Package ‘MRIaggr’

March 10, 2015

Type Package

Title Management, Display, and Processing of Medical Imaging Data

Version 1.0

Date 2015-02-27

Author Brice Ozenne

Maintainer Brice Ozenne <brice.ozenne@orange.fr>

Description Provide a compact storage for medical imaging data with access and display possibilities. Basic tools for processing brain imaging data are proposed like extraction of brain voxels, morphological image segmentation, and filtering / normalization of contrast parameters. Specific tools are also provided for blood perfusion imaging to calculate hypoperfusion and reperfusion volumes.

License GPL-3

Depends R (>= 2.10), Rcpp

Imports methods, stats4, RANN, spam, Matrix, ROCR, fields, mritc, oro.dicom, oro.nifti

Suggests rgl

LazyLoad yes

LinkingTo Rcpp, RcppArmadillo


NeedsCompilation yes

Repository CRAN

Date/Publication 2015-03-10 09:24:57

R topics documented:

MRIaggr-package ................................................................. 4
affectClinic ................................................................. 4
affectContrast .............................................................. 5
affectDescStats .............................................................. 7
R topics documented:

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>affectHemisphere</td>
<td>8</td>
</tr>
<tr>
<td>affectNormalization</td>
<td>10</td>
</tr>
<tr>
<td>affectTable</td>
<td>11</td>
</tr>
<tr>
<td>array2df</td>
<td>13</td>
</tr>
<tr>
<td>boxplotMask</td>
<td>14</td>
</tr>
<tr>
<td>calcAUPRC</td>
<td>16</td>
</tr>
<tr>
<td>calcBrainMask</td>
<td>17</td>
</tr>
<tr>
<td>calcContralateral</td>
<td>20</td>
</tr>
<tr>
<td>calcContro_cpp</td>
<td>22</td>
</tr>
<tr>
<td>calcCriteriaGR</td>
<td>23</td>
</tr>
<tr>
<td>calcDistMask</td>
<td>23</td>
</tr>
<tr>
<td>calcDistTissues</td>
<td>25</td>
</tr>
<tr>
<td>calcFilter</td>
<td>26</td>
</tr>
<tr>
<td>calcGR</td>
<td>29</td>
</tr>
<tr>
<td>calcGroupsCoords</td>
<td>32</td>
</tr>
<tr>
<td>calcGroupsCoords_cpp</td>
<td>34</td>
</tr>
<tr>
<td>calcGroupsMask</td>
<td>34</td>
</tr>
<tr>
<td>calcGroupsW</td>
<td>36</td>
</tr>
<tr>
<td>calcGroupsW_cpp</td>
<td>37</td>
</tr>
<tr>
<td>calcHemisphere</td>
<td>37</td>
</tr>
<tr>
<td>calcHemi_cpp</td>
<td>40</td>
</tr>
<tr>
<td>calcNormalization</td>
<td>40</td>
</tr>
<tr>
<td>calcRadius_cpp</td>
<td>42</td>
</tr>
<tr>
<td>calcRegionalContrast</td>
<td>42</td>
</tr>
<tr>
<td>calcROCthreshold</td>
<td>44</td>
</tr>
<tr>
<td>calcSigmaGR</td>
<td>46</td>
</tr>
<tr>
<td>calcSmoothMask</td>
<td>48</td>
</tr>
<tr>
<td>calcTableHypoReperf</td>
<td>50</td>
</tr>
<tr>
<td>calcTableLesion</td>
<td>53</td>
</tr>
<tr>
<td>calcThreshold</td>
<td>54</td>
</tr>
<tr>
<td>calcTissueType</td>
<td>57</td>
</tr>
<tr>
<td>calcW</td>
<td>58</td>
</tr>
