object has the site-process specific parameter with the given id.

#### Usage

```r
## S3 method for class 'Process'
hasSiteSpecificParameter(this, id, ...)
```
Arguments

  this    A Process object.
  id      The identifier of the site-process specific parameter of interest.
  ...     Not used.

Value

  TRUE or FALSE.

Author(s)

  Botond Sipos, Gregory Jordan

See Also

  For more information see Process.

Examples

  # create a process object
  p <- Process()
  # check whether it has the "rate.multiplier" site-process specific parameter
  hasSiteSpecificParameter(p,"rate.multiplier"); # TRUE
  # check whether it has the "omega" site-process specific parameter
  hasSiteSpecificParameter(p,"omega"); # FALSE
hasUndefinedRate GeneralInDel

Check whether the general rate of a GeneralInDel object is undefined

Description
Check whether the general rate of a GeneralInDel object is undefined.

Usage
## S3 method for class 'GeneralInDel'
hasUndefinedRate(this, ...)

Arguments
this A GeneralInDel object.
... Not used.

Value
TRUE or FALSE.
Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see GeneralIndel.

Examples
C create a GeneralIndel object
o<-'GeneralIndel()
# check if the general rate is undefined
hasUndefinedRate(o)
# set general rate
o$rate<-1
# check rate again
hasUndefinedRate(o)

Description
Check if a GeneralSubstitution object has undefined rates.

Usage
## S3 method for class 'GeneralSubstitution'
hasUndefinedRate(this, ...)

Arguments
this A GeneralSubstitution object.
... Not used.

Value
TRUE or FALSE.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see GeneralSubstitution.
hasUndefinedRate.Process

Examples

```r
c create a GeneralSubstitution object
p<-GeneralSubstitution(alphabet=BinaryAlphabet())
c check if it has undefined rates
hasUndefinedRate(p) # TRUE
c set the missing rates
p$rateList<-list("0->1"=1,"1->0"=2)
c check for undefined rates again
hasUndefinedRate(p) # FALSE
```

Description

Check if the Process object has undefined rate parameters.

For the instances of the Process class this method always returns FALSE. Descendant classes should implement more meaningful methods.

Usage

```r
# S3 method for class 'Process'
hasUndefinedRate(this, ...)
```

Arguments

- `this`: A Process object.
- `...`: Not used.

Value

FALSE

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Process.
Examples

```java
# create object
p<Process()
# check if has undefined rates
hasUndefinedRate(p) # return FALSE
```

---

**HKY**

**The HKY class**

Description

This class implements the HKY GTR-submodel.

The rate parameters are the following: "Alpha", "Beta". Package:

Class HKY

```
Object
~~|
~~+-PSRoot
~~~~~~|
~~~~~~~~+-Process
~~~~~~~~~~|
~~~~~~~~~~~~+-GeneralSubstitution
~~~~~~~~~~~~~~|
~~~~~~~~~~~~~~~~+-UNREST
~~~~~~~~~~~~~~~~~~|
~~~~~~~~~~~~~~~~~~~+-GTR
~~~~~~~~~~~~~~~~~~~~|
~~~~~~~~~~~~~~~~~~~~~~+-HKY
```

Directly known subclasses:

- public static class **HKY**
  - extends GTR

Usage

```java
HKY(
    name="Anonymous", rate.params=list(Alpha = 1, Beta = 1),
    base.freqs=c(0.25, 0.25, 0.25, 0.25), ...)
```
Arguments

name  Object name.
rate.params  Rate parameters.
base.freqs  Base frequency parameters.
...  Not used.

Fields and Methods

Methods:

checkConsistency  -
getBaseFreqs  -
getRateParam  -
getRateParamList  -
setBaseFreqs  -
setRateParam  -
setRateParamList  -
summary  -

Methods inherited from GTR:
checkConsistency, getBaseFreqs, getRateParam, getRateParamList, setBaseFreqs, setRateParam, setRateParamList, summary

Methods inherited from UNREST:
checkConsistency, summary

Methods inherited from GeneralSubstitution:
as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is.na, is, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
[., [<-., $, $<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save
Author(s)
Botond Sipos, Gregory Jordan

References

See Also
GTR UNREST GeneralSubstitution TN93

Examples
```r
# create substitution process object
p<-HKY(rate.params=list("Alpha"=10,"Beta"=2),
base.freqs=c(4,3,2,1)/10
)
  # get a summary
  summary(p)
  # display a bubble plot
  plot(p)

# The following code demonstrates how to use
# the process in a simulation.

# create a sequence, attach process p
s<-NucleotideSequence(length=20,processes=list(list(p)))
# sample states
sampleStates(s)
# make the first five positions invariable
setRateMultipliers(s,p,0,1:5)
# get rate multipliers
getRateMultipliers(s,p)
# create a simulation object
sim<->PhyloSim(root.seq=s,phylo=rcal(2))
# run simulation
Simulate(sim)
# print alignment
sim$alignment
```

---

**insertSequence.Sequence**

*Insert a Sequence object into another Sequence object after a specified position*

---

**Description**

Insert a Sequence object into another Sequence object after a specified position.
**Usage**

```r
## S3 method for class 'Sequence'
insertSequence(this, insert, position, process=NA, paranoid=FALSE, ...)
```

**Arguments**

- `this`: The target `Sequence` object.
- `insert`: The `Sequence` object to be inserted.
- `position`: The position after the `Sequence` object will be inserted.
- `process`: The `Process` object performing the insertion (optional).
- `paranoid`: If `TRUE`, then the consistency of the target objects is checked more rigorously after insertion.
- `...`: Not used.

**Value**

The `Sequence` object (invisible).

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `Sequence`.

**Examples**

```r
# create some sequence objects.
target<-NucleotideSequence(string="AAAAAAAAAAA")
insert<-NucleotideSequence(string="GGGGGGGGGGG")
# insert after position 5
insertSequence(target,insert,5)
# print the target sequence
target
```

---

**intersect.list.PSRoot**  
*Utility method returning the intersection of two lists*

**Description**

Utility method returning the intersection of two lists. Duplicated elements are not considered.
Usage

```r
# S3 method for class 'PSRoot'
intersect.list(this, one, two, ...)
```

Arguments

- **this**: A PSRoot object.
- **one**: A list of objects.
- **two**: A list of objects.
- **...**: Not used.

Value

The list containing the intersection.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `PSRoot`.

Examples

```r
# create some lists
a <- list(1, 2, 3);
b <- c(a, list("a", "b", "c"))
# get the intersection of a and b
PSRoot$intersect.list(a, b)
```

---

**is.Alphabet**

*Check if an object is an instance of the Alphabet class*

Description

Check if an object is an instance of the Alphabet class.

Usage

```r
# Default S3 method:
is.Alphabet(this, ...)
```

Arguments

- **this**: An object.
- **...**: Not used.
is.Codon Alphabet

Value
TRUE or FALSE

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see Alphabet.

Examples

```R
# create an alphabet object
a <- Alphabet()
# create a PSRoot object
o <- PSRoot()
# check if they are alphabet objects
is.Alphabet(a)
is.Alphabet(o)
```

Description
Check if an object inherits from Codon Alphabet.

Usage
```R
## Default S3 method:
is.CodonAlphabet(this, ...)
```

Arguments

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>this</td>
<td>An object.</td>
</tr>
<tr>
<td>...</td>
<td>Not used.</td>
</tr>
</tbody>
</table>

Value
TRUE or FALSE.

Author(s)
Botond Sipos, Gregory Jordan
Examples

    # create some objects
    a<-CodonAlphabet()
    p<-Process()
    # check if they inherit from CodonAlphabet
    is.CodonAlphabet(a)
    is.CodonAlphabet(p)

is.CodonUNREST   Check whether an object inherits from CodonUNREST

Description

Check whether an object inherits from CodonUNREST.

Usage

    ## Default S3 method:
    is.CodonUNREST(this, ...)

Arguments

    this  An object.
    ...  Not used.

Value

    TRUE or FALSE.

Author(s)

    Botond Sipos, Gregory Jordan

Examples

    # create some objects
    p<-CodonUNREST()
    pp<-GTR()
    # check if they inherit from CodonUNREST
    is.CodonUNREST(p)
    is.CodonUNREST(pp)
is.Event

Check whether an object inherits from the class Event

Description

Check whether an object inherits from the class Event.

Usage

```r
## Default S3 method:
is.Event(this, ...)
```

Arguments

- **this**: An object.
- **...**: Not used.

Value

TRUE or FALSE.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Event`.

Examples

```r
# create some objects
e <- Event()
a <- Alphabet()
# check if they inherit from Event
is.Event(e)
is.Event(a)
```
is.GeneralDeletor  
Check whether an object inherits from GeneralDeletor

Description
Check whether an object inherits from GeneralDeletor.

Usage
```r
## Default S3 method:
is.GeneralDeletor(this, ...)
```

Arguments
- `this`: An object.
- `...`: Not used.

Value
TRUE or FALSE.

Author(s)
Botond Sipos, Gregory Jordan

Examples
```r
# create some objects
d <- GeneralDeletor()
i <- GeneralInsertor()
# check if they inherit from GeneralDeletor
is.GeneralDeletor(d)
is.GeneralDeletor(i)
```

is.GeneralInDel  
Check if an object inherits from the GeneralInDel class

Description
Check if an object inherits from the GeneralInDel class.

Usage
```r
## Default S3 method:
is.GeneralInDel(this, ...)
```
is.GeneralInsertor

Arguments

   this            An object.
   ...            Not used.

Value

   TRUE or FALSE.

Author(s)

   Botond Sipos, Gregory Jordan

Examples

   # create some objects
   o <- GeneralInDel(rate=1, propose.by=function(process){sample(c(1:10), 1)});
   x <- GTR()
   # check if they inherit from GeneralInDel
   is.GeneralInDel(o)
   is.GeneralInDel(x)

is.GeneralInsertor

Check whether an object inherits from GeneralInsertor

Description

Check whether an object inherits from GeneralInsertor.

Usage

   ## Default S3 method:
   is.GeneralInsertor(this, ...)

Arguments

   this            An object.
   ...            Not used.

Value

   TRUE or FALSE.

Author(s)

   Botond Sipos, Gregory Jordan
is.\textit{GeneralSubstitution}

\textit{Check if an object is an instance of the GeneralSubstitution class}

\section*{Description}

Check if an object is an instance of the GeneralSubstitution class.

\section*{Usage}

\texttt{## Default S3 method:}
\texttt{is.\textit{GeneralSubstitution}(this, \ldots)}

\section*{Arguments}

\begin{itemize}
  \item \texttt{this} \hspace{1cm} An object.
  \item \texttt{\ldots} \hspace{1cm} Not used.
\end{itemize}

\section*{Value}

TRUE or FALSE.

\section*{Author(s)}

Botond Sipos, Gregory Jordan

\section*{See Also}

For more information see \texttt{\textit{GeneralSubstitution}}.

\section*{Examples}

\begin{verbatim}
# create some objects
d <- \textit{GeneralDeletor}()
i <- \textit{GeneralInsertor}()
# check if they inherit from GeneralInsertor
is.\textit{GeneralInsertor}(i)
is.\textit{GeneralInsertor}(d)
\end{verbatim}

\begin{verbatim}

# create some objects
p <- \textit{GeneralSubstitution}()
pp <- \textit{Process}()
# check if they inherit from GeneralSubstitution
is.\textit{GeneralSubstitution}(p)
is.\textit{GeneralSubstitution}(pp)
\end{verbatim}
is.GY94

Check whether an object inherits from GY94

Description
Check whether an object inherits from GY94.

Usage
## Default S3 method:
is.GY94(this, ...)

Arguments
- this: An object.
- ...: Not used.

Value
TRUE or FALSE.

Author(s)
Botond Sipos, Gregory Jordan

Examples
# create some objects
p <- CodonUNREST()
pp <- GY94()
# check if they inherit from CodonUNREST
is.GY94(p)
is.GY94(pp)

is.na.PSRoot

Check if a PSRoot object is NA

Description
Check if a PSRoot object is NA. PSRoot objects cannot be NA, so this method always returns FALSE.

Usage
## S3 method for class 'PSRoot'
is.na(x, ...)


is.phylo

Arguments

x A PSRoot object.

Value

FALSE

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see PSRoot.

Examples

is.na(PSRoot());

Description

Check if an object is an instance of the phylo class. Phylo objects are created by the APE package. This method just return the value of inherits(this,"phylo").

Usage

## Default S3 method:
is.phylo(this, ...)

Arguments

this An object.

Value

TRUE or FALSE.

Author(s)

Botond Sipos, Gregory Jordan
See Also

The ape package.

Examples

# load APE
library(ape);
# create some objects
o1<-Object();
o2<-rcoal(3);
# check if they are phylo objects
is.phylo(o1);
is.phylo(o2);

Description

Check if an object is an instance of the Process class.

Usage

## Default S3 method:
is.Process(this, ...)

Arguments

this An object.

... Not used.

Value

TRUE or FALSE

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Process.
Examples

```r
# create some objects
a <- Site();
p <- Process();
# check if they inherit from Process
is.Process(a)
is.Process(p)
```

---

is.PSRoot  
*Check if an object inherits from PSRoot*

Description

Check if an object inherits from PSRoot.

Usage

```r
## Default S3 method:
is.PSRoot(this, ...)
```

Arguments

- `this`: An object.
- `...`: Not used.

Value

TRUE or FALSE.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `PSRoot`.

Examples

```r
# create some objects
o <- PSSRoot()
a <- Alphabet()
x <- Object()
# check if they inherit form PSRoot
is.PSRoot(o)
is.PSRoot(a)
is.PSRoot(x)
```
is.QMatrix  

Check if an object is an instance of the QMatrix class

Description

Check if an object is an instance of the QMatrix class.

Usage

```r
## Default S3 method:
is.QMatrix(this, ...)
```

Arguments

- `this`: An object.
- `...`: Not used.

Value

TRUE or FALSE.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `QMatrix`.

Examples

```r
# create some objects
m<-QMatrix()
p<-Process()
# check if they inherit from QMatrix
is.GeneralSubstitution(m)
is.GeneralSubstitution(p)
```
is.Sequence  

**Check whether an object inherits from the Sequence class**

Description

Check whether an object inherits from the Sequence class.

Usage

```r
## Default S3 method:
is.Sequence(this, ...)
```

Arguments

- `this`: An object.
- `...`: Not used.

Value

TRUE or FALSE

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Sequence`.

Examples

```r
# create some objects
seq<-Sequence(length=10)
a<-Alphabet()
# check if they inherit from Sequence
is.Sequence(seq)
is.Sequence(a)
```
is.Site  

Check if an object is an instance of the Site class

Description

Check if an object is an instance of the Site class.

Usage

```r
## Default S3 method:
is.Site(this, ...)
```

Arguments

- **this**: An object.
- **...**: Not used.

Value

TRUE or FALSE.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Site`.

Examples

```r
# create an object
s <- Site();
# check whether is a Site object
is.Site(s)
# the same with an Event object
is.Site(Event());
```
is.tip.PhyloSim

Check if a node is a tip

Description

Check if a node is a tip.

Usage

```r
## S3 method for class 'PhyloSim'
is.tip(this, node=NA, ...)
```

Arguments

- `this`: A PhyloSim object.
- `node`: A node identifier (integer vector of length one).
- `...`: Not used.

Value

TRUE or FALSE

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `PhyloSim`.

Examples

```r
# create a PhyloSim object
sim <- PhyloSim(phylo=rcoal(5));
# check if node 4 is a tip
is.tip(sim, 4)
# check if node 6 is a tip
is.tip(sim, 6)
```
isAttached.Site

Check whether a Process object is attached to a Site object

Description

Check whether a Process object is attached to a Site object.

Usage

```r
## S3 method for class 'Site'
isAttached(this, process, ...)
```

Arguments

- `this`: A Site object.
- `process`: A Process object.
- `...`: Not used.

Value

TRUE or FALSE.

Author(s)

Botond Sipos, Gregory Jordan

See Also

Site Process attachProcess detachProcess getProcesses setProcesses

Examples

```r
# create a Site object
s<-Site(alphabet=NucleotideAlphabet())
# create a Process object
p<-JC69()
# check if p is attached to s
isAttached(s,p)
# attach p to s
s$processes<-list(p)
isAttached(s,p)
```
is Empty. Alphabet  

*Check if the symbol set is empty*

**Description**

Check if the symbol set is empty.

**Usage**

```r
## S3 method for class 'Alphabet'
isEmpty(this, ...)
```

**Arguments**

- `this`: An Alphabet object.
- `...`: Not used.

**Value**

TRUE or FALSE

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see *Alphabet*.

**Examples**

```r
# create an empty alphabet
a <- Alphabet()
# check whether it is empty
isEmpty(a)
# specify a new symbol set
a$symbols <- c(0, 1)
isEmpty(a)
```
isStartCodon.CodonAlphabet

Check if a codon is a start codon

Description

Check if a codon is a start codon.

Usage

## S3 method for class 'CodonAlphabet'
isStartCodon(this, codon, ...)

Arguments

- **this**: A CodonAlphabet object.
- **codon**: The codon to be checked.
- **...**: Not used.

Value

TRUE or FALSE.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see CodonAlphabet.

Examples

```r
# create a CodonAlphabet object
a <- CodonAlphabet()
# check some codons
isStartCodon(a,"ATG")
isStartCodon(a,"TGA")
```
isStopCodon.CodonAlphabet

Check if a codon is a stop codon

Description

Check if a codon is a stop codon.

Usage

```r
## S3 method for class 'CodonAlphabet'
isStopCodon(this, codon, ...)
```

Arguments

- `this`: A CodonAlphabet object.
- `codon`: The codon to be checked.
- `...`: Not used.

Value

TRUE or FALSE.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `CodonAlphabet`.

Examples

```r
# create a CodonAlphabet object
a<-CodonAlphabet()
# check some codons
isStopCodon(a,"ATG")
isStopCodon(a,"TGA")
```
The JC69 class

Description

This class implements Jukes-Cantor nucleotide substitution model.

Package:

Class JC69

Object

|--

|--PSRoot

|+-Process

|-------------GeneralSubstitution

|------------------UNREST

|-------------------JC69

Directly known subclasses:

public static class JC69

extends UNREST

Usage

JC69(name="Anonymous", ...)

Arguments

name Object name.

... Additional arguments.

Fields and Methods

Methods:

checkConsistency -
summary -
Methods inherited from UNREST:
checkConsistency, summary

Methods inherited from GeneralSubstitution:
as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is.na, is, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
[, ][<-, [[<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)
Botond Sipos, Gregory Jordan

References

See Also
GeneralSubstitution UNREST GTR

Examples

```r
p<-JC69()
# get a summary
summary(p)
# display a bubble plot
plot(p)

# The following code demonstrates how to use
# the process in a simulation.

# create a sequence, attach process p
s<-NucleotideSequence(length=20,processes=list(list(p)))
# sample states
sampleStates(s)
# make the first five positions invariable
```
setRateMultipliers(s,p,0,1:5)  # get rate multipliers
getRateMultipliers(s,p)  # create a simulation object
sim<-PhyloSim(root.seq=s,phylo=rcoral(2))  # run simulation
Simulate(sim)  # print alignment
sim$alignment

---

**The JTT empirical amino acid substitution model**

**Description**

Package: 
Class JTT

Object
~~
~~+-PSRoot
~~~~~~~
~~~~~~~~+-Process
~~~~~~~~~~
~~~~~~~~~~~~+-GeneralSubstitution
~~~~~~~~~~~~~~
~~~~~~~~~~~~~~+-AminoAcidSubst
~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~+-JTT

**Directly known subclasses:**

public static class **JTT**
extends **AminoAcidSubst**

**Usage**

JTT(equ.dist=NA, ...)

**Arguments**

equ.dist　　Equilibrium distribution.
...　　Not used.
Fields and Methods

Methods:
No methods defined.

Methods inherited from AminoAcidSubst:
buildFromPAML, checkConsistency, newAAMatrix, setEquDist, summary

Methods inherited from GeneralSubstitution:
as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is.na, is, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
[, [, $<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)
Botond Sipos, Gregory Jordan

References

See Also
AminoAcidSubst GeneralSubstitution UNREST

Examples

# create substitution model object
p<-JTT()
# get object summary
summary(p)
# display a bubble plot
plot(p)

# The following code demonstrates how to use
# the process in a simulation.
# create a sequence, attach process p
s<-AminoacidSequence(length=10,processes=list(list(p)))
# sample states
sampleStates(s)
  # make the first three positions invariable
  setRateMultipliers(s,p,0,1:3)
  # get rate multipliers
  getRateMultipliers(s,p)
  # create a simulation object
  sim<-PhyloSim(root.seq=s,phylo=rcoal(2))
  # run simulation
  Simulate(sim)
  # print alignment
  sim$alignment

---

**JTT.dcmut**

*The JTT.dcmut empirical amino acid substitution model*

---

**Description**

Package:  
Class *JTT.dcmut*

**Object**

```
|--PSRoot
|   |--Process
|       |--GeneralSubstitution
|              |--AminoAcidSubst
|                  |--JTT.dcmut
```

**Directly known subclasses:**

```
public static class JTT.dcmut
extends AminoAcidSubst
```

**Usage**

```
JTT.dcmut(equ.dist=NA, ...)
```
Arguments

- `equ.dist` Equilibrium distribution.
- ... Not used.

Fields and Methods

**Methods:**

*No methods defined.*

**Methods inherited from AminoAcidSubst:**

- `buildFromPAML`, `checkConsistency`, `newAAMatrix`, `setEquDist`, `summary`

**Methods inherited from GeneralSubstitution:**

- `as.character`, `checkConsistency`, `clone`, `getAlphabet`, `getEquDist`, `getEventRate`, `getEventRateAtSite`, `getEventsAtSite`, `getQMatrix`, `getRate`, `getRateList`, `hasUndefinedRate`, `is`, `plot`, `rescaleQMatrix`, `sampleState`, `setAlphabet`, `setEquDist`, `setQMatrix`, `setRate`, `setRateList`, `summary`

**Methods inherited from Process:**

- `!=`, `===`, `as.character`, `checkConsistency`, `clone`, `getAlphabet`, `getEventsAtSite`, `getId`, `getName`, `getParameterAtSite`, `getSiteSpecificParamIds`, `getSiteSpecificParamList`, `getWriteProtected`, `hasSiteSpecificParameter`, `hasUndefinedRate`, `is`, `setAlphabet`, `setId`, `setName`, `setParameterAtSite`, `setSiteSpecificParamIds`, `setSiteSpecificParamList`, `setWriteProtected`, `summary`

**Methods inherited from PSRoot:**

- `checkConsistency`, `enableVirtual`, `getComments`, `getMethodsList`, `globalConsistencyCheck`, `intersect.list`, `is.na`, `is`, `ll`, `my.all.equal`, `plot`, `setComments`, `setMethodsList`, `summary`, `virtualAssignmentForbidden`

**Methods inherited from Object:**

- `[`, `[<-`, `S`. `<-`, `as.character`, `attach`, `attachLocally`, `clearCache`, `clearLookupCache`, `clone`, `detach`, `equals`, `extend`, `finalize`, `gc`, `getEnvironment`, `getFieldModifier`, `getFieldModifiers`, `getFields`, `getInstanceiationTime`, `getStaticInstance`, `hasField`, `hashCode`, `ll`, `load`, `objectSize`, `print`, `registerFinalizer`, `save`

**Author(s)**

Botond Sipos, Gregory Jordan

**References**


**See Also**

- `AminoAcidSubst`
- `GeneralSubstitution`
- `UNREST`
Examples

```r
# create substitution model object
p <- JTT.dcmut()
# get object summary
summary(p)
# display a bubble plot
plot(p)

# The following code demonstrates how to use
# the process in a simulation.

# create a sequence, attach process p
s <- AminoAcidSequence(length = 10, processes = list(list(p)))
# sample states
sampleStates(s)
  # make the first three positions invariable
setRateMultipliers(s, p, 0, 1:3)
# get rate multipliers
getRateMultipliers(s, p)
# create a simulation object
sim <- PhyloSim(root.seq = s, phylo = rcoal(2))
# run simulation
Simulate(sim)
# print alignment
sim$alignment
```

---

**K80**

*The K80 class*

**Description**

This class implements the K80 (Kimura 2-parameter) GTR-submodel. The rate parameters are the following: "Alpha", "Beta". Package:

Class K80

Object

```
> k80 <- PSRoot
> > Process
> >> GeneralSubstitution
> >>> UNREST
> >>>>> GTR
```
Directly known subclasses:

public static class **K80**
extends GTR

Usage

**K80**(name="Anonymous", rate.params=list(Alpha = 1, Beta = 1), ...)

Arguments

- **name**: Object name.
- **rate.params**: Rate parameters.
- **...**: Not used.

Fields and Methods

Methods:

```
checkConsistency
getBaseFreqs
getRateParam
getRateParamList
setBaseFreqs
setRateParam
setRateParamList
summary
```

Methods inherited from GTR:
checkConsistency, getBaseFreqs, getRateParam, getRateParamList, setBaseFreqs, setRateParam, setRateParamList, summary

Methods inherited from UNREST:
checkConsistency, summary

Methods inherited from GeneralSubstitution:
as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSite-
SpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersectList, is.na, is, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
[, , [<-, $, $<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldType, getMethodModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)
Botond Sipos, Gregory Jordan

References

See Also
GTR UNREST GeneralSubstitution TN93

Examples

```r
# create substitution process object
p <- K80(rate.params = list("Alpha" = 6, "Beta" = 2),
        base.freqs = c(4, 3, 2, 1) / 10)

# get a summary
summary(p)
# display a bubble plot
plot(p)

# The following code demonstrates how to use
# the process in a simulation.

# create a sequence, attach process p
s <- NucleotideSequence(length = 20, processes = list(list(p)))
# sample states
sampleStates(s)
# make the first five positions invariant
setRateMultipliers(s, p, 0, 1:5)
# get rate multipliers
gRateMultipliers(s, p)
# create a simulation object
sim <- PhyloSim(root = s, phylo = rcoal(2))
# run simulation
Simulate(sim)
```
The K81 class

Description

This class implements the K81 (Kimura 3-parameter) GTR-submodel. The rate parameters are the following: "Alpha", "Beta", "Gamma". Package: Class K81

Object
  ~|  
  ~---PSRoot
  ~~~~~~|  
  ~~~~~~~~~Process
  ~~~~~~~~|  
  ~~~~~~~~~~~GeneralSubstitution
  ~~~~~~~~~~~|  
  ~~~~~~~~~~~~UNREST
  ~~~~~~~~~~~~~|
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~GTR
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|  
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~K81

Directly known subclasses:

public static class K81
  extends GTR

Usage

K81(name="Anonymous", rate.params=list(Alpha = 1, Beta = 1, Gamma = 1), ...)

Arguments

name  Object name.
rate.params  Rate parameters.
...  Not used.
Fields and Methods

Methods:
Methods inherited from GTR:
checkConsistency, getBaseFreqs, getRateParam, getRateParamList, setBaseFreqs, setRateParam, setRateParamList, summary

Methods inherited from UNREST:
checkConsistency, summary

Methods inherited from GeneralSubstitution:
as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is.na, is, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
[, $<-$, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)
Botond Sipos, Gregory Jordan

References

See Also
GTR UNREST GeneralSubstitution HKY
Examples

```r
# create substitution process object
p<-K81(rate.params=list(  "Alpha"=10,"Beta"=2,"Gamma"=5))
  # get a summary
  summary(p)
  # display a bubble plot
  plot(p)

  # The following code demonstrates how to use
  # the process in a simulation.

  # create a sequence, attach process p
  s<-NucleotideSequence(length=20,processes=list(list(p)))
  # sample states
  sampleStates(s)
  # make the first five positions invariable
  setRateMultipliers(s,p,0,1:5)
  # get rate multipliers
  getRateMultipliers(s,p)
  # create a simulation object
  sim<PhyloSim(root.seq=s,phylo=rcoal(2))
  # run simulation
  Simulate(sim)
  # print alignment
  sim$alignment
```

---

**LG**  
*The LG empirical amino acid substitution model*

---

Description

Package:

Class LG

```
Object
  ~-PSRoot
  ~~~~~~~~~-Process
  ~~~~~~~~~~~~~~~-GeneralSubstitution
  ~~~~~~~~~~~~~~~~-AminoAcidSubst
  ~~~~~~~~~~~~~~~~~~~~-LG
```
Directly known subclasses:

public static class LG
extends AminoAcidSubst

Usage

LG(equ.dist=NA, ...)

Arguments

equ.dist   Equilibrium distribution.
...  Not used.

Fields and Methods

Methods:
No methods defined.

Methods inherited from AminoAcidSubst:
buildFromPAML, checkConsistency, newAAMatrix, setEquDist, summary

Methods inherited from GeneralSubstitution:
as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is.na, is, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
[, [[]<-, $<$-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)

Botond Sipos, Gregory Jordan

References

See Also

AminoAcidSubst GeneralSubstitution UNREST

Examples

```r
# create substitution model object
p <- LG()
# get object summary
summary(p)
# display a bubble plot
plot(p)

# The following code demonstrates how to use
# the process in a simulation.

# create a sequence, attach process p
s <- AminoAcidSequence(length=10, processes=list(list(p)))
# sample states
sampleStates(s)
    # make the first three positions invariable
    setRateMultipliers(s, p, 0, 1:3)
    # get rate multipliers
    getRateMultipliers(s, p)
    # create a simulation object
    sim <- PhyloSim(root.seq=s, phylo=coal(2))
    # run simulation
    Simulate(sim)
    # print alignment
    sim$alignment
```

11.PSRoot

`Display detailed information about the virtual fields and methods defined for a given object`

Description

Display detailed information about the virtual fields and methods defined for a given object. The method prints the class of the object, all the parent classes, and the virtual fields and methods defined in the immediate class.

This method provides a "quick and minimal" documentation for PhyloSim classes.

Usage

```
## S3 method for class 'PSRoot'
ll(this, quiet=FALSE, ...)
```
Arguments

this A PSRoot object.
quiet Do not print out methods list.
... Not used.

Value

Text.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see PSRoot.

Examples

# create a Site object
s<-Site()
l1(s)
# get information about the Process class
l1(Process())

---

Log.PhyloSim  Save a message in the PhyloSim log file

Description

Save a message in the PhyloSim log file.
The message is written to the log file only if the log level is non-negative. You can use this method for logging in the case you write classes for PhyloSim.

Usage

## S3 method for class 'PhyloSim'
Log(this, message, ...)

Arguments

this A PhyloSim object.
message A character vector of length one.
... Not used.
Value

The message (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see PhylSim.

Examples

```
# create a PhyloSim object,
# with logLevel set to zero
sim<-PhyloSim(log.level=0);
# log a message
Log(sim,"Hiya there!");
# close log connection
close(sim$.log.connection)
# print out the log file
cat(paste(scan(file=sim$LogFile,what=character(),sep="\n"),collapse="\n");cat("\n");
# clean up
unlink(sim$LogFile)
```
Directly known subclasses:

public static class mtArt
extends AminoAcidSubst

Usage

mtArt(equ.dist=NA, ...)

Arguments

equ.dist Equilibrium distribution.
... Not used.

Fields and Methods

Methods:
No methods defined.

Methods inherited from AminoAcidSubst:
buildFromPAML, checkConsistency, newAAMatrix, setEquDist, summary

Methods inherited from GeneralSubstitution:
as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is.na, is, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
[[], [[-], $, $<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)

Botond Sipos, Gregory Jordan

References

See Also
AminoAcidSubst GeneralSubstitution UNREST

Examples

```r
# create substitution model object
p <- mtArt()
# get object summary
summary(p)
# display a bubble plot
plot(p)

    # The following code demonstrates how to use
    # the process in a simulation.

# create a sequence, attach process p
s <- AminoAcidSequence(length=10, processes=list(list(p)))
# sample states
sampleStates(s)
    # make the first three positions invariable
setRateMultipliers(s, p, 0:3)
    # get rate multipliers
getRateMultipliers(s, p)
    # create a simulation object
sim <- PhyloSim(root.seq=s, phylo=rcial(2))
    # run simulation
Simulate(sim)
    # print alignment
sim$alignment
```

---

**mtMam**

The *mtMam* empirical amino acid substitution model

**Description**

Package: mtMam

Class mtMam

- **Object**
  - |PSRoot
  - |Process
  - |GeneralSubstitution
  - |AminoAcidSubst
Directly known subclasses:

public static class mtMam
extends AminoAcidSubst

Usage

mtMam(equ.dist=NA, ...)

Arguments

equ.dist      Equilibrium distribution.
...
            Not used.

Fields and Methods

Methods:
No methods defined.

Methods inherited from AminoAcidSubst:
buildFromPAML, checkConsistency, newAAMatrix, setEquDist, summary

Methods inherited from GeneralSubstitution:
as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is, na, na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
[<-, $<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)

Botond Sipos, Gregory Jordan
References


See Also

AminoAcidSubst GeneralSubstitution UNREST

Examples

```r
# create substitution model object
p <- mtREV24()
# get object summary
summary(p)
# display a bubble plot
plot(p)

  # The following code demonstrates how to use
  # the process in a simulation.

# create a sequence, attach process p
s <- AminoAcidSequence(length=10, processes=list(list(p)))
# sample states
sampleStates(s)
  # make the first three positions invariable
  setRateMultipliers(s, p, 0, 1:3)
  # get rate multipliers
  getRateMultipliers(s, p)
  # create a simulation object
  sim <- PhyloSim(root.seq=s, phylo=coal(2))
  # run simulation
  Simulate(sim)
  # print alignment
  sim$alignment
```

**mtREV24**

*The mtREV24 empirical amino acid substitution model*

Description

Package:

Class mtREV24

Object

```r
~
```

```r
```
public static class mtREV24
extends AminoAcidSubst

Usage

mtREV24(equ.dist=NA, ...)

Arguments

equ.dist Equilibrium distribution.
... Not used.

Fields and Methods

Methods:
No methods defined.

Methods inherited from AminoAcidSubst:
buildFromPAML, checkConsistency, newAAMatrix, setEquDist, summary

Methods inherited from GeneralSubstitution:
as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSMRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is.na, is, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
[, [, <-, $, $<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach,
equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)

Botond Sipos, Gregory Jordan

References


See Also

AminoAcidSubst GeneralSubstitution UNREST

Examples

```r
# create substitution model object
p<-mtREV24()
# get object summary
summary(p)
# display a bubble plot
plot(p)

  # The following code demonstrates how to use
  # the process in a simulation.

  # create a sequence, attach process p
  s<-AminoAcidSequence(length=10,processes=list(list(p)))

  # sample states
  sampleStates(s)
      # make the first three positions invariable
      setRateMultipliers(s,p,0:1:3)
  # get rate multipliers
  getRateMultipliers(s,p)
  # create a simulation object
  sim<-PhyloSim(root.seq=s,phylo=rcal(2))
  # run simulation
  Simulate(sim)
  # print alignment
  sim$alignment
```
The MtZoa empirical amino acid substitution model

Description

Package:

Class MtZoa

Object

~~KMMpsroot