<tr>
<td>Carto3D-class</td>
<td>60</td>
</tr>
<tr>
<td>Carto3D2MRIaggr</td>
<td>61</td>
</tr>
<tr>
<td>constCarto3D</td>
<td>62</td>
</tr>
<tr>
<td>constCompressMRIaggr</td>
<td>63</td>
</tr>
<tr>
<td>constMRIaggr</td>
<td>65</td>
</tr>
<tr>
<td>constReduceMRIaggr</td>
<td>66</td>
</tr>
<tr>
<td>constSweave</td>
<td>68</td>
</tr>
<tr>
<td>df2array</td>
<td>71</td>
</tr>
<tr>
<td>EDK</td>
<td>73</td>
</tr>
<tr>
<td>filtrage2Dmed_cpp</td>
<td>74</td>
</tr>
<tr>
<td>filtrage2D_cpp</td>
<td>74</td>
</tr>
<tr>
<td>filtrage3Dmed_cpp</td>
<td>75</td>
</tr>
<tr>
<td>filtrage3D_cpp</td>
<td>75</td>
</tr>
<tr>
<td>GRalgo</td>
<td>76</td>
</tr>
<tr>
<td>heatmapMRIaggr</td>
<td>77</td>
</tr>
<tr>
<td>initCol</td>
<td>79</td>
</tr>
<tr>
<td>R topics documented:</td>
<td></td>
</tr>
<tr>
<td>---------------------------------------------------------</td>
<td>---</td>
</tr>
<tr>
<td>initDisplayWindow</td>
<td>80</td>
</tr>
<tr>
<td>initFilter</td>
<td>81</td>
</tr>
<tr>
<td>initGR</td>
<td>82</td>
</tr>
<tr>
<td>initIndex</td>
<td>83</td>
</tr>
<tr>
<td>initNeighborhood</td>
<td>84</td>
</tr>
<tr>
<td>initNum</td>
<td>85</td>
</tr>
<tr>
<td>initParameter</td>
<td>86</td>
</tr>
<tr>
<td>initWindow</td>
<td>86</td>
</tr>
<tr>
<td>legendMRI</td>
<td>88</td>
</tr>
<tr>
<td>MRIaggr-class</td>
<td>88</td>
</tr>
<tr>
<td>MRIaggr.Pat1_red</td>
<td>91</td>
</tr>
<tr>
<td>multiplot</td>
<td>92</td>
</tr>
<tr>
<td>outline</td>
<td>97</td>
</tr>
<tr>
<td>outlineMRIaggr</td>
<td>98</td>
</tr>
<tr>
<td>plotDistClass</td>
<td>101</td>
</tr>
<tr>
<td>plotLesion3D</td>
<td>103</td>
</tr>
<tr>
<td>plotMRI</td>
<td>105</td>
</tr>
<tr>
<td>plotTableLesion</td>
<td>106</td>
</tr>
<tr>
<td>pointsHemisphere</td>
<td>107</td>
</tr>
<tr>
<td>pointsOutline</td>
<td>108</td>
</tr>
<tr>
<td>readMRI</td>
<td>109</td>
</tr>
<tr>
<td>selectClinic</td>
<td>110</td>
</tr>
<tr>
<td>selectContrast</td>
<td>111</td>
</tr>
<tr>
<td>selectCoords</td>
<td>115</td>
</tr>
<tr>
<td>selectDefault_value</td>
<td>117</td>
</tr>
<tr>
<td>selectDescStats</td>
<td>118</td>
</tr>
<tr>
<td>selectHemispheres</td>
<td>119</td>
</tr>
<tr>
<td>selectHistory</td>
<td>120</td>
</tr>
<tr>
<td>selectIdentifier</td>
<td>121</td>
</tr>
<tr>
<td>selectMidplane</td>
<td>122</td>
</tr>
<tr>
<td>selectN</td>
<td>122</td>
</tr>
<tr>
<td>selectNormalization</td>
<td>123</td>
</tr>
<tr>
<td>selectParameter</td>
<td>125</td>
</tr>
<tr>
<td>selectTable</td>
<td>126</td>
</tr>
<tr>
<td>selectVoxelDim</td>
<td>127</td>
</tr>
<tr>
<td>selectVoxelSize</td>
<td>128</td>
</tr>
<tr>
<td>summary-methods</td>
<td>129</td>
</tr>
<tr>
<td>supprContrast</td>
<td>129</td>
</tr>
<tr>
<td>supprDescStats</td>
<td>130</td>
</tr>
</tbody>
</table>

**Index** 132
affectClinic

Description

Affect clinical data to a MRIaggr object.

Usage

```r
## S4 replacement method for signature 'MRIaggr'
affectClinic(object, add=FALSE, overwrite=FALSE, trace=TRUE) <- value
```

Arguments

- **object**: an object of class MRIaggr. REQUIRED.
- **value**: the clinical data. A one row data.frame. REQUIRED.
- **add**: should value be added to the existing clinical slot? logical.
affectContrast

overwrite if clinical parameters with the same names are already stored in `object@clinic`, can they be overwritten? logical.

trace should the execution of the function be traced? logical.

Value
None.

See Also
`selectClinic` to extract the clinical data.

Examples
```r
## load a mriaggr object
data("MRIaggr.Pat1_red", package="MRIaggr")

## affect clinical data
affectClinic(MRIaggr.Pat1_red) <- data.frame(Age=32,Gender="Male",NIHSS_H0="5")
selectClinic(MRIaggr.Pat1_red,param="Age")

## add a new parameter
affectClinic(MRIaggr.Pat1_red,add=TRUE,overwrite=TRUE) <- data.frame(City="Lyon")
selectClinic(MRIaggr.Pat1_red)
```

---

**Description**
Affect one or several contrast parameters to a `MRIaggr` object.

**Usage**
```r
## S4 replacement method for signature 'MRIaggr'
affectContrast(object, param=NULL, default_value=NULL, overwrite=FALSE, trace=TRUE) <- value
```

**Arguments**
- **object**: an object of class `MRIaggr`. REQUIRED.
- **value**: the value of each contrast parameter (in columns) at each voxel (in rows). `data.frame`. REQUIRED.
- **param**: the names of the contrast parameters. `character vector` or NULL leading to use the name of the value argument.
- **default_value**: the reference values of the contrast parameters (e.g background values). A one row `data.frame` where the column length must match the length of the `param` argument.
overwrite if a contrast parameters with the same names are already stored in object@data, can they be overwritten? logical.

trace should the execution of the function be traced? logical.

Details

FUNCTION:
If the param argument is not specified then the names of the value argument will be used to define the parameter names.
If the default_value argument is NULL then default values "undefined" are attributed to each parameter.
If a parameter named "mask" is intended to be affected it must be done alone and it must be of type logical.
Parameter names "index", "i", "j" and "k" are reserved and cannot be modified. Nevertheless if value contains parameters "i", "j" and "k", the correspondence between these coordinates and the object coordinates is tested.

Value
None.

See Also
calcControlateral, calcRegionalContrast, calcFilter and calcTissueType to compute, modify and affect cartography.
selectContrast to select contrast parameters in the MRIaggr object.

Examples

```r
## load nifti files and convert them to MRIaggr
path <- system.file(file.path("nifti"),package = "MRIaggr")
ls.array <- list()
ls.array[[1]] <- readMRI(file=file.path(path,"DWI_t0"),format="nifti")
ls.array[[2]] <- readMRI(file=file.path(path,"MASK_DWI_t0"),format="nifti")
MRIaggr.Pat1 <- constMRIaggr(ls.array,identifier="Pat1",param=c("DWI_t0","MASK_DWI_t0"))

## affect a new contrast parameters
affectContrast(MRIaggr.Pat1,param="noise",overwrite=TRUE) <- rnorm(selectN(MRIaggr.Pat1))

## perform operations on a contrast parameters and store the results
myCarto <- selectContrast(MRIaggr.Pat1,param="DWI_t0")
myCarto <- myCarto+2+1
affectContrast(MRIaggr.Pat1,param="myCarto",overwrite=TRUE) <- myCarto

## import a contrast parameters in an already existing MRIaggr object
nifti.MTT_t0 <- readMRI(file=file.path(path,"MTT_t0"),format="nifti")
df.MTT_t0 <- array2df(nifti.MTT_t0,name_newparam="MTT_t0")$MTT_t0
affectContrast(MRIaggr.Pat1,param="MTT_t0",overwrite=TRUE) <- df.MTT_t0

## some calc methods automatically save results in the @data slot
```
calcFilter(MRIaggr.Pat1,param="MTT_t0",filter="2D_G3", update.object=TRUE,overwrite=TRUE)
res <- selectContrast(MRIaggr.Pat1,param="MTT_t0_2D_G3")

---

### affectDescStats

Affect non standard elements

#### Description

Affect non standard elements to a `MRIaggr` object.