~~~~~~~~KMMprocess

~~~~~~~~~~~~~~KMMgeneralsubstitution

~~~~~~~~~~~~~~~KMMaminoacidsubst

~~~~~~~~~~~~~~~~~KMMmtzoa

Directly known subclasses:

public static class MtZoa

extends AminoAcidSubst

Usage

MtZoa(equ.dist=NA, ...)

Arguments

equ.dist       Equilibrium distribution.

...            Not used.

Fields and Methods

Methods:

No methods defined.

Methods inherited from AminoAcidSubst:

buildFromPAML, checkConsistency, newAAMatrix, setEquDist, summary

Methods inherited from GeneralSubstitution:

as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary
Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is.na, is, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
[,[<-, $, $<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)
Botond Sipos, Gregory Jordan

References

See Also
AminoAcidSubst GeneralSubstitution UNREST

Examples

# create substitution model object
p<-MtZoa()
# get object summary
summary(p)
# display a bubble plot
plot(p)

    # The following code demonstrates how to use
    # the process in a simulation.

# create a sequence, attach process p
s<-AminoAcidSequence(length=10,processes=list(list(p)) )
# sample states
sampleStates(s)
    # make the first three positions invariable
setRateMultipliers(s,p,0,1:3)
    # get rate multipliers
getRateMultipliers(s,p)
    # create a simulation object
sim<-PhyloSim(root.seq=s,phylo=rcol(2))
# run simulation
Simulate(sim)
# print alignment
sim$alignment

my.all.equal.PSRoot  Test if two objects are nearly equal

Description
Test if two objects are nearly equal.
This method simply calls all.equal.default with the tolerance parameter set to .Machine$double.eps ^ 0.5. More useful as a static method.

Usage
## S3 method for class 'PSRoot'
my.all.equal(static, target, current, ...)

Arguments
- **static**: A PSRoot object.
- **target**: R object.
- **current**: Other R object, to be compared with target.
- **...**: Not used.

Value
TRUE or FALSE.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see PSRoot.

Examples
PSRoot$my.all.equal(0.0, 0.0001)
PSRoot$my.all.equal(0.0, 0.000000001)
newAAMatrix.AminoAcidSubst

Undocumented method

Description

Undocumented method.

Usage

```r
## S3 method for class 'AminoAcidSubst'
newAAMatrix(name=NA, paml.file=NA, equ.dist=NA, ...)
```

Arguments

- `name` Object name
- `paml.file` PAML file.
- `equ.dist` Equilibrium distribution.
- `...` Not used.

Value

A process object inheriting from AminoAcidSubst.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `AminoAcidSubst`.

---

newMatrix.CodonSubst  New codon substitution matrix from PAML file

Description

New codon substitution matrix from PAML file.

Usage

```r
## S3 method for class 'CodonSubst'
newMatrix(name=NA, paml.file=NA, equ.dist=NA, ...)
```
NucleotideAlphabet

Arguments

name Object name
paml.file PAML file.
equ.dist Equilibrium distribution.
... Not used.

Value

A process object inheriting from CodonSubst.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see CodonSubst.

NucleotideAlphabet  The NucleotideAlphabet class

Description

Class of Alphabet objects with the c("T","C","A","G") symbol set, representing nucleotides.

Package: Class NucleotideAlphabet

Object

```
| ~
| ~~~~|PSRoot
| ~~~~~~~~~|alphabet
| ~~~~~~~~~|NucleotideAlphabet
```

Directly known subclasses:

public static class NucleotideAlphabet
extends Alphabet

Usage

NucleotideAlphabet(...)
NucleotideSequence

Arguments

... Not used.

Fields and Methods

Methods:
No methods defined.

Methods inherited from Alphabet:
!=, ==, as.character, checkConsistency, getSize, getSymbolLength, getSymbols, getType, getWriteProtected, hasSymbols, is, isEmpty, setSize, setSymbolLength, setSymbols, setType, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, na, is, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
[, [[<-, $, $<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)
Botond Sipos, Gregory Jordan

See Also
Alphabet

Examples

# create a nucleotide alphabet
b <- NucleotideAlphabet()
# get alphabet summary
summary(b)

The NucleotideSequence class

Description

Sequence objects aggregating Site objects having a NucleotideAlphabet attached by default.

Package:
Class NucleotideSequence

Object

~|
Directly known subclasses:

```java
class NucleotideSequence
    extends Sequence
```

**Usage**

```java
NucleotideSequence(name=NA, string=NA, length=NA, processes=NA, ancestral.obj=NA, ...)
```

**Arguments**

- `name` Name of the Sequence object.
- `string` A string specifying the length and the states of the Sequence object.
- `length` The length of the sequence. Mutually exclusive with "string".
- `processes` A list of lists of Process objects, to be attached to the aggregated Site objects. Recycled if shorter than the length of the sequence.
- `ancestral.obj` The ancestral object of the Sequence object (a valid Sequence or Process object).
- `...` Not used.

**Fields and Methods**

**Methods:**

```java
revComp
```

**Methods inherited from Sequence:**

as.character, attachProcess, checkConsistency, clearStates, clone, copySubSequence, deleteSubSequence, detachProcess, getAlphabets, getAncestral, getBigRate, getCumulativeRates, getCumulativeRatesFromRange, getDeletionTolerance, getEvents, getId, getInsertionTolerance, getLength, getName, getOmegas, getParameterAtSites, getProcesses, getRateMultipliers, getStates, getStrings, getSymbolFreqs, getTotalRates, getUniqueAlphabets, getUniqueProcesses, getWriteProtected, insertSequence, is, plot, plotParametersAtSites, plusGamma, plusInvGamma, sampleStates, setAlphabets, setAncestral, setBigRate, setCumulativeRates, setDeletionTolerance, setName, setInsertionTolerance, setLength, setOmegas, setParameterAtSites, setProcesses, setRateMultipliers, setStates, setStrings, setTotalRates, setUniqueAlphabets, setUniqueProcesses, setWriteProtected, summary

**Methods inherited from PSRoot:**
omegaHist.CodonSequence

checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is.na, is, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
[1] [, [, $, $<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstanceationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)
Botond Sipos, Gregory Jordan

See Also
Sequence NucleotideAlphabet

Examples

# create an empty NucleotideSequence object
s <- NucleotideSequence(length=50)
s
# set states
s$states <- c("A","A","G","T")
s
# create a sequence object by specifying a string
s <- NucleotideSequence(string="ATGCCGATTAGAAA")
s

omegaHist.CodonSequence

Plot a histogram of omega values from a range

Description
Plot a histogram of omega values from a range.

Usage

## S3 method for class 'CodonSequence'
omegaHist(this, process, breaks, index, ...)

Arguments

  this         A CodonSequence object.
  process      A process object inheriting from GY94.
  breaks       breaks parameter for hist().
  index        A vector of positions.
  ...          Not used.
Value

The CodonSequence object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see CodonSequence.

Examples

```r
# create a GY94 process
p<-GY94()
# create a CodonSequence object,
# attach a process p
s<-CodonSequence(length=20,processes=list(list(p)))
  # set omega values through omegaVarM0.CodonSequence
  omegaVarM0(s,p0=0.5,p1=0.2,omega=1.5)
  # get a histogram of omega values from the range 1:15
  omegaHist(s,p,breaks=10,1:15)
```

omegaVarM0.CodonSequence

*The M0 (one-ratio) model of variable omega ratios among sites*

Description

The M0 (one-ratio) model of variable omega ratios among sites.

This method sets the omega site-process specific parameter in the specified range to values sampled from the M0 (one-ratio) model of variable omega ratios among sites.

Distribution of omega values:

<table>
<thead>
<tr>
<th>CATEGORY</th>
<th>PROBABILITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>omega 1</td>
<td></td>
</tr>
</tbody>
</table>

Usage

```r
## S3 method for class 'CodonSequence'
omegaVarM0(this, process, omega, index, ...)
```
Arguments

this A CodonSequence object.
process A process object inheriting from GY94.
omega The fixed omega value.
index A vector of positions.
... Not used.

Value

Invisible TRUE.

Author(s)

Botond Sipos, Gregory Jordan

References


See Also

For more information see CodonSequence.

Examples

# create a GY94 object
p<-GY94(kappa=2)
# create a CodonSequence object, attach process p
s<-CodonSequence(length=20, processes=list(list(p)))
# sample states
sampleStates(s)
# sample omegas in range 1:5 from model M0
omegaVarM0(s,p,omega=2,1:5)
# get omega values
getOmegas(s,p)
# get a histogram of omega values in range 1:5
omegaHist(s,p,breaks=50,1:5)
omegaVarM1.CodonSequence

The M1 (neutral) model of variable omega ratios among sites

Description

The M1 (neutral) model of variable omega ratios among sites.
This method sets the omega site-process specific parameter in the specified range to values sampled from the M1 (neutral) model of variable omega ratios among sites.

Distribution of omega values:

<table>
<thead>
<tr>
<th>CATEGORY</th>
<th>PROBABILITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>omega_0</td>
<td>p0</td>
</tr>
<tr>
<td>omega_1</td>
<td>1 - p0</td>
</tr>
</tbody>
</table>

Usage

```r
## S3 method for class 'CodonSequence'
omegaVarM1(this, process, p0, index, ...)
```

Arguments

- **this**: A CodonSequence object.
- **process**: A process object inheriting from GY94.
- **p0**: See above.
- **index**: A vector of positions.
- ... Not used.

Value

Invisible TRUE.

Author(s)

Botond Sipos, Gregory Jordan

References


omegaVarM2.CodonSequence

See Also

For more information see CodonSequence.

Examples

```r
# create a GY94 object
p <- GY94(kappa=2)
# create a CodonSequence object, attach process p
s <- CodonSequence(length=25, processes=list(list(p)))
# sample states
sampleStates(s)
# sample omegas in range 1:20 from model M1
omegaVarM1(s, p, p0=0.5, 1:20)
# get omega values
getOmeegas(s, p)
# get a histogram of omega values in range 1:20
omegaHist(s, p, breaks=50, 1:20)
```

omegaVarM2.CodonSequence

The M2 (selection) model of variable omega ratios among sites

Description

The M2 (selection) model of variable omega ratios among sites.

This method sets the omega site-process specific parameter in the specified range to values sampled from the M2 (selection) model of variable omega ratios among sites.

Distribution of omega values:

<table>
<thead>
<tr>
<th>CATEGORY</th>
<th>PROBABILITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>omega_0</td>
<td>0 p0</td>
</tr>
<tr>
<td>omega_1</td>
<td>1 p1</td>
</tr>
<tr>
<td>omega_2</td>
<td>1-p0-p1</td>
</tr>
</tbody>
</table>

Usage

```r
## S3 method for class 'CodonSequence'
omegaVarM2(this, process, p0, p1, omega_2, index, ...)
```

Arguments

- **this**: A CodonSequence object.
- **process**: A process object inheriting from GY94.
- **p0**: See above.
omegaVarM2.CodonSequence

p1 See above.
omega_2 See above.
index A vector of positions.
... Not used.

Value

Invisible TRUE.

Author(s)

Botond Sipos, Gregory Jordan

References


See Also

For more information see CodonSequence.

Examples

# create a GY94 object
p<-GY94(kappa=2)
# create a CodonSequence object, attach process p
s<-CodonSequence(length=25, processes=list(list(p)))
# sample states
sampleStates(s)
# sample omegas in range 1:20 from model M2
omegaVarM2(s,p,p0=0.2,p1=0.3,omega_2=4,1:20)
# get omega values
getOmegas(s,p)
# get a histogram of omega values in range 1:20
omegaHist(s,p,breaks=50,1:20)
The M3 (discrete) model of variable omega ratios among sites.

This method sets the omega site-process specific parameter in the specified range to values sampled from the M3 (discrete) model of variable omega ratios among sites.

Distribution of omega values:

<table>
<thead>
<tr>
<th>CATEGORY</th>
<th>PROBABILITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>omega_0</td>
<td>p0</td>
</tr>
<tr>
<td>omega_1</td>
<td>p1</td>
</tr>
<tr>
<td>omega_2</td>
<td>p2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>omega_k</td>
<td>pk</td>
</tr>
</tbody>
</table>

Usage

```r
## S3 method for class 'CodonSequence'
omegaVarM3(this, process, omegas, probs, index, ...)
```

Arguments

- **this**: A CodonSequence object.
- **process**: A process object inheriting from GY94.
- **omegas**: A vector of omega values (omega_0 ... omega_k).
- **probs**: A vector of probabilities (p0 ... pk).
- **index**: A vector of positions.
- **...**: Not used.

Value

Invisible TRUE.

Author(s)

Botond Sipos, Gregory Jordan
omegaVarM4.CodonSequence

References


See Also

For more information see CodonSequence.

Examples

# create a GY94 object
p<-GY94(kappa=2)
# create a CodonSequence object, attach process p
s<-CodonSequence(length=25, processes=list(list(p)))
# sample states
sampleStates(s)
# sample omegas in range 1:20 from model M3
omegaVarM4(s,p,omegas=c(0,2,4),probs=c(1/3,1/3,1/3),1:20)
# get omega values
getOmegas(s,p)
# get a histogram of omega values in range 1:20
omegaHist(s,p,breaks=50,1:20)

_________________________________________________________________________

omegaVarM4.CodonSequence

The M4 (freqs) model of variable omega ratios among sites

_________________________________________________________________________

Description

The M4 (freqs) model of variable omega ratios among sites.

This method sets the omega site-process specific parameter in the specified range to values sampled from the M4 (freqs) model of variable omega ratios among sites.

Distribution of omega values:

<table>
<thead>
<tr>
<th>CATEGORY</th>
<th>PROBABILITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>omega_0 = 0</td>
<td>p0</td>
</tr>
<tr>
<td>omega_1 = 1/3</td>
<td>p1</td>
</tr>
<tr>
<td>omega_2 = 2/3</td>
<td>p2</td>
</tr>
<tr>
<td>omega_3 = 1</td>
<td>p3</td>
</tr>
<tr>
<td>omega_4 = 3</td>
<td>p4</td>
</tr>
</tbody>
</table>
Usage

## S3 method for class 'CodonSequence'
omegaVarM4(this, process, probs, index, ...)

Arguments

- **this**: A CodonSequence object.
- **process**: A process object inheriting from GY94.
- **probs**: A vector of probabilities (p0 ... p4).
- **index**: A vector of positions.
- **...**: Not used.

Value

Invisible TRUE.

Author(s)

Botond Sipos, Gregory Jordan

References


See Also

For more information see CodonSequence.

Examples

```r
# create a GY94 object
p<-GY94(kappa=2)
# create a CodonSequence object, attach process p
s<-CodonSequence(length=25, processes=list(list(p)))
# sample states
sampleStates(s)
# sample omegas in range 1:20 from model M4
omegaVarM4(s,p,probs=c(2/5,1/5,1/5,1/10,1/10),1:20)
# get omega values
getOmeegas(s,p)
# get a histogram of omega values in range 1:20
omegaHist(s,p,breaks=50,1:20)
```
The PAM empirical amino acid substitution model

Description

Package:
Class PAM

Object
~~|  
~~+--PSRoot  
~~~~~~~~|  
~~~~~~~~+--Process  
~~~~~~~~~~|  
~~~~~~~~~~+--GeneralSubstitution  
~~~~~~~~~~~~|  
~~~~~~~~~~~~+--AminoAcidSubst  
~~~~~~~~~~~~~~|  
~~~~~~~~~~~~~~+--PAM

Directly known subclasses:

public static class PAM
extends AminoAcidSubst

Usage

PAM(equ.dist=NA, ...)

Arguments

equ.dist Equilibrium distribution.
... Not used.

Fields and Methods

Methods:
No methods defined.

Methods inherited from AminoAcidSubst:
builtFromPAML, checkConsistency, newAAMatrix, setEquDist, summary

Methods inherited from GeneralSubstitution:
as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary
Methods inherited from Process:
!=, ==, as.character, check Consistency, clone, get Alphabet, get Events At Site, getId, getName, get Parameter At Site, get Site Specific Param Ids, get Site Specific Param List, get Write Protected, has Site Specific Parameter, has Undefined Rate, is, set Alphabet, setId, setName, set Parameter At Site, set Site Specific Param Ids, set Site Specific Param List, set Write Protected, summary

Methods inherited from PSRoot:
check Consistency, enable Virtual, get Methods List, global Consistency Check, intersect list, is.na, is, ll, my.all.equal, plot, set Comments, set Methods List, summary, virtual Assignment Forbidden

Methods inherited from Object:
[[], [<-, $<-, as.character, attach, attach Locally, clear Cache, clear Lookup Cache, clone, detach, equals, extend, finalize, gc, get Environment, get Field Modifier, get Field Modifiers, get Fields, get Instantiation Time, get Static Instance, has Field, hash Code, ll, load, object Size, print, register Finalizer, save

Author(s)
Botond Sipos, Gregory Jordan

References

See Also
Amino Acid Subst General Substitution UNREST

Examples

# create substitution model object
p<-PAM()
# get object summary
summary(p)
# display a bubble plot
plot(p)

    # The following code demonstrates how to use
    # the process in a simulation.

# create a sequence, attach process p
s<-Amino Acid Sequence(length=10,processes=list(list(p)))
# sample states
sample States(s)
    # make the first three positions invariable
set Rate Multipliers(s,p,0,1:3)
    # get rate multipliers
get Rate Multipliers(s,p)
    # create a simulation object
sim<-Phylot Sim(root.seq=s,phylo=rc oal(2))
    # run simulation
Simulate(sim)
The PAM.dcmut empirical amino acid substitution model

Description

Package: PAM.dcmut

Class PAM.dcmut

Object
~~|~~|~~=PSRoot
~~~~~|~~~~~~~|~~~~~~~Process
~~~~~~~~~~|~~~~~~~~~~~~~|~~~~~~~~~~~~~GeneralSubstitution
~~~~~~~~~~~~~~|~~~~~~~~~~~~~AminoAcidSubst
~~~~~~~~~~~~~~~~|~~~~~~~~~~~~~~~~PAM.dcmut

Directly known subclasses:

public static class PAM.dcmut
extends AminoAcidSubst

Usage

PAM.dcmut(equ.dist=NA, ...)

Arguments

equ.dist   Equilibrium distribution.
...
   Not used.

Fields and Methods

Methods:
No methods defined.

Methods inherited from AminoAcidSubst:
buildFromPAML, checkConsistency, newAAMatrix, setEquDist, summary
Methods inherited from `GeneralSubstitution`:
as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

Methods inherited from `Process`:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from `PSRoot`:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, na, is, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from `Object`:
[, [[, $, $<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)
Botond Sipos, Gregory Jordan

References

See Also
AminoAcidSubst GeneralSubstitution UNREST

Examples
# create substitution model object
p<-PAM.dcmut()
# get object summary
summary(p)
# display a bubble plot
plot(p)

# The following code demonstrates how to use
# the process in a simulation.

# create a sequence, attach process p
s<-AminoAcidSequence(length=10,processes=list(list(p)) )
# sample states
sampleStates(s)
# make the first three invariable
setRateMultipliers(s,p,0,1:3)
# get rate multipliers
Perform an event

Description

Performing an event means that the modifications described by the Event object are actually made by calling the event handler function as returned by `getHandler` with the Event object as the first argument.

The event won’t be performed if the handler function is invalid, if there is no associated Site object, if the site position is undefined, if the rate is undefined, or if the generator process is invalid.

The handler function will be overwritten after performing an event, so the `Perform` method should be called only once for every Event object.

Usage

```r
## S3 method for class 'Event'
Perform(this, ...)  # Arguments

this An Event object.

... Not used.
```

Value

The value returned by the handler function.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Event`.  

---

```
getRateMultipliers(s,p)  # create a simulation object
simc-PhyloSim(root.seq=s,phylo=rcoal(2))  # run simulation
Simulate(sim)  # print alignment
sim$alignment
```
Examples

# create a sequence and attach a process
s<-NucleotideSequence(string="ATGC",processes=list(list(JC69()))))
# get the first active event from the first site
# only sequence methods set .position,
# so s$sites[[1]]$events[[1]] wouldn't work.
e<-getEvents(s,1)[[1]]
# perform e
Perform(e)
# check the effect of the event on s
s

PhyloSim  The PhyloSim class

Description

PhyloSim is an extensible object-oriented framework for the Monte Carlo simulation of sequence evolution written in 100 percent R. It is built on the top of the R.oo and ape packages and uses Gillespie’s direct method to simulate substitutions, insertions and deletions.

Key features offered by the framework:

- Simulation of the evolution of a set of discrete characters with arbitrary states evolving by a continuous-time Markov process with an arbitrary rate matrix.
- Explicit implementations of the most popular substitution models (for nucleotides, amino acids and codons).
- Simulation under the popular models of among-sites rate variation, like the gamma (+G) and invariants plus gamma (+I+G) models.
- The possibility to simulate with arbitrarily complex patterns of among-sites rate variation by setting the site specific rates according to any R expression.
- Simulation with one or more separate insertion and/or deletion processes acting on the sequences, which sample the insertion/deletion length from an arbitrary discrete distribution or an R expression (so all the probability distributions implemented in R are readily available for this purpose).
- Simulation of the effects of variable functional constraints over the sites by site-process-specific insertion and deletion tolerance parameters, which determine the rejection probability of a proposed insertion/deletion.
- The possibility of having a different set of processes and site-process-specific parameters for every site, which allow for an arbitrary number of partitions in the simulated data.
- Simulation of heterotachy and other cases of non-homogeneous evolution by allowing the user to set "node hook" functions altering the site properties at internal nodes of the phylogeny.
- The possibility to export the counts of various events ("branch statistics") as phylo objects (see exportStatTree.PhyloSim).
General notes:

- The Sequence objects have no "immortal links". The simulation is aborted if the sequence length shrinks to zero. It is up to the user to choose sensible indel rates and sequence lengths to prevent that.

- The sites near the beginning and end of the sequences have less sites proposing insertion and deletion events around the so the insertion and deletion processes have an "edge effect". The user can simulate realistic flanking sequences to alleviate the edge effect in the simulation settings where it may be an issue.

Notes on performance:

- The pure R implementation offers flexibility, but also comes with a slower simulation speed. If the PSIM_FAST object is present in the environment, a "fast & careless" mode is enabled. In this mode most of the error checking is skipped, increasing the speed. It is recommended that simulations are only run in fast mode if you are sure that the simulation settings are free from errors. It is probably a good practice to set up the simulations in normal mode with short sequences and enable fast mode when running the actual simulation with long sequences.

- Please note, that no "branch statistics" are saved in fast mode.

- Logging also has a negative impact on performance, so it’s not a good idea to run large simulations with the logging enabled.

- The time needed to run a simulation depends not only on the number of the sites, but also on the length of the tree.

- Constructing Sequence objects with large number of sites is expensive. Avoid doing that inside a cycle.

- In the case of Sequence objects with a large number of sites (more than 10 000) the amount of available memory can be limiting as well.

The examples below demonstrate only some more common simulation settings, the framework offers much more flexibility. See the package vignette (vignette("PhyloSim", package="phylosim")) and the examples directory (http://github.com/sbotond/phylosim/tree/master/examples/) for additional examples.

Package:
Class PhyloSim

Object
```
| ~ | ~ | ~
| ~ | ~ | ~
```

Directly known subclasses:

public static class PhyloSim extends PSRoot
**Usage**

PhyloSim(phylo=NA, root.seq=NA, name=NA, log.file=NA, log.level=-1, ...)

**Arguments**

- **phylo** A rooted phylo object, constructed by the APE package.
- **root.seq** A valid Sequence object with Process objects attached. Used as the starting sequence during simulation.
- **name** The name of the object (a character vector of length one).
- **log.file** Name of the file used for logging.
- **log.level** An integer specifying the verbosity of logging (see `setLogLevel.PhyloSim`).
- ... Not used.

**Fields and Methods**

**Methods:**

- `as.character` -
- `attachHookToNode` -
- `attachSeqToNode` -
- `checkConsistency` -
- `Debug` -
- `detachHookFromNode` -
- `detachSeqFromNode` -
- `exportStatTree` -
- `getAlignment` -
- `getAlignmentLength` -
- `getBranchEvents` -
- `getEdge` -
- `getEdges` -
- `getId` -
- `getLogFile` -
- `getLogLevel` -
- `getName` -
- `getNedges` -
- `getNodes` -
- `getNtips` -
- `getPhylo` -
- `getRootNode` -
- `getRootSeq` -
- `getSeqFromNode` -
- `getSequences` -
- `getTipLabels` -
- `getTips` -
- `getTreeLength` -
- `is.tip` -
Methods inherited from PSRoot:
- checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, is, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
- [, [<-., $, $<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)
- Botond Sipos, Gregory Jordan

References
See Also

PSRoot Alphabet AminoAcidAlphabet AminoAcidSequence AminoAcidSubst AnyAlphabet BinaryAlphabet BinarySubst BrownianInsertor CodonAlphabet CodonSequence CodonUNREST ContinuousDelelor ContinuousInsertor cpREV DiscreteDelelor DiscreteInsertor Event F81 F84 FastFieldDelelor GeneralDelelor GeneralInDel GeneralInsertor GeneralSubstitution GTR GY94 HKY JC69 JTT JTT.dcmut K80 K81 LG mtArt mtMam mtREV24 MtZoa NucleotideAlphabet NucleotideSequence PAM PAM.dcmut PhyloSim Process QMatrix Sequence Site T92 TN93 UNREST WAG

Examples

```r
set.seed(1)
## The following examples demonstrate
## the typical use of the framework.
## See the package vignette and
## `url("http://github.com/sbotond/phylosim/tree/master/examples/)
##
## The `ll()` method gives information about the methods defined
## in the immediate class of an object.
## Useful when exploring the framework.

ts<-Sequence()
ll(s)
ll(PhyloSim())
ll(GTR())

## Example 1 - A short simulation:
## simulate nucleotide sequences and display
## the resulting alignment matrix.

Simulate(
  PhyloSim(phy=rcrcoal(3),
    root=NucleotideSequence(string="ATGC", proc=list(list(JC69())) )
)
)

# Construct a phylo object for the following
# simulations, scale total tree length to 1:

tmpp<-PhyloSim(phylo=rcrcoal(3))
scaleTree(tmp,1/tmp$treeLength)
tmp$treeLength
t<-tmp$phylo

## Example 3 - simulating rate variation,
## insertions and deletions.
## See the examples/example_3_clean.R file
## in the phylosim GitHub repository.

# construct a GTR process object
gtr<-GTR(
  name="MyGTR",
  rate.params=list(
    "a"=1, "b"=2, "c"=3, 
```


```r

# construct root sequence object
s <- NucleotideSequence(length=40)
# attach process via virtual field
s$processes <- list(list(gtr))
# sample states from the equilibrium distribution of the attached processes
sampleStates(s)
# create among-site rate variation by sampling the "rate.multiplier" site-process-specific parameter
# from a discrete gamma distribution (GTR+G).
plusGamma(s, gtr, shape=0.1)
# make the range 11:20 invariable
setRateMultipliers(s, gtr, 0, 11:20)
# get the rate multipliers for s and gtr
getRateMultipliers(s, gtr)

# proposing lengths in the range 1:3
d <- DiscreteDeleter(
rate=0.1,
name="MyDel",
sizes=c(1:3),
probs=c(3/6, 2/6, 1/6)
)
# get object
summary(d)
# plot deletion length distribution
plot(d)
# attach deletion process d to sequence s
attachProcess(s, d)
  # create a region rejecting all deletions
setDeletionTolerance(s, d, 0, 11:20)

# construct an insertion process object
# proposing lengths in the range 1:3
i <- DiscreteInsertor(
rate=0.1,
name="MyDel",
sizes=c(1:2),
probs=c(1/2, 1/2),
template.seq = NucleotideSequence(length=1, processes=list(list(JC69())))
)
  # states will be sampled from the JC69 equilibrium distribution
# get object
summary(i)
```
# plot insertion length distribution
plot(i)
# attach insertion process i to sequence s
attachProcess(s,i)
# create a region rejecting all insertions
setInsertionTolerance(s,i,0,1:20)

# plot total site rates
plot(s)
# construct simulation object
sim<-PhyloSim(root.seq=s, phylo=t)
# get object summary
summary(sim)
# plot tree
plot(sim)
# run simulation
Simulate(sim)
# get the list of recorded per-branch event counts
getBranchEvents(sim)
# export the number of substitutions as a phylo object
subst<-exportStatTree(sim,"substitution")
# plot the exported phylo object
plot(subst)
# plot tree and alignment
plot(sim)
# save and display alingment
file<-paste("PhyloSim_dummy_fasta_",Sys.getenv(".").fas",sep="")
saveAlignment(sim,file=file,paranoid=TRUE);
# print out the Fasta file
cat(paste(scan(file=file,what=character(),sep="\n"),collapse="\n"));cat("\n");
# delete Fasta file
unlink(file)

# See \url{http://github.com/sbotond/phylosim/tree/master/examples/examples_class.R}
# for the full list of PhyloSim constructor examples.

## See the package vignette and
## the GitHub repository for even more examples.
Usage

```r
## S3 method for class 'ContinuousDeletor'
plot(x, sample.size=NA, ...)
```

Arguments

- `x` A ContinuousDeletor object.
- `sample.size` Number of lengths sampled for the plot.
- `...` Not used.

Value

The process object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `ContinuousDeletor`.

Examples

```r
# create object
o<-ContinuousDeletor(rate=1)
# set length sampling expression via virtual field
o$dist<-expression(rnorm(1,mean=10, sd=4))
# set maxLength
setMaxLength(o, 30)
# plot density
plot(o)
```

---

**plot.ContinuousInsertor**

*Plot the density of proposed lengths*

Description

Plot the density of proposed lengths.

Usage

```r
## S3 method for class 'ContinuousInsertor'
plot(x, sample.size=NA, ...)
```
Arguments

- `x` A ContinuousInsertor object.
- `sample.size` Number of lengths sampled for the plot.
- `...` Not used.

Value

The process object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `ContinuousInsertor`.

Examples

```r
# create object
o <- ContinuousInsertor(rate=1)
# set length sampling expression via virtual field
o$dist <- expression(rnorm(1, mean=10, sd=4))
# set max length
setMaxLength(o, 30)
# plot density
plot(o)
```

plot.DiscreteDeletor  
*Plot the deletion length distribution*

Description

Plot the deletion length distribution.

Usage

```r
## S3 method for class 'DiscreteDeletor'
plot(x, ...)
```

Arguments

- `x` A DiscreteDeletor object.
- `...` Not used.
**Value**

The DiscreteDeleter object (invisible).

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `DiscreteDeleter`.

**Examples**

```r
d <- DiscreteDeleter(
  name = "MyDiscDel",
  sizes = 1:6,
  probs = c(0.25000000, 0.16666667, 0.08333333, 0.08333333, 0.16666667, 0.25000000)
)
# plot the deletion length distribution
plot(d)
```

---

**Description**

Plot the insertion length distribution.

**Usage**

```r
## S3 method for class 'DiscreteInsertor'
plot(x, ...)
```

**Arguments**

- `x` A DiscreteInsertor object.
- `...` Not used.

**Value**

The DiscreteInsertor object (invisible).

**Author(s)**

Botond Sipos, Gregory Jordan
plot.GeneralSubstitution

See Also
For more information see DiscreteInsertor.

Examples

```r
discinsertor <- DiscreteInsertor(
  name = "MyDiscIns",
  sizes = 1:6,
  probs = c(0.2500000, 0.1666667, 0.0833333, 0.0833333, 0.1666667, 0.2500000)
)
# plot the insertion length distribution
plot(discinsertor)
```

Description
Create a bubble plot of the substitution process.

Bubble plots ([http://biowiki.org/BubblePlots](http://biowiki.org/BubblePlots)) visualize the characteristics of the substitution process. The area of the circles is proportional to the rates/probabilities. The plot is not produced if the rate matrix or the equilibrium distribution has undefined elements.

Usage

```r
## S3 method for class 'GeneralSubstitution'
plot(x, scale=1, ...)
```

Arguments

- `x` An object inheriting from GeneralSubstitution.
- `scale` A scale factor affecting the area of the circles.
- `...` Not used.

Value
The process object (invisible).

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see GeneralSubstitution.
Examples

plot(BinarySubst(rate.list=list("0->1"=1,"1->0"=1.5)))
plot(JC69())
# get smaller circles
plot(JC69(),scale=0.5)
plot(F84(base.freqs=c(3/6,1/6,1/6,1/6)))
plot(WAG())

Description

Plot a PhyloSim object.

This method plots the aggregated alignment alongside the tree used for simulation. Various options allow for control over the plot style.

Usage

```r
## S3 method for class 'PhyloSim'
plot(
  x, plot.tree, plot.ancestors, plot.chars, plot.legend,
  plot.labels, aspect.ratio, num.pages, char.text.size,
  axis.text.size, color.scheme, color.branches, tree.xlim,
  aln.xlim, tracks, aln.length.tolerance, plot.nongap.bl, ...
)
```

Arguments

- `x`: A PhyloSim object.
- `plot.tree`: Whether to plot the tree alongside the alignment. TRUE or FALSE; defaults to TRUE.
- `plot.ancestors`: Whether to plot the ancestral sequences. TRUE or FALSE; defaults to TRUE.
- `plot.chars`: Whether to plot the actual text of the characters.
- `plot.legend`: Whether to plot the legend showing the character-to-color mapping.
- `plot.labels`: Whether to plot the sequence labels along the y-axis.
- `aspect.ratio`: (Experimental; when set, this option forces the num.pages value to 1) Constrains the alignment residues to have a certain aspect ratio; values above 1 cause vertically-stretched blocks. FALSE disables aspect ratio control, numerical values set the aspect ratio; defaults to FALSE.
- `num.pages`: Optionally split the alignment over a number of vertically-stacked pages. This is useful for long alignments. 'auto' chooses a sensible number of pages, numerical values specify a number; defaults to 'auto'.
**char.text.size**  
Text size for the aligned characters. This may require tweaking depending on the DPI and output format. Defaults to 'auto'.

**axis.text.size**  
Text size for the sequence labels along the y-axis. This may require tweaking depending on the DPI and output format. Defaults to 'auto'.

**color.scheme**  
Color scheme to use ("auto", "binary", "dna", "protein", "codon", "combined", "combined_codon"). Defaults to 'auto'. When set to 'auto', the function will choose an appropriate coloring scheme based on the alignment content.

**color.branches**  
The event count used to color the branches ("substitutions" by default). See `getBranchEvents.PhyloSim`.

**tree.xlim**  
The x-axis limits of the tree panel.

**aln.xlim**  
The x-axis limits of the alignment panel (in alignment column coordinates).

**tracks**  
Tracks to display above or below the alignment as colored blocks.  
The input format for tracks is a list of data frames with the following possible fields, all of which are optional and can be omitted:

- **pos** - the sequence position (starting with 1) of the feature. Defaults to NULL.
- **score** - the score (between 0 and 1) of the feature. Scores above 1 or below zero will be truncated. Defaults to 1.
- **y_lo** - the lower Y offset (between 0 and 1) of the feature. Defaults to 0. Use a y_lo and y_hi value for each row in the track data frame to create a wiggle plot like effect.
- **y_hi** - the upper Y offset (between 0 and 1) of the feature. Defaults to 1. Use just a y_hi value for each row in the track data frame to create a bar plot like effect.

The fields below are considered unique per track; the values from the first row in the track data frame are used.

- **id** - the display ID for the track. Defaults to 'Track'.
- **layout** - set to 'above' to put the track above the alignment, 'below' for below.
- **height** - the number of alignment rows for the track to span in height. Defaults to 3.
- **color.gradient** - a comma-separated list of colors to interpolate between when coloring the blocks. Examples: 'white,red' 'blue,gray,red' '#FF00FF,#FFFFFF'. Defaults to 'white,black'.
- **color** - a single color to use when coloring the blocks. Mutually exclusive with color.gradient, and if a color.gradient value exists then this value will be ignored. Defaults to black.
- **background** - a color for the background of the track. Defaults to white.

**aln.length.tolerance**  
The desired alignment/sequence length ratio (A/S ratio) to trim the alignment to. The A/S ratio is defined as the ratio between the alignment length and the mean ungapped sequence length, and the alignment trimming procedure will remove blocks of indel-containing columns (in a sensible order) until either (a) the desired indel tolerance is reached, or (b) no more columns can be removed without yielding an empty alignment. A track is added below the alignment to indicate how many indels each resulting alignment column used used to harbor,
and black squares are overlaid onto the alignment where extant sequence data has been trimmed. Defaults to NULL (no trimming); values in the range of 0.9 to 1.3 tend to work well at improving the legibility of very gappy alignments.

plot.nongap.bl If set to TRUE, plots the non-gap branch length (defined as the branch length of the subtree of non-gapped sequences) as a track below the alignment. Defaults to FALSE.

Value
The PhyloSim object (invisible).

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see PhyloSim.

Examples

# Create a PhyloSim object.
# Provide the phylo object
# and the root sequence.
sim<-PhyloSim(
  name="TinySim",
  phylo=rcoal(3),
  root.seq=NucleotideSequence(string="ATGCTAGCTAGG",processes=list(list(JC69()))))

  # plot the aggregated phylo object
  plot(sim)
# run simulation
Simulate(sim)

  # Plot the alignment without the tree or ancestral sequences.
  plot(sim, plot.ancestors=FALSE, plot.tree=FALSE)
  # Force a DNA-based color scheme
  # (default is 'auto' to auto-detect based on the sequence composition)
  plot(sim, color.scheme='dna', plot.legend=TRUE)

plot.Sequence
Plot the total site rates for a collection of Site objects aggregated by a Sequence object.

Description
Plot the total site rates for a collection of Site objects aggregated by a Sequence object.
Usage

### S3 method for class 'Sequence'
plot(x, index=NA, ...)

Arguments

x  
A Sequence object.

index  
An integer vector specifying a set of positions. It is set to 1:seq$length if omitted.

...  
Not used.

Value

Invisible TRUE or FALSE.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Sequence.

Examples

# create a nucleotide sequence with a process attached
s<-NucleotideSequence(string="ATGGCCA",processes=list(list(JC69())))
# plot total rates in range 1:4
plot(s,1:4)
# plot all total rates
plot(s)

plotParametersAtSites.Sequence

Plot the value of a site-process specific parameter for a collection of Site objects aggregated by a Sequence object

Description

Plot the value of a site-process specific parameter for a collection of Site objects aggregated by a Sequence object. The type of the parameter must be numeric. The Process object must be attached to all positions specified in the index vector.

Usage

### S3 method for class 'Sequence'
plotParametersAtSites(this, process, id, index, ...)
plusGamma.Sequence

Sample the rate multiplier parameters of a Process from a Gamma distribution for a collection of Site objects aggregated by a Sequence object.

Description

Sample the rate multiplier parameters of a Process from a Gamma distribution for a collection of Site objects aggregated by a Sequence object.

The gamma distribution is discretized by calculating the means of the portions corresponding to the categories having equal probabilities. If the ncat argument is not numeric, the rates are sampled from the continuous gamma distribution.

Arguments

- **this**: A Sequence object.
- **process**: A Process object.
- **id**: The identifier of the site-process specific parameter.
- **index**: An integer vector specifying a set of positions. It is set to 1:seq$length if omitted.
- **...**: Not used.

Value

Invisible TRUE or FALSE.

Author(s)

Botond Sipos, Gregory Jordan

See Also

Site Process Sequence

Examples