#### Usage

```r
## S4 replacement method for signature 'MRIaggr'
affectDescStats(object,name,overwrite=FALSE,trace=TRUE) <- value
```

#### Arguments

- `object`: an object of class `MRIaggr`. REQUIRED.
- `value`: any R object. REQUIRED.
- `name`: the name of the element storing `value`. character. REQUIRED.
- `overwrite`: if an element with the same name is already stored in `object@ls_descStats`, can it be overwritten? logical.
- `trace`: should the execution of the function be traced? logical.

#### Details

**FUNCTION:**
Contrary to all other `affect` methods that impose restrictions on the objects that can be affected, this function enable to affect freely a R element. However these elements will have no interaction with the methods of this package.

There is an exception for the element "W.euclidean" that is a reserved space for a neighboring matrix. It must be a `dgCMatrix` with row size and column size corresponding to the number of observations.

#### Value

None.

#### See Also

- `calcW` to compute the neighboring matrix.
- `selectDescStats` to extract non standard elements.
Examples

```r
## load a MRIaggr object
data("MRIaggr.Pat1_red", package="MRIaggr")

## affect a vector
affectDescStats(MRIaggr.Pat1_red,name="spatial_res") <- c(1.875,1.875,6)

## select the corresponding element
selectDescStats(MRIaggr.Pat1_red,"spatial_res")

## some calc methods automatically save results in the ls_descStats slot
# find spatial groups
calcGroupsMask(MRIaggr.Pat1_red,mask=c("MASK_DWI_t0","MASK_T2_FLAIR_t2"),
               distband=6,
               spatial_res=selectDescStats(MRIaggr.Pat1_red,"spatial_res"),
               update.object=TRUE,overwrite=TRUE)

# extract spatial groups
selectDescStats(MRIaggr.Pat1_red,"GroupsLesion")
```

---

affectHemisphere  Affect the position of the mid-saggital plan

Description

Affect information about the position of the mid-saggital plan to a MRIaggr object.

Usage

```r
## S4 replacement method for signature 'MRIaggr'
affectHemisphere(object,overwrite=FALSE,trace=TRUE) <- value
```

Arguments

- **object**: an object of class MRIaggr. REQUIRED.
- **value**: a list of data.frame. Names must be among "midplane", "hemispheres" "data". See the Details section. REQUIRED.
- **overwrite**: if the characteristics of a mid-saggital plan are already stored in object, can they be overwritten? logical.
- **trace**: should the execution of the function be traced? logical.

Details

ARGUMENTS:
The "midplane" element indicates the position of the observations of the mid-saggital plan. It has to be a two columns data.frame with the coordinates ("i", "j") in columns and the observations in rows.
affectHemisphere

The "data" element must contains the position of each voxel relatively to the mid-saggital plan (column "i_hemisphere" and "j_hemisphere") the hemisphere ("left", "right" or "undefined") to which the voxel belongs (column "hemisphere").

The "hemispheres" element indicates in which hemisphere is the lesion (denoted by "lesion"). The others hemispheres are denoted by "controlateral". It has to be a one line two columns data.frame with names "left" "right".

Value

None.

See Also

calcHemisphere to obtain the position of the mid-saggital plan and the position of the lesion in the hemispheres.
selectParameter to extract the previous elements.