```r
# create a nucleotide sequence with a process attached
p<-JC69()
s<-NucleotideSequence(string="ATGGCCA",processes=list(list(p)))
# plot rate multipliers in range 1:4
plotParametersAtSites(s,p,"rate.multiplier",1:4)
# plot rate multiplier for the full sequence
plotParametersAtSites(s,p,"rate.multiplier")
```
Usage

```r
## S3 method for class 'Sequence'
plusGamma(this, process, shape, index, ncat=4, ...)
```

Arguments

- `this`: A Sequence object.
- `process`: A Process object.
- `shape`: The shape parameter of the gamma distribution.
- `index`: A vector of positions.
- `ncat`: Number of categories in the discretized gamma distribution (4 by default).
- `...`: Not used.

Value

The Sequence object (invisible);

Author(s)

Botond Sipos, Gregory Jordan

References


See Also

For more information see `Sequence`.

Examples

```r
# create a sequence
s<-NucleotideSequence(length=20)
# attach a process
p<-JC69()
attemptProcess(s,p)
# get rate multipliers
getRateMultipliers(s,p) # the default value is 1.0
# sample rate multipliers in range 1:5 from a discrete gamma distribution with shape parameter 0.5
plusGamma(s,p,0.5,1:5)
# get rate multipliers
getRateMultipliers(s,p) # the default value is 1.0
# sample rates from a continuous gamma distribution with shape parameter 0.5
plusGamma(s,p,0.5,ncat="cont")
# get rate multipliers
```
plusInvGamma.Sequence

Sample the rate multiplier parameters of a Process from an I+Gamma distribution for a collection of Site objects aggregated by a Sequence object.

Description

Sample the rate multiplier parameters of a Process from an I+Gamma distribution for a collection of Site objects aggregated by a Sequence object.

The gamma distribution is discretized by calculating the means of the portions corresponding to the categories having equal probabilities. If the ncat argument is not numeric, the rates are sampled from the continuous gamma distribution.

Usage

```r
## S3 method for class 'Sequence'
plusInvGamma(this, process, pinv, shape, index, ncat=4, ...)
```

Arguments

- `this`: A Sequence object.
- `process`: A Process object.
- `pinv`: The proportion of invariant sites.
- `shape`: The shape parameter of the gamma distribution.
- `index`: A vector of positions.
- `ncat`: Number of categories in the discretized gamma distribution (4 by default).
- `...`: Not used.

Value

The Sequence object (invisible);

Author(s)

Botond Sipos, Gregory Jordan

References


print.PSRootSummary

See Also
For more information see Sequence.

Examples
# create a sequence
s <- NucleotideSequence(length=20)
# attach a process
p <- JC69()
attachProcess(s,p)
# get rate multipliers
gRateMultipliers(s,p) # the default value is 1.0
# sample rate multipliers in range 1:5 from I+discrete Gamma
plusInvGamma(s,p, pinv=0.5, shape=0.5, 1:5)
# get rate multipliers
gRateMultipliers(s,p) # the default value is 1.0
# sample rates from an I+continuos Gamma model
plusInvGamma(s,p, pinv=0.5, shape=0.5, 1:5, ncat="cont")
# get rate multipliers
gRateMultipliers(s,p)

print.PSRootSummary | Print out a PSRootSummary object

Description
Print out a PSRootSummary object.

Usage
## S3 method for class 'PSRootSummary'
print(x, ...)

Arguments
x A PSRootSummary object.
... Not used.

Value
The summary object (invisible).

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see PSRootSummary.
print.QMatrix

Print the character representation of a QMatrix object

Description

Print the character representation of a QMatrix object.

Usage

## S3 method for class 'QMatrix'
print(x, ...)

Arguments

x 
A QMatrix object.

... 
Not used.

Value

The character representation of the QMatrix object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see QMatrix.

Examples

# create a QMatrix object
m<-QMatrix()
# print the character representation
print(m)
# the same, but implicitly
m
**Description**

This is the class representing a generic process acting on Site and Sequence objects. Process objects can be attached to Site objects if the associated Alphabet objects match.

The processes can have site-process-specific parameters. The templates for site-process-specific parameters and their default values are stored in the Process objects and copied into the Site object when the process is attached. See the documentation of the Site class for more details.

The rate multiplier parameter (id="rate.multiplier") is present in the Process class and is inherited by all descendant classes.

Package:

**Class Process**

```java
public static class Process extends PSRoot
```

**Usage**

```java
Process(name=NA, alphabet=NA, ...)
```

**Arguments**

- **name**
  - The name of the Process object: a character vector of length one.

- **alphabet**
  - The associated Alphabet object.

- **...**
  - Not used.

**Fields and Methods**

**Methods:**
Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Process.

Examples

# create a Process object
p<-Process(name="MyProc",alphabet=AminoAcidAlphabet())
# check if it's a Process object
is.Process(p)
# check object consistency
checkConsistency(p)
# set process name
p$name="Ppppprocce"
# get process name
p$name
# get unique process identifier
p$id
# get the list of site specific parameters and parameter IDs
p$siteSpecificParamList
proposeLength.GenerallnDel

Propose indel length

**Description**

Propose indel length.

This method simply calls the function returned by the `getProposeBy` method.

**Usage**

```r
## S3 method for class 'GenerallnDel'
proposeLength(this, ...)  
```

**Arguments**

- `this`  
  A `GenerallnDel` object.

- `...`  
  Not used.

**Value**

A numeric vector of length one (the indel length).
Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see GeneralIndel.

Examples

```r
# create a GeneralIndel object
# proposing event lengths in the range 1:10
o <- GeneralIndel(rate = 1, propose.by = function(process) {
  sample(c(1:10), 1)
});
# propose indel length
proposeLength(o)
```

---

**PSRoot**

*The root class for all phylosim objects*

---

Description

The root class for all phylosim objects containing some utility methods. Package: 

Class **PSRoot**

```r
Object
```

```r
~~|
~~++---PSRoot
```

Directly known subclasses:

Alphabet, AminoAcidAlphabet, AminoAcidSequence, AminoAcidSubst, AnyAlphabet, BinaryAlphabet, BinarySequence, BinarySubst, BrownianInsertor, CodonAlphabet, CodonSequence, CodonUNREST, ContinuousDeletor, ContinuousInsertor, cpREV, DiscreteDeletor, DiscreteInsertor, Event, F81, F84, FastFieldDeletor, GeneralDeletor, GeneralIndel, GeneralInsertor, GeneralSubstitution, GTR, GY94, HKY, JC69, JTT, JTT.dcmut, K80, K81, LG, mtArt, mtMam, mtREV24, MtZoa, NucleotideAlphabet, NucleotideSequence, PAM, PAM.dcmut, PhyloSim, Process, QMatrix, Sequence, Site, T92, TN93, UNREST, WAG

public static class **PSRoot**
extends **Object**

Usage

PSRoot(...)

**Arguments**

... Not used.

**Fields and Methods**

**Methods:**

- checkConsistency
- enableVirtual
- getComments
- getMethodsList
- globalConsistencyCheck
- intersect.list
- is.na
- is
- ll
- my.all.equal
- plot
- setComments
- setMethodsList
- summary
- virtualAssignmentForbidden

**Methods inherited from Object:**

[<-, $, $<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

Object

**Examples**

```r
obj <- PSRoot();
ll(obj);
```
The PSRootSummary class

Description

PSRootSummary objects are blessed lists containing summary entries created by summary.~\* methods.

Package:

Class PSRootSummary

```
+--PSRootSummary
```

Directly known subclasses:

```
public class PSRootSummary
```

Usage

```
PSRootSummary(summary=list(), ...)
```

Arguments

summary A list.
...
Not used.

Fields and Methods

Methods:

```
checkConsistency -
print -
```

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see PSRootSummary.
QMatrix

The QMatrix class

Description

This is the class representing rate matrices. The QMatrix objects store two rate matrices: one contains the rates provided by the user (unscaled rate matrix), the other matrix (scaled rate matrix) is rescaled so the expected number of substitutions per unit time is equal to one when the process is at equilibrium. The scaled rate matrices, along with the site-process-specific rate multiplier parameters define the rates of the Event objects generated by the associated GeneralSubstitution objects.

The QMatrix objects have a bidirectional relationship with the GeneralSubstitution object as they store a reference to the associated process objects. QMatrix objects do not have a write-protection field, but they use the one from the associated GeneralSubstitution object.

Usually there is no need to interact with the QMatrix objects directly, so this class is mainly used internally.

Package:

Class QMatrix

Usage

QMatrix(name="Anonymous", alphabet=NA, rate.list=NA, process=NA, ...)

Arguments

name The name of the QMatrix object (a character vector of length one).
alphabet An Alphabet object.
rate.list A list of unscaled event rates and the associated event names. It will be passed to the setRateList.QMatrix method.
process A GeneralSubstitution object to be associated with the QMatrix object.
... Not used.
Fields and Methods

Methods:

as.character -
checkConsistency -
getAlphabet -
getEventRate -
getId -
getMatrix -
getName -
getProcess -
getRate -
getRateList -
getScaledMatrix -
getWriteProtected -
is -
print -
Scale -
setAlphabet -
setId -
setMatrix -
setName -
setProcess -
setRate -
setRateList -
setScaledMatrix -
setWriteProtected -
summary -

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is.na, is, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
[. [<-,. $, $<-,. as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)

Botond Sipos, Gregory Jordan

See Also

GeneralSubstitution Alphabet Process
Examples

```r
# create a QMatrix object by providing an Alphabet object and rates
m <- QMatrix(name = "Susie Q", alphabet = BinaryAlphabet(), rate.list = list("1->0" = 2, "0->1" = 3))
# get object summary
summary(m)

# create a GeneralSubstitution object by
# providing an Alphabet object and the rates
p <- GeneralSubstitution(alphabet = BinaryAlphabet(), rate.list = list("1->0" = 1, "0->1" = 3))
# get the QMatrix object from p
m <- p$QMatrix
# get various object properties
m
is.QMatrix(m)
m$name
m$id
m$alphabet
# get the associated process
m$process
# get the unscaled rate of "0->1"
getRate(m, "0->1")
# get the scaled rate of "0->1"
getEventRate(m, "0->1")
# get the list of unscaled rates
m$rateList
# get unscaled rate matrix
m$matrix
# get scaled rate matrix
m$scaledMatrix
```

readAlignment.PhyloSim

*Read alignment from file*

Description

Read alignment from file.

This method reads an alignment by using the `read.dna` function from the `ape` package and stores in the PhyloSim object. If a tree is already attached to the PhyloSim object, the alignment must at least contain the sequences corresponding to tip nodes (but it may also contain additional ancestral sequences).

Usage

```r
## S3 method for class 'PhyloSim'
readAlignment(this, file, format = "fasta", …)
```
Arguments

this A PhyloSim object.

file A file name specified by either a variable of mode character, or a double-quoted string.

format a character string specifying the format of the DNA sequences. Four choices are possible: "interleaved", "sequential", "clustal", or "fasta", or any unambiguous abbreviation of these.

... Not used.

Value

The PhyloSim object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see PhyloSim.

Examples

# get a safe file name
fname<-paste("PhyloSim_dummy_fas_",Sys.getpid(),sep"")
# write out a fasta alignment
cat("> t\nGTCTTT-CG\n",file=fname);
cat("> t\nG--TC-TCGG\n",file=fname,append=TRUE);
cat("> t\nG--TC-TCGG\n",file=fname,append=TRUE);
cat("> t\nGTC-GTCGG",file=fname,append=TRUE);
# construct a PhyloSim object,
# set the phylo object
sim<-PhyloSim(phylo=rcal(4))
# read the alignment
readAlignment(sim,fname)
# remove alignment file
unlink(fname)
# plot the tree & alignment
plot(sim)
Description

Read tree from file.

This method reads a tree by using the `read.tree` function from the `ape` package and stores it in the `PhyloSim` object. If an alignment is already attached to the `PhyloSim` object, it must contain all sequences corresponding to tip nodes.

Usage

```r
## S3 method for class 'PhyloSim'
readTree(this, file, ...)
```

Arguments

- `this`: A `PhyloSim` object.
- `file`: A file name specified by either a variable of mode character, or a double-quoted string.
- `...`: Not used.

Value

The `PhyloSim` object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `PhyloSim`.

Examples

```r
# get a safe file name
fname <- paste("PhylloSim_dummy_fas_", Sys.getpid(), sep="")
# write out a fasta alignment
cat("(a,b,c);", file=fname);
# construct a PhyloSim object:
sim <- PhyloSim()
# read the alignment
readTree(sim, fname)
# remove alignment file
unlink(fname)
# plot the tree
plot(sim)
```
Rescale the scaled rate matrix of a QMatrix object aggregated by a GeneralSubstitution object

Description
Rescale the scaled rate matrix of a QMatrix object aggregated by a GeneralSubstitution object. The QMatrix objects aggregated by the GeneralSubstitution objects store two rate matrices: one contains the rates provided by the user (unscaled rate matrix), the other matrix (scaled rate matrix) is rescaled to have the expected number of substitutions per unit time equal to one when the process is at equilibrium. This method performs the re-scaling of the scaled rate matrix in the associated QMatrix object based on the equilibrium distribution and the unscaled rate matrix. This method is mainly used internally as the scaled matrix is rescaled every time the unscaled matrix is modified.

Usage
```r
## S3 method for class 'GeneralSubstitution'
rescaleQMatrix(this, ...)
```

Arguments
- `this` A GeneralSubstitution object.
- `...` Not used.

Value
Invisible TRUE.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see `GeneralSubstitution`.

Examples
```r
# create a GeneralSubstitution object
# provide an Alphabet object and the rates
p <- GeneralSubstitution(alphabet=BinaryAlphabet(), rate.list=list("1->0"=1,"0->1"=3))
# rescale rate matrix
rescaleQMatrix(p)
```
**Description**

Reverse complement a NucleotideSequence object.

The method reverse complements the sequence "in place", no object cloning is performed.

**Usage**

```r
## S3 method for class 'NucleotideSequence'
revComp(this, ...)
```

**Arguments**

- `this` A NucleotideSequence object
- `...` Not used.

**Value**

Invisible TRUE.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `NucleotideSequence`.

**Examples**

```r
s <- NucleotideSequence(string = "ATGC")
s
revComp(s)
s
```
sampleState.GeneralSubstitution

Sample a state from the equilibrium distribution of a GeneralSubstitution object

Description

Sample a state from the equilibrium distribution of a GeneralSubstitution object.

Usage

```r
## S3 method for class 'GeneralSubstitution'
sampleState(this, ...)
```

Arguments

- `this` A GeneralSubstitution object.
- `...` Not used.

Value

A character vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `GeneralSubstitution`.

Examples

```r
# create a GeneralSubstitution object
# provide an Alphabet object and the rates
p <- GeneralSubstitution(alphabet=BinaryAlphabet(), rate.list=list("1->0"=1,"0->1"=1))
# get equilibrium distribution
getEquilDist(p)
# get equilibrium distribution via virtual field
p$equilDist
# sample from equilibrium distribution
sampleState(p)
```
Sample the states for a collection of Site objects aggregated by a Sequence object

Description

Sample the states for a collection of Site objects aggregated by a Sequence object. This method samples new states from the equilibrium distribution of the attached process(es) for sites having undefined states (NA). If a site has more than one substitution process attached, then the method samples the new state from the mixture of equilibrium distributions. The weight of each equilibrium distribution is proportional to the site-process specific rate multiplier of the corresponding process at the given site.

Sites having defined states are not touched. All sites with undefined states must have at least one substitution process (object inheriting from GeneralSubstitution) attached.

Usage

```r
## S3 method for class 'Sequence'
sampleStates(this, index, ...)
```

Arguments

- `this`: A Sequence object.
- `index`: An integer vector specifying a set of positions. It is set to 1:seq$length if omitted.
- `...`: Not used.

Value

The Sequence object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

`sampleState.GeneralSubstitution`, `GeneralSubstitution`

Examples

```r
# create a nucleotide sequence
s<-NucleotideSequence(length=80)
# create some processes
jc69<-JC69();  # Jukes-Cantor
hky<-HKY(base.freqs=c(0.1,0.4,0.1,0.4))  # "GC-rich" HKY
# attach the processes
```
saveAlignment.PhyloSim

Save the alignment stored in a PhyloSim object in a Fasta file.

Description

Save the alignment stored in a PhyloSim object in a Fasta file.

Usage

```
## S3 method for class 'PhyloSim'
saveAlignment(this, file="phylosim.fas", skip.internal=FALSE, paranoid=FALSE, ...)
```

Arguments

- `this`: A PhyloSim object.
- `file`: The name of the output file.
- `skip.internal`: Do not save sequences corresponding to internal nodes.
- `paranoid`: Check the consistency of the alignment.
- `...`: Not used.

Value

The PhyloSim object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `PhyloSim`.
Examples

```
# Create a PhyloSim object.
# Provide the phylo object
# and the root sequence.
sim<-PhyloSim(
  name="TinySim",
  phylo=rcoal(3),
  root.seq=NucleotideSequence(string="ATG",processes=list(list(JC69())))
);
# run the simulation
Simulate(sim)
# save the alignment
file<-paste("PhyloSim_dummy_fasta_",Sys.getpid(),".fas",sep="");
saveAlignment(sim,file=file,paranoid=TRUE);
# print out the Fasta file
cat(paste(scan(file=file,what=character(),sep="\n"),collapse="\n"));
cat("\n");
# delete Fasta file
unlink(file);
```

Scale.QMatrix

Scale the scaled rate matrix stored in a QMatrix object by the provided factor.

Description

Scale the scaled rate matrix stored in a QMatrix object by the provided factor.
This method sets the scaled rate matrix to unscaled_matrix * constant.

Usage

```r
## S3 method for class 'QMatrix'
Scale(this, constant=NA, ...)
```

Arguments

- `this` A QMatrix object.
- `constant` The scaling factor (a numeric vector of length one).
- `...` Not used.

Value

The QMatrix object (invisible).

Author(s)

Botond Sipos, Gregory Jordan
See Also

For more information see `QMatrix`.

Examples

```r
# create a QMatrix object
# , provide Alphabet object and rates
m <- QMatrix(name = "Susie Q", alphabet = BinaryAlphabet(), rate.list = list("1->0" = 2, "0->1" = 3))
# get object summary
summary(m)
# perform scaling
Scale(m, 1/0.666)
# get object summary
summary(m)
```

---

scaleTree.PhyloSim  

*Scale the branch lengths of a phylo object aggregated by a PhyloSim object*

Description

Scale the branch lengths of a phylo object aggregated by a PhyloSim object. This method multiples all the edge lengths by the specified factor.

Usage

```r
## S3 method for class 'PhyloSim'
scaleTree(this, factor, ...)
```

Arguments

- `this`  
  A PhyloSim object.

- `factor`  
  A numeric vector of length one.

- `...`  
  Not used.

Value

The PhyloSim object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `PhyloSim`.  

---

scaleTree.PhyloSim  

*Scale the branch lengths of a phylo object aggregated by a PhyloSim object*
**Examples**

```r
# create a PhyloSim object
sim<-PhyloSim(phylo=coal(5));
# get the tree length
sim$treeLength
# scale tree
scaleTree(sim,10)
# get the scaled tree length
sim$treeLength
```

---

**Description**

This is the class representing a sequence. The backbone of the Sequence objects are lists aggregating Site objects. The class has fields for keeping track of cumulative site rates, the sum of all active event rates and methods for performing actions on a collection of sites (positions).

The Sequence objects have a field specifying an ancestral object, which can be a Sequence object (when the object is obtained through clone()) or the "Root insertion process" object (for newly created objects).

Package:

**Class Sequence**

```
Object
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
Arguments

- **name**: The name of the Sequence object.
- **string**: A string containing symbols belonging to the associated Alphabet object. It can be used to set the initial states of the aggregated Site objects. It also specifies the length of the sequence.
- **length**: The length of the sequence. It cannot be used when 'string' is specified.
- **alphabets**: A list of Alphabet objects to be associated with the Site objects. The list is recycled in the case it is shorter than the sequence length.
- **processes**: A list of lists of Process objects to be attached (recycled if shorter than sequence length).
- **ancestral.obj**: The ancestral object (Sequence or Process).

Fields and Methods

Methods:

- `as.character`
- `attachProcess`
- `checkConsistency`
- `clearStates`
- `clone`
- `copySubSequence`
- `deleteSubSequence`
- `detachProcess`
- `getAlphabets`
- `getAncestral`
- `getBigRate`
- `getCumulativeRates`
- `getCumulativeRatesFromRange`
- `getDeletionTolerance`
- `getEvents`
- `getId`
- `getInsertionTolerance`
- `getLength`
- `getName`
- `getOmegas`
- `getParameterAtSites`
- `getProcesses`
- `getRateMultipliers`
- `getSites`
- `getStates`
- `getString`
- `getSymbolFreqs`
- `getTotalRates`
- `getTotalRatesFromRange`
Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is.na, is, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
[, [[<-*, $, $<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldName, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)
Botond Sipos, Gregory Jordan

See Also
Alphabet Site Process Event
Examples

```r
# create a sequence object by
# providing alphabets, processes and states
s.one<-Sequence(
  name="Seq",
  string="AATTCCTTAAGGCCTAA",
  alphabets=list(NucleotideAlphabet()),
  processes=list(list(JC69()))
)
s.one
# check if inherits from Sequence
is.Sequence(s.one)
# get object summary
summary(s.one)
# create a sequence object,
# specifying length, alphabets
# and ancestral object
s<-Sequence(
  name="Seq",
  length=20,
  ancestral.obj=s.one
)
# get sequence string
s$string
# get the list of site objects
s$sites
# get object id
s$id
# set and get name
s$name<-"SeqSeq"
s$seq
# get length
s$length
# get and set ancestral object
s$ancestral
s$ancestral<-Sequence();
# set alphabets
setAlphabets(s,list(NucleotideAlphabet()))
# set states
# "A"-s in the range 1:10
setStates(s,"A",1:10)
# a pattern of "ATGC"-s in the range 11:20
setStates(s,c("A","T","G","C"),11:20)
# get states from range 10:12
getStates(s,10:12)
# attach a JC69 process to range 1:10
jc69<-JC69()
attachProcess(s,jc69,1:10)
# set the rate multiplier site-process specific parameter for jc69
setParameterAtSites(s,jc69,"rate.multiplier",2,1:10)
# get "rate.multiplier" for jc69 from range 1:2
getParameterAtSites(s, jc69, "rate.multiplier",1:2)
```
setAcceptBy.GeneralInDel

Set the function used for accepting/rejecting indel events
Description

Set the function used for accepting/rejecting indel events.

The function object must have the following arguments: process (the caller process), sequence (the target sequence), window (a vector containing the positions affecting acceptance).

Usage

```r
## S3 method for class 'GeneralInDel'
setAcceptBy(this, value, ...)
```

Arguments

- `this` A GeneralInDel object.
- `value` A function object.
- `...` Not used.

Value

The function object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `GeneralInDel`.

Examples

```r
# create a GeneralInDel object
# rejecting half of the events
o<-GeneralInDel(
  rate=1,
  propose.by=function(process){return(5)},
  accept.by=function(){sample(c(TRUE,FALSE),1)}
);
# set/get the acceptBy function
setAcceptBy(o,value=function(){return(FALSE)}) # reject all events
getAcceptBy(o)
# set/get acceptBy function via virtual field
o$acceptBy<-function(){return(TRUE)} # accept all events
o$acceptBy
```
Description

Set the size of the acceptance window

This parameter determines the number of sites neighbouring the position (from left and right) of
the insertion considered when accepting/rejecting a proposed insertion. The "insertion.tolerance"
parameter is retrieved from sites falling in the window specified by this parameter. The default value
is 1, so the two neighbouring sites are considered by default.

Usage

```r
## S3 method for class 'GeneralInsertor'
setAcceptWin(this, value, ...)
```

Arguments

- `this`: A GeneralInsertor object.
- `value`: An integer vector of length one.
- `...`: Not used.

Value

The new value.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `GeneralInsertor`.

Examples

```r
# create a GeneralInsertor object
i <- GeneralInsertor(rate=0.5);
# get acceptance window size
gAcceptWin(i)
# get acceptance window size via virtual field
i$acceptWin
# set acceptance window size
setAcceptWin(i, 2)
# set acceptance window size via virtual field
i$acceptWin <- 3
i$acceptWin
```
Forbidden action: setting the alignment stored in a PhyloSim object.

Usage

```r
## S3 method for class 'PhyloSim'
setAlignment(this, value, ...)
```

Arguments

- `this`: An object.
- `value`: Not used.
- `...`: Not used.

Value

Throws an error.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `PhyloSim`.

Set the Alphabet object aggregated by a GeneralSubstitution object.

Usage

```r
## S3 method for class 'GeneralSubstitution'
setAlphabet(this, value, ...)
```
Arguments

this A GeneralSubstitution object.
value An Alphabet object.
... Not used.

Value

The Alphabet object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `GeneralSubstitution`.

Examples

```r
# create a GeneralSubstitution object with an attached BinaryAlphabet object
p <- GeneralSubstitution(alphabet=BinaryAlphabet())
# get object summary
summary(p)
# get alphabet
getAlphabet(p)
# get alphabet via virtual field
p$alphabet
# set a new alphabet
setAlphabet(p, NucleotideAlphabet())
summary(p)
# set alphabet via virtual field
p$alphabet <- BinaryAlphabet()
p$alphabet
```

---

**setAlphabet.Process**  
**Associate an Alphabet object with a Process object**

Description

Associate an Alphabet object with a Process object.

Usage

```r
## S3 method for class 'Process'
setAlphabet(this, new.alphabet, ...)
```
Arguments

  this          A Process object.
  new.alphabet A valid Alphabet object.
  ...         Not used.

Value

  The Alphabet object (invisible).

Author(s)

  Botond Sipos, Gregory Jordan

See Also

  For more information see Process.

Examples

  # create objects
  a<-AminoAcidAlphabet()
  p<-Process()
  # associate p with Alphabet object a
  setAlphabet(p,a)
  p$alphabet
  # associate p with a new NucleotideAlphabet via virtual field
  p$alphabet<-NucleotideAlphabet()
  p$alphabet

setAlphabet.QMatrix  Set the Alphabet object for a QMatrix object

Description

  Set the Alphabet object for a QMatrix object.
  This method rebuilds the scaled and unscaled rate matrices and so sets all rates to NA.

Usage

  ## S3 method for class 'QMatrix'
  setAlphabet(this, value, ...)

Arguments

  this          A QMatrix object.
  value         An Alphabet object.
  ...           Not used.
setAlphabet.Site

Value

The Alphabet object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see QMatrix.

Examples

```r
# create a QMatrix object
m<-QMatrix()
# set the alphabet
setAlphabet(m,NucleotideAlphabet())
# get the alphabet
getAlphabet(m)
# set alphabet via virtual field
m$alphabet<-BinaryAlphabet()
summary(m)
```

---

**setAlphabet.Site**  Attach an Alphabet object to a Site object

Description

Attach an Alphabet object to a Site object. If the ancestral site is not NA, then the symbol set of the ancestral Alphabet object and the new Alphabet object must be the same. The current state must be in the symbol set of the new Alphabet object, unless it is NA.

Usage

```r
## S3 method for class 'Site'
setAlphabet(this, new.alphabet, ...)
```

Arguments

- **this**  A Site object.
- **new.alphabet**  A valid Alphabet object.
- **...**  Not used.

Value

Returns the new Alphabet object (invisible).
setAlphabets.Sequence

Associate Alphabet objects to a set of Site objects aggregated by a Sequence object.

Description

Associate Alphabet objects to a set of Site objects aggregated by a Sequence object.

Usage

```
## S3 method for class 'Sequence'
setAlphabets(this, value, index, ...)
```

Arguments

- **this**: A Sequence object.
- **value**: A list of Alphabet objects, recycled if shorter than the index vector.
- **index**: An integer vector specifying a set of positions. It is set to 1:seq$length if omitted.
- **...**: Not used.

Examples

```r
# create a site object
s <- Site()
# create an Alphabet object
a <- Alphabet(c("A", "T", "G"))
# attach alphabet to site object
setAlphabet(s, a)
# set site state
s$state <- "A"
# clone the alphabet object
b <- clone(a)
# modify symbol set in b
b$symbols <- c(b$symbols, "C")
# attach b to s via virtual field
s$alphabet <- b
s$alphabet
```
Value

The Sequence object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Sequence.

Examples

```r
# create a Sequence object
s <- Sequence(length=10)
# set the alphabets for range 1:5
setAlphabets(s, list(NucleotideAlphabet(), BinaryAlphabet()), 1:5)
# set the alphabets for range 6:10
setAlphabets(s, list(AminoAcidAlphabet()), 6:10)
# get the list of attached Alphabet objects
getAlphabets(s)
# get Alphabets from a range
getAlphabets(s, c(2:3, 5))
# set alphabets via virtual field
s$alphabets <- list(BinaryAlphabet(), NucleotideAlphabet())
# get alphabets via virtual field
s$alphabets
```

---

`setAncestral.Sequence`  
Set the ancestral object of a Sequence object

Description

Set the ancestral object of a Sequence object.

Usage

```r
## S3 method for class 'Sequence'
setAncestral(this, value, ...)
```

Arguments

- **this**: A Sequence object.
- **value**: A Sequence or a Process object.
- **...**: Not used.
Value

The new ancestral object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Sequence.

Examples

# create a nucleotide sequence and a process object
s<-NucleotideSequence(string="AGCT")
p<-Process(name="MyProcess")
# set the p as the ancestral of s
setAncestral(s,p)
s$ancestral
# clone s
cs<-clone(s)
# set cs as ancestral of s via virtual field
s$ancestral<-cs
# get ancestral ids
s$ancestral$id
cs$ancestral$id

setAncestral.Site  Forbidden action: setting the ancestral object for a Site object

Description

Forbidden action: setting the ancestral object for a Site object.

Usage

## S3 method for class 'Site'
setAncestral(this, value, ...)

Arguments

this  A Site object.
value  Not used.
...  Not used.

Value

Throws an error.
**setBaseFreqs.F81**

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see *Site*.

---

`setBaseFreqs.F81`  
*Set the base frequency parameters*

**Description**

Set the base frequency parameters.

The order of the frequency parameters must match with the order of symbols in the NucleotideAlphabet objects.

**Usage**

```r
## S3 method for class 'F81'
setBaseFreqs(this, value, ...)
```

**Arguments**

- `this`  
  A F81 object.

- `value`  
  A vector of base frequencies.

- `...`  
  Not used.

**Value**

`value` (invisible)

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see *F81*. 
Examples

```
# construct a F81 object
p <- F81()
# set/get base frequency parameters
setBaseFreqs(p, c(2, 1, 2, 1)/6)
getBaseFreqs(p)
# set/get base frequency parameters via virtual field
p$baseFreqs <- c(4, 4, 1, 1)/10
p$baseFreqs
# get object summary
summary(p)
```

setBaseFreqs.F84  Set the base frequency parameters

Description

Set the base frequency parameters.

The order of the frequency parameters must match with the order of symbols in the NucleotideAlphabet objects.

Usage

```
## S3 method for class 'F84'
setBaseFreqs(this, value, ...)
```

Arguments

- **this**: A F84 object.
- **value**: A vector of base frequencies.
- **...**: Not used.

Value

value (invisible)

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see F84.
setBaseFreqs.GTR

Examples

```r
# construct object
p <- F84()
# set/get base frequency parameters
setBaseFreqs(p, c(2, 1, 2, 1)/6)
getBaseFreqs(p)
# set/get base frequency parameters via virtual field
p$dbasefreqs <- c(4, 4, 1, 1)/10
p$dbasefreqs
# get object summary
summary(p)
```

Description

Set the base frequency parameters.

The order of the frequency parameters must match with the order of symbols in the NucleotideAlphabet objects.

Usage

```r
## S3 method for class 'GTR'
setBaseFreqs(this, value, ...)
```

Arguments

- `this` A GTR object.
- `value` A vector of base frequencies.
- `...` Not used.

Value

- `value (invisible)`

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `GTR`.
Examples

# construct a GTR object
p<-GTR()
# set/get base frequency parameters
setBaseFreqs(p,c(2,1,2,1)/6)
getBaseFreqs(p)
# set/get base frequency parameters via virtual field
p$baseFreqs<-c(4,4,1,1)/10
p$baseFreqs
# get object summary
summary(p)

setBaseFreqs.HKY  Set the base frequency parameters

Description

Set the base frequency parameters.

The order of the frequency parameters must match with the order of symbols in the NucleotideAlphabet objects.

Usage

## S3 method for class 'HKY'
setBaseFreqs(this, value, ...)

Arguments

this  An HKY object.
value  A vector of base frequencies.
...  Not used.