Examples

```r
## load a MRIaggr object
data("MRIaggr.Pat1_red", package="MRIaggr")

## automatic affectation
## Not run:
calcHemisphere(MRIaggr.Pat1_red,param="T2_GRE_t0",num=1,i_test=2,j_test=2,angle_test=2,
update.object=TRUE,overwrite=TRUE)

## End(Not run)

## manual affectation
## Not run:
resHemi <- calcHemisphere(MRIaggr.Pat1_red,param="T2_GRE_t0",num=1,i_test=2,j_test=2,angle_test=2)

## End(Not run)
affectHemisphere(MRIaggr.Pat1_red,overwrite=TRUE) <- list(midplane=resHemi$midplane,
data=resHemi$data)

## display
index1 <- data.frame(selectMidplane(MRIaggr.Pat1_red),15)
names(index1) <- c("i","j","k")
multiplot(MRIaggr.Pat1_red,param="T2_GRE_t0",num=1,midplane=TRUE,window=FALSE,
index1=list(coords=index1,pch=20,cex=2,col="purple")
)

selectMidplane(MRIaggr.Pat1_red)
selectHemispheres(MRIaggr.Pat1_red)
```
affectNormalization  Affect normalization values

Description

Affect normalization values to a `mriaggr` object.

Usage

```r
## S4 replacement method for signature 'MRIaggr'
affectNormalization(object, overwrite=FALSE, trace=TRUE) <- value
```

Arguments

- `object`: an object of class `MRIaggr`. REQUIRED.
- `value`: the normalisation values. A list of `data.frame`. REQUIRED.
- `overwrite`: if normalization values are already stored in object, can they be overwritten? logical.
- `trace`: should the execution of the function be traced? logical.

Details

ARGUMENTS:
To enable other methods of the package to use the normalization values the value argument should match the result of the `calcNormalization` function. This function only partially control the validity of the value argument.

A valid value argument should be composed a list of `data.frame` with the following names:

- "norm_global": it should have 6 rows named "mu_both" "mu_left" "mu_right" "sigma_both" "sigma_left" "sigma_right".
- "normMu_slice_both":
- "normSigma_slice_both"
- "normMu_slice_left"
- "normSigma_slice_left"
- "normMu_slice_right"
- "normSigma_slice_right"
- "normMu_3slices_both"
- "normSigma_3slices_both"
- "normMu_3slices_left"
- "normSigma_3slices_left"
- "normMu_3slices_right"
- "normSigma_3slices_right".
affectTable

All the elements of the list must have column names that match the contrast parameters present in the object. Apart from the first element of the list, all elements should have as many rows as slices contained in the object.

Value
None.

See Also
calcNormalization to compute the normalisation values.
selectNormalization to extract the normalisation values.

Examples

```r
## load a MRIaggr object
data("MRIaggr.Pat1_red", package="MRIaggr")

## parameters to normalize
param <- c("DWI_t0","T2_FLAIR_t2","T2_GRE_t0","TTP_t0")

## manual affectation
resNormaliation <- calcNormalization(MRIaggr.Pat1_red,param=param)
affectNormaliation(MRIaggr.Pat1_red,overwrite=TRUE) <- resNormaliation

## automatic affectation
resNormaliation <- calcNormalization(MRIaggr.Pat1_red,param=param,
                                          update.object=TRUE,overwrite=TRUE)
```

Description
Affect volumic information to a MRIaggr object.

Usage

```r
## S4 replacement method for signature 'MRIaggr'
affectTable(object,type,overwrite=FALSE,trace=TRUE) <- value
```

Arguments

- **object**: an object of class MRIaggr. REQUIRED.
- **value**: the volumic information. data.frame. REQUIRED.
- **type**: the type of volumic information. Can be "lesion" "reperfusion" "hypoperfusion". REQUIRED.
overwrite if tables are already stored in object@normalization, can they be overwritten? logical.

trace should the execution of the function be traced? logical.

Details

ARGUMENTS:
The validity of the value object is not checked. A valid format should match the result of the calcTableHypReperf and calcTableLesion functions.

Value

None.

See Also

calcTableHypReperf to compute the hypoperfusion and reperfusion tables.
calcTableLesion to compute the lesion table.
selectTable to extract the tables.
plotTableLesion to display the lesion volume by slice.