Value

value (invisible)

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see HKY.
Examples

# construct object
p <- K8Y()

# set/get base frequency parameters
setBaseFreqs(p, c(2, 1, 2, 1)/6)
getBaseFreqs(p)

# set/get base frequency parameters via virtual field
p$baseFreqs <- c(4, 4, 1, 1)/10
p$baseFreqs

# get object summary
summary(p)

---

**setBaseFreqs.K80**  
*Forbidden action: setting the base frequency parameters for a K80 object*

### Description

Forbidden action: setting the base frequency parameters for a K80 object.

### Usage

```r
## S3 method for class 'K80'
setBaseFreqs(this, value, ...)
```

### Arguments

- **this**: An object.
- **value**: Not used.
- **...**: Not used.

### Value

Throws an error.

### Author(s)

Botond Sipos, Gregory Jordan

### See Also

For more information see K80.
**setBaseFreqs.K81**

*Forbidden action: setting the base frequency parameters for a K81 model*

---

**Description**

Forbidden action: setting the base frequency parameters for a K81 model.

**Usage**

```r
## S3 method for class 'K81'
setBaseFreqs(this, value, ...)
```

**Arguments**

- `this`: An object.
- `value`: Not used.
- `...`: Not used.

**Value**

Throws an error.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `K81`.

---

**setBaseFreqs.TN93**

*Set the base frequency parameters*

---

**Description**

Set the base frequency parameters.

The order of the frequency parameters must match with the order of symbols in the NucleotideAlphabet objects.

**Usage**

```r
## S3 method for class 'TN93'
setBaseFreqs(this, value, ...)
```
Arguments

this: A TN93 object.
value: A vector of base frequencies.
...: Not used.

Value

value (invisible)

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see TN93.

Examples

# construct a TN93 object
p<-TN93()
# set/get base frequency parameters
setBaseFreqs(p,c(2,1,2,1)/6)
getBaseFreqs(p)
# set/get base frequency parameters via virtual field
p$baseFreqs<-c(4,4,1,1)/10
p$baseFreqs
# get object summary
summary(p)

setBigRate.Sequence

Forbidden action: setting the sum of total active event rates for a Sequence object

Description

Forbidden action: setting the sum of total active event rates for a Sequence object.

Usage

## S3 method for class 'Sequence'
setBigRate(this, value, ...)

Arguments

this: An object.
value: Not used.
...: Not used.
setBranchEvents.PhyloSim

Value

Throws an error.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Sequence.

setBranchEvents.PhyloSim

Forbidden action: setting the list of events having per-branch statistics recorded

Description

Forbidden action: setting the list of events having per-branch statistics recorded.

Usage

## S3 method for class 'PhyloSim'
setBranchEvents(this, value, ...)

Arguments

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>this</td>
<td>An object.</td>
</tr>
<tr>
<td>value</td>
<td>Not used.</td>
</tr>
<tr>
<td>...</td>
<td>Not used.</td>
</tr>
</tbody>
</table>

Value

Throws an error.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see PhyloSim.
Get codon frequencies.

Usage

```r
## S3 method for class 'GY94'
setCodonFreqs(this, value, ...)
```

Arguments

- `this`: A GY94 object.
- `value`: A vector of codon frequencies.
- `...`: Not used.

Value

- `value` (invisible)

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `GY94`.

Examples

```r
# create a GY94 object
p <- GY94()
# get codon frequencies
getCodonFreqs(p)
p$codonFreqs
# set codon frequencies
p$codonFreqs <- rep(c(1, 2, 3, 4), length.out = 61)
p$codonFreqs
```
setComments.PSRoot  Set the comments associated with an object

Description
Set the comments associated with an object.
The comment field can contain any type of object.

Usage
```r
## S3 method for class 'PSRoot'
setComments(this, new_value, ...)
```

Arguments
- `this`: A PSRoot object.
- `new_value`: An object.
- `...`: Not used.

Value
The new value of the comment field (invisible).

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see `PSRoot`.

Examples
```r
# create an object
o<-PSRoot()
# add some comments
setComments(o,"Random comment")
# get the comment
getComments(o)
# get/set the comment via virtual fields
o$comments<-"Second random comment"
o$comments
```
setCumulativeRates.Sequence

Forbidden action: setting the cumulative rates for the sites aggregated by a Sequence object

Description

Forbidden action: setting the cumulative rates for the sites aggregated by a Sequence object.

Usage

```r
## S3 method for class 'Sequence'
setCumulativeRates(this, value, ...)
```

Arguments

- **this**: An object.
- **value**: Not used.
- **...**: Not used.

Value

Throws an error.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Sequence`.

setDeletionTolerance.Sequence

Set the deletion tolerance site-process specific parameter for a collection of Site objects aggregated by a Sequence object

Description

Set the deletion tolerance site-process specific parameter for a collection of Site objects aggregated by a Sequence object. This method does some error checking and the calls `setParameterAtSites(this=this, process=process, id=id, index=index, value=value)`.

See `setParameterAtSites.Sequence` for more details.
setDist.ContinuousDeletor

Usage

## S3 method for class 'Sequence'
setDeletionTolerance(this, process, value, index, ...)

Arguments

- **this**: A Sequence object.
- **process**: A Process object.
- **value**: A numeric vector, recycled if shorter than the index vector.
- **index**: An integer vector specifying a set of positions. It is set to 1:seq$length if omitted.
- **...**: Not used.

Value

The Sequence object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

setParameterAtSites.Sequence

---

setDist.ContinuousDeletor

*Set the length sampling expression*

Description

Set the length sampling expression.

The length sampling expression can be any valid R expression returning a numeric vector of length one. The value returned by the expression will be rounded.

Usage

## S3 method for class 'ContinuousDeletor'
setDist(this, value, ...)

Arguments

- **this**: A ContinuousDeletor object.
- **value**: An R expression.
- **...**: Not used.
Value

An R expression object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `ContinuousDeletor`.

Examples

```r
# create object
o <- ContinuousDeleter(rate=1)
# set/get length sampling expression
setDist(o, expression(rnorm(1, mean=3, sd=2)))
getDist(o)
# set/get length sampling expression via virtual field
do$dist <- expression(rnorm(1, mean=6, sd=3))
do$dist
# set maxLength
o$maxLength <- 10
# propose a length
proposeLength(o)
```

Description

Set the length sampling expression.

The length sampling expression can be any valid R expression returning a numeric vector of length one. The value returned by the expression will be rounded.

Usage

```r
## S3 method for class 'ContinuousInsertor'
setDist(this, value, ...)
```

Arguments

- `this` A ContinuousInsertor object.
- `value` An R expression.
- `...` Not used.
Value

An R expression object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see ContinuousInsertor.

Examples

# create object
o <- ContinuousInsertor(rate = 1)
# set/get length sampling expression
setDist(o, expression(rnorm(1, mean = 3, sd = 2)))
getDist(o)
# set/get length sampling expression via virtual field
o$dist <- expression(rnorm(1, mean = 6, sd = 3))
o$dist
# set maxLength
o$maxLength <- 10
# propose a length
proposeLength(o)
setEquDist. GeneralSubstitution

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see PhyloSim.

---

setEquDist. GeneralSubstitution

*Set the equilibrium distribution for a GeneralSubstitution object*

Description
Set the equilibrium distribution for a GeneralSubstitution object.

In the case the equilibrium distribution cannot be guessed from the rate matrix one should provide a valid equilibrium distribution. The equilibrium distribution must be compatible with the rate matrix. The provided numeric vector will be rescaled in the case the sum of the elements is not one.

Usage
```r
## S3 method for class 'GeneralSubstitution'
setEquDist(this, value, force=FALSE, silent=FALSE, ...)
```

Arguments

- `this`: A GeneralSubstitution object.
- `value`: A numeric vector containing the equilibrium symbol frequencies. The order of the frequencies must be the same as in the symbol vector of the attached Alphabet object.
- `force`: Do not check compatibility with the rate matrix.
- `silent`: Do not print out warnings.
- `...`: Not used.

Value
The new equilibrium distribution (invisible).

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see GeneralSubstitution.
Examples

# create a GeneralSubstitution object
# provide an Alphabet object and the rates
p <- GeneralSubstitution(
    alphabet = BinaryAlphabet(),
    rate.list = list("1->0" = 1, "0->1" = 1)
)

# get equilibrium distribution
getEquidist(p)

# get equilibrium distribution via virtual field
p$equidist

# re-set the equilibrium distribution
dist <- p$equidist * 3
dist
setEquidist(p, dist)
p$equidist

# re-set equilibrium distribution via virtual field
p$equidist <- p$equidist * 2
p$equidist

---

setEvents.Site

Forbidden action: setting the list of active events for a Site object

Description

Forbidden action: setting the list of active events for a Site object.

Usage

## S3 method for class 'Site'
setEvents(this, value, ...)

Arguments

- **this**: An object.
- **value**: Not used.
- **...**: Not used.

Value

Throws an error.

Author(s)

- Botond Sipos, Gregory Jordan

See Also

For more information see Site.
setGenerateBy.GeneralInsertor

Set the function object used for generating inserts

Description

Set the function object used for generating inserts. The provided function must return a Sequence object when called and must have the following arguments: process, length, target.seq, event.pos, insert.pos (see `generateInsert.GeneralInsertor`).

Usage

```r
## S3 method for class 'GeneralInsertor'
setGenerateBy(this, value, ...)
```

Arguments

- `this` A GeneralInsertor object.
- `value` A function object.
- `...` Not used.

Value

The function object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `GeneralInsertor`.

Examples

```r
# create a GeneralInsertor object
i<-GeneralInsertor(
  rate=0.5,
  propose.by=function(process){sample(c(5:10),1)}, # inserts between 5 and 10
  template.seq=NucleotideSequence(string="AAAAAAA")
)

# save insert generator
old.gen<-getGenerateBy(i)
# set a new insert generator
i$generateBy<-function(
  process,
  length,
```
Forbidden action: setting the handler function of an Event object

Description

Forbidden action: setting the handler function of an Event object. The handler function is typically set by a `getEventsAtSite.*` method generating the Event object by directly modifying the `this$.handler` field or by the `.setHandler()` method.

Usage

```ruby
## S3 method for class 'Event'
setHandler(this, value, ...)
```

Arguments

- this: An object.
- value: Not used.
- ...: Not used.

Value

Throws an error.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Event`. 
**setId.PhyloSim**

Forbidden action: setting the unique identifier of a PhyloSim object.

**Description**

Forbidden action: setting the unique identifier of a PhyloSim object.

**Usage**

```
## S3 method for class 'PhyloSim'
setId(this, value, ...)
```

**Arguments**

- **this**: An object.
- **value**: Not used.
- **...**: Not used.

**Value**

Throws an error.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `PhyloSim`.

---

**setId.Process**

Forbidden action: setting the unique Process object identifier.

**Description**

Forbidden action: setting the unique Process object identifier.

**Usage**

```
## S3 method for class 'Process'
setId(this, value, ...)
```
Arguments

this       An object.
value      Not used.
...        Not used.

Value

Throws an error.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `process`.

---

```r
setId.QMatrix
```

Forbidden action: setting the unique identifier for a QMatrix object

Description

Forbidden action: setting the unique identifier for a QMatrix object.

Usage

```r
## S3 method for class 'QMatrix'
setId(this, value, ...)
```

Arguments

this       An object.
value      Not used.
...        Not used.

Value

Throws an error.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `QMatrix`. 
**setid.Sequence**

Forbidden action: setting the unique identifier of a Sequence object.

**Description**

Forbidden action: setting the unique identifier of a Sequence object.

**Usage**

```r
## S3 method for class 'Sequence'
setId(this, value, ...)
```

**Arguments**

- **this**
  - An object.
- **value**
  - Not used.
- **...**
  - Not used.

**Value**

Throws an error.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `Sequence`.

---

**setInsertHook.GeneralInsertor**

*Set the insert hook function*

**Description**

Set the insert hook function.

The insert hook allows to make various modifications on the insert before performing the insertion. The function must have the following arguments: `seq` (the sequence object to insert), `target.seq` (the target Sequence object), `event.pos` (the position of the site which generated the insertion event), `insert.pos` (the position of the insertion).

The insert hook function is called by the insertion event handler function. The insert hook takes the sequence generated by the `generateInsert` method throught the "seq" argument. The function must return a Sequence object, which will be inserted in the target sequence.
Usage

```r
## S3 method for class 'GeneralInsertor'
setInsertHook(this, value, ...)
```

Arguments

- **this**: A GeneralInsertor object.
- **value**: A function object.
- **...**: Not used.

Value

The function object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `GeneralInsertor`.

Examples

```r
# create a GeneralInsertor object
i <- GeneralInsertor(
  rate = 0.5,
  propose.by = function(process){sample(c(5:10),1)}, # inserts between 5 and 10
  template.seq = NucleotideSequence(length = 5, processes = list(list(JC69())))
)
# set a dummy insert hook
setInsertHook(i, function(seq){return(seq)})
# set a new insert hook via virtual field
i$insertHook <- function(seq){
  seq$processes <- list(list(GTR())) # replace the subsitution process
  return(seq)
}
# get the insert hook via virtual field
i$insertHook
# get the insert hook
getInsertHook(i)
```
setInsertionTolerance.Sequencedefine

Set the insertion tolerance site-process specific parameter for a collection of Site objects aggregated by a Sequence object

Description

Set the insertion tolerance site-process specific parameter for a collection of Site objects aggregated by a Sequence object. This method does some error checking and the calls setParameterAtSites(this=this, process=process, index=index, value=value). See setParameterAtSites.Sequence for more details.

Usage

## S3 method for class 'Sequence'
setInsertionTolerance(this, process, value, index, ...)

Arguments

this A Sequence object.
process A Process object.
value A numeric vector, recycled if shorter than the index vector.
index An integer vector specifying a set of positions. It is set to 1:seq$length if omitted.
... Not used.

Value

The Sequence object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

setParameterAtSites.Sequence
Description

Get the transition transversion rate ratio.

Usage

```r
## S3 method for class 'F84'
setKappa(this, value, ...)
```

Arguments

- `this`: A `F84` object.
- `value`: A numeric vector of length one.
- `...`: Not used.

Value

The new value of Kappa (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `fXT`.

Examples

```r
# construct a F84 object
p<-F84();
# set/get Kappa
setKappa(p, 2)
getKappa(p)
# set/get Kappa via virtual field
p$kappa<-4
p$kappa
# get object summary
summary(p)
```
Description
Set the transition/transversion rate ratio.

Usage
## S3 method for class 'GY94'
setKappa(this, value, ...)

Arguments
this
A GY94 object.
value
A numeric vector of length one.
... Not used.

Value
The new value of kappa.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see GY94.

Examples
# create a GY94 object
p<-GY94()
# set/get kappa
setKappa(p,2)
getKappa(p)
# set/get kappa via virtual field
p$kappa<-3
p$kappa
setLengthParam1.FastFieldDeletor

setLength.Sequence

Forbidden action: setting the length of a Sequence object

Description
Forbidden action: setting the length of a Sequence object. The length of the Sequence objects can be specified when the object is constructed, or modified later by the "insertSequence" and "deleteSubSequence" methods.

Usage

### S3 method for class 'Sequence'
setLength(this, value, ...)

Arguments

- this: An object.
- value: Not used.
- ...: Not used.

Value

Throws an error.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Sequence.

---

setLengthParam1.FastFieldDeletor

Set the first length parameter

Description

Set the first length parameter.

Usage

### S3 method for class 'FastFieldDeletor'
setLengthParam1(this, value, ...)

Arguments

- **this**: A FastFieldDeletor object.
- **value**: A numeric vector of length one.
- **...**: Not used.

Value

- **value**: (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `FastFieldDeletor`.

Examples

```r
# create a geometric FastFieldDeletor
p <- FastFieldDeletor()
# set/get the first length parameter
setLengthParam1(p, 0.5)
getLengthParam1(p)
# set/get the first length parameter via virtual field
p$lengthParam1 <- 0.2
p$lengthParam1
```

Description

Set the second length parameter.

Usage

```r
## S3 method for class 'FastFieldDeletor'
setLengthParam2(this, value, ...)
```

Arguments

- **this**: A FastFieldDeletor object.
- **value**: A numeric vector of length one.
- **...**: Not used.
Value

value (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see FastFieldDeletor.

Examples

# create a geometric FastFieldDeletor
p<-FastFieldDeletor()
# set/get the second length parameter
setLengthParam2(p,0.5)
getLengthParam2(p)
# set/get the second length parameter via virtual field
p$lengthParam2<-0.2
p$lengthParam2

setLogFile.PhyloSim  Set the name of the file used for logging

Description

Set the name of the file used for logging.

Usage

## S3 method for class 'PhyloSim'
setLogFile(this, value, ...)

Arguments

this A PhyloSim object.

value The name of the file used for logging.

... Not used.

Value

The new logfile.

Author(s)

Botond Sipos, Gregory Jordan
setLogLevel.PhyloSim

See Also

For more information see PhyloSim.

Examples

```r
# Create a PhyloSim object
sim <- PhyloSim();
# get the name of the log file
getLogFile(sim)
# modify log file name
setLogFile(sim, "OldLog.txt")
# get/set log file name via virtual field
sim$logFile
sim$logFile <- "NewLog"
sim$logFile
```

Description

Set the log level for a given PhyloSim object.

No logging is performed if the log level is negative. If the log level is zero, the messages passed to the Log method will be written in the log file. If the log level is positive, the messages passed to the Debug method are saved as well.

The default log level is -1. The specified file will be truncated in the case it already exists.

Usage

```r
## S3 method for class 'PhyloSim'
setLogLevel(this, value, ...)
```

Arguments

- **this**: A PhyloSim object.
- **value**: The new log level as an integer.
- **...**: Not used.

Value

The new level as an integer vector of length one.

Author(s)

Botond Sipos, Gregory Jordan
setMatrix.QMatrix

See Also

gLogLevel PhyloSim

Examples

setMatrix.QMatrix

Forbidden action: setting the unscaled rate matrix for a QMatrix object.

Description

Forbidden action: setting the unscaled rate matrix for a QMatrix object.

Usage

## S3 method for class 'QMatrix'
setMatrix(this, value, ...)

Arguments

this An object.

value Not used.

... Not used.

Value

Throws an error.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see QMatrix.
setMaxLength.ContinuousDeletor

Set the maximum length

Description

Set the maximum length.

Usage

```r
## S3 method for class 'ContinuousDeletor'
setMaxLength(this, value, ...)
```

Arguments

- `this` A ContinuousDeletor object.
- `value` A numeric (integer) vector of length one.
- `...` Not used.

Value

The new maximum length.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `ContinuousDeletor`.

Examples

```r
# create object
o <- ContinuousDeletor(rate=1)
# set length sampling expression via virtual field
o$dist <- expression(rnorm(1, mean=6, sd=3))
# set/get maxLength
setMaxLength(o, 3)
getMaxLength(o)
# set/get maxLength via virtual field
o$maxLength <- 10
o$maxLength
# propose a length
proposeLength(o)
```
Description

Set the maximum length.

Usage

```r
## S3 method for class 'ContinuousInsertor'
setMaxLength(this, value, ...)
```

Arguments

- `this`: A ContinuousInsertor object.
- `value`: A numeric (integer) vector of length one.
- `...`: Not used.

Value

The new maximum length.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `ContinuousInsertor`.

Examples

```r
# create object
o <- ContinuousInsertor(rate=1)
# set length sampling expression via virtual field
o$dist <- expression(rnorm(1,mean=6,sd=3))
# set/get maxLength
setMaxLength(o, 3)
getMaxLength(o)
# set/get maxLength via virtual field
o$maxLength <- 10
o$maxLength
# propose a length
proposeLength(o)
```
setMethodsList.PSRoot  
*Forbidden action: setting the list of applicable methods for an object*

**Description**

Forbidden action: setting the list of applicable methods for an object.

**Usage**

```r
## S3 method for class 'PSRoot'
setMethodsList(this, value, ...)
```

**Arguments**

- `this`  
  An object.
- `value`  
  Not used.
- `...`  
  Not used.

**Value**

Throws an error.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `PSRoot`.

---

setName.Event  
*Set the name of an Event object*

**Description**

Set the name of an Event object.

**Usage**

```r
## S3 method for class 'Event'
setName(this, new.name, ...)
```
Arguments

this  An Event object.
new.name  A character vector of length one.
...  Not used.

Value

The new name (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Event.

Examples

# create an Event object
e<-Event()
# set event name
setName(e,"Insertion")
# get event name
e$name
# set name via virtual field
e$name<-"Deletion"
e$name

setName.PhylolSim  Set the name of a PhylolSim object

Description

Set the name of a PhylolSim object.

Usage

## S3 method for class 'PhylolSim'
setName(this, new.name, ...)

Arguments

this  A PhylolSim object.
new.name  A character vector of length one.
...  Not used.
Value

The new name.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see PhyloSim.

Examples

```r
# create a PhyloSim object
o <- PhyloSim();
# set/get name
setName(o, "MySim");
getName(o, "MySim");
# set/get name via virtual field
o$process$name <- "George";
o$name
```

Description

Set the name of a Process object.

This method also updates the unique identifier of the Process object.

Usage

```r
## S3 method for class 'Process'
setName(this, new.name, ...)
```

Arguments

- `this` A Process object.
- `new.name` A character vector of length one.
- `...` Not used.

Value

Returns the new name (invisible).
Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see Process.

Examples

```r
# create object
p <- Process()
# get name
p$name
# set new name
getName(p)
# get name and id
p$name
p$id
# set new name via virtual field
p$name <- "Procey"
p$name
```

---

**setName.QMatrix**  
*Set the name of a QMatrix object*

**Description**
Set the name of a QMatrix object.

**Usage**
```r
## S3 method for class 'QMatrix'
setName(this, value, ...)
```

**Arguments**
- **this**  
  A QMatrix object.
- **value**  
  A character vector of length one.
- **...**  
  Not used.

**Value**
The new object name.

**Author(s)**
Botond Sipos, Gregory Jordan
See Also

For more information see QMatrix.

Examples

```r
# create a QMatrix object
m <- QMatrix()
# set/get name
setName(m, "Susie Q")
getName(m)
# set/get name via virtual field
m$name <- "Q"
```

Description

Set the name of a Sequence object.

Usage

```r
## S3 method for class 'Sequence'
setName(this, new.name, ...)
```

Arguments

- `this` A Sequence object.
- `new.name` A character vector of length one.
- `...` Not used.

Value

The new object name (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Sequence.
Examples

```r
# create a Sequence object
s <- Sequence(name = "MySeq", length = 4)

# get sequence name
s$name

# rename object
setName(s, "50")

# rename via virtual field
s$name <- "SeqSeq"

s$name
```

**Description**

Forbidden action: setting the number of edges for phylo object aggregated by a PhyloSim object.

**Usage**

```r
## S3 method for class 'PhyloSim'
setNedges(this, value, ...)
```

**Arguments**

- `this` An object.
- `value` Not used.
- `...` Not used.

**Value**

Throws an error.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `PhyloSim`.
Description
Forbidden action: setting the node identifiers for a PhyloSim object.

Usage
```
## S3 method for class 'PhyloSim'
setNodes(this, value, ...)
```

Arguments
- `this` An object.
- `value` Not used.
- `...` Not used.

Value
Throws an error.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see `PhyloSim`.

---

Description
Forbidden action: setting the number of the tips for a phylo object aggregated by a PhyloSim object.

Usage
```
## S3 method for class 'PhyloSim'
setNtips(this, value, ...)
```
Arguments

this    An object.
value   Not used.
...     Not used.

Value

Throws an error.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see PhyloSim.

---

setOmegas.CodonSequence

Set the omegas for a collection of sites

Description

Set the omegas for a collection of sites.

Usage

```r
## S3 method for class 'CodonSequence'
setOmegas(this, process, value, index, ...)
```

Arguments

this    A CodonSequence object.
process A process object inheriting from GY94.
value   A vector containing the new values of the site-process specific parameter, recy-
cled if shorter than the index vector.
index   A vector of positions. It is set to 1:seq$length if omitted.
...     Not used.

Value

A numeric vector.

Author(s)

Botond Sipos, Gregory Jordan
setParameterAtSite.Process

See Also

For more information see `CodonSequence`.

Examples

```r
# create a GY94 process
p<-GY94()
# create a CodonSequence object,
# attach a process p
s<-CodonSequence(length=20,processes=list(list(p)))
# set omega values in range 1:5
setOmegas(s,p,c(0.5,1,1.5),1:5)
# get omega values from sites 1,2,3,10, and 20
getOmegas(s,p,c(1:3,10,20))
```

---

**setParameterAtSite.Process**

*Set the value of a site-process specific parameter in a Site object attached to a Process object*

---

Description

Set the value of a site-process specific parameter in a Site object attached to a Process object. The Process object must be attached to the Site object. The new value must be compatible with the type of the site-process specific parameter.

Usage

```r
## S3 method for class 'Process'
setParameterAtSite(this, site, id, value, ...)
```

Arguments

- `this` A Process object.
- `site` A Site object.
- `id` The identifier of the site-process specific parameter.
- `value` The new value for the parameter.
- `...` Not used.

Value

The site-process specific parameter, which is a list containing the following keys: id, name, value, type.
setParameterAtSites.Sequence

Set the values of a site-process specific parameter for a process and a collection of Site objects aggregated by a Sequence object.

Description

Set the values of a site-process specific parameter for a process and a collection of Site objects aggregated by a Sequence object.

Usage

```r
## S3 method for class 'Sequence'
setParameterAtSites(this, process, id, value, index, ...)
```

Arguments

- `this` A Sequence object.
- `process` A valid Process object.
- `id` The identifier of the site-process specific parameter.
- `value` A vector containing the new values of the site-process specific parameter, recycled if shorter than the index vector. It should be consistent with the type of the parameter.
- `index` An integer vector specifying a set of positions. It is set to 1:seq$length if omitted.
- `...` Not used.
Value

The Sequence object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

Site Process For more information see Sequence.

Examples

# create a sequence, attach a process
p<-k80()
s<-Sequence(length=6,alphabets=list(NucleotideAlphabet()),processes=list(list(p)))
# set a new pattern of rate multipliers in the range 1:3,
# the default value is 1.0 by the way
setParameterAtSites(s,p,"rate.multiplier",c(2,3),1:3)
# get rate multipliers
getParameterAtSites(s,p,"rate.multiplier")
# set a new value for the whole sequence
setParameterAtSites(s,p,"rate.multiplier",0.5)
# get rate multipliers
getParameterAtSites(s,p,"rate.multiplier")
Value

A phylo object or FALSE.

Author(s)

Botond Sipos, Gregory Jordan

See Also

The PhyloSim class, the ape package.

Examples

#create a PhyloSim object
sim<-PhyloSim();
# creat a phylo object
tree<-rcoal(3);
# get/set phylo object
setPhylo(sim,tree);
getPhylo(sim,tree);
# get/set phylo object via virtual field
sim$tree<-rcoal(5);
sim$tree;

setPosition.Event  Set the position of the Site object associated to an Event object

Description

Set the position of the Site object associated to an Event object.

The position field is usually not modified directly, but set by the getEvents.Sequence method. The position is *not* set by Site methods as getEventsAtSite.Site.

Usage

## S3 method for class 'Event'
setPosition(this, value, ...)

Arguments

this  An Event object.
value  The position.
...  Not used.

Value

The new position (invisible).
**setProbs.DiscreteDeletor**

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `Event`.

**Examples**

```r
# Note: the following example is not too useful

# create a sequence and attach a process
s<-NucleotideSequence(string="ATGC",processes=list(list(JC69())))
# get the first active event from the first site
e<-getEvents(s,1)[[1]]
# get event position
e$position
# set the position of the site associated with e
setPosition(e,2)
# get position via virtual field
e$position
# set position via virtual field
e$position<-1
e$position
```

---

**Description**

Set the deletion length probabilities.  
The `sizes` virtual field must be set before setting the length probabilities. The length of the provided numeric vector must match with the length of the vector stored in the `sizes` virtual field. The vector is rescaled if the values do not sum to one and a warning is issued.

**Usage**

```r
## S3 method for class 'DiscreteDeletor'
setProbs(this, value, ...)
```

**Arguments**

- `this`: A `DiscreteDeletor` object.
- `value`: A numeric vector containing the length probabilities.
- `...`: Not used.
Value

The vector of probabilities.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see DiscreteDeletor.

Examples

```r
# create a DiscreteDeleter object
d <- DiscreteDeleter(rate = 1, sizes = 1:3)
# set/get length probabilities
setProbs(d, c(1/3, 1/3, 1/3))  # equal probabilities
getProbs(d)
# set/get length probabilities via virtual field
x <- c(2, 2, 1)
# normalize x
x <- x / sum(x)
d$probs <- x
d$probs
```

Description

Set the insertion length probabilities.

The sizes virtual field must be set before setting the length probabilities. The length of the provided numeric vector must match with the length of the vector stored in the sizes virtual field. The vector is rescaled if the values do not sum to one and a warning is issued.

Usage

```r
## S3 method for class 'DiscreteInsertor'
setProbs(this, value, ...)
```

Arguments

- **this**: A DiscreteInsertor object.
- **value**: A numeric vector containing the length probabilities.
- **...**: Not used.
**setProcess.Event**

**Value**

The vector of probabilities.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see *DiscreteInsertor*.

**Examples**

```r
# create a DiscreteInsertor object
i <- DiscreteInsertor(rate=1, sizes=1:3)
# set/get length probabilities
setProbs(i, c(1/3, 1/3, 1/3))  # equal probabilities
getProbs(i)
# set/get length probabilities via virtual field
x <- c(2, 2, 1)
# normalize x
x <- x/sum(x)
i$probs <- x
i$probs
```

---

**setProcess.Event**

*Set the generator process for an Event object*

**Description**

Set the generator process for an Event object.

**Usage**

```r
## S3 method for class 'Event'
setProcess(this, new.proc, ...)
```

**Arguments**

- **this**
  - An Event object.
- **new.proc**
  - A valid Process object.
- **...**
  - Not used.

**Value**

A Process object.
Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Event.

Examples

```r
# create an Event object
e <- Event()
# set a generator process for e
setProcess(e, Process())
# get generator process
e$process
# set process via virtual field
e$process <- K80()
e$process
```

setProcess.QMatrix  
Associate a process object with a QMatrix object

Description

Associate a process object with a QMatrix object.

Usage

```r
## S3 method for class 'QMatrix'
setProcess(this, value, ...)
```

Arguments

- **this**: A QMatrix object.
- **value**: A Process object.
- **...**: Not used.

Value

The Process object.

Author(s)

Botond Sipos, Gregory Jordan
setProcesses.Sequence

See Also

For more information see QMatrix.

Examples

# Create a GeneralSubstitution object
p <- GeneralSubstitution(alphabet=BinaryAlphabet())
p
# get the associated QMatrix object from p
m <- p$qMatrix
summary(m)
# get the associated process from m
m$process
# clone p
pp <- clone(p)
# associate m with pp
pp$qMatrix <- m
# associate pp with m
m$process <- pp
m$process

setProcesses.Sequence  Specify a set of Process objects to be attached to a set of Site objects aggregated by a Sequence object

Description

Specify a set of Process objects to be attached to a set of Site objects aggregated by a Sequence object. The Process objects in a given inner list correspond to the set of processes to be attached to one Site object. Process objects already attached to a given Site are skipped. Attached processes which are not members of the list are detached, so specifying an empty list will detach all processes.

Usage

## S3 method for class 'Sequence'
setProcesses(this, value, index, ...)

Arguments

this  A Sequence object.
value  A list of list of Process objects, recycled if shorter than the index vector.
index  An integer vector specifying a set of positions. It is set to 1:seq$length if omitted.
...  Not used.
Value

The Sequence object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Sequence.

Examples

# create a sequence of length 5
s <- Sequence(length=5, alphabets=list(NucleotideAlphabet()));
# set a pattern of processes
setProcesses(s, list(list(JC69(), K81())))
# get attached processes
s$processes
# detach all processes from range 1:3
setProcesses(s, list(list()), 1:3)
s$processes
# detach all processes via virtual field
s$processes <- list(list())
# create a process pattern in the full sequence via virtual field
s$processes <- list(JC69(), list(GTR(), K88()))
s$processes

---

setProcesses.Site Specify the list of Process objects attached to a Site object

Description

Specify the list of Process objects attached to a Site object. The Process objects in the "value" list correspond to the set of processes to be attached to the Site object. Process objects already attached to a given Site are skipped. Attached processes which are not members of the list are detached, so specifying an empty list will detach all processes.

This method is an alternative to attachProcess.Site and detachProcess.Site, working with more than one process object.

Usage

## S3 method for class 'Site'
setProcesses(this, value, ...)


setProposeBy GeneralInDel

Arguments

this          A Site object.
value         A list of valid Process objects.
...           Not used.

Value

The Site object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Site.

Examples

# create a Site object
s<-Site(alphabet=NucleotideAlphabet())
# create some processes
p1<-JC69(); p2<-K80(); p3<-DiscreteInsertor(rate=1);
# attach the processes
setProcesses(s,list(p1,p2,p3))
# attach one more process via virtual field
s$processes<-c(s$processes,list(GTR))
# get the list of attached processes
s$processes
# detach all processes via virtual field
s$processes<-list()

Description

Set the function used for proposing indel lengths.

Usage

## S3 method for class 'GeneralInDel'
setProposeBy(this, value, ...)
Arguments

  this  A GeneralInDel object.
  value A function object returning a numeric vector of length one.
  ...  Not used.

Value

  The function object (invisible).

Author(s)

  Botond Sipos, Gregory Jordan

See Also

  For more information see GeneralInDel.

Examples

```r
# create a GeneralInDel object
# proposing events with a constant length of 5
o <- GeneralInDel(rate = 1, propose.by = function(process) { return(5) });
# set/get the proposeBy function
setProposeBy(o, value = function(process) { return(6) })
getProposeBy(o)
# set/get proposeBy function via virtual field
o$proposeBy <- function(process) { return(3) }
```

Description

  Set the QMatrix object aggregated by a GeneralSubstitution object.

  This method is mostly used internally.

Usage

```r
## S3 method for class 'GeneralSubstitution'
setQMatrix(this, value, ...)
```
Arguments

this A GeneralSubstitution object.
value A QMatrix object.
... Not used.

Value

The QMatrix object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see GeneralSubstitution.

Examples

# create a GeneralSubstitution object
# provide an Alphabet object and the rates
p< GeneralSubstitution(alphabet=BinaryAlphabet(), rate.list=list("1->0"=1,"0->1"=1))
# get the QMatrix object
gQM(p)
# get the QMatrix object via virtual field
q<-p$qMatrix
# tweak with the QMatrix
setRate(q,"0->1",2)
# set a new QMatrix for p
setQMatrix(p,q)
summary(p)
# set new QMatrix via virtual field
setRate(q,"1->0",2)
p$qMatrix<-q
summary(p)

setDescription(Event, Set the rate of an Event object)

Set the rate of an Event object.

Usage

## S3 method for class 'Event'
setRate(this, value, ...)

Arguments

- **this**: An Event object.
- **value**: The event rate.
- **...**: Not used.

Value

The new value of the rate (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Event.

Examples

```r
# create an Event object
e <- Event(rate = 0.1)
# set a new rate
setRate(e, 0.2)
# get rate via virtual field
e$rate
# set rate via virtual field
e$rate <- 0.5
e$rate
```

---

setRate.GeneralIndel  
*Set the general rate*

Description

Set the general rate.

Usage

```r
## S3 method for class 'GeneralIndel'
setRate(this, value, ...)
```

Arguments

- **this**: A GeneralIndel object.
- **value**: The new general rate (a numeric vector of length one).
- **...**: Not used.
**Value**

The new general rate.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see GeneralIndel.