Examples

```r
## load a MRIaggr object
data("MRIaggr.Pat1_red", package="MRIaggr")

### 1- lesion ###
## manual affectation
maskN <- c("MASK_DWI_t0","MASK_T2_FLAIR_t2")
restable <- calcTableLesion(MRIaggr.Pat1_red,maskN=maskN,as.logical=TRUE)
affectTable(MRIaggr.Pat1_red,type="lesion",overwrite=TRUE) <- restable

## automatic affectation
restable <- calcTableLesion(MRIaggr.Pat1_red,maskN=maskN,
as.logical=TRUE,update.object=TRUE,overwrite=TRUE)

## display
selectTable(MRIaggr.Pat1_red,type="lesion")

### 2- hypoperfusion and reperfusion ###
## manual affectation
restable <- calcTableHypoReperf(MRIaggr.Pat1_red,param=c("TTP","MTT"),time=c("t0","t1"))
affectTable(MRIaggr.Pat1_red,type="hypoperfusion",overwrite=TRUE) <- restable$volume_hypo
affectTable(MRIaggr.Pat1_red,type="reperfusion",overwrite=TRUE) <- restable$volume_reperf

## automatic affectation
restable <- calcTableHypoReperf(MRIaggr.Pat1_red,param=c("TTP","MTT"),time=c("t0","t1"),
update.object=TRUE,overwrite=TRUE)

## display
selectTable(MRIaggr.Pat1_red,type="reperfusion")
```
**array2df**

**Array to data.frame converter**

**Description**
Convert observations stored in the array format into the data.frame format.

**Usage**

```r
array2df(array, coords=NULL, name_newparam="res",
          names_coords=letters[9:(8+ncol(coords))], na.rm=TRUE)
```

**Arguments**
- `array` the array that should be converted into a data.frame. *array or matrix*. REQUIRED.
- `coords` the spatial coordinates of the observations contained in `array`. *matrix or NULL.*
- `name_newparam` the name of the contrast parameter to which corresponds `array`. *character.*
- `names_coords` the name of the coordinates. *character vector.*
- `na.rm` should observations with missing values be removed? *logical.*

**Details**
ARGUMENTS:
- If `coords` is set to `NULL`, the coordinates will be defined by the position of the observations in `array`.
- If `na.rm` is set to `TRUE`, the `coord` argument will only contain the coordinates of the non-NA observations of `array`.

**Value**
A data frame with in columns the coordinates and the parameter values, and in rows the observations.

**Examples**

```r
## load a nifti file (array format)
path <- system.file(file.path("nifti"), package = "MRIaggr")
nifti.Pat1_TTP_t0 <- readMri(file=file.path(path,"TTP_t0"), format="nifti")
dim(nifti.Pat1_TTP_t0)

## conversion to data frame format
res128 <- array2df(array=nifti.Pat1_TTP_t0, name_newparam="TTP_t0")
dim(res128)
head(res128)
```
## boxplotMask

### Boxplot spatial group characteristics

#### Description

Display a boxplot of the contrast parameter values inside and outside a spatial group.

#### Usage

```r
# S4 method for signature 'MRIaggr'
boxplotMask(object, param, mask, num=NULL, hemisphere="both",
            norm_mu=FALSE, norm_sigma=FALSE, scale=TRUE, as.logical=FALSE,
            window=FALSE, ylim=NULL, col=c("white","purple"),
            main=NULL, mgp=c(2,0.5,0), x.legend="topright",
            y.legend=NULL, cex.legend=0.8,
            filename=paste(object@identifier,"boxplotMask",sep="_"),
            width=1000, height=700, path=NULL, unit="px", res=NULL)
```

#### Arguments

- **object**: an object of class `MRIaggr`. REQUIRED.
- **param**: the contrast parameter(s) associated with the lesion mask(s). character vector. REQUIRED.
- **mask**: the binary contrast parameter(s) defining the spatial group(s). character vector. REQUIRED.
- **num**: the slices to consider. numeric vector or NULL. REQUIRED.
- **hemisphere**: the hemisphere to consider. character.
- **norm_mu**: the type of centering to apply on the parameter values. character.
- **norm_sigma**: the type of scaling to apply on the parameter values. character.
- **scale**: should the contrast parameters be scaled? logical.
- **as.logical**: should mask be convert to logical? logical.
- **window**: the type of device on which the plot will be displayed. logical, NULL or character.
- **ylim**: the y limits of the plot. numeric vector of size 2 or NULL leading to automatic setting of the y limits.
- **col**: the colors of the boxplots for observations inside and outside the mask(s). character vector of size 2.
- **main**: an overall title for the plot. character.
ARGUMENTS:

- `num`: Information about the `num` argument can be found in the details section of `initNum`.
- `hemisphere`, `norm_mu` and `norm_sigma`: Information about the `hemisphere`, `norm_mu` and `norm_sigma` arguments can be found in the details section of `selectContrast`.
- `window`, `filename`, `width`, `height`, `path`, `unit` and `res`: Information about the `window`, `filename`, `width`, `height`, `path`, `unit` and `res` arguments can be found in the details section of `initWindow`.
- `mgp`: Information about the `mgp` argument can be found in `par`.
- `x.legend`, `y.legend`, `cex.legend`: Information about the `x.legend`, `y.legend`, `cex.legend` arguments can be found in `legend` (cex.legend is the cex argument of legend).

Value

None.

Examples

```r
## load a MRIaggr object
data("MRIaggr.Pat1_red", package="MRIaggr")

## display
boxplotMask(MRIaggr.Pat1_red,param=c("DWI_t0","TTP_t0","MTT_t0"),mask="MASK_T2 FLAIR_t2", as.logical=TRUE)
```
Description

Compute the area under the precision recall curve by numerical integration.

Usage

calcauprc(x, y, subdivisions = 10000, performance = NULL)

Arguments

- **x**  
  the biomarker values. *numeric vector*. REQUIRED.

- **y**  
  the class labels. *numeric vector, character vector or logical vector*. REQUIRED.

- **subdivisions**  
  the maximum number of subintervals used for the integration. *positive integer*.

- **performance**  
  an object of class *performance* can be supplied instead of arguments x and y.

Details

ARGUMENTS:

- **y** must have exactly two levels.

- If **performance** is set to **NULL**, the codex and **y** will be used to form the **performance** object.

Value

- A *numeric* between 0 and 1.

Examples

```r
### 1- with MRIaggr data ###  
# load a MRIaggr object  
data(MRIaggr.Pat1_red, package = "MRIaggr")

# select parameter and binary outcome  
cartoT2 <- selectContrast(MRIaggr.Pat1_red, param = "T2_FLAIR_t2")  
cartoMASK <- selectContrast(MRIaggr.Pat1_red, param = "MASK_T2_FLAIR_t2")

# compute AUPRC  
T2.AUPRC <- calcaUPRC(x = cartoT2, y = cartoMASK)

# compute AUC  
# if(require(pROC)){  
# T2.AUC <- auc(roc(cartoMASK ~ cartoT2))  
# }  

# display  
multiplot(MRIaggr.Pat1_red, param = "T2_FLAIR_t2", num = 1,
```
### Description

Seek to form two groups of observations (brain observations and background observations) using a threshold approach or a k-means algorithm.

### Usage

```r
## S4 method for signature 'MRIaggr'
calcBrainMask(object, param, type="kmeans",
              th.breaks=100, th.smoothing=TRUE, th.select_optima=1, th.upper=TRUE, plot=TRUE,
              kmeans.n.groups=2:4, kmeans.Neighborhood=3, skull.param=NULL, skull.n.groups=3,
              window=FALSE, filename=paste("calcBrainMask", type, object@identifier, sep="_"),
              width=1000, height=700, path=NULL, unit="px", res=NA,
              trace=TRUE, update.object=FALSE, overwrite=FALSE)
```

### Arguments

- **object**: an object of class `MRIaggr`. **REQUIRED.**
- **param**: the contrast parameter(s) that should be used to identify the brain observations. **character vector. REQUIRED**
- **type**: the method to use. Can be "threshold" or "kmeans".
- **th.breaks**: the number of thresholds to use. **positive integer.**
- **th.smoothing**: should the derivative be smoothed? **logical.**
- **th.select_optima**: the rank of the optimum to retain. **positive integer.**
- **th.upper**: should the observations above the selected threshold be retained? Else the observations below will the selected threshold be retained. **logical.**