**Examples**

```r
# create a GeneralInDel object
o<-GeneralInDel(rate=0.5)
# get/set general rate
getRate(o)
setRate(o, 1.5)
# get/set rate via virtual field
o$rate
o$rate<-0.3
o$rate
```

---

**Description**

Set an unscaled rate for an event from a GeneralSubstitution object.

This method sets the element corresponding to a given event in the unscaled Q matrix. The event can be specified by the initial and target states ("from" and "to" arguments), or by the event name ("from->to"). The event name takes precedence over the "from" and "to" arguments.

Modifying any rate in the unscaled Q matrix will trigger the re-scaling of the whole matrix. The rescaled rates (used during simulations) are returned by the getEventRate method.

This method doesn’t modify the site specific rate multipliers.

**Usage**

```r
## S3 method for class 'GeneralSubstitution'
setRate(this, name=NA, value, from=NA, to=NA, ...)
```
Arguments

- `this`: A GeneralSubstitution object.
- `name`: The name of the event.
- `from`: The initial state.
- `value`: The new value of the rate.
- `to`: Target state.
- `...`: Not used.

Value

A Numeric vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `GeneralSubstitution`.

Examples

```r
# create a GeneralSubstitution object
# provide an Alphabet object and the rates
p <- GeneralSubstitution(alphabet=BinaryAlphabet(), rate=list("1->0"=1,"0->1"=1))
# set the unscaled rate by event name
setRate(p,"0->1",2)
# get the unscaled rate of "0->1" by name
getRate(p,"0->1")
# set the unscaled rate by states
setRate(p,"0->1",0.5)
# get the unscaled rate of "0->1" by states
getRate(p,from="0",to="1")
```

Description

Set an unscaled rate for an event from a GY94 object.

See `setRate.Generalsubstitution`.

Usage

```r
## S3 method for class 'GY94'
setRate(this, name=NA, value, from=NA, to=NA, ...)
```
Arguments

- **this**: A GeneralSubstitution object.
- **name**: The name of the event.
- **from**: The initial state.
- **value**: The new value of the rate.
- **to**: Target state.
- **...**: Not used.

Value

A Numeric vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see GY94.

---

**setRate.QMatrix**

*Set an unscaled rate in a QMatrix object*

**Description**

Set an unscaled rate in a QMatrix object.

This method sets the element corresponding to a given event in the unscaled rate matrix. The event can be specified by the initial and target states ("from" and "to" arguments), or by the event name ("from->to"). The event name takes precedence over the "from" and "to" arguments.

**Usage**

```r
## S3 method for class 'QMatrix'
setRate(this, name=NA, value, from=NA, to=NA, scale=TRUE, diag=TRUE, guess.equ=TRUE, ...)
```

Arguments

- **this**: A QMatrix object.
- **name**: The name of the event.
- **value**: The new value of the rate.
- **from**: The initial state.
- **to**: Target state.
- **scale**: Call rate rescaling.
- **diag**: Calculate diagonal elements.
- **guess.equ**: Guess equilibrium distribution.
- **...**: Not used.
Value

A Numeric vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see QMatrix.

Examples

# create a QMatrix object
# provide an Alphabet object and the rates
m<-QMatrix(alphabet=BinaryAlphabet(), rate.list=list("1->0"=1,"0->1"=1))
# set the unscaled rate by event name
setRate(m,"0->1",2)
# get the unscaled rate of "0->1" by name
getRate(m,"0->1")
# set the unscaled rate by states
setRate(m,"0->1",0.5)
# get the unscaled rate of "0->1" by states
getRate(m,"0->1",0.5)
# set the unscaled rate of "0->1" by states
getRate(m,from="0",to="1")
Value

The GeneralSubstitution object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see GeneralSubstitution.

Examples

```r
# create a GeneralSubstitution object
# provide an Alphabet object and the rates
p <- GeneralSubstitution(alphabet=BinaryAlphabet(), rate.list=list("1->0"=1,"0->1"=3))
# get the event rates from the unscaled Q matrix
getRateList(p)
# get rates from the unscaled rate matrix via virtual field
p$rateList
# set rates in the unscaled rate matrix
setRateList(p, list("0->1"=1,"1->0"=1))
p$rateList
# set rates in the unscaled rate matrix via virtual field
p$rateList<-list("0->1"=3,"1->0"=1);
# check the content of the associated QMatrix object
summary(p$QMatrix)
```

```r
setRateList.QMatrix Setting the unscaled rates stored in a QMatrix object

Description

Setting the unscaled rates stored in a QMatrix object.

This method set the rates in the unscaled Q matrix based on the provided list containing even names and the associated rates. The rate must be specified for every event!

Usage

```r
## S3 method for class 'QMatrix'
setRateList(this, value, ...)
```

Arguments

- `this` A QMatrix object.
- `value` A list with the events names and the associated rates.
- `...` Not used.
Value

The QMatrix object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `QMatrix`.

Examples

```r
# create a GeneralSubstitution object
# provide an Alphabet object and the rates
p <- GeneralSubstitution(alphabet=BinaryAlphabet(), rate.list=list("1->0"=1,"0->1"=3))
# get the QMatrix object from p
m <- p$QMatrix
# get the event rates from the unscaled Q matrix
generateList(m)
# get rates from the unscaled rate matrix via virtual field
m$rateList
# set rates in the unscaled rate matrix
setRateList(m, list("0->1"=1,"1->0"=1))
m$rateList
# set rates in the unscaled rate matrix via virtual field
m$rateList <- list("0->1"=3,"1->0"=1);
m$rateList
```

---

`setRateMultipliers.Sequence`

*Set the values of the rate multiplier parameters for a given Process object and a collection of Site object aggregated by a Sequence object*

Description

Set the values of the rate multiplier parameters for a given Process object and a collection of Site object aggregated by a Sequence object. This method just calls `setParameterAtSites(this=this, process=process, id="rate", index=index, ...)`.

See `setParameterAtSites.Sequence` for details.

Usage

```r
## S3 method for class 'Sequence'
setRateMultipliers(this, process, value, index, ...)
```
Arguments

- **this**: A Sequence object.
- **process**: A valid Process object.
- **value**: A numeric vector containing the new values of the site-process specific parameter, recycled if shorter than the index vector.
- **index**: An integer vector specifying a set of positions. It is set to 1:seq$length if omitted.
- **...**: Not used.

Value

The Sequence object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

setParameterAtSites.Sequence

---

Forbidden action: getting the list of rate parameters

Description

Forbidden action: getting the list of rate parameters.

This model has no rate parameters.

Usage

```r
## S3 method for class 'F81'
setRateParam(this, name, value, ...)
```

Arguments

- **this**: An object.
- **name**: Not used.
- **value**: Not used.
- **...**: Not used.

Value

The object.
**setRateParam.F84**  
*Set the value of a rate parameter*

**Description**

Set the value of a rate parameter.  
The rate parameters are: Kappa.

**Usage**

```r  
## S3 method for class 'F84'
setRateParam(this, name, value, ...)
```

**Arguments**

- `this` A F84 object.
- `name` The name of the rate parameter.
- `value` A numeric vector of length one.
- `...` Not used.

**Value**

The new value of the rate parameter (invisible).

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `F84`.

**Examples**

```r  
# construct a F84 object
p<-F84();
# set/get a rate parameter
setRateParam(p,"Kappa",4)
getRateParam(p,"Kappa")
# get object summary
summary(p)
```
setRateParam.GTR

Set the value of a rate parameter

Description

Set the value of a rate parameter.

The rate parameters are named as in PAML (see PAML documentation: http://bit.ly/9SQK2f).

Usage

## S3 method for class 'GTR'
setRateParam(this, name, value, ...)

Arguments

- **this**: A GTR object.
- **name**: The name of the rate parameter.
- **value**: A numeric vector of length one.
- **...**: Not used.

Value

The new value of the rate parameter (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see GTR.

Examples

# construct a GTR object
p<-GTR();
# set/get a rate parameter
setRateParam(p,"a",4)
getRateParam(p,"a")
# get object summary
summary(p)
setRateParam.HKY  

**Set the value of a rate parameter**

### Description

Set the value of a rate parameter.

The rate parameters are: Alpha, Beta.

### Usage

```r
## S3 method for class 'HKY'
setRateParam(this, name, value, ...)
```

### Arguments

- `this`  
  An HKY object.
- `name`  
  The name of the rate parameter.
- `value`  
  A numeric vector of length one.
- `...`  
  Not used.

### Value

The new value of the rate parameter (invisible).

### Author(s)

Botond Sipos, Gregory Jordan

### See Also

For more information see [HKY](https://example.com).

### Examples

```r
# construct HKY object
p<-HKY();
# set/get a rate parameter
setRateParam(p,"Alpha",4)
getRateParam(p,"Beta")
# get object summary
summary(p)
```
setRateParam.K80

Set the value of a rate parameter

Description

Set the value of a rate parameter.

The rate parameters are: Alpha, Beta.

Usage

## S3 method for class 'K80'
setRateParam(this, name, value, ...)

Arguments

this A K80 object.
name The name of the rate parameter.
value A numeric vector of length one.
... Not used.

Value

The new value of the rate parameter (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see K80.

Examples

# construct a K80 object
p<-K80();
# set/get a rate parameter
setRateParam(p,"Alpha",4)
getRateParam(p,"Beta")
# get object summary
summary(p)
setRateParam.K81

Set the value of a rate parameter

Description
Set the value of a rate parameter.
The rate parameters are: Alpha, Beta, Gamma.

Usage

```r
## S3 method for class 'K81'
setRateParam(this, name, value, ...)
```

Arguments

- **this**: A K8 object.
- **name**: The name of the rate parameter.
- **value**: A numeric vector of length one.
- **...**: Not used.

Value
The new value of the rate parameter (invisible).

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see `K81`.

Examples

```r
# construct a K80 object
p<-K81();
# set/get a rate parameter
setRateParam(p,"Alpha",4)
getRateParam(p,"Gamma")
# get object summary
summary(p)
```
setRateParam.T92

Set the value of a rate parameter

Description

Set the value of a rate parameter.

The rate parameters are: Alpha, Beta.

Usage

```r
## S3 method for class 'T92'
setRateParam(this, name, value, ...)
```

Arguments

- **this**: A T92 object.
- **name**: The name of the rate parameter.
- **value**: A numeric vector of length one.
- **...**: Not used.

Value

The new value of the rate parameter (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `T92`.

Examples

```r
# construct a T92 object
p<-T92();
# set/get a rate parameter
setRateParam(p,"Alpha",4)
getRateParam(p,"Beta")
# get object summary
summary(p)
```
**setRateParam.TN93**  
*Set the value of a rate parameter*

**Description**
Set the value of a rate parameter.  
The rate parameters are: Alpha1, Alpha2, Beta.

**Usage**
```r
## S3 method for class 'TN93'
setRateParam(this, name, value, ...)
```

**Arguments**
- `this`: A TN93 object.  
- `name`: The name of the rate parameter.  
- `value`: A numeric vector of length one.  
- `...`: Not used.

**Value**
The new value of the rate parameter (invisible).

**Author(s)**
Botond Sipos, Gregory Jordan

**See Also**
For more information see [TN93](##).

**Examples**
```r
# construct a TN93 object
p<-TN93();
# set/get a rate parameter
setRateParam(p,"Beta",4)
generateParam(p,"Beta")
# get object summary
summary(p)
```
**setRateParamList.F81**  
*Forbidden action: getting the list of rate parameters*

### Description
Forbidden action: getting the list of rate parameters.
This model has no rate parameters.

### Usage
```r
## S3 method for class 'F81'
setRateParamList(this, value, ...)
```

### Arguments
- `this`: An object.
- `value`: Not used.
- `...`: Not used.

### Value
The object.

### Author(s)
Botond Sipos, Gregory Jordan

### See Also
For more information see F81.

---

**setRateParamList.F84**  
*Set the rate parameters*

### Description
Set the rate parameters.
The rate parameters are: Kappa.

### Usage
```r
## S3 method for class 'F84'
setRateParamList(this, value, ...)
```
Arguments

this A F84 object.
value A list containing the rate parameters.
... Not used.

Value

The list of rate parameters (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see F84.

Examples

# create F84 object
p<-F84()
# set/get rate parameters
setRateParamList(p,list("Kappa"=3))
getRateParamList(p)
# set/get rate parameters via virtual field
p$rateParamList<-list("Kappa"=2.5)
p$rateParamList
# get object summary
summary(p)

Description

Set the rate parameters.

The rate parameters are named as in PAML (see PAML documentation: http://bit.ly/9SQK2f).

Usage

## S3 method for class 'GTR'
setRateParamList(this, value, ...)
setRateParamList.HKY

Arguments

- **this**: A GTR object.
- **value**: A list containing the rate parameters.
- **...**: Not used.

Value

The list of rate parameters (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see GTR.

Examples

```r
# create GTR object
p<-GTR()
# set/get rate parameters
setRateParamList(p, list(
  "a"=1, "b"=2, "c"=3,
  "d"=1, "e"=2, "f"=3
))
getRateParamList(p)
# set/get rate parameters via virtual field
p$rateParamList<-list(
  "a"=4, "b"=1, "c"=4,
  "d"=1, "e"=4, "f"=1
)
p$rateParamList
# get object summary
summary(p)
```

---

setRateParamList.HKY   Set the rate parameters

Description

Set the rate parameters.

The rate parameters are: Alpha, Beta.

Usage

```r
## S3 method for class 'HKY'
setRateParamList(this, value, ...)
```
Arguments

    this       An HKY object.
    value      A list containing the rate parameters.
    ...        Not used.

Value

The list of rate parameters (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see HKY.

Examples

```r
# create HKY object
p <- HKY()
# set/get rate parameters
setRateParamList(p, list(
  "Alpha" = 1,
  "Beta" = 0.5
))
getRateParamList(p)
# set/get rate parameters via virtual field
p$rateParamList <- list(
  "Alpha" = 1,
  "Beta" = 3
)
p$rateParamList
# get object summary
summary(p)
```

Description

Set the rate parameters.

The rate parameters are: Alpha, Beta.

Usage

```r
## S3 method for class 'K80'
setRateParamList(this, value, ...)
```
Arguments

this A K80 object.
value A list containing the rate parameters.
... Not used.

Value

The list of rate parameters (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see K80.

Examples

```r
# create K80 object
p <- K80()
# set/get rate parameters
setRateParamList(p, list(
  "Alpha"=1,
  "Beta"=0.5
))
getRateParamList(p)
# set/get rate parameters via virtual field
p$rateParamList <- list(
  "Alpha"=1,
  "Beta"=3
)
p$rateParamList
# get object summary
summary(p)
```

**Description**

Set the rate parameters.

The rate parameters are: Alpha, Beta, Gamma.

**Usage**

```
## S3 method for class 'K80'
setRateParamList(this, value, ...)
```
Arguments

- **this** A K81 object.
- **value** A list containing the rate parameters.
- **...** Not used.

Value

The list of rate parameters (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `K81`.

Examples

```r
# create K81 object
p<-K81()
# set/get rate parameters
setRateParamList(p,list(
  "Alpha"=1,
  "Beta"=0.5,
  "Gamma"=2
))
getRateParamList(p)
# set/get rate parameters via virtual field
p$rateParamList<-list(
  "Alpha"=1,
  "Beta"=3,
  "Gamma"=2
)
p$率ateParamList
# get object summary
summary(p)
```

Description

Set the rate parameters.

The rate parameters are: Alpha, Beta.
Usage

```r
## S3 method for class 'T92'
setRateParamList(this, value, ...)
```

Arguments

- **this**: A T92 object.
- **value**: A list containing the rate parameters.
- **...**: Not used.

Value

The list of rate parameters (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `T92`.

Examples

```r
# create a T92 object
p <- T92()
# set/get rate parameters
setRateParamList(p, list(
  "Alpha"=1,
  "Beta"=0.5
))
getRateParamList(p)
# set/get rate parameters via virtual field
p$rateParamList <- list(
  "Alpha"=1,
  "Beta"=3
)
p$rateParamList
# get object summary
summary(p)
```
setRateParamList.TN93  Set the rate parameters

Description
Set the rate parameters.
The rate parameters are: Alpha1, Alpha2, Beta.

Usage
### S3 method for class 'TN93'
setRateParamList(this, value, ...)

Arguments
- **this**: A TN93 object.
- **value**: A list containing the rate parameters.
- **...**: Not used.

Value
The list of rate parameters (invisible).

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see TN93.

Examples
# create TN93 object
p<-TN93()
# set/get rate parameters
setRateParamList(p,list(  
  "Alpha1"=1,  
  "Alpha2"=2,  
  "Beta"=0.5  
))
getRateParamList(p)
# set/get rate parameters via virtual field
p$rateParamList<-list(  
  "Alpha1"=1,  
  "Alpha2"=1,  
  "Beta"=3  
)
Forbidden action: setting the identifier of the root node for a PhyloSim object.

Usage

## S3 method for class 'PhyloSim'
setRootNode(this, value, ...)

Arguments

- **this**: An object.
- **value**: Not used.
- **...**: Not used.

Value

Throws an error.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see PhyloSim.
setRootSeq.PhyloSim  

Set the root sequence for a PhyloSim object

Description

Set the root sequence for a PhyloSim object.

The root sequence will be used as a starting point for the simulation. The phylo object must be set before trying to set the root sequence object.

Usage

## S3 method for class 'PhyloSim'
setRootSeq(this, value, ...)

Arguments

- **this**: A PhyloSim object.
- **value**: A valid Sequence object.
- **...**: Not used.

Value

The root Sequence object if successful, FALSE otherwise.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see PhyloSim. Sequence Process

Examples

# create some objects
sim<-PhyloSim(phylo=rcoal(3));
seq<-NucleotideSequence(string="ATGCC");
# set/get root sequence
setRootSeq(sim, seq);
getRootSeq(sim, seq);
# set/get root sequence via virtual field
sim$rootSeq<-BinarySequence(string="111000111000");
sim$rootSeq;
setScale.BrownianInsertor

\textit{Set scale parameter}

\section*{Description}
Set scale parameter.

\section*{Usage}
\begin{verbatim}
## S3 method for class 'BrownianInsertor'
setScale(this, value, ...)
\end{verbatim}

\section*{Arguments}
\begin{itemize}
\item \textbf{this} A BrownianInsertor object.
\item \textbf{value} A numeric vector of length one.
\item \textbf{...} Not used.
\end{itemize}

\section*{Value}
value (invisible).

\section*{Author(s)}
Botond Sipos, Gregory Jordan

\section*{See Also}
For more information see \texttt{BrownianInsertor}.

\section*{Examples}
\begin{verbatim}
# create a BrownianInsertor object
p<-BrownianInsertor(scale=0.002)
# set/get scale parameter
setScale(p,0.1)
getAddress(p)
# set/get scale parameter via virtual field
p$scale<-0.1
p$scale
\end{verbatim}
setScaledMatrix.QMatrix

Forbidden action: setting the scaled rate matrix for a QMatrix object

Description
Forbidden action: setting the scaled rate matrix for a QMatrix object.

Usage

## S3 method for class 'QMatrix'
setScaledMatrix(this, value, ...)

Arguments

this        An object.
value       Not used.
...         Not used.

Value
Throws an error.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see QMatrix.

setSequence.Site

Assotiate a Sequence object with a Site object

Description
Assotiate a Sequence object with a Site object.

Usage

## S3 method for class 'Site'
setSequence(this, new.seq, ...)


Arguments

this A Site object.
new.seq A valid Sequence object.
... Not used.

Value

The Sequence object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Site.

Examples

# create a site object
s <- Site()
# get associated Sequence object
s$sequence
# set associated Sequence object
setSequence(s, Sequence())
s$sequence

Forbidden action: setting the Sequence objects associated with the nodes of a phylo object aggregated by a PhyloSim object

Description

Forbidden action: setting the Sequence objects associated with the nodes of a phylo object aggregated by a PhyloSim object.

Usage

## S3 method for class 'PhyloSim'
setSequences(this, value, ...)

Arguments

this An object.
value Not used.
... Not used.
**Value**

Throws an error.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see *PhyloSim*.

---

**setSite.Event**  
*Associate an Event object with a Site object*

**Description**

Associate an Event object with a Site object.

**Usage**

```r
## S3 method for class 'Event'
setSite(this, new.site, ...)
```

**Arguments**

- `this`  
  An Event object.

- `new.site`  
  A valid Site object.

- `...`  
  Not used.

**Value**

The new associated Site object (invisible).

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see *Event*. 
Examples

```r
# create an Event object
e <- Event()
# create some Site objects
s1 <- Site(alphabet = NucleotideAlphabet(), state = "A")
s2 <- clone(s1); s2$state <- "T"
# associate s1 with e
setSite(e, s1)
e$site
# associate s2 with e via virtual field
e$site <- s2
e$site
```

```r
setSiteSpecificParamIds.Process

Forbidden action: setting the parameter identifiers of the site specific parameters from a Process object
```

Description

Forbidden action: setting the parameter identifiers of the site specific parameters from a Process object.

Usage

```r
## S3 method for class 'Process'
setSiteSpecificParamIds(this, value, ...)
```

Arguments

- `this`: An object.
- `value`: Not used.
- `...`: Not used.

Value

Throws an error.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Process`.  
setSiteSpecificParamList.Process
Forbidden action: setting the site specific parameter list for a Process object

Description
Forbidden action: setting the site specific parameter list for a Process object. Use .addSiteSpecificParameter to add new site specific parameters when implementing new processes.

Usage
## S3 method for class 'Process'
setSiteSpecificParamList(this, value, ...)

Arguments
- this: An object.
- value: Not used.
- ...: Not used.

Value
Throws an error.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see Process.

setSize.Alphabet
Forbidden action: setting the symbol set size of an Alphabet object

Description
Forbidden action: setting the symbol set size of an Alphabet object.

Usage
## S3 method for class 'Alphabet'
setSize(this, value, ...)

Description
Set the sizes of the proposed deletions.
The provided numeric vector is rounded.

Usage
```r
## S3 method for class 'DiscreteDeletor'
setSizes(this, value, ...)
```

Arguments
- **this**: A DiscreteDeletor object.
- **value**: A numeric vector.
- ... Not used.

Value
A vector of integers (invisible).

Author(s)
Botond Sipos, Gregory Jordan
See Also

For more information see `DiscreteDeletor`.

Examples

```r
# create a DiscreteDeletor object
d <- DiscreteDeletor(rate = 1)
# set deletion sizes
setSizes(d, c(1, 2, 3))
# get deletion sizes
getSizes(d)
# set/get sizes via virtual field
d$sizes <- 1:10
d$sizes
```

---

`setSizes.DiscreteInsertor`

*Set the sizes of the proposed insertions*

Description

Set the sizes of the proposed insertions.

The provided numeric vector is rounded.

Usage

```r
## S3 method for class 'DiscreteInsertor'
setSizes(this, value, ...)
```

Arguments

- `this` A DiscreteInsertor object.
- `value` A numeric vector.
- `...` Not used.

Value

A vector of integers (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `DiscreteInsertor`. 
Examples

```r
# create a DiscreteInsertor object
i<-DiscreteInsertor(rate=1)
# set insertion sizes
setSizes(i,c(1,2,3))
# get insertion sizes
getSizes(i)
# set/get sizes via virtual field
i$sizes<-1:10
i$sizes
```

---

**setState.Site**  
*Set the state of a Site object*

---

**Description**

Set the state of a Site object.

**Usage**

```r
## S3 method for class 'Site'
setState(this, new.state, ...)
```

**Arguments**

- `this`  
  A Site object.

- `new.state`  
  A character vector of length one, containing a symbol belonging to the attached Alphabet object.

- `...`  
  Not used.

**Value**

Returns the new state (invisible).

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `Site`. 
**setStates.Sequence**

*Set the states for a set of Site objects aggregated by a Sequence object*

**Description**

Set the states for a set of Site objects aggregated by a Sequence object. The value vector is recycled, which is useful when creating repeated patterns.

**Usage**

```r
## S3 method for class 'Sequence'
setStates(this, value, index, ...)
```

**Arguments**

- `this` A Sequence object.
- `value` A character vector containing the states (recycled if shorter than the index vector). The states must be compatible with the corresponding Alphabet object.
- `index` An integer vector specifying a set of positions. It is set to 1:seq$length if omitted.
- `...` Not used.

**Value**

The Sequence object (invisible).

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `Sequence`. 

**Examples**

```r
# create a Site object with an Alphabet object attached
s <- Site(alphabet = Alphabet(symbols = c(0, 1)), state = 1);
# set a new state
setState(s, "0")
# get state via virtual field
s$state
# set a new state via virtual field
s$state <- 1
s$state
```
Examples

```r
# create a sequence object of length 10
s<-Sequence(alphabets=list(NucleotideAlphabet()),length=10)
# set the states in some ranges
setStates(s,c("A","T","A"),index=1:5)
setStates(s,c("G","C"),index=6:10)
# display sequence
s
# set the states for the whole Sequence object
setStates(s,c("A","T","T","A"))
# set states via virtual field
s$states<-c("A","T")
s
```

### setString.SetSequence

**Forbidden action: setting the string representation of a Sequence object**

---

**Description**

Forbidden action: setting the string representation of a Sequence object.

**Usage**

```r
## S3 method for class 'Sequence'
setString(this, value, ...)
```

**Arguments**

- **this**: An object.
- **value**: Not used.
- **...**: Not used.

**Value**

Throws an error.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `Sequence`. 
Description
Forbidden action: setting the symbol length for an Alphabet object.

Usage
## S3 method for class 'Alphabet'
setSymbolLength(this, value, ...)

Arguments
- **this**: An Alphabet object.
- **value**: Not used.
- **...**: Not used.

Value
Throws an error.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see Alphabet.

Description
Specify a new symbol set for an Alphabet object.

Usage
## S3 method for class 'Alphabet'
setSymbols(this, set, ...)

Arguments

this An Alphabet object.
set The vector containing the new symbols set, automatically converted into a character vector.
... Not used.

Value

The new symbol set as a character vector.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Alphabet.

Examples

# create a new alphabet object
a<-Alphabet()
a
# specify a new symbol set
setSymbols(a,c(0,1))
a
# the same, but now use the virtual field
a$symbols<-c("A","T","G","C")
a

setTableId.CodonAlphabet

Forbidden action: setting the genetic code id

Description

Forbidden action: setting the genetic code id. Use the table.id constructor argument to set the genetic code.

Usage

## S3 method for class 'CodonAlphabet'
setTableId(this, value, ...)

Forbidden action: setting the genetic code id
Arguments

this An object.
value Not used.
... Not used.

Value

Throws an error.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see CodonAlphabet.

---

**setTemplateSeq-GeneralInsertor**

*Set the template sequence object*

Description

Set the template sequence object. The template sequence object is used by the default `generateBy` function to generate insert sequences.

Usage

```r
## S3 method for class 'GeneralInsertor'
setTemplateSeq(this, value, ...)
```

Arguments

- **this**: A GeneralInsertor object.
- **value**: A Sequence object.
- **...**: Not used.

Value

The Sequence object.

Author(s)

Botond Sipos, Gregory Jordan
See Also

For more information see \texttt{GeneralInsertor}.

Examples

\begin{verbatim}
# create a GeneralInsertor object
i<-GeneralInsertor(
  rate=0.5,
  propose.by=function(process){sample(c(5:10),1)}, # inserts between 5 and 10
  template.seq=NucleotideSequence(string="AAAAAA")
)
# get template sequence
getTemplateSeq(i)
# get template sequence via virtual field
i$templateSeq
# set template sequence
setTemplateSeq(i, NucleotideSequence(string="C"));
# generate insert
generateInsert(i)
# set template sequence via virtual field
i$templateSeq<-NucleotideSequence(string="G")
# generate insert
generateInsert(i)
\end{verbatim}

---

\texttt{setTheta.T92} \hspace{1cm} \textit{Set the GC content}

Description

Set the GC content.

Usage

\texttt{## S3 method for class 'T92'
setTheta(this, value, ...)}

Arguments

\begin{verbatim}
  this \hspace{1cm} A T92 object.
  value \hspace{1cm} A numeric vector of length one.
  ... \hspace{1cm} Not used.
\end{verbatim}

Value

The new value of theta (invisible).
setTipLabels.PhyloSim

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see T92.

Examples

```r
# construct a T92 object
p <- T92()
# set/get GC content
setTheta(p, 0.6)
getTheta(p)
# set/get GC content via virtual field
p$theta <- 0.3
p$theta
# get object summary
summary(p)
```

Description

Forbidden action: setting the tip labels for a phylo object aggregated by a PhyloSim object.

Usage

```r
## S3 method for class 'PhyloSim'
setTipLabels(this, value, ...)
```

Arguments

- this
  - An object.
- value
  - Not used.
- ... 
  - Not used.

Value

Throws an error.

Author(s)
Botond Sipos, Gregory Jordan
See Also

For more information see PhyloSim.

---

**setTips.PhyloSim**

*Forbidden action: setting the node identifiers of the tip nodes for a PhyloSim object*

### Description

Forbidden action: setting the node identifiers of the tip nodes for a PhyloSim object.

### Usage

```r
## S3 method for class 'PhyloSim'
setTips(this, value, ...)
```

### Arguments

- `this`: An object.
- `value`: Not used.
- `...`: Not used.

### Value

Throws an error.

### Author(s)

Botond Sipos, Gregory Jordan

### See Also

For more information see PhyloSim.
setToleranceMargin.FastFieldDeletor

*Set the tolerance margin*

**Description**

Set the tolerance margin.

**Usage**

```r
## S3 method for class 'FastFieldDeletor'
setToleranceMargin(this, value, ...)
```

**Arguments**

- `this` A FastFieldDeletor object.
- `value` A numeric vector of length one.
- `...` Not used.

**Value**

value (invisible).

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `FastFieldDeletor`.

**Examples**

```r
# create a geometric FastFieldDeletor
p <- FastFieldDeletor()
# set/get tolerance margin
setToleranceMargin(p, 0.8)
getToleranceMargin(p)
# set/get tolerance margin via virtual field
p$toleranceMargin <- 0.75
p$toleranceMargin
```
setTotalRate.Site

Forbidden action: setting the total active event rate for a Site object

Description
Forbidden action: setting the total active event rate for a Site object.

Usage
## S3 method for class 'Site'
setTotalRate(this, value, ...)

Arguments
- this: An object.
- value: Not used.
- ...: Not used.

Value
Throws an error.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see Site.

setTotalRates.Sequence

Forbidden action: setting the list of total site rates for a Sequence object

Description
Forbidden action: setting the list of total site rates for a Sequence object.

Usage
## S3 method for class 'Sequence'
setTotalRates(this, value, ...)

Arguments

- **this**: An object.
- **value**: Not used.
- **...**: Not used.

Value

Throws an error.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see [Sequence](#).

---

## setTransTable.CodonAlphabet

*Forbidden action: setting the genetic code table for a CodonAlphabet object*

Description

Forbidden action: setting the genetic code table for a CodonAlphabet object. Use the `table.id` constructor argument to set the genetic code.

Usage

```r
## S3 method for class 'CodonAlphabet'
setTransTable(this, value, ...)
```

Arguments

- **this**: An object.
- **value**: Not used.
- **...**: Not used.

Value

Throws an error.

Author(s)

Botond Sipos, Gregory Jordan
setTreeLength.PhyloSim

See Also

For more information see CodonAlphabet.

Description

Forbidden action: setting the tree length for a PhyloSim object.

Usage

```r
## S3 method for class 'PhyloSim'
setTreeLength(this, value, ...)
```

Arguments

- **this**: An object.
- **value**: Not used.
- **...**: Not used.

Value

Throws an error.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see PhyloSim.
Description

Set Alphabet object type.

Usage

## S3 method for class 'Alphabet'
setParameter(this, new_type, ...)

Arguments

- `this`: An Alphabet object.
- `new_type`: A character vector of length one.
- `...`: Not used.

Value

The new type (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Alphabet`.

Examples

```r
# create an alphabet object
a <- Alphabet(symbols=c(1,2,3))
# set a new type
setParameter(a,"MyAlphabet")
a$type
# set type via virtual field
a$type <- "MorphChars"
```
**Description**

Forbidden action: setting the type of a BrownianInsertor object.
The type can be set only from the type constructor argument and cannot be changed later.

**Usage**

```r
## S3 method for class 'BrownianInsertor'
setType(this, value, ...)
```

**Arguments**

- `this`: An object.
- `value`: Not used.
- `...`: Not used.

**Value**

Throws an error.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `BrownianInsertor`.

---

**Description**

Forbidden action: setting the type of a FastFieldDeletor object.
The type can be set only through the type constructor argument.

**Usage**

```r
## S3 method for class 'FastFieldDeletor'
setType(this, value, ...)
```
Arguments

this         An object.
value       Not used.
...         Not used.

Value

Throws an error.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `FastFieldDeletor`.

Forbidden action: setting the list of unique Alphabet objects attached to the Site object aggregated by a Sequence object

## Description

Forbidden action: setting the list of unique Alphabet objects attached to the Site object aggregated by a Sequence object.

## Usage

```ruby
## S3 method for class 'Sequence'
setUniqueAlphabets(this, value, ...)
```

Arguments

this         An object.
value       Not used.
...         Not used.

Value

Throws an error.

Author(s)

Botond Sipos, Gregory Jordan
**See Also**

For more information see [Sequence](#).

---

**Description**

Forbidden action: setting the list of unique Process instances attached to the sites of a Sequence object.

**Usage**

```r
## S3 method for class 'Sequence'
setUniqueProcesses(this, value, ...)
```

**Arguments**

- **this**: An object.
- **value**: Not used.
- **...**: Not used.

**Value**

Throws an error.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see [Sequence](#).
setWriteProtected.Alphabet

Set the write protection field for an object

Description
Set the write protection field for an object. Write protected objects cannot be modified through get/set methods and virtual fields.

Usage
```r
## S3 method for class 'Alphabet'
setWriteProtected(this, value, ...)
```

Arguments
- **this**: An object.
- **value**: A logical vector of size one.
- **...**: Not used.

Value
Invisible TRUE or FALSE.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see Alphabet.

Examples
```r
# create an object
o<-Alphabet()
# toggle write protection
setWriteProtected(o,TRUE)
# check write protection
o$writeProtected
# set write protection via virtual field
o$writeProtected<-FALSE
o$writeProtected
```
setWriteProtected.Event

Set the write protection field for an object

Description

Set the write protection field for an object. Write protected objects cannot be modified through get/set methods and virtual fields.

Usage

```r
## S3 method for class 'Event'
setWriteProtected(this, value, ...)
```

Arguments

- `this`: An object.
- `value`: A logical vector of size one.
- `...`: Not used.

Value

Invisible TRUE or FALSE.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Event`.

Examples

```r
# create an object
o <- Event()
# toggle write protection
setWriteProtected(o, TRUE)
# check write protection
o$writeProtected
# set write protection via virtual field
o$writeProtected <- FALSE
o$writeProtected
```
setWriteProtected.Process

Set the write protection field for an object

Description
Set the write protection field for an object. Write protected objects cannot be modified through get/set methods and virtual fields.

Usage
```r
## S3 method for class 'Process'
setWriteProtected(this, value, ...)
```

Arguments
- `this`: A Process object.
- `value`: A logical vector of size one.
- `...`: Not used.

Value
Invisible TRUE or FALSE.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see `Process`.

Examples
```r
# create an object
o <- Process()
# toggle write protection
setWriteProtected(o, TRUE)
# check write protection
o$writeProtected
# set write protection via virtual field
o$writeProtected <- FALSE
o$writeProtected
```
setWriteProtected.QMatrix

Set the write protection field for a QMatrix object

Description

Set the write protection field for a QMatrix object.

QMatrix objects do not have a write protection flag of their own, but they use the one from the associated Process object. Write protected objects cannot be modified through get/set methods and virtual fields.

Usage

### S3 method for class 'QMatrix'

setWriteProtected(this, value, ...)

Arguments

- **this**: A QMatrix object.
- **value**: A logical vector of size one.
- **...**: Not used.

Value

TRUE or FALSE

Author(s)

Botond Sipos, Gregory Jordan

See Also

setWriteProtected.Process

setWriteProtected.Sequence

Set the write protection field for an object

Description

Set the write protection field for an object. Write protected objects cannot be modified through get/set methods and virtual fields.
Usage

```r
## S3 method for class 'Sequence'
setWriteProtected(this, value, ...)
```

Arguments

- `this` An object.
- `value` A logical vector of size one.
- `...` Not used.

Value

Invisible TRUE or FALSE.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Sequence`.

Examples

```r
# create an object
o <- Sequence()
# toggle write protection
setWriteProtected(o, TRUE)
# check write protection
o$writeProtected
# set write protection via virtual field
o$writeProtected <- FALSE
o$writeProtected
```

---

**Simulate.PhyloSim**

Run a simulation according to a PhyloSim object

Description

Run a simulation according to a PhyloSim object.

The phylo object and the root sequence must be set before attempting to run a simulation. Also the bigRate of the root sequence must not be NA or zero, so at least one sane Process object must be attached to the root sequence object.
Usage

```r
## S3 method for class 'PhyloSim'
Simulate(this, quiet=FALSE, ...)
```

Arguments

- `this`: A PhyloSim object.
- `quiet`: TRUE or FALSE (default).
- `...`: Not used.

Value

The PhyloSim object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `phylosim`.

Examples

```r
# Create a PhyloSim object.
# Provide the phylo object
# and the root sequence.
sim<-PhyloSim(
  name="TinySim",
  phylo=rcoal(3),
  root.seq=NucleotideSequence(string="ATGC",processes=list(list(JC69()))))

# Run the simulation
Simulate(sim);
# Print the resulting sequences
sim$sequences
# Print the resulting alignment
sim$alignment
```
Description

This is the class representing a site. Site objects can have one associated Alphabet object and one or more Process objects that act on their states. The associated Process and Site objects must have associated Alphabet objects with the same symbols set, or at least one of the Alphabet objects should inherit from the class AnyAlphabet.

Site objects store the site-process-specific parameters of the attached Process objects. A site-process-specific parameter is a list containing: the identifier, the name, the value and type of the parameter. For example the ubiquitous rate multiplier site-process-specific parameter looks like `list(id="rate.multiplier",name="Rate multiplier",value=1,type="numeric")`.

Templates for site-process-specific parameters and their default values are stored in the Process objects and copied into the Site object when the process is attached.

Site objects have fields for associated ancestral Site objects and Sequence objects.

Package:

Class Site

```
Object
~~
~~+-PSRoot
~~~~~~|
~~~~~~~~-Site

Directly known subclasses:

public static class Site
extends PSRoot

Usage

Site(state=NA, alphabet=NA, ancestral=NA, processes=NA, sequence=NA, ...)

Arguments

state A symbol belonging to the specified alphabet.
alphabet An alphabet object.
ancestral The ancestral Site object.
processes A list of Process objects.
sequence The Sequence object to which the Site object belongs.
... Not used.

Fields and Methods

Methods:
Site

as.character  -
attachProcess  -
checkConsistency  -
detachProcess  -
flagTotalRate  -
getAlphabet  -
getAncestral  -
getEvents  -
getProcesses  -
getSequence  -
getState  -
getTotalRate  -
is  -
isAttached  -
setAlphabet  -
setAncestral  -
setEvents  -
setProcesses  -
setSequence  -
setState  -
setTotalRate  -
summary  -

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is.na, is, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
[. [, [[<-, $, $<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)
Botond Sipos, Gregory Jordan

See Also
Alphabet, Process, Event

Examples

# create a site object
site<-Site()
# print the character representation (state)
print(site);
# get a summary
summary(site);
# create a site with a nucleotide alphabet
site <- Site(state = "A", alphabet = NucleotideAlphabet());

# set site state
site$state <- "G"

# manipulate the associated Sequence object
site$sequence <- Sequence()

# attach a substitution process
site$processes <- list(JC69())

# add one more substitution process
attachProcess(site, K80())

# get a list of active Event objects
site$events

# get a summary of the Site object
summary(site);

---

### `stringLength`

*Returns the string length of the character representation of an object*

---

### Description

Returns the string length of the character representation of an object. More useful as a static method.

### Usage

```r
## Default S3 method:
stringLength(this, ...)  
```

### Arguments

- `this`: An object.

- `...`: Not used.

### Value

An integer vector of length one.

### Author(s)

Botond Sipos, Gregory Jordan

### See Also

For more information see `PSRoot`. 
Examples

```r
x <- "character representation"
# get the string length of x
stringLength(x)
```

### stringLengthVector

*Returns the string lengths of the character representations of a collection of objects*

**Description**

Returns the string lengths of the character representations of a collection of objects. More useful as a static method.

**Usage**

```r
## Default S3 method:
stringLengthVector(this, ...)
```

**Arguments**

- `this`: An array or a list of object.
- `...`: Not used.

**Value**

An integer vector with the corresponding lengths.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see [PSRoot](#).

**Examples**

```r
x <- c("character representation", "other string");
# get the string length of x
stringLengthVector(x)
```
summarize.Alphabet  

### Description

Summarize the properties of an object.

### Usage

```r
## S3 method for class 'Alphabet'
summary(object, ...)
```

### Arguments

- **object**: An object
- **...**: Not used.

### Value

Returns a PSRootSummary object.

### Author(s)

Botond Sipos, Gregory Jordan

### See Also

For more information see `Alphabet`.

### Examples

```r
# create an object
a <- NucleotideAlphabet()
# get a summary
summary(a)
```
summarize.AminoAcidSubst

**Description**

Summarize the properties of an object.

**Usage**

```r
## S3 method for class 'AminoAcidSubst'
summary(object, ...)
```

**Arguments**

- `object` An object
- `...` Not used.

**Value**

Returns a PSRootSummary object.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `AminoAcidSubst`.

**Examples**

```r
# create an object
a <- AminoAcidSubst()
# get a summary
summary(a)
```
Summary

Summarize the properties of an object.

Usage

```r
## S3 method for class 'BinarySubst'
summary(object, ...)
```

Arguments

- `object`: An object
- `...`: Not used.

Value

Returns a PSRootSummary object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `BinarySubst`.

Examples

```r
# create an object
p <- BinarySubst(rate=0.25, name="Binary", rate.list=list("0->1"=2,"1->0"=1))
# get a summary
summary(p)
```
**Description**

Summarize the properties of an object.

**Usage**

```r
## S3 method for class 'BrownianInsertor'
summary(object, ...)
```

**Arguments**

- `object` : An object
- `...` : Not used.

**Value**

Returns a PSRootSummary object.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `BrownianInsertor`.

**Examples**

```r
# create an object
p <- BrownianInsertor(
  type = "discrete",
  scale = 0.05,
  sizes = 1:4,
  probs = c(3/6, 1/6, 1/6, 1/6),
  rate = 0.05
)

# get a summary
summary(p)
```
**summary.CodonAlphabet**  
*Summarize the properties of an object*

**Description**

Summarize the properties of an object.

**Usage**

```r
## S3 method for class 'CodonAlphabet'
summary(object, ...)
```

**Arguments**

- `object`: An object
- `...`: Not used.

**Value**

Returns a PSRootSummary object.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `CodonAlphabet`.

**Examples**

```r
# create an object
a <- CodonAlphabet()
# get a summary
summary(a)
```
### Description

Summarize the properties of an object.

### Usage

```r
## S3 method for class 'CodonSubst'
summary(object, ...)
```

### Arguments

- `object` An object
- `...` Not used.

### Value

Returns a PSRootSummary object.

### Author(s)

Botond Sipos, Gregory Jordan

### See Also

For more information see `CodonSubst`.

### Examples

```r
# create an object
a<-CodonSubst()
# get a summary
summary(a)
```
**Summary**

ContinuousDeletor

*Summarize the properties of an object*

**Description**

Summarize the properties of an object.

**Usage**

```r
## S3 method for class 'ContinuousDeletor'
summary(object, ...)
```

**Arguments**

- `object`  An object
- `...` Not used.

**Value**

Returns a PSRootSummary object.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `ContinuousDeletor`.

**Examples**

```r
# create an object
a <- ContinuousDeletor(rate=1, dist=expression(rnorm(1, mean=5, sd=3)), max.length=10)

# get a summary
summary(a)
```
**summary.ContinuousInsertor**

*Summarize the properties of an object*

**Description**

Summarize the properties of an object.

**Usage**

```r
## S3 method for class 'ContinuousInsertor'
summary(object, ...)
```

**Arguments**

- `object`  
  An object
- `...`  
  Not used.

**Value**

Returns a PSRootSummary object.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `ContinuousInsertor`.

**Examples**

```r
# create an object
a <- ContinuousInsertor(rate=1, dist=expression(rnorm(1, mean=5, sd=2)), max.length=7)
# get a summary
summary(a)
```
summary.DiscreteDeletor

*Summarize the properties of an object*

**Description**

Summarize the properties of an object.

**Usage**

```r
## S3 method for class 'DiscreteDeletor'
summary(object, ...)
```

**Arguments**

- `object` An object
- `...` Not used.

**Value**

Returns a PSRootSummary object.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `DiscreteDeleter`.

**Examples**

```r
# create an object
a <- DiscreteDeletor(rate=1, sizes=c(1,2), probs=c(1/2,1/2))
# get a summary
summary(a)
```
Summary

DiscreteInsertor

Description

Summarize the properties of an object.

Usage

```r
## S3 method for class 'DiscreteInsertor'
summary(object, ...)
```

Arguments

- `object`:
  - An object
- `...`:
  - Not used.

Value

Returns a PSRootSummary object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `DiscreteInsertor`.

Examples

```r
# create an object
a <- DiscreteInsertor(rate=1, sizes=c(1,2), probs=c(1/2,1/2))
# get a summary
summary(a)
```
summary.Event  

**Summarize the properties of an object**

---

**Description**

Summarize the properties of an object.

**Usage**

```r
## S3 method for class 'Event'
summary(object, ...)
```

**Arguments**

- **object**: An object
- **...**: Not used.

**Value**

Returns a PSRootSummary object.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `Event`.

**Examples**

```r
# create an object
e <- Event()
# get a summary
summary(e)
```
Summary the properties of an object

Description

Summarize the properties of an object.

Usage

```r
## S3 method for class 'F81'
summary(object, ...)
```

Arguments

- `object`: An object
- `...`: Not used.

Value

Returns a PSRootSummary object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `F81`.

Examples

```r
# create an object
a <- NucleotideAlphabet()
# get a summary
summary(a)
```
Description

Summarize the properties of an object.

Usage

```r
## S3 method for class 'F84'
summary(object, ...)
```

Arguments

- `object` An object
- `...` Not used.

Value

Returns a PSRootSummary object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `F84`.

Examples

```r
# create an object
da <- F84(rate.params=list("Kappa"=3))
# get a summary
summary(da)
```
**Summary**

`summary.FastFieldDeletor`

*Summarize the properties of an object*

**Description**

Summarize the properties of an object.

**Usage**

```r
## S3 method for class 'FastFieldDeletor'
summary(object, ...)
```

**Arguments**

- `object` An object
- `...` Not used.

**Value**

Returns a PSRootSummary object.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `FastFieldDeletor`.

**Examples**

```r
# create an object
a <- FastFieldDeletor()
# get a summary
summary(a)
```
summarize. GeneralDeleter

Summarize the properties of an object

Description

Summarize the properties of an object.

Usage

```r
## S3 method for class 'GeneralDeleter'
summary(object, ...)
```

Arguments

- `object`: An object
- `...`: Not used.

Value

Returns a PSRootSummary object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `GeneralDeleter`.

Examples

```r
# create an object
a <- GeneralDeleter(rate=1, name="Del Bosque")
# get a summary
summary(a)
```
Summary

**GeneralInDel**

Summarize the properties of an object.

**Description**

Summarize the properties of an object.

**Usage**

```r
## S3 method for class 'GeneralInDel'
summary(object, ...)
```

**Arguments**

- `object` An object
- `...` Not used.

**Value**

Returns a PSRootSummary object.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `GeneralInDel`.

**Examples**

```r
# create an object
a<-GeneralInDel(rate=1,propose.by=function(process){sample(c(1,2,3),1)})
# get a summary
summary(a)
```
Summary

Summarize the properties of an object.

Description

Summarize the properties of an object.

Usage

```r
## S3 method for class 'GeneralInsertor'
summary(object, ...)
```

Arguments

- `object`: An object
- `...`: Not used.

Value

Returns a PSRootSummary object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `GeneralInsertor`.

Examples

```r
# create an object
a <- GeneralInsertor(rate=1)
# get a summary
summary(a)
```
**summarize**.GeneralSubstitution

*Summary of the properties of an object*

---

**Description**

Summarize the properties of an object.

**Usage**

```r
## S3 method for class 'GeneralSubstitution'
summary(object, ...)
```

**Arguments**

- `object` An object
- `...` Not used.

**Value**

Returns a PSRootSummary object.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `GeneralSubstitution`.

**Examples**

```r
# create an object
a <- GeneralSubstitution(alphabet=BinaryAlphabet(), rate.list=list("0->1"=1, "1->0"=2))
# get a summary
summary(a)
```
**Summary**

Summarize the properties of an object.

**Usage**

```r
## S3 method for class 'GTR'
summary(object, ...)
```

**Arguments**

- `object`: An object
- `...`: Not used.

**Value**

Returns a PSRootSummary object.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `GTR`.

**Examples**

```r
# create an object
a <- GTR()
# get a summary
summary(a)
```
Summary

Summarize the properties of an object.

Usage

```r
## S3 method for class 'GY94'
summary(object, ...)
```

Arguments

- `object`: An object
- `...`: Not used.

Value

Returns a PSRootSummary object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see GY94.

Examples

```r
# create an object
a <- GY94(kappa=2)
# get a summary
summary(a)
```
Summary of the properties of an object

Description

Summarize the properties of an object.

Usage

```r
## S3 method for class 'HKY'
summary(object, ...)
```

Arguments

- `object`: An object
- `...`: Not used.

Value

Returns a PSRootSummary object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see HKY.

Examples

```r
# create an object
a <- NucleotideAlphabet()
# get a summary
summary(a)
```
Description
Summarize the properties of an object.

Usage

## S3 method for class 'JC69'
summary(object, ...)

Arguments

<table>
<thead>
<tr>
<th>object</th>
<th>An object</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>Not used</td>
</tr>
</tbody>
</table>

Value
Returns a PSRootSummary object.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see JC69.

Examples

```r
# create an object
p <- JC69()
# get a summary
summary(p)
```

---

Description
Summarize the properties of an object.

Usage

## S3 method for class 'K80'
summary(object, ...)

```r
# create an object
p <- K80()
# get a summary
summary(p)
```
Arguments

object          An object
...             Not used.

Value

Returns a PSRootSummary object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see \texttt{K80}.

Examples

\begin{verbatim}
# create an object
a<-NucleotideAlphabet()
# get a summary
summary(a)
\end{verbatim}

\begin{verbatim}
summary.K81            \textit{Summarize the properties of an object}
\end{verbatim}

Description

Summarize the properties of an object.

Usage

\begin{verbatim}
## S3 method for class 'K81'
summary(object, ...)
\end{verbatim}

Arguments

object          An object
...             Not used.

Value

Returns a PSRootSummary object.

Author(s)

Botond Sipos, Gregory Jordan
See Also

For more information see \texttt{k81}.

Examples

```r
# create an object
a <- NucleotideAlphabet()
# get a summary
summary(a)
```

\begin{verbatim}
summary.PhyloSim  Summarize the properties of an object
\end{verbatim}

Description

Summarize the properties of an object.

Usage

\begin{verbatim}
## S3 method for class 'PhyloSim'
summary(object, ...)
\end{verbatim}

Arguments

- \texttt{object} \hspace{1cm} An object
- \texttt{...} \hspace{1cm} Not used.

Value

Returns a PSRootSummary object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see \texttt{PhyloSim}.
**Examples**

```r
# Create a PhyloSim object.
# Provide the phylo object
# and the root sequence.
sim <- PhyloSim(
  name = "TinySim",
  phylo = coal(3),
  root.seq = NucleotideSequence(string = "ATG", processes = list(JC69())))

  # get a summary
  summary(sim)
```

---

**summary.Process**

**Summarize the properties of an object**

**Description**

Summarize the properties of an object.

**Usage**

```r
## S3 method for class 'Process'
summary(object, ...)
```

**Arguments**

- `object`:
  - An object
- `...`:
  - Not used.

**Value**

Returns a PSRootSummary object.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `Process`.

**Examples**

```r
# create an object
a <- Nucleotide Alphabet()
# get a summary
summary(a)
```
**summary.PSRoot**

*Summarize the properties of an object*

---

**Description**

Summarize the properties of an object.

**Usage**

```r
## S3 method for class 'PSRoot'
summary(object, ...)
```

**Arguments**

- **object**: An object
- **...**: Not used.

**Value**

Returns a PSRootSummary object.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `PSRoot`.

**Examples**

```r
# create an object
a <- PSRoot()
# get a summary
summary(a)
```
### Description

Summarize the properties of an object.

### Usage

```r
define S3 method for class 'QMMatrix'
summary(object, ...)
```

### Arguments

- **object**: An object
- **...**: Not used.

### Value

Returns a PSRootSummary object.

### Author(s)

Botond Sipos, Gregory Jordan

### See Also

For more information see QMatrix.

### Examples

```r
# create an object
a <- QMatrix(alphabet=BinaryAlphabet(), rate.list=list("0->1"=1,"1->0"=3))
# get a summary
summary(a)
```
**summary.Sequence**

*Summarize the properties of an object*

**Description**

Summarize the properties of an object.

**Usage**

```r
## S3 method for class 'Sequence'
summary(object, ...)
```

**Arguments**

- `object`: An object.
- `...`: Not used.

**Value**

Returns a PSRootSummary object.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `Sequence`.

**Examples**

```r
# create an object
a<-Sequence()
# get a summary
summary(a)
```
Description

Summarize the properties of an object.

Usage

```r
## S3 method for class 'Site'
summary(object, ...)
```

Arguments

- `object` An object
- `...` Not used.

Value

Returns a PSRootSummary object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Site`.

Examples

```r
# create an object
o<-Site()
# get a summary
summary(o)
```
Summarize the properties of an object

Description

Summarize the properties of an object.

Usage

```r
## S3 method for class 'T92'
summary(object, ...)
```

Arguments

- **object**: An object
- **...**: Not used.

Value

Returns a PSRootSummary object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `T92`.

Examples

```r
# create an object
a <- T92(theta=0.8)
# get a summary
summary(a)
```
Summary

Summarize the properties of an object.

Description

Summarize the properties of an object.

Usage

```r
## S3 method for class 'TN93'
summary(object, ...)
```

Arguments

- **object**: An object
- **...**: Not used.

Value

Returns a PSRootSummary object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `TN93`.

Examples

```r
# create an object
a <- NucleotideAlphabet()
# get a summary
summary(a)
```
**Summary**

Summarize the properties of an object.

**Description**

Summarize the properties of an object.

**Usage**

```r
## S3 method for class 'UNREST'
summary(object, ...)
```

**Arguments**

- `object`: An object
- `...`: Not used.

**Value**

Returns a PSRootSummary object.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `UNREST`.

**Examples**

```r
# create an object
p<-UNREST(rate.list=list(
  "T->C"=1, "T->A"=2, "T->G"=3, "C->T"=4, "C->A"=1,
  "C->G"=2, "A->T"=3, "A->C"=4, "A->G"=1, "G->T"=2,
  "G->C"=3, "G->A"=4)
))

# get a summary
summary(p)
```
The T92 class

Description

This class implements the T92 GTR-submodel. The rate parameters are the following: "Alpha", "Beta","Gamma". The theta virtual field stores the GC content parameter.

Package:

Class T92

Object
```
~~~~~~~~|   
~~~~~~~~--PSRoot
~~~~~~~~~~|   
~~~~~~~~~~--Process
~~~~~~~~~~~~|   
~~~~~~~~~~~~--GeneralSubstitution
~~~~~~~~~~~~~~|   
~~~~~~~~~~~~~~--UNREST
~~~~~~~~~~~~~~~|   
~~~~~~~~~~~~~~~--GTR
~~~~~~~~~~~~~~~~|   
~~~~~~~~~~~~~~~~--T92
```

Directly known subclasses:

public static class T92
extends GTR

Usage

T92(name="Anonymous", rate.params=list(Alpha = 1, Beta = 1), theta=0.5, ...)

Arguments

<table>
<thead>
<tr>
<th>name</th>
<th>Object name.</th>
</tr>
</thead>
<tbody>
<tr>
<td>rate.params</td>
<td>Rate parameters.</td>
</tr>
<tr>
<td>theta</td>
<td>GC content (0.5 by default).</td>
</tr>
<tr>
<td>...</td>
<td>Not used.</td>
</tr>
</tbody>
</table>
Fields and Methods

Methods:

- checkConsistency
- getRateParam
- getRateParamList
- getTheta
- setRateParam
- setRateParamList
- setTheta
- summary

Methods inherited from GTR:
- checkConsistency
- getBaseFreqs
- getRateParam
- getRateParamList
- setBaseFreqs
- setRateParam
- setRateParamList
- summary

Methods inherited from UNREST:
- checkConsistency
- summary

Methods inherited from GeneralSubstitution:
- as.character
- checkConsistency
- clone
- getAlphabet
- getEquDist
- getEventRate
- getEventRateAtSite
- getEventsAtSite
- getQMatrix
- getRate
- getRateList
- hasUndefinedRate
- is
- plot
- rescaleQMatrix
- sampleState
- setAlphabet
- setEquDist
- setQMatrix
- setRate
- setRateList
- summary

Methods inherited from Process:
- !=
- ==
- as.character
- checkConsistency
- clone
- getAlphabet
- getEventsAtSite
- getId
- getName
- getParameterAtSite
- getSiteSpecificParamIds
- getSiteSpecificParamList
- getWriteProtected
- hasSiteSpecificParameter
- hasUndefinedRate
- is
- setName
- setParameterAtSite
- setSiteSpecificParamIds
- setSiteSpecificParamList
- setWriteProtected
- summary

Methods inherited from PSRoot:
- checkConsistency
- enableVirtual
- getComments
- getMethodsList
- globalConsistencyCheck
- intersect.list
- is
- is.na
- ll
- my.all.equal
- plot
- setComments
- setMethodsList
- summary
- virtualAssignmentForbidden

Methods inherited from Object:
- [<-
- $<-
- as.character
- attach
- attachLocally
- clearCache
- clearLookupCache
- clone
- detach
- equals
- extend
- finalize
- gc
- getEnvironment
- getFieldModifier
- getFieldModifiers
- getFields
- getInstantiationTime
- getStaticInstance
- hasField
- hashCode
- ll
- load
- objectSize
- print
- registerFinalizer
- save

Author(s)

Botond Sipos, Gregory Jordan

References

See Also

GTR UNREST GeneralSubstitution HKY

Examples

```r
# create substitution process object
p <- TN93(rate.params=list("Alpha"=10,"Beta"=2),theta=0.8)
# get a summary
summary(p)
# display a bubble plot
plot(p)

# The following code demonstrates how to use
# the process in a simulation.

# create a sequence, attach process p
s <- NucleotideSequence(length=20,processes=list(list(p)))
# sample states
sampleStates(s)
# make the first five positions invariable
setRateMultipliers(s,p,0,1:5)
# get rate multipliers
getRateMultipliers(s,p)
# create a simulation object
sim <- PhyloSim(root.seq=s,phylo=rcoal(2))
# run simulation
Simulate(sim)
# print alignment
sim$alignment
```

TN93

*The TN93 class*

Description

This class implements the Tamura-Nei 93 GTR-submodel.
The rate parameters are the following: "Alpha1", "Alpha2","Beta". Package:
Class TN93

Object

~~|  
~~---PSRoot
~~~~~~~~|
~~~~~~~~---Process
~~~~~~~~~~~~|
~~~~~~~~~~~~~~~~~~~~~~~~---GeneralSubstitution
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|
Directly known subclasses:

public static class TN93
extends GTR

Usage

TN93(
    name="Anonymous", rate.params=list(Alpha1 = 1, Alpha2 = 1, Beta = 1),
    base.freqs=c(0.25, 0.25, 0.25, 0.25), ...
)

Arguments

name            Object name.
rate.params     Rate parameters.
base.freqs      Base frequency parameters.
...             Not used.

Fields and Methods

Methods:

    checkConsistency -
    getBaseFreqs -
    getRateParam -
    getRateParamList -
    setBaseFreqs -
    setRateParam -
    setRateParamList -
    summary -

Methods inherited from GTR:
checkConsistency, getBaseFreqs, getRateParam, getRateParamList, setBaseFreqs, setRateParam, setRateParamList, summary

Methods inherited from UNREST:
checkConsistency, summary
Methods inherited from GeneralSubstitution:
as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is.na, is, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
[[], [[<-, $, $<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstanceationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)
Botond Sipos, Gregory Jordan

References

See Also
GTR HKY UNREST GeneralSubstitution

Examples
# create substitution process object
p<-'TN93(rate.params=list( "Alpha1"=4,"Alpha2"=3,"Beta"=2),
base.freqs=c(2,2,1,3)/9)
# get a summary
summary(p)
# display a bubble plot
plot(p)

# The following code demonstrates how to use
# the process in a simulation.

# create a sequence, attach process p
s<-NucleotideSequence(length=20,processes=list(list(p)))
# sample states
sampleStates(s)
Description

Translate a CodonSequence object.

Usage

```r
## S3 method for class 'CodonSequence'
Translate(this, ...)
```

Arguments

- `this`: A CodonSequence object
- `...`: Not used.

Value

The translation as an Amino AcidSequence object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `CodonSequence`.

Examples

```r
# create a CodonSequence object
s <- CodonSequence(string = "ATCTTTGGAATGGGCCCTCCGA")
# get the translation as an Amino AcidSequence object
as <- Translate(s)
```

# make the first five positions invariable
setRateMultipliers(s, p, 0, 1:5)
# get rate multipliers
getRateMultipliers(s, p)
# create a simulation object
sim <- PhyloSim(root.seq = s, phylo = rcoal(2))
# run simulation
Simulate(sim)
# print alignment
sim$alignment
translateCodon.CodonAlphabet

Translate a codon

Description

Translate a codon.

Usage

```r
## S3 method for class 'CodonAlphabet'
translateCodon(this, codon, ...)
```

Arguments

- `this`: A CodonAlphabet object.
- `codon`: The codon to be translated.
- `...`: Not used.

Value

A character vector containing an amino acid IUPAC code.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `CodonAlphabet`.

Examples

```r
# create a CodonAlphabet object
a <- CodonAlphabet()
# translate a codon
translateCodon(a,"AGG")
```
Undocumented object (PhyloSim package)

Description

Undocumented object (PhyloSim package).
See the corresponding specific methods if applicable.

Usage

### S3 method for class 'PhyloSim'

Undocumented(...)

Arguments

... Not used.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `PhyloSim`.

UNREST The UNREST class

Description

This class implements the UNRESTricted nucleotide substitution model. UNREST objects are basically a GeneralSubstitution process acting on a nucleotide alphabet.

Package:

Class UNREST

Object

|↓
|---PSRoot
|-----Process
|--------GeneralSubstitution
|---------UNREST
Directly known subclasses:
F81, F84, GTR, HKY, JC69, K80, K81, T92, TN93

public static class UNREST
extends GeneralSubstitution

Usage
UNREST(name="Anonymous", rate.list=NA, equ.dist=NA, ...)

Arguments
name Object name.
rate.list A list of unscaled rates (see setRateList.GeneralSubstitution).
equ.dist Equilibrium distribution.
... Additional arguments.

Fields and Methods
Methods:

checkConsistency -
summary -

Methods inherited from GeneralSubstitution:
as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is.na, is, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
[. [<- $, $<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save
virtualAssignmentForbidden.PSRoot

Throws an error message informing the user about forbidden action on virtual a field
Description

Throws an error message informing the user about forbidden action on virtual a field.

Usage

```r
## S3 method for class 'PSRoot'
virtualAssignmentForbidden(this, ...)
```

Arguments

- `this`: A PSRoot object.
- `...`: Not used.

Value

Throws an error.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `PSRoot`.