### Example Script

```r
# with simulated data
n0 <- 1000
n1 <- c(10, 100, 1000)
for(iter_n in 1:length(n1)){
  X <- c(rnorm(n0, 0), rnorm(n1[iter_n], 2))
  Y <- c(rep(0, n0), rep(1, n1[iter_n]))
  print(calcAUPRC(X,Y))
}
```

```r
## alternative way using a performance object
perfXY <- ROC::performance(ROC::prediction(X, Y), x.measure="rec", measure="prec")
calcAUPRC(performance=perfXY, subdivisions=10000)
```
plot should the results be plotted? logical.
kmeans.n_groups the number of groups to use in the kmeans algorithm. positive integer vector.
kmeans.Neighborhood the range of the neighborhood. positive integer.
skull.param the parameter used to identify the skull. character.
skull.n_groups the number of groups to use in the kmeans algorithm to obtain the skull.
window the type of device on which the plot will be displayed. logical, NULL or character.
filename the name of the file used to export the plot. character.
width the width of the device used to export the plot. positive numeric.
height the height of the device used to export the plot. positive numeric.
path the directory where the plot file will be created. character.
unit the units in which height and width are given. character.
res the nominal resolution in ppi which will be recorded in the bitmap file. positive integer.
trace should the execution of the function be traced? logical.
update.object should the resulting mask be stored in object as a "mask" parameter? logical.
overwrite if a mask is already stored in object@data, can it be overwritten? logical.

Details

ARGUMENTS:
Information about the window, filename, width, height, path, unit and res arguments can be found in the details section of initWindow.

Setting skull.param to NULL leads to skip the skull stripping step.

FUNCTION:
The threshold approach searches the best break point of the function that maps thresholds to the number of observations. For this, it find the optima of the first derivative of this function (possibly smoothed).
The arguments th.breaks, th.smoothing, th.select_optima and th.upper are only active if type is "threshold".

th.smoothing can be set to an integer to specify the width of the smoothing kernel.

The k-means approach seeks the most spatially coherent partition of the observations, among the possible partitions defined by the kmeans.n_groups argument. The mean number of neighbors averaged over observations (spatial potential) is used as a metric of the spatial coherency of the partition.
The arguments kmeans.n_groups, kmeans.Neighborhood and upper are only active if type is "kmeans".

The skull step consists in identifying the skull with an additional parameter (T1 sequence appears well suited for this purpose), and remove the corresponding observations from the brain mask. It is the partition that gives the best spatial coherency for the final brain mask that is retained, leaving the possibility of no skull stripping.
The plot is active only if type is "threshold".
calcBrainMask

Value

If type is "threshold", a list containing:

- {[[analysis]]}: A matrix containing the parameter thresholds (column "threshold"), the number of observations inside the mask (column "Nb"), the first and its smoothed version (column "dNb" and "dNb.filtered") and indicator of optima (column "optima"). matrix.

- {[[th_opt]]}: A matrix containing the number of observations inside the mask and its derivative (in lines) for each optimum (in columns).

- {[[best_group]]}: An indicator variable giving the observations that belong to the mask. logical vector.

- {[[mask_name]]}: the mask name. character.

If type is "kmeans", a list containing:

- {[[kmeans]]}: the optimal kmeans partition (result of the kmeans function). list.

- {[[potential]]}: the spatial potential for the various brain partitions. matrix.

- {[[best_V]]}: the highest potential. numeric.

- {[[best_group]]}: An indicator variable giving the observations that belong to the mask. logical vector.

- {[[mask_name]]}: the mask name. character.

- {[[potential_skull]]}: the spatial potential for the various skull partitions (only if skull.param is not NULL). matrix.

See Also

selectContrast to select the mask parameter.
calcSmoothMask to spatially regularized the obtained mask.

Examples

```r
# load nifti files and convert them to MRIaggr
path <- system.file(file.path("nifti"), package = "MRIaggr")
ls.array <- list(readMRI(file=file.path(path,"T1_t0"),format="nifti"),
    readMRI(file=file.path(path,"T2_GRE_t0"),format="nifti"))
MRIaggr.Pat1 <- constMRIaggr(ls.array,identifier="Pat1",param=c("T1_t0","T2_GRE_t0"))

### 1- thresholding approach ###
res <- calcBrainMask(MRIaggr.Pat1,param="T2_GRE_t0",type="threshold",
    th.select_optima