---

**WAG**

*The WAG empirical amino acid substitution model*

---

Description

Package:

```
```

```
Class WAG

```

```
Object
```

```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
Directly known subclasses:

public static class WAG
extends AminoAcidSubst

Usage

WAG(equ.dist=NA, ...)

Arguments

equ.dist Equilibrium distribution.

... Not used.

Fields and Methods

Methods:
No methods defined.

Methods inherited from AminoAcidSubst:
builtFromPAML, checkConsistency, newAAMatrix, setEquDist, summary

Methods inherited from GeneralSubstitution:
as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is.na, is, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
[., [[<-, $<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFields, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)
Botond Sipos, Gregory Jordan
References


See Also

AminoAcidSubst GeneralSubstitution UNREST

Examples

# create substitution model object
p<-WAG()
# get object summary
summary(p)
# display a bubble plot
plot(p)

    # The following code demonstrates how to use
    # the process in a simulation.

# create a sequence, attach process p
s<-AminoAcidSequence(length=10,processes=list(list(p)) )
# sample states
sampleStates(s)
    # make the first three positions invariable
setRateMultipliers(s,p,0,1:3)
    # get rate multipliers
getRateMultipliers(s,p)
    # create a simulation object
sim<-PhyloSim(root.seq=s,phylo=rcoul(2))
    # run simulation
Simulate(sim)
    # print alignment
sim$alignment
Index

AminoAcidSequence, 18, 359, 378, 393
AminoAcidSubst, 19, 21, 45, 48, 89, 90, 129, 313, 315, 323, 324, 327–330, 332, 334, 337, 350, 352, 359, 375, 378, 521, 562, 563
AminoAcidSubst.buildFromPAML
(buildFromPAML.AminoAcidSubst), 45
AminoAcidSubst.checkConsistency
(checkConsistency.AminoAcidSubst), 48
AminoAcidSubst.newAAAMatrix
(newAAAMatrix.AminoAcidSubst), 337
AminoAcidSubst.summary
(summary.AminoAcidSubst), 521
AnyAlphabet, 14, 22, 359, 378
ape, 355, 383, 385
areSynonymous (Undocumented.PhyloSim), 559
areSynonymous,CodonAlphabet-method
(areSynonymous.CodonAlphabet), 23
areSynonymous.CodonAlphabet, 23
as.character, Alphabet-method
(as.character.Alphabet), 24
as.character, Event-method
(as.character.Event), 25
as.character, GeneralSubstitution-method
(as.character.GeneralSubstitution), 26
as.character, PhyloSim-method
(as.character.PhyloSim), 27
as.character, Process-method
(as.character.Process), 28
as.character, QMatrix-method
(as.character.QMatrix), 29
as.character, Sequence-method
(as.character.Sequence), 30

all (Undocumented.PhyloSim), 559
Alphabet, 12, 14, 16, 17, 22, 24, 36, 47, 76, 248, 255, 256, 267, 272, 285, 293, 308, 338, 359, 378, 491, 496, 497, 506, 510, 520
Alphabet.== (==.Alphabet), 12
Alphabet.as.character
(as.character.Alphabet), 24
Alphabet.checkConsistency
(checkConsistency.Alphabet), 47
Alphabet.getSize (getSize.Alphabet), 248
Alphabet.getSymbolLength
(getSymbolLength.Alphabet), 255
Alphabet.getSymbols
(getSymbols.Alphabet), 256
Alphabet.getType (getType.Alphabet), 267
Alphabet.WRITEPROTECTED
(getWriteProtected.Alphabet), 272
Alphabet.hasSymbols
(hasSymbols.Alphabet), 284
Alphabet.isEmpty (isEmpty.Alphabet), 308
Alphabet.setSize (setSize.Alphabet), 490
Alphabet.setSymbolLength
(setSymbolLength.Alphabet), 496
Alphabet.setSymbols
(setSymbols.Alphabet), 496
Alphabet.setType (setType.Alphabet), 506
Alphabet.WRITEPROTECTED
(setWriteProtected.Alphabet), 510
Alphabet.summary (summary.Alphabet), 520
AminoAcidAlphabet, 14, 16, 359, 378
as.character, Site-method
   (as.character.Site), 31
as.character.Alphabet, 24
as.character.Event, 25
as.character.GeneralSubstitution, 26
as.character.PhyloSim, 27
as.character.Process, 28
as.character.QMatrix, 29
as.character.Sequence, 30
as.character.Site, 31
attachHookToNode
   (Undocumented.PhyloSim), 559
attachHookToNode.PhyloSim-method
   (attachHookToNode.PhyloSim), 32
attachHookToNode.PhyloSim, 32
attachProcess (Undocumented.PhyloSim),
   559
attachProcess.Sequence-method
   (attachProcess.Sequence), 33
attachProcess.Site-method
   (attachProcess.Site), 34
attachProcess.Sequence, 33
attachProcess.Site, 34
attachSeqToNode
   (Undocumented.PhyloSim), 559
attachSeqToNode.PhyloSim-method
   (attachSeqToNode.PhyloSim), 35
attachSeqToNode.PhyloSim, 35
BinaryAlphabet, 14, 36, 359, 378
BinarySequence, 37, 359, 378, 393
BinarySubst, 39, 49, 129, 359, 375, 378, 522
BinarySubst.checkConsistency
   (checkConsistency.BinarySubst),
   48
BinarySubst.summary
   (summary.BinarySubst), 522
BrownianInsertor, 41, 44, 49, 100, 123, 126,
   239, 268, 359, 375, 378, 485, 507,
   523
BrownianInsertor.BrownianPath
   (BrownianPath.BrownianInsertor),
   44
BrownianInsertor.checkConsistency
   (checkConsistency.BrownianInsertor),
   49
BrownianInsertor.getScale
   (getScale.BrownianInsertor),
   239
BrownianInsertor.getType
   (getType.BrownianInsertor), 268
BrownianInsertor.setScale
   (setScale.BrownianInsertor),
   485
BrownianInsertor.setType
   (setType.BrownianInsertor), 507
BrownianInsertor.summary
   (summary.BrownianInsertor), 523
BrownianPath (Undocumented.PhyloSim),
   559
BrownianPath.BrownianInsertor-method
   (BrownianPath.BrownianInsertor),
   44
BrownianPath.BrownianInsertor, 44
buildFromPAML (Undocumented.PhyloSim),
   559
buildFromPAML.AminoAcidSubst-method
   (buildFromPAML.AminoAcidSubst),
   45
buildFromPAML.CodonSubst-method
   (buildFromPAML.CodonSubst), 46
buildFromPAML.AminoAcidSubst, 45
buildFromPAML.CodonSubst, 46
checkConsistency
   (Undocumented.PhyloSim), 559
checkConsistency.Alphabet-method
   (checkConsistency.Alphabet), 47
checkConsistency.AminoAcidSubst-method
   (checkConsistency.AminoAcidSubst),
   48
checkConsistency.BinarySubst-method
   (checkConsistency.BinarySubst),
   48
checkConsistency.BrownianInsertor-method
   (checkConsistency.BrownianInsertor),
   49
checkConsistency.CodonAlphabet-method
   (checkConsistency.CodonAlphabet),
   50
checkConsistency.CodonSequence-method
   (checkConsistency.CodonSequence),
   50
checkConsistency.CodonSubst-method
   (checkConsistency.CodonSubst),
   51
checkConsistency.CodonUNREST-method
   (checkConsistency.CodonUNREST),
checkConsistency, ContinuousDeleter-method (checkConsistency.ContinuousDeleter), 52
checkConsistency, ContinuousInsertor-method (checkConsistency.ContinuousInsertor), 53
checkConsistency, DiscreteDeleter-method (checkConsistency.DiscreteDeleter), 54
checkConsistency, DiscreteInsertor-method (checkConsistency.DiscreteInsertor), 54
checkConsistency, Event-method (checkConsistency.Event), 55
checkConsistency, F81-method (checkConsistency.F81), 56
checkConsistency, F84-method (checkConsistency.F84), 56
checkConsistency, FastFieldDeleter-method (checkConsistency.FastFieldDeleter), 57
checkConsistency, GeneralDeleter-method (checkConsistency.GeneralDeleter), 58
checkConsistency, GeneralInDel-method (checkConsistency.GeneralInDel), 58
checkConsistency, GeneralInsertor-method (checkConsistency.GeneralInsertor), 59
checkConsistency, GeneralSubstitution-method (checkConsistency.GeneralSubstitution), 60
checkConsistency, GTR-method (checkConsistency.GTR), 60
checkConsistency, GY94-method (checkConsistency.GY94), 61
checkConsistency, HKY-method (checkConsistency.HKY), 62
checkConsistency, JC69-method (checkConsistency.JC69), 62
checkConsistency, K80-method (checkConsistency.K80), 63
checkConsistency, K81-method (checkConsistency.K81), 64
checkConsistency, PhyloSim-method (checkConsistency.PhyloSim), 64
checkConsistency, Process-method (checkConsistency.Process), 65
checkConsistency, PSRoot-method (checkConsistency.PSRoot), 66
checkConsistency, PSRootSummary-method (checkConsistency.PSRootSummary), 66
checkConsistency, QMatrix-method (checkConsistency.QMatrix), 67
checkConsistency, Sequence-method (checkConsistency.Sequence), 68
checkConsistency, Site-method (checkConsistency.Site), 68
checkConsistency, T92-method (checkConsistency.T92), 69
checkConsistency, TN93-method (checkConsistency.TN93), 70
checkConsistency, UNREST-method (checkConsistency.UNREST), 70
checkConsistency, Alphabet, 47
checkConsistency, AminoAcidSubst, 48
checkConsistency, BinarySubst, 48
checkConsistency, BrownianInsertor, 49
checkConsistency, CodonAlphabet, 50
checkConsistency, CodonSequence, 50
checkConsistency, CodonSubst, 51
checkConsistency, CodonUNREST, 52
checkConsistency, ContinuousDeleter, 52
checkConsistency, ContinuousInsertor, 53
checkConsistency, DiscreteDeleter, 54
checkConsistency, DiscreteInsertor, 54
checkConsistency, Event, 55
checkConsistency, F81, 56
checkConsistency, F84, 56
checkConsistency, FastFieldDeleter, 57
checkConsistency, GeneralDeleter, 58
checkConsistency, GeneralInDel, 58
checkConsistency, GeneralInsertor, 59
checkConsistency, GeneralSubstitution, 60
checkConsistency, GTR, 60
checkConsistency, GY94, 61
checkConsistency, HKY, 62
checkConsistency, JC69, 62
checkConsistency, K80, 63
checkConsistency, K81, 64
checkConsistency, PhyloSim, 64
checkConsistency.Process, 65
checkConsistency.PSRoot, 66
checkConsistency.PSRootSummary, 66
checkConsistency.QMatrix, 67
checkConsistency.Sequence, 68
checkConsistency.Site, 68
checkConsistency.TN93, 70
checkConsistency.UNREST, 70
clearStates (Undocumented.PhylSim), 559
clearStates.Sequence-method
(clearStates.Sequence), 71
clearStates.Sequence, 71
codeAlphabet
(buildFromPAML, 46, 51, 79, 81, 102–105, 338, 359, 525)
codeAlphabet.arity
(arity, 23)
codeAlphabet.checkConsistency
(checkConsistency, 50)
codeAlphabet.getRelativeTo
(getRelativeTo, 257)
codeAlphabet.isStartCodon
(isStartCodon, 309)
codeAlphabet.isStopCodon
(isStopCodon, 310)
codeAlphabet.setRelative
(setRelative, 497)
codeAlphabet.setTransTable
(setTransTable, 504)
codonAlphabet.summary
(summary, 524)
codonAlphabet.translateCodon
(translateCodon, 558)
codonSequence
(checkConsistency, 51, 77, 79, 201, 342, 343, 345, 346, 348, 349, 359, 378, 393, 447, 557)
codonSequence.checkConsistency
(checkConsistency, 50)
codonSequence.getOmeegas
(getOmeegas, 201)
codonSequence.getOmegasHist
(getOmegasHist, 341)
codonSequence.omegaVarM0
(omegaVarM0, 342)
codonSequence.omegaVarM1
(omegaVarM1, 344)
codonSequence.omegaVarM2
(omegaVarM2, 345)
codonSequence.omegaVarM3
(omegaVarM3, 347)
codonSequence.omegaVarM4
(omegaVarM4, 348)
codonSequence.setOmeegas
(setOmegas, 446)
codonSequence.Translate
(Translate, 557)
codonSubst.
builtFromPAML
(builtFromPAML, 46)
codonSubst.checkConsistency
(checkConsistency, 51)
codonSubst.newMatrix
(newMatrix, 337)
codonSubst.summary
(summary, 525)
codonUNREST
(arity, 81, 129, 281, 359, 375, 378)
codonUNREST.checkConsistency
(checkConsistency, 52)
continuousDeleter
(getDist, 52)
ContinuousDeleter.getMaxLength
(getMaxLength.ContinuousDeleter), 190
ContinuousDeleter.plot
(plot.ContinuousDeleter), 361
ContinuousDeleter.setDist
(setDist.ContinuousDeleter), 418
ContinuousDeleter.setMaxLength
(setMaxLength.ContinuousDeleter), 437
ContinuousDeleter.summary
(summary.ContinuousDeleter), 526
ContinuousInsertor, 53, 85, 123, 126, 159, 191, 359, 363, 375, 378, 420, 438, 527
ContinuousInsertor.checkConsistency
(checkConsistency.ContinuousInsertor), 53
ContinuousInsertor.getDist
(getDist.ContinuousInsertor), 159
ContinuousInsertor.getMaxLength
(getMaxLength.ContinuousInsertor), 191
ContinuousInsertor.plot
(plot.ContinuousInsertor), 362
ContinuousInsertor.setDist
(setDist.ContinuousInsertor), 419
ContinuousInsertor.setMaxLength
(setMaxLength.ContinuousInsertor), 438
ContinuousInsertor.summary
(summary.ContinuousInsertor), 527
copySubSequence
(Undocumented.PhyloSim), 559
copySubSequence.Sequence-method
(copySubSequence.Sequence), 88
copySubSequence.Sequence, 88
cpREV, 20, 89, 129, 359, 375, 378
Debug (Undocumented.PhyloSim), 559
Debug.PhyloSim-method (Debug.PhyloSim), 91
Debug.PhyloSim, 91
deleteSubSequence
(Undocumented.PhyloSim), 559
deleteSubSequence.Sequence-method
deleteSubSequence.Sequence, 92
detachHookFromNode
(Undocumented.PhyloSim), 559
detachHookFromNode.PhyloSim-method
detachHookFromNode.PhyloSim, 93
detachHookFromNode.PhyloSim, 93
detachProcess (Undocumented.PhyloSim), 559
detachProcess.Sequence-method
detachProcess.Sequence, 94
detachProcess.Site-method
detachProcess.Site, 95
detachProcess.Sequence, 94
detachProcess.Site, 95
detachSeqFromNode
(Undocumented.PhyloSim), 559
detachSeqFromNode.PhyloSim-method
detachSeqFromNode.PhyloSim, 96
detachSeqFromNode.PhyloSim, 96
DiscreteDeleter, 54, 97, 116, 120, 123, 207, 249, 359, 364, 375, 378, 452, 492, 528
DiscreteDeleter.checkConsistency
(checkConsistency.DiscreteDeleter), 54
DiscreteDeleter.getProbs
(getProbs.DiscreteDeleter), 207
DiscreteDeleter.getSizes
(getSizes.DiscreteDeleter), 249
DiscreteDeleter.plot
(plot.DiscreteDeleter), 363
DiscreteDeleter.setProbs
(setProbs.DiscreteDeleter), 451
DiscreteDeleter.setSizes
(setSizes.DiscreteDeleter), 491
DiscreteDeleter.summary
(summary.DiscreteDeleter), 528
DiscreteInsertor, 42, 55, 99, 123, 126, 208, 250, 359, 365, 375, 378, 453, 492,
DiscreteInsertor.checkConsistency
(checkConsistency.DiscreteInsertor), 54
DiscreteInsertor.getProbs
(getProbs.DiscreteInsertor), 208
DiscreteInsertor.getSizes
(getSizes.DiscreteInsertor), 250
DiscreteInsertor.plot
(plot.DiscreteInsertor), 364
DiscreteInsertor.setProbs
(setProbs.DiscreteInsertor), 452
DiscreteInsertor.setSizes
(setSizes.DiscreteInsertor), 492
DiscreteInsertor.summary
(summary.DiscreteInsertor), 529
ECMrest, 80, 102
ECMunrest, 80, 104
enableVirtual (Undocumented.PhyloSim), 559
ableVirtual, PSRoot-method
(enableVirtual.PSRoot), 106
equal (Undocumented.PhyloSim), 559
Event.as.character
(as.character.Event), 25
Event.checkConsistency
(checkConsistency.Event), 55
Event.getHandler (getHandler.Event), 175
Event.getName (getName.Event), 193
Event.getPosition (getPosition.Event), 206
Event.getProcess (getProcess.Event), 209
Event.getRate (getRate.Event), 215
Event.getSite (getSite.Event), 244
Event.setWriteProtected
(setWriteProtected.Event), 273
Event.Perform (Perform.Event), 354
Event.setHandler (setHandler.Event), 424
Event.setName (setName.Event), 439
Event.setPosition (setPosition.Event), 450
Event.setProcess (setProcess.Event), 453
Event.setRate (setRate.Event), 459
Event.setSite (setSite.Event), 488
Event.setWriteProtected
(setWriteProtected.Event), 511
Event.summary (summary.Event), 530
exportStatTree (Undocumented.PhyloSim), 559
exportStatTree, PhyloSim-method
(exportStatTree.PhyloSim), 109
exportStatTree.PhyloSim, 109, 355
F81, 56, 111, 129, 144, 222, 229, 278, 359, 375, 378, 407, 468, 475, 531, 560
F81.checkConsistency
(checkConsistency.F81), 56
F81.getBaseFreqs (getBaseFreqs.F81), 144
F81.getRateParam (getRateParam.F81), 222
F81.getRateParamList
(getRateParamList.F81), 229
F81.setBaseFreqs (setBaseFreqs.F81), 407
F81.setRateParam (setRateParam.F81), 467
F81.setRateParamList
(setRateParamList.F81), 475
F81.summary (summary.F81), 531
F84.checkConsistency
(checkConsistency.F84), 56
F84.getBaseFreqs (getBaseFreqs.F84), 145
F84.getKappa (getKappa.F84), 182
F84.getRateParam (getRateParam.F84), 222
F84.getRateParamList
(getRateParamList.F84), 230
F84.setBaseFreqs (setBaseFreqs.F84), 408
F84.setKappa (setKappa.F84), 430
F84.setRateParam (setRateParam.F84), 468
F84.setRateParamList
(setRateParamList.F84), 475
F84.summary (summary.F84), 532
FastFieldDeleter, 57, 116, 118, 120, 123, 185, 186, 262, 269, 359, 375, 378, 433, 434, 502, 508, 533
FastFieldDeleter.checkConsistency
(checkConsistency.FastFieldDeleter), 57
(setAcceptWin.GeneralInsertor), 399
GeneralInsertor.setGenerateBy
(setGenerateBy.GeneralInsertor), 423
GeneralInsertor.setInsertHook
(setInsertHook.GeneralInsertor), 427
GeneralInsertor.setTemplateSeq
(setTemplateSeq.GeneralInsertor), 498
GeneralInsertor.summary
(summary.GeneralInsertor), 536
GeneralSubstitution.as.character
(as.character.GeneralSubstitution), 26
GeneralSubstitution.checkConsistency
(checkConsistency.GeneralSubstitution), 60
GeneralSubstitution.clone
(clone.GeneralSubstitution), 72
GeneralSubstitution.getAlphabet
(getAlphabet.GeneralSubstitution), 137
GeneralSubstitution.getEquDist
(getEquDist.GeneralSubstitution), 162
GeneralSubstitution.getEventRate
(getEventRate.GeneralSubstitution), 163
GeneralSubstitution.getEventRateAtSite
(getEventRateAtSite.GeneralSubstitution), 165
GeneralSubstitution.getEventsAtSite
(getEventsAtSite.GeneralSubstitution), 171
GeneralSubstitution.getQMatrix
(getQMatrix.GeneralSubstitution), 214
GeneralSubstitution.getRate
(getRate.GeneralSubstitution), 217
GeneralSubstitution.getRateList
(getRateList.GeneralSubstitution), 219
GeneralSubstitution.hasUndefinedRate
(hasUndefinedRate.GeneralSubstitution), 286
GeneralSubstitution.plot
(plot.GeneralSubstitution), 365
GeneralSubstitution.rescaleQMatrix
(rescaleQMatrix.GeneralSubstitution), 386
GeneralSubstitution.sampleState
(sampleState.GeneralSubstitution), 388
GeneralSubstitution.setAlphabet
(setAlphabet.GeneralSubstitution), 400
GeneralSubstitution.setEquDist
(setEquDist.GeneralSubstitution), 421
GeneralSubstitution.setQMatrix
(setQMatrix.GeneralSubstitution), 458
GeneralSubstitution.setRate
(setRate.GeneralSubstitution), 461
GeneralSubstitution.setRateList
(setRateList.GeneralSubstitution), 464
GeneralSubstitution.summary
(summary.GeneralSubstitution), 537
GeneralSubstitution.generateInsert
(Undocumented.PhyloSim), 559
GeneralSubstitution.generateInsert, GeneralInsertor-method
(generateInsert.GeneralInsertor), 132
GeneralSubstitution.generateInsert, GeneralInsertor-131
getAcceptBy(Undocumented.PhyloSim), 559
getAcceptBy, GeneralInDel-method
(getAcceptBy, GeneralInDel), 133
generalInsert, GeneralInsertor
(getAcceptBy, GeneralInDel), 133
generalInsert, GeneralInsertor
(getAcceptBy, GeneralInDel), 133
getAcceptWin(Undocumented.PhyloSim), 559
getAcceptWin, GeneralInsertor-method  
   (getAcceptWin, GeneralInsertor), 134
getAcceptWin, GeneralInsertor, 134
getAlignment (Undocumented.PhylSim), 559
getAlignment, PhyloSim-method  
   (getAlignment, PhyloSim), 135
getAlignment, PhyloSim, 135
getAlignmentLength  
   (Undocumented.PhylSim), 559
getAlignmentLength, PhyloSim-method  
   (getAlignmentLength, PhyloSim), 136
getAlignmentLength, PhyloSim, 136
getAlignment (Undocumented.PhylSim), 559
getAlignment, GeneralSubstitution-method  
   (getAlignment, GeneralSubstitution), 137
getAlignment, Process-method  
   (getAlignment, Process), 138
getAlignment, QMatrix-method  
   (getAlignment, QMatrix), 139
getAlignment, Site-method  
   (getAlignment, Site), 140
getAlignment, GeneralSubstitution, 137
getAlignment, Process, 138
getAlignment, Site, 140
getAlphabets (Undocumented.PhylSim), 559
getAlphabets, Sequence-method  
   (getAlphabets, Sequence), 141
getAlphabets, Sequence, 141
getAncestral (Undocumented.PhylSim), 559
getAncestral, Sequence-method  
   (getAncestral, Sequence), 142
getAncestral, Site-method  
   (getAncestral, Site), 143
getAncestral, Sequence, 142
getAncestral, Site, 143
getBaseFreqs (Undocumented.PhylSim), 559
getBaseFreqs, F81-method  
   (getBaseFreqs, F81), 144
getBaseFreqs, F84-method  
   (getBaseFreqs, F84), 145
getBaseFreqs, GTR-method  
   (getBaseFreqs, GTR), 146
getBaseFreqs, HKY-method  
   (getBaseFreqs, HKY), 147
getBaseFreqs, K80-method  
   (getBaseFreqs, K80), 148
getBaseFreqs, K81-method  
   (getBaseFreqs, K81), 149
getBaseFreqs, TN93-method  
   (getBaseFreqs, TN93), 150
getBaseFreqs, F81, 144
getBaseFreqs, F84, 145
getBaseFreqs, GTR, 146
getBaseFreqs, HKY, 147
getBaseFreqs, K80, 148
getBaseFreqs, K81, 149
getBaseFreqs, TN93, 150
getBigRate (Undocumented.PhylSim), 559
getBigRate, Sequence-method  
   (getBigRate, Sequence), 151
getBigRate, Sequence, 151
getBranchEvents (Undocumented.PhylSim), 559
getBranchEvents, PhyloSim-method  
   (getBranchEvents, PhyloSim), 152
getBranchEvents, PhyloSim, 152, 367
getCodonFreqs (Undocumented.PhylSim), 559
getCodonFreqs, GY94-method  
   (getCodonFreqs, GY94), 153
getCodonFreqs, GY94, 153
getComments (Undocumented.PhylSim), 559
getComments, PSRoot-method  
   (getComments, PSRoot), 154
getComments, PSRoot, 154
getCumulativeRates (Undocumented.PhylSim), 559
getcumulativeRates, Sequence-method  
   (getCumulativeRates, Sequence), 155
getcumulativeRates, Sequence, 155
getcumulativeRatesFromRange (Undocumented.PhylSim), 559
getcumulativeRatesFromRange, Sequence-method  
   (getCumulativeRatesFromRange, Sequence), 156
getcumulativeRatesFromRange, Sequence, 156
getDeletionTolerance
  (Undocumented.PhylomSim), 559
getDeletionTolerance.Sequence-method
  (getDeletionTolerance.Sequence), 157
getDeletionTolerance.Sequence, 157
getDist (Undocumented.PhylomSim), 559
getDist.ContinuousDeleter-method
  (getDist.ContinuousDeleter), 158
getDist.ContinuousDeleter, 158
getDist.ContinuousInsertor-method
  (getDist.ContinuousInsertor), 159
getDist.ContinuousInsertor, 159
getEdge (Undocumented.PhylomSim), 559
getEdge.PhylomSim-method
  (getEdge.PhylomSim), 160
getEdge.PhylomSim, 160
getEdges (Undocumented.PhylomSim), 559
getEdges.PhylomSim-method
  (getEdges.PhylomSim), 161
getEdges.PhylomSim, 161
getEquDist (Undocumented.PhylomSim), 559
getEquDist.GeneralSubstitution-method
  (getEquDist.GeneralSubstitution), 162
getEquDist.GeneralSubstitution, 162
getEventRate (Undocumented.PhylomSim), 559
getEventRate.GeneralSubstitution-method
  (getEventRate.GeneralSubstitution), 163
getEventRate.QMatrix-method
  (getEventRate.QMatrix), 164
getEventRate.QMatrix, 164
getEventRateAtSite
  (Undocumented.PhylomSim), 559
getEventRateAtSite.GeneralSubstitution-method
  (getEventRateAtSite.GeneralSubstitution), 165
getEventRateAtSite, 165
getEvents (Undocumented.PhylomSim), 559
getEvents.Sequence-method
  (getEvents.Sequence), 166
getEvents.Site-method (getEvents.Site), 167
getEvents.Sequence, 166
getEvents.Site, 167
getEventsAtSite
  (Undocumented.PhylomSim), 559
getEventsAtSite.FastFieldDeleter-method
  (getEventsAtSite.FastFieldDeleter), 168
getEventsAtSite, 168
getEventsAtSite.GeneralDeleter-method
  (getEventsAtSite.GeneralDeleter), 169
getEventsAtSite, 169
getEventsAtSite.GeneralInsertor-method
  (getEventsAtSite.GeneralInsertor), 170
getEventsAtSite, 170
getEventsAtSite.GeneralSubstitution-method
  (getEventsAtSite.GeneralSubstitution), 171
getEventsAtSite, 171
getEventsAtSite.GY94-method
  (getEventsAtSite.GY94), 172
getEventsAtSite, 172
getEventsAtSite.Process-method
  (getEventsAtSite.Process), 173
getEventsAtSite, 173
getEventsAtSite.FastFieldDeleter, 168
getEventsAtSite, 168
getEventsAtSite.GeneralDeleter, 169
getEventsAtSite, 169
getEventsAtSite.GeneralInsertor, 170
getEventsAtSite, 170
getEventsAtSite.GeneralSubstitution, 171
getEventsAtSite, 171
getGenerateBy (Undocumented.PhylomSim), 559
getGenerateBy, GeneralInsertor-method
  (getGenerateBy, GeneralInsertor), 174
getGenerateBy, 174
getGenerateBy. GeneralInsertor, 174
getHandler (Undocumented.PhylomSim), 559
getHandler.Event-method
  (getHandler.Event), 175
getHandler, 175
getHandler.Event, 175
getId (Undocumented.PhylomSim), 559
getId.Process-method (getId.Process), 176
getId, 176
getId, 177
getId.Process, 177
getId.QMatrix-method (getId.QMatrix), 178
getId, 178
getId.QMatrix, 178
getId.Sequence-method (getId.Sequence), 179
getId, 179
getId.Sequence, 179
getId.PhylomSim, 176
getId.PhylomSim, 176
getidNprocess, 177
getidNqmatrix, 178
getidNsequence, 179
getinserthook (Undocumented.PhylomSim), 559
getinserthook, GeneralInsertor-method
   (getinserthook.GeneralInsertor), 180
getinserthook.GeneralInsertor, 180
getinsertontolerance
   (Undocumented.PhylomSim), 559
getinsertontolerance, Sequence-method
   (getinsertontolerance.Sequence), 181
getinsertontolerance.Sequence, 181
getkappa (Undocumented.PhylomSim), 559
getkappa, F84-method (getkappa.F84), 182
getakappa, GY94-method (getkappa.GY94), 183
getakappa.F84, 182
getakappa.GY94, 183
gelong (Undocumented.PhylomSim), 559
getelengt, Sequence-method
   (getelengt.Sequence), 184
getelengt.Sequence, 184
getlengthparam1
   (Undocumented.PhylomSim), 559
getlengthparam1, FastFieldDeleter-method
   (getlengthparam1.FastFieldDeleter), 185
getlengthparam1.FastFieldDeleter, 185
getlengthparam2
   (Undocumented.PhylomSim), 559
getlengthparam2, FastFieldDeleter-method
   (getlengthparam2.FastFieldDeleter), 186
getlengthparam2.FastFieldDeleter, 186
getlogfile (Undocumented.PhylomSim), 559
getlogfile, PhylomSim-method
   (getlogfile.PhylomSim), 187
getlogfile.PhylomSim, 187
getloglevel (Undocumented.PhylomSim), 559
getloglevel, PhylomSim-method
   (getloglevel.PhylomSim), 188
getloglevel.PhylomSim, 188
getmatrix (Undocumented.PhylomSim), 559
getmatrix, QMatrix-method
   (getmatrix.QMatrix), 189
getmaxlength (Undocumented.PhylomSim), 559
getmaxlength, ContinuousDeleter-method
   (getmaxlength.ContinuousDeleter), 190
getmaxlength, ContinuousInsertor-method
   (getmaxlength.ContinuousInsertor), 191
getmaxlength.ContinuousDeleter, 190
getmaxlength.ContinuousInsertor, 191
getmethodlist (Undocumented.PhylomSim), 559
getmethodlist, PRoot-method
   (getmethodlist.PRoot), 192
getmethodlist.PRoot, 192
getname, Event-method (getname.Event), 193
getname, PhylomSim-method
   (getname.PhylomSim), 194
getname, Process-method
   (getname.Process), 195
getname, QMatrix-method
   (getname.QMatrix), 196
getname, Sequence-method
   (getname.Sequence), 197
getname.Event, 193
gename.PhylomSim, 194
gename.Process, 195
gename.QMatrix, 196
gename.Sequence, 197
getnedges (Undocumented.PhylomSim), 559
getnedges, PhylomSim-method
   (getnedges.PhylomSim), 198
getnedges.PhylomSim, 198
getnodes (Undocumented.PhylomSim), 559
getnodes, PhylomSim-method
   (getnodes.PhylomSim), 199
getnodes.PhylomSim, 199
getntips (Undocumented.PhylomSim), 559
getntips, PhylomSim-method
   (getntips.PhylomSim), 200
getntips.PhylomSim, 200
getomegas (Undocumented.PhylomSim), 559
getomegas, CodonSequence-method
   (getomegas.CodonSequence), 201
getomegas.CodonSequence, 201
getomegas.Sequence
(getSiteSpecificParamList.Process), 247
gSiteSpecificParamList.Process, 247
gSiteSpecificParamList.Process, 247
gSize (Undocumented.PhyloSim), 559
gSize, Alphabet-method
  (getSize.Alphabet), 248
gSize.Alphabet, 248
gSizes (Undocumented.PhyloSim), 559
gSizes, DiscreteDeleter-method
  (getSize.DiscreteDeleter), 249
gSizes, DiscreteInsertor-method
  (getSize.DiscreteInsertor), 250
gSizes.DiscreteDeleter, 249
gSizes.DiscreteInsertor, 250
gState (Undocumented.PhyloSim), 559
gState, Site-method (getState.Site), 251
gState.Site, 251
gStates (Undocumented.PhyloSim), 559
gStates, Sequence-method
  (getStates.Sequence), 252
gStates.Sequence, 252
gString (Undocumented.PhyloSim), 559
gString, Sequence-method
  (getString.Sequence), 253
gString.Sequence, 253
gSymbolFreqs (Undocumented.PhyloSim), 559
gSymbolFreqs, Sequence-method
  (getSymbolFreqs.Sequence), 254
gSymbolFreqs.Sequence, 254
gSymbolLength
  (Undocumented.PhyloSim), 559
gSymbolLength, Alphabet-method
  (getSymbolLength.Alphabet), 255
gSymbolLength.Alphabet, 255
gSymbols (Undocumented.PhyloSim), 559
gSymbols, Alphabet-method
  (getSymbols.Alphabet), 256
gSymbols.Alphabet, 256
gTableId (Undocumented.PhyloSim), 559
gTableId, CodonAlphabet-method
  (getTableId.CodonAlphabet), 257
gTableId.CodonAlphabet, 257
gTargetState (Undocumented.PhyloSim), 559
gTemplateSeq (Undocumented.PhyloSim),
  559
gTemplateSeq, GeneralInsertor-method
  (getTemplateSeq.GeneralInsertor), 258
gTemplateSeq.GeneralInsertor, 258
gTheta (Undocumented.PhyloSim), 559
gTheta,T2-method (getTheta.T2), 259
gTheta.T2, 259
gTipLabels (Undocumented.PhyloSim), 559
gTipLabels, PhyloSim-method
  (getTipLabels.PhyloSim), 260
gTipLabels.PhyloSim, 260
gTips (Undocumented.PhyloSim), 559
gTips, PhyloSim-method
  (getTips.PhyloSim), 261
gTips.PhyloSim, 261
gToleranceMargin
  (Undocumented.PhyloSim), 559
gToleranceMargin, FastFieldDeleter-method
  (getToleranceMargin.FastFieldDeleter), 262
gToleranceMargin.FastFieldDeleter, 262
gTotalRate (Undocumented.PhyloSim), 559
gTotalRate, Site-method
  (getTotalRate.Site), 263
gTotalRate.Site, 263
gTotalRates (Undocumented.PhyloSim), 559
gTotalRates, Sequence-method
  (getTotalRates.Sequence), 264
gTotalRates.Sequence, 264
gTotalRatesFromRange
  (Undocumented.PhyloSim), 559
gTotalRatesFromRange, Sequence-method
  (getTotalRatesFromRange.Sequence), 264
gTotalRatesFromRange.Sequence, 264
gTransTable (Undocumented.PhyloSim), 559
gTransTable, CodonAlphabet-method
  (getTransTable.CodonAlphabet), 265
gTransTable.CodonAlphabet, 265
gTreeLength (Undocumented.PhyloSim), 559
getTreeLength, PhyloSim-method
  (getTreeLength.PhylomSim), 266
getTreeLength, PhyloSim, 266
getType (Undocumented, PhyloSim), 559
getType, Alphabet-method
  (getType.Alphabet), 267
getType, BrownianInsertor-method
  (getType.BrownianInsertor), 268
ggetType, FastFieldDeleter-method
  (getType.FastFieldDeleter), 269
ggetType, Alphabet, 267
ggetType, BrownianInsertor, 268
ggetType, FastFieldDeleter, 269
getUniqueAlphabets
  (Undocumented, PhyloSim), 559
getUniqueAlphabets, Sequence-method
  (getUniqueAlphabets.Sequence), 270
gUniqueAlphabets.Sequence, 270
getUniqueProcesses
  (Undocumented, PhyloSim), 559
getUniqueProcesses, Sequence-method
  (getUniqueProcesses.Sequence), 271
gUniqueProcesses.Sequence, 271
getWriteProtected
  (Undocumented, PhyloSim), 559
getWriteProtected, Alphabet-method
  (getWriteProtected.Alphabet), 272
getWriteProtected, Event-method
  (getWriteProtected.Event), 273
getWriteProtected, Process-method
  (getWriteProtected.Process), 274
getWriteProtected, QMatrix-method
  (getWriteProtected.QMatrix), 275
gWriteProtected, Sequence-method
  (getWriteProtected.Sequence), 276
gWriteProtected.Alphabet, 272
gWriteProtected.Event, 273
gWriteProtected.Process, 274
gWriteProtected.QMatrix, 275
gWriteProtected.Sequence, 276
globalConsistencyCheck
  (Undocumented, PhyloSim), 559

globalConsistencyCheck, PSRoot-method
  (globalConsistencyCheck.PSRoot), 277
globalConsistencyCheck.PSRoot, 277
GTR, 61, 111, 114, 129, 146, 224, 231, 278, 288, 317, 318, 320, 359, 375, 378, 409, 469, 477, 538, 552, 555, 560
GTR.checkConsistency
  (checkConsistency.GTR), 60
GTR.getBaseFreqs (getBaseFreqs.GTR), 146
GTR.getRateParam (getRateParam.GTR), 223
GTR.getRateParamList
  (getRateParamList.GTR), 231
GTR.setBaseFreqs (setBaseFreqs.GTR), 409
GTR.setRateParam (setRateParam.GTR), 469
GTR.setRateParamList
  (setRateParamList.GTR), 476
GTR.summary (summary.GTR), 538
GY94.checkConsistency
  (checkConsistency.GY94), 61
GY94.getBaseFreqs
  (getBaseFreqs.GY94), 153
GY94.getEventsAtSite
  (getEventsAtSite.GY94), 172
GY94.getKappa (getKappa.GY94), 183
GY94.getOmegaScalingFactor
  (getOmegaScalingFactor.GY94), 202
GY94.setBaseFreqs
  (setBaseFreqs.GY94), 415
GY94.setKappa (setKappa.GY94), 431
GY94.setRate (setRate.GY94), 462
GY94.summary (summary.GY94), 539

hasSiteSpecificParameter
  (Undocumented, PhyloSim), 559
hasSiteSpecificParameter, Process-method
  (hasSiteSpecificParameter.Process), 283
hasSiteSpecificParameter.Process, 283
hasSymbols (Undocumented, PhyloSim), 559
hasSymbols, Alphabet-method
  (hasSymbols.Alphabet), 284
hasSymbols.Alphabet, 284
hasUndefinedRate
  (Undocumented, PhyloSim), 559
hasUndefinedRate, GeneralInDel-method
(hasUndefinedRate.GeneralInDel), 285
hasUndefinedRate, GeneralSubstitution-method
(hasUndefinedRate.GeneralSubstitution), 286
hasUndefinedRate, Process-method
(hasUndefinedRate.Process), 287
hasUndefinedRate, GeneralInDel, 285
hasUndefinedRate, GeneralSubstitution, 286
hasUndefinedRate, Process, 287
HKY.checkConsistency
(checkConsistency.HKY), 62
HKY.getBaseFreqs (getBaseFreqs.HKY), 147
HKY.getRateParam (getRateParam.HKY), 224
HKY.getRateParamList
(getRateParamList.HKY), 232
HKY.setBaseFreqs (setBaseFreqs.HKY), 410
HKY.setRateParam (setRateParam.HKY), 470
HKY.setRateParamList
(setRateParamList.HKY), 477
HKY.summary (summary.HKY), 540
insertSequence (Undocumented.PhyloSim), 559
insertSequence, Sequence-method
(insertSequence.Sequence), 290
insertSequence.Sequence, 290
intersect (Undocumented.PhyloSim), 559
intersect.list, PSRoot-method
(intersect.list.PSRoot), 291
intersect.list.PSRoot, 291
is (Undocumented.PhyloSim), 559
is.Alphabet, 292
is.CodonAlphabet, 293
is.CodonUNREST, 294
is.Event, 295
is.GeneralDeleter, 296
is.GeneralInDel, 296
is.GeneralInsertor, 297
is.GeneralSubstitution, 298
is.GY94, 299
is.na, PSRoot-method (is.na.PSRoot), 299
is.na.PSRoot, 299
is.phylo, 300
is.Process, 301
is.PSRoot, 302
is.QMatrix, 303
is.Sequence, 304
is.Site, 305
is.tip, PhyloSim-method
(is.tip.PhyloSim), 306
isAttached (Undocumented.PhyloSim), 559
isAttached, Site-method
(isAttached.Site), 307
isAttached.Site, 307
isEmpty (Undocumented.PhyloSim), 559
isEmpty, Alphabet-method
(isEmpty.Alphabet), 308
isEmpty.Alphabet, 308
isStartCodon (Undocumented.PhyloSim), 559
isStartCodon, CodonAlphabet-method
(isStartCodon.CodonAlphabet), 309
isStartCodon.CodonAlphabet, 309
isStopCodon (Undocumented.PhyloSim), 559
isStopCodon, CodonAlphabet-method
(isStopCodon.CodonAlphabet), 310
isStopCodon.CodonAlphabet, 310
JC69, 63, 129, 311, 359, 375, 378, 541, 560
JC69.checkConsistency
(checkConsistency.JC69), 62
JC69.summary (summary.JC69), 541
JTT, 20, 129, 313, 359, 375, 378
JTT.dcmut, 20, 129, 315, 359, 375, 378
K80.checkConsistency
(checkConsistency.K80), 63
K80.getBaseFreqs (getBaseFreqs.K80), 148
K80.getRateParam (getRateParam.K80), 225
K80.getRateParamList
(getRateParamList.K80), 233
K80.setBaseFreqs (setBaseFreqs.K80), 411
K80.setRateParam (setRateParam.K80), 471
K80.setRateParamList
(setRateParamList.K80), 478
K80.summary (summary.K80), 541
K81, 64, 129, 149, 226, 234, 278, 320, 359, 375, 378, 412, 472, 480, 543, 560
PhyloSim.attachSeqToNode
   (attachSeqToNode.PhyloSim), 35
PhyloSim.checkConsistency
   (checkConsistency.PhyloSim), 64
PhyloSim.Debug
   (Debug.PhyloSim), 91
PhyloSim.detachHookFromNode
   (detachHookFromNode.PhyloSim), 93
PhyloSim.detachSeqFromNode
   (detachSeqFromNode.PhyloSim), 96
PhyloSim.exportStatTree
   (exportStatTree.PhyloSim), 109
PhyloSim.getAlignment
   (getAlignment.PhyloSim), 135
PhyloSim.getAlignmentLength
   (getAlignmentLength.PhyloSim), 136
PhyloSim.getBranchEvents
   (getBranchEvents.PhyloSim), 152
PhyloSim.getId
   (getId.PhyloSim), 176
PhyloSim.getLogFile
   (getLogFile.PhyloSim), 187
PhyloSim.getLogLevel
   (getLogLevel.PhyloSim), 188
PhyloSim.getName
   (getName.PhyloSim), 194
PhyloSim.getNedges
   (getNedges.PhyloSim), 198
PhyloSim.getNodes
   (getNodes.PhyloSim), 199
PhyloSim.getNtips
   (getNtips.PhyloSim), 200
PhyloSim.getPhylo
   (getPhylo.PhyloSim), 205
PhyloSim.getRootNode
   (getRootNode.PhyloSim), 237
PhyloSim.getRootSeq
   (getRootSeq.PhyloSim), 238
PhyloSim.getSeqFromNode
   (getSeqFromNode.PhyloSim), 241
PhyloSim.getSequences
   (getSequences.PhyloSim), 243
PhyloSim.getTipLabels
   (getTipLabels.PhyloSim), 260
PhyloSim.getTips
   (getTips.PhyloSim), 261
PhyloSim.getTreeLength
   (getTreeLength.PhyloSim), 266
PhyloSim.is.tip
   (is.tip.PhyloSim), 306
PhyloSim.Log
   (Log.PhyloSim), 326
PhyloSim.plot
   (plot.PhyloSim), 366
PhyloSim.readAlignment
   (readAlignment.PhyloSim), 383
PhyloSim.readTree
   (readTree.PhyloSim), 384
PhyloSim.saveAlignment
   (saveAlignment.PhyloSim), 390
PhyloSim.scaleTree
   (scaleTree.PhyloSim), 392
PhyloSim.setAlignment
   (setAlignment.PhyloSim), 400
PhyloSim.setBranchEvents
   (setBranchEvents.PhyloSim), 414
PhyloSim.setEdges
   (setEdges.PhyloSim), 420
PhyloSim.setID
   (setID.PhyloSim), 425
PhyloSim.setLogFile
   (setLogFile.PhyloSim), 434
PhyloSim.setLogLevel
   (setLogLevel.PhyloSim), 435
PhyloSim.setName
   (setName.PhyloSim), 440
PhyloSim.setNedges
   (setNedges.PhyloSim), 444
PhyloSim.setNodes
   (setNodes.PhyloSim), 445
PhyloSim.setNtips
   (setNtips.PhyloSim), 445
PhyloSim.setPhylo
   (setPhylo.PhyloSim), 449
PhyloSim.setRootNode
   (setRootNode.PhyloSim), 483
PhyloSim.setRootSeq
   (setRootSeq.PhyloSim), 484
PhyloSim.setSequences
   (setSequences.PhyloSim), 487
PhyloSim.setTipLabels
   (setTipLabels.PhyloSim), 500
PhyloSim.setTips
   (setTips.PhyloSim), 501
PhyloSim.setTreeLength
   (setTreeLength.PhyloSim), 505
PhyloSim.Simulate
   (Simulate.PhyloSim), 514
PhyloSim.summary
   (summary.PhyloSim), 543
PhyloSim.Undocumented
INDEX

plot (Undocumented.PhyloSim), 559
plot,ContinuousDeleter-method
 (plot.ContinuousDeleter), 361
plot,ContinuousInsertor-method
 (plot.ContinuousInsertor), 362
plot,DiscreteDeleter-method
 (plot.DiscreteDeleter), 363
plot,DiscreteInsertor-method
 (plot.DiscreteInsertor), 364
plot,GeneralSubstitution-method
 (plot.GeneralSubstitution), 365
plot,PhyloSim-method (plot.PhyloSim), 366
plot,Sequence-method (plot.Sequence), 368
plot,ContinuousDeleter, 361
plot,ContinuousInsertor, 362
plot,DiscreteDeleter, 363
plot,DiscreteInsertor, 364
plot,GeneralSubstitution, 365
plot,PhyloSim, 366
plot,Sequence, 368
plotParametersAtSites
 (Undocumented.PhyloSim), 559
plotParametersAtSites,Sequence-method
 (plotParametersAtSites,Sequence), 369
plotParametersAtSites,Sequence, 369
plusGamma (Undocumented.PhyloSim), 559
plusGamma,Sequence-method
 (plusGamma,Sequence), 370
plusGamma,Sequence, 370
plusInvGamma (Undocumented.PhyloSim), 559
plusInvGamma,Sequence-method
 (plusInvGamma,Sequence), 372
plusInvGamma,Sequence, 372
print,PSRootSummary-method
 (print.PSRootSummary), 373
print,PMatrix-method (print.PMMatrix), 374
print.PSRootSummary, 373
print.PMatrix, 374
Process.== (=.Process), 13
Process.as.character
 (as.character,Process), 28
Process.checkConsistency
 (checkConsistency,Process), 65
Process.clone (clone,Process), 73
Process.getAlphabet
 (getAlphabet,Process), 138
Process.getEventsAtSite
 (getEventsAtSite,Process), 173
Process.getId (getId,Process), 177
Process.getName (getName,Process), 195
Process.getParameterAtSite
 (getParameterAtSite,Process), 203
Process.getSiteSpecificParamIds
 (getSiteSpecificParamIds,Process), 246
Process.getSiteSpecificParamList
 (getSiteSpecificParamList,Process), 247
Process.getWriteProtected
 (getWriteProtected,Process), 274
Process.hasSiteSpecificParameter
 (hasSiteSpecificParameter,Process), 283
Process.hasUndefinedRate
 (hasUndefinedRate,Process), 287
Process.setAlphabet
 (setAlphabet,Process), 401
Process.setId (setId,Process), 425
Process.setName (setName,Process), 441
Process.setParameterAtSite
 (setParameterAtSite,Process), 447
Process.setSiteSpecificParamIds
 (setSiteSpecificParamIds,Process), 489
Process.setSiteSpecificParamList
 (setSiteSpecificParamList,Process), 490
Process.setWriteProtected
Process.summary (summary.Process), 544
proposeLength (Undocumented.PhylSim), 559
proposeLength, GeneralInDel-method (proposeLength.GeneralInDel), 377
proposeLength, GeneralInDel, 377
PSRoot.checkConsistency (checkConsistency.PSRoot), 66
PSRoot.enableVirtual (enableVirtual.PSRoot), 106
PSRoot.getComments (getComments.PSRoot), 154
PSRoot.getMethodsList (getMethodsList.PSRoot), 192
PSRoot.globalConsistencyCheck (globalConsistencyCheck.PSRoot), 277
PSRoot.intersect.list (intersect.list.PSRoot), 291
PSRoot.is.na (is.na.PSRoot), 299
PSRoot.11 (11.PSRoot), 325
PSRoot.my.all.equal (my.all.equal.PSRoot), 336
PSRoot.setComments (setComments.PSRoot), 416
PSRoot.setMethodsList (setMethodsList.PSRoot), 439
PSRoot.summary (summary.PSRoot), 545
PSRoot.virtualAssignmentForbidden (virtualAssignmentForbidden.PSRoot), 561
PSRootSummary, 67, 373, 380, 380
PSRootSummary.checkConsistency (checkConsistency.PSRootSummary), 66
PSRootSummary.print (print.PSRootSummary), 373
QMatrix.as.character (as.character.QMatrix), 29
QMatrix.checkConsistency (checkConsistency.QMatrix), 67
QMatrix.getAlphabet (getAlphabet.QMatrix), 139
QMatrix.getEventRate (getEventRate.QMatrix), 164
QMatrix.getId (getId.QMatrix), 178
QMatrix.getMatrix (getMatrix.QMatrix), 189
QMatrix.getName (getName.QMatrix), 196
QMatrix.getProcess (getProcess.QMatrix), 210
QMatrix.getRate (getRate.QMatrix), 218
QMatrix.getRateList (getRateList.QMatrix), 220
QMatrix.getScaledMatrix (getScaledMatrix.QMatrix), 240
QMatrix.getWriteProtected (getWriteProtected.QMatrix), 275
QMatrix.print (print.QMatrix), 374
QMatrix.Scale (Scale.QMatrix), 391
QMatrix.setAlphabet (setAlphabet.QMatrix), 402
QMatrix.setId (setId.QMatrix), 426
QMatrix.setMatrix (setMatrix.QMatrix), 436
QMatrix.setName (setName.QMatrix), 442
QMatrix.setProcess (setProcess.QMatrix), 454
QMatrix.setRate (setRate.QMatrix), 463
QMatrix.setRateList (setRateList.QMatrix), 465
QMatrix.setScaledMatrix (setScaledMatrix.QMatrix), 486
QMatrix.setWriteProtected (setWriteProtected.QMatrix), 513
QMatrix.summary (summary.QMatrix), 546
INDEX

Sequence\textunderscore as\textunderscore character (as\textunderscore character\textunderscore Sequence), 30
Sequence\textunderscore attach\textunderscore process (attach\textunderscore process\textunderscore Sequence), 33
Sequence\textunderscore check\textunderscore consistency (check\textunderscore consistency\textunderscore Sequence), 68
Sequence\textunderscore clear\textunderscore states (clear\textunderscore states\textunderscore Sequence), 71
Sequence\textunderscore clone (clone\textunderscore Sequence), 74
Sequence\textunderscore copy\textunderscore sub\textunderscore sequence (copy\textunderscore sub\textunderscore sequence\textunderscore Sequence), 88
Sequence\textunderscore delete\textunderscore sub\textunderscore sequence (delete\textunderscore sub\textunderscore sequence\textunderscore Sequence), 92
Sequence\textunderscore detach\textunderscore process (detach\textunderscore process\textunderscore Sequence), 94
Sequence\textunderscore get\textunderscore alphabets (get\textunderscore alphabets\textunderscore Sequence), 141
Sequence\textunderscore get\textunderscore ancestral (get\textunderscore ancestral\textunderscore Sequence), 142
Sequence\textunderscore get\textunderscore big\textunderscore rate (get\textunderscore big\textunderscore rate\textunderscore Sequence), 151
Sequence\textunderscore get\textunderscore cumulative\textunderscore rates (get\textunderscore cumulative\textunderscore rates\textunderscore Sequence), 155
Sequence\textunderscore get\textunderscore cumulative\textunderscore rates\textunderscore from\textunderscore range (get\textunderscore cumulative\textunderscore rates\textunderscore from\textunderscore range\textunderscore Sequence), 156
Sequence\textunderscore get\textunderscore deletion\textunderscore tolerance (get\textunderscore deletion\textunderscore tolerance\textunderscore Sequence), 157
Sequence\textunderscore get\textunderscore events (get\textunderscore events\textunderscore Sequence), 166
Sequence\textunderscore get\textunderscore id (get\textunderscore id\textunderscore Sequence), 179
Sequence\textunderscore get\textunderscore insertion\textunderscore tolerance (get\textunderscore insertion\textunderscore tolerance\textunderscore Sequence), 181
Sequence\textunderscore get\textunderscore length (get\textunderscore length\textunderscore Sequence), 184
Sequence\textunderscore get\textunderscore name (get\textunderscore name\textunderscore Sequence), 197
Sequence\textunderscore get\textunderscore parameter\textunderscore at\textunderscore sites

R.\textunderscore oo, 355
readAlignment (Undocumented\textunderscore PhylodSim), 559
readAlignment\textunderscore PhylodSim-method (readAlignment\textunderscore PhylodSim), 383
readAlignment\textunderscore PhylodSim, 383
readTree (Undocumented\textunderscore PhylodSim), 559
readTree\textunderscore PhylodSim-method (readTree\textunderscore PhylodSim), 384
readTree\textunderscore PhylodSim, 384
rescale\textunderscore qMatrix (Undocumented\textunderscore PhylodSim), 559
rescale\textunderscore qMatrix\textunderscore General\textunderscore substitution\textunderscore method (rescale\textunderscore qMatrix\textunderscore General\textunderscore substitution), 386
rescale\textunderscore qMatrix\textunderscore General\textunderscore substitution, 386
revComp (Undocumented\textunderscore PhylodSim), 559
revComp\textunderscore nucleotide\textunderscore sequence\textunderscore method (revComp\textunderscore nucleotide\textunderscore sequence), 387
revComp\textunderscore nucleotide\textunderscore sequence, 387
sampleState (Undocumented\textunderscore PhylodSim), 559
sampleState\textunderscore General\textunderscore substitution\textunderscore method (sampleState\textunderscore General\textunderscore substitution), 388
sampleState\textunderscore General\textunderscore substitution, 388
sampleStates (Undocumented\textunderscore PhylodSim), 559
sampleStates\textunderscore sequence\textunderscore method (sampleStates\textunderscore sequence), 389
sampleStates\textunderscore sequence, 389
saveAlignment (Undocumented\textunderscore PhylodSim), 559
saveAlignment\textunderscore PhylodSim-method (saveAlignment\textunderscore PhylodSim), 390
saveAlignment\textunderscore PhylodSim, 390
saveLoadReference (Undocumented\textunderscore PhylodSim), 559
Scale (Undocumented\textunderscore PhylodSim), 559
Scale\textunderscore qMatrix\textunderscore method (Scale\textunderscore qMatrix), 391
Scale\textunderscore qMatrix, 391
scaleTree (Undocumented\textunderscore PhylodSim), 559
scaleTree\textunderscore PhylodSim-method (scaleTree\textunderscore PhylodSim), 392
scaleTree\textunderscore PhylodSim, 392
(getParameterAtSites.Sequence), 204
Sequence.getProcesses (getProcesses.Sequence), 211
Sequence.getRateMultipliers (getRateMultipliers.Sequence), 221
Sequence.getSites (getSites.Sequence), 245
Sequence.getStates (getStates.Sequence), 252
Sequence.getString (getString.Sequence), 253
Sequence.getSymbolFreqs (getSymbolFreqs.Sequence), 254
Sequence.getTotalRates (getTotalRates.Sequence), 264
Sequence.getTotalRatesFromRange (getTotalRatesFromRange.Sequence), 264
Sequence.getUniqueAlphabets (getUniqueAlphabets.Sequence), 270
Sequence.getUniqueProcesses (getUniqueProcesses.Sequence), 271
Sequence.WRITE_PROTECTED (getWriteProtected.Sequence), 276
Sequence.insertSequence (insertSequence.Sequence), 290
Sequence.plot (plot.Sequence), 368
Sequence.plotParametersAtSites (plotParametersAtSites.Sequence), 369
Sequence.plusGamma (plusGamma.Sequence), 370
Sequence.plusInvGamma (plusInvGamma.Sequence), 372
Sequence.sampleStates (sampleStates.Sequence), 389
Sequence.setAlphabets (setAlphabets.Sequence), 404
Sequence.setAncestral (setAncestral.Sequence), 405
Sequence.setBigRate (setBigRate.Sequence), 413
Sequence.setCumulativeRates (setCumulativeRates.Sequence), 417
Sequence.setDeletionTolerance (setDeletionTolerance.Sequence), 417
Sequence.setId (setId.Sequence), 427
Sequence.setInsertionTolerance (setInsertionTolerance.Sequence), 429
Sequence.setLength (setLength.Sequence), 432
Sequence.setName (setName.Sequence), 443
Sequence.setParameterAtSites (setParameterAtSites.Sequence), 448
Sequence.setProcesses (setProcesses.Sequence), 455
Sequence.setRateMultipliers (setRateMultipliers.Sequence), 466
Sequence.setStates (setStates.Sequence), 494
Sequence.setString (setString.Sequence), 495
Sequence.setTotalRates (setTotalRates.Sequence), 503
Sequence.setUniqueAlphabets (setUniqueAlphabets.Sequence), 508
Sequence.setUniqueProcesses (setUniqueProcesses.Sequence), 509
Sequence.setWriteProtected (setWriteProtected.Sequence), 513
Sequence.summary (summary.Sequence), 547
setAcceptBy (Undocumented.PhylotSim), 559
setAcceptBy, GeneralIndel-method (setAcceptBy, GeneralIndel), 397
setAcceptBy, GeneralIndel (setAcceptBy, GeneralIndel), 397
setAcceptWin (Undocumented.PhylotSim), 559
setAcceptWin, GeneralInsertor-method (setAcceptWin, GeneralInsertor), 399
setAcceptWin, GeneralInsertor, 399
setAlignment (Undocumented.PhyloSim), 559
setAlignment,PhyloSim-method
(setAlignment.PhyloSim), 400
setAlignment.PhyloSim, 400
setAlphabet (Undocumented.PhyloSim), 559
setAlphabet,GeneralSubstitution-method
(setAlphabet.GeneralSubstitution), 400
setAlphabet,Process-method
(setAlphabet.Process), 401
setAlphabet,QMatrix-method
(setAlphabet.QMatrix), 402
setAlphabet,Site-method
(setAlphabet.Site), 403
setAlphabet.GeneralSubstitution, 400
setAlphabet.Process, 401
setAlphabet.QMatrix, 402
setAlphabet.Site, 403
setAlphabets (Undocumented.PhyloSim), 559
setAlphabets,Sequence-method
(setAlphabets.Sequence), 404
setAlphabets.Sequence, 404
setAncestral (Undocumented.PhyloSim), 559
setAncestral,Sequence-method
(setAncestral.Sequence), 405
setAncestral,Site-method
(setAncestral.Site), 406
setAncestral.Sequence, 405
setAncestral.Site, 406
setBaseFreqs (Undocumented.PhyloSim), 559
setBaseFreqs,F81-method
(setBaseFreqs.F81), 407
setBaseFreqs,F84-method
(setBaseFreqs.F84), 408
setBaseFreqs,GTR-method
(setBaseFreqs.GTR), 409
setBaseFreqs,HKY-method
(setBaseFreqs.HKY), 410
setBaseFreqs,K80-method
(setBaseFreqs.K80), 411
setBaseFreqs,K81-method
(setBaseFreqs.K81), 412
setBaseFreqs,TN93-method
(setBaseFreqs.TN93), 412
setBaseFreqs,F81, 407
setBaseFreqs,F84, 408
setBaseFreqs,GTR, 409
setBaseFreqs,HKY, 410
setBaseFreqs,K80, 411
setBaseFreqs,K81, 412
setBaseFreqs,TN93, 412
setBigRate (Undocumented.PhyloSim), 559
setBigRate,Sequence-method
(setBigRate.Sequence), 413
setBigRate.Sequence, 413
setBranchEvents
(Undocumented.PhyloSim), 559
setBranchEvents,PhyloSim-method
(setBranchEvents.PhyloSim), 414
setBranchEvents.PhyloSim, 414
setCodonFreqs (Undocumented.PhyloSim), 559
setCodonFreqs,GY94-method
(setCodonFreqs.GY94), 415
setCodonFreqs.GY94, 415
setComments (Undocumented.PhyloSim), 559
setComments,PSRoot-method
(setComments.PSRoot), 416
setComments.PSRoot, 416
setCumulativeRates
(Undocumented.PhyloSim), 559
setCumulativeRates,Sequence-method
(setCumulativeRates.Sequence), 417
setCumulativeRates.Sequence, 417
setDeletionTolerance
(Undocumented.PhyloSim), 559
setDeletionTolerance,Sequence-method
(setDeletionTolerance.Sequence), 417
setDeletionTolerance.Sequence, 417
setDist (Undocumented.PhyloSim), 559
setDist,ContinuousDeleter-method
(setDist.ContinuousDeleter), 418
setDist.ContinuousDeleter, 418
setDist,ContinuousInsertor-method
(setDist.ContinuousInsertor), 419
setDist.ContinuousInsertor, 419
setEdges (Undocumented.Insertor), 559
setEdges,PhyloSim-method
setEdges,Undocumented.PhyloSim, 559
INDEX

(setEdges.PhyloSim), 420
setEdges.PhyloSim, 420
setEquDist (Undocumented.PhyloSim), 559
setEquDist, GeneralSubstitution-method (setEquDist.GeneralSubstitution), 421
setEquDist, AminoAcidSubst (setEquDist.GeneralSubstitution), 421
setEvents (Undocumented.PhyloSim), 559
setEvents, Site-method (setEvents.Site), 422
setGenerateBy (Undocumented.PhyloSim), 559
setGenerateBy, GeneralInsertor-method (setGenerateBy.GeneralInsertor), 423
setHandler (Undocumented.PhyloSim), 559
setHandler, Event-method (setHandler.Event), 424
setId (Undocumented.PhyloSim), 559
setId, PhyloSim-method (setId.PhyloSim), 425
setId, Process-method (setId.Process), 425
setId, QMatrix-method (setId.QMatrix), 426
setId, Sequence-method (setId.Sequence), 427
setId, PhyloSim, 425
setId, Process, 425
setId, QMatrix, 426
setId, Sequence, 427
setInsertHook (Undocumented.PhyloSim), 559
setInsertHook, GeneralInsertor-method (setInsertHook.GeneralInsertor), 427
setInsertHook, GeneralInsertor, 427
setInsertionTolerance (Undocumented.PhyloSim), 559
setInsertionTolerance, Sequence-method (setInsertionTolerance.Sequence), 429
setInsertionTolerance.Sequence, 429
setKappa (Undocumented.PhyloSim), 559
setKappa, F84-method (setKappa.F84), 430
setKappa, GY94-method (setKappa.GY94), 431
setKappa.F84, 430
setKappa.GY94, 431
setLength (Undocumented.PhyloSim), 559
setLength, Sequence-method (setLength.Sequence), 432
setLength.Sequence, 432
setLengthParam1 (Undocumented.PhyloSim), 559
setLengthParam1, FastFieldDeleter-method (setLengthParam1.FastFieldDeleter), 432
setLengthParam1.FastFieldDeleter, 432
setLengthParam2 (Undocumented.PhyloSim), 559
setLengthParam2, FastFieldDeleter-method (setLengthParam2.FastFieldDeleter), 433
setLengthParam2.FastFieldDeleter, 433
setLogFile (Undocumented.PhyloSim), 559
setLogFile, PhyloSim-method (setLogFile.PhyloSim), 434
setLogFile.PhyloSim, 434
setLogLevel (Undocumented.PhyloSim), 559
setLogLevel, PhyloSim-method (setLogLevel.PhyloSim), 435
setLogLevel.PhyloSim, 357, 435
setMatrix (Undocumented.PhyloSim), 559
setMatrix, QMatrix-method (setMatrix.QMatrix), 436
setMatrix.QMatrix, 436
setMaxLength (Undocumented.PhyloSim), 559
setMaxLength, ContinuousDeleter-method (setMaxLength.ContinuousDeleter), 437
setMaxLength, ContinuousInsertor-method (setMaxLength.ContinuousInsertor), 438
setMaxLength.ContinuousDeleter, 437
setMaxLength.ContinuousInsertor, 438
setMethodsList (Undocumented.PhyloSim), 559
setMethodsList, PSRoot-method
(setMethodsList.PSRoot), 439
setMethodsList.PSRoot, 439
setName (Undocumented.PhylSim), 559
setName, Event-method (setName.Event), 439
setName, PhylSim-method
(setName.PhylSim), 440
setName, Process-method
(setName.Process), 441
setName, QMatrix-method
(setName.QMatrix), 442
setName, Sequence-method
(setName.Sequence), 443
setName.Event, 439
setName.PhylSim, 440
setName.Process, 441
setName.QMatrix, 442
setName.Sequence, 443
setNedges (Undocumented.PhylSim), 559
setNedges, PhylSim-method
(setNedges.PhylSim), 444
setNedges.PhylSim, 444
setNodes (Undocumented.PhylSim), 559
setNodes, PhylSim-method
(setNodes.PhylSim), 445
setNodes.PhylSim, 445
setNtips (Undocumented.PhylSim), 559
setNtips, PhylSim-method
(setNtips.PhylSim), 445
setNtips.PhylSim, 445
setOmegas (Undocumented.PhylSim), 559
setOmegas, CodonSequence-method
(setOmegas.CodonSequence), 446
setOmegas.CodonSequence, 446
setOmegas.Sequence
(setOmegas.CodonSequence), 446
setParameterAtSite
(Undocumented.PhylSim), 559
setParameterAtSite, Process-method
setParameterAtSite.Process, 447
setParameterAtSite.FastFieldDeleter
setParameterAtSite.Process, 447
setParameterAtSite.Process, 447
setParameterAtSites
(Undocumented.PhylSim), 559
setParameterAtSites, Sequence-method
setParameterAtSites.Sequence, 448
setParameterAtSites.Sequence, 448
setPhylo (Undocumented.PhylSim), 559
setPhylo, PhylSim-method
(setPhylo.PhylSim), 449
setPhylo.PhylSim, 449
setPosition (Undocumented.PhylSim), 559
setPosition, Event-method
setPosition.Event, 450
setPosition.Event, 450
setProbs (Undocumented.PhylSim), 559
setProbs, DiscreteDeleter-method
(setProbs.DiscreteDeleter), 451
setProbs.DiscreteDeleter, 451
setProbs, DiscreteInsertor-method
(setProbs.DiscreteInsertor), 452
setProbs.DiscreteDeleter, 452
setProbs.DiscreteInsertor, 452
setqMatrix (Undocumented.PhylSim), 559
setqMatrix, GeneralSubstitution-method
(setqMatrix.Generalsubstitution), 458
setqMatrix.Generalsubstitution, 458
setRate (Undocumented.PhylorSim), 559
setRate.Event-method (setRate.Event), 459
setRate, GeneralInDel-method (setRate.GeneralInDel), 460
setRate, GeneralSubstitution-method (setRate.GeneralSubstitution), 461
setRate, GY94-method (setRate.GY94), 462
setRate, QMatrix-method (setRate.QMatrix), 463
setRate.Event, 459
setRate, GeneralInDel, 460
setRate, GeneralSubstitution, 461, 462
setRate, GY94, 462
setRate, QMatrix, 463
setRateList (Undocumented.PhylorSim), 559
setRateList, GeneralSubstitution-method (setRateList.GeneralSubstitution), 464
setRateList, QMatrix-method (setRateList.QMatrix), 465
setRateList, GeneralSubstitution, 464
setRateList, QMatrix, 465
setRateMultipliers (Undocumented.PhylorSim), 559
setRateMultipliers, Sequence-method (setRateMultipliers.Sequence), 466
setRateMultipliers.Sequence, 466
setRateParam (Undocumented.PhylorSim), 559
setRateParam.F81-method (setRateParam.F81), 467
setRateParam.F84-method (setRateParam.F84), 468
setRateParam, GTR-method (setRateParam.GTR), 469
setRateParam, HKY-method (setRateParam.HKY), 470
setRateParam, K80-method (setRateParam.K80), 471
setRateParam, K81-method (setRateParam.K81), 472
setRateParam, T92-method (setRateParam.T92), 473
setRateParam, TN93-method (setRateParam.TN93), 474
setRateParam.F81, 467
setRateParam.F84, 468
setRateParam.GTR, 469
setRateParam.HKY, 470
setRateParam.K80, 471
setRateParam.K81, 472
setRateParam.T92, 473
setRateParam.TN93, 474
setRateParamList (Undocumented.PhylorSim), 559
setRateParamList, F81-method (setRateParamList.F81), 475
setRateParamList, F84-method (setRateParamList.F84), 475
setRateParamList, GTR-method (setRateParamList.GTR), 476
setRateParamList, HKY-method (setRateParamList.HKY), 477
setRateParamList, K80-method (setRateParamList.K80), 478
setRateParamList, K81-method (setRateParamList.K81), 479
setRateParamList, T92-method (setRateParamList.T92), 480
setRateParamList, TN93-method (setRateParamList.TN93), 482
setRateParamList.F81, 475
setRateParamList.F84, 475
setRateParamList, GTR, 476
setRateParamList, HKY, 477
setRateParamList, K80, 478
setRateParamList, K81, 479
setRateParamList, T92, 480
setRateParamList, TN93, 482
setRootNode (Undocumented.PhylorSim), 559
setRootNode, PhyloSim-method (setRootNode.PhylorSim), 483
setRootNode, PhyloSim, 483
setRootSeq (Undocumented.PhylorSim), 559
setRootSeq, PhyloSim-method (setRootSeq.PhylorSim), 484
setRootSeq, PhyloSim, 484
setScale (Undocumented.PhylorSim), 559
setScale, BrownianInsertor-method (setScale.BrownianInsertor), 485
setScale.BrownianInsertor, 485
setScaleMatrix
(setTotalRate.Site), 503
(setTotalRate.Site), 503
(setTotalRates (Undocumented.PhyloSim), 559
(setTotalRates.Sequence), 503
(setTotalRates.Sequence), 503
(setTransTable (Undocumented.PhyloSim), 559
(setTransTable.CodonAlphabet-method
(setTransTable.CodonAlphabet), 504
(setTransTable.CodonAlphabet), 504
(setTreeLength (Undocumented.PhyloSim), 559
(setTreeLength.PhyloSim), 505
(setTreeLength.PhyloSim), 505
(setType (Undocumented.PhyloSim), 559
(setType.Alphabet-method
(setType.Alphabet), 506
(setType.BrownianInsertor-method
(setType.BrownianInsertor), 507
(setType.FastFieldDeletor-method
(setType.FastFieldDeletor), 507
(setUniqueAlphabets
(setUniqueAlphabets.Sequence), 508
(setUniqueAlphabets.Sequence), 508
(setUniqueProcesses
(setUniqueProcesses.Sequence), 509
(setUniqueProcesses.Sequence), 509
(setUniqueProcesses.Sequence), 509
(setWriteProtected
(setWriteProtected.Sequence), 510
(setWriteProtected.Alphabet-method
(setWriteProtected.Alphabet), 510
(setWriteProtected.Event-method
(setWriteProtected.Event), 511
(setWriteProtected.Process-method
(setWriteProtected.QMatrix-method
(setWriteProtected.QMatrix), 513
(setWriteProtected.Sequence-method
(setWriteProtected.Sequence), 513
(setWriteProtected.Alphabet), 510
(setWriteProtected.Event), 511
(setWriteProtected.Process), 512
(setWriteProtected.QMatrix-method
(setWriteProtected.QMatrix), 513
(setWriteProtected.Sequence-method
(setWriteProtected.Sequence), 513
Simulate (Undocumented.PhyloSim), 559
Simulate.PhyloSim, 514
Site, 31, 69, 96, 119, 140, 143, 167, 212, 242,
251, 263, 305, 359, 378, 404, 407,
422, 457, 487, 493, 503, 515, 548
Site.as.character (as.character.Site), 31
Site.attachProcess
(attachProcess.Site), 34
Site.checkConsistency
(checkConsistency.Site), 68
Site.detachProcess
(detachProcess.Site), 95
Site.flagTotalRate
(flagTotalRate.Site), 119
Site.getAlphabet (getAlphabet.Site), 140
Site.getAncestral (getAncestral.Site), 143
Site.getEvents (getEvents.Site), 167
Site.getProcesses (getProcesses.Site), 212
Site.getSequence (getSequence.Site), 242
Site.getState (getState.Site), 251
Site.getTotalRate (getTotalRate.Site), 263
Site.isAttached (isAttached.Site), 307
Site.setAlphabet (setAlphabet.Site), 403
Site.setAncestral (setAncestral.Site), 406
Site.setEvents (setEvents.Site), 422
Site.setProcesses (setProcesses.Site), 456
Site.setSequence (setSequence.Site), 486
INDEX

Site.setState(setState.Site), 493
Site.setTotalRate(setTotalRate.Site), 503
Site.summary(summary.Site), 548
stringLength, 518
stringLengthVector, 519
summary, Alphabet-method
(summary.Alphabet), 520
summary, AminoAcidSubst-method
(summary.AminoAcidSubst), 521
summary, BinarySubst-method
(summary.BinarySubst), 522
summary, BrownianInsertor-method
(summary.BrownianInsertor), 523
summary, CodonAlphabet-method
(summary.CodonAlphabet), 524
summary, CodonSubst-method
(summary.CodonSubst), 525
summary, ContinuousDeleter-method
(summary.ContinuousDeleter), 526
summary, ContinuousInsertor-method
(summary.ContinuousInsertor), 527
summary, DiscreteDeleter-method
(summary.DiscreteDeleter), 528
summary, DiscreteInsertor-method
(summary.DiscreteInsertor), 529
summary, Event-method (summary.Event), 530
summary, F81-method (summary.F81), 531
summary, F84-method (summary.F84), 532
summary, FastFieldDeleter-method
(summary.FastFieldDeleter), 533
summary, GeneralDeleter-method
(summary.GeneralDeleter), 534
summary, GeneralIndel-method
(summary.GeneralIndel), 535
summary, GeneralInsertor-method
(summary.GeneralInsertor), 536
summary, GeneralSubstitution-method
(summary.GeneralSubstitution), 537
summary, GTR-method (summary.GTR), 538
summary, GY94-method (summary.GY94), 539
summary, HKY-method (summary.HKY), 540
summary, JC69-method (summary.JC69), 541
summary, K80-method (summary.K80), 541
summary, K81-method (summary.K81), 542
summary, PhyloSim-method
(summary.PhyloSim), 543
summary, Process-method
(summary.Process), 544
summary, PSRoot-method (summary.PSRoot), 545
summary, QMatrix-method
(summary.QMatrix), 546
summary, Sequence-method
(summary.Sequence), 547
summary, Site-method (summary.Site), 548
summary, T92-method (summary.T92), 549
summary, TN93-method (summary.TN93), 550
summary, UNREST-method (summary.UNREST), 551
summary, Alphabet, 520
summary, AminoAcidSubst, 521
summary, BinarySubst, 522
summary, BrownianInsertor, 523
summary, CodonAlphabet, 524
summary, CodonSubst, 525
summary, ContinuousDeleter, 526
summary, ContinuousInsertor, 527
summary, DiscreteDeleter, 528
summary, DiscreteInsertor, 529
summary, Event, 530
summary, F81, 531
summary, F84, 532
summary, FastFieldDeleter, 533
summary, GeneralDeleter, 534
summary, GeneralIndel, 535
summary, GeneralInsertor, 536
summary, GeneralSubstitution, 537
summary, GTR, 538
summary, GY94, 539
summary, HKY, 540
summary, JC69, 541
summary, K80, 541
summary, K81, 542
summary, PhyloSim, 543
summary, Process, 544
summary, PSRoot, 545
summary, QMatrix, 546
summary, Sequence, 547
summary, Site, 548
summary, T92, 549
summary, TN93, 550
summary.UNREST, 551
T92.checkConsistency (checkConsistency.T92), 69
T92.getRateParam (getRateParam.T92), 227
T92.getRateParamList (getRateParamList.T92), 235
T92.getTheta (getTheta.T92), 259
T92.setRateParam (setRateParam.T92), 473
T92.setRateParamList (setRateParamList.T92), 480
T92.setTheta (setTheta.T92), 499
T92.summary (summary.T92), 549
tip (Undocumented.PhyloSim), 559
TN93, 70, 129, 150, 228, 236, 278, 359, 375, 378, 413, 474, 482, 550, 554, 560
TN93.checkConsistency (checkConsistency.TN93), 70
TN93.getBaseFreqs (getBaseFreqs.TN93), 150
TN93.getRateParam (getRateParam.TN93), 228
TN93.getRateParamList (getRateParamList.TN93), 236
TN93.setBaseFreqs (setBaseFreqs.TN93), 412
TN93.setRateParam (setRateParam.TN93), 474
TN93.setRateParamList (setRateParamList.TN93), 482
TN93.summary (summary.TN93), 550
Translate (Undocumented.PhyloSim), 559
Translate.CodonSequence-method (Translate.CodonSequence), 557
Translate.CodonSequence, 557
translateCodon (Undocumented.PhyloSim), 559
translateCodon.CodonAlphabet-method (translateCodon.CodonAlphabet), 558
translateCodon.CodonAlphabet, 558
Undocumented (Undocumented.PhyloSim), 559
Undocumented.PhyloSim-method (Undocumented.PhyloSim), 559
Undocumented.PhyloSim, 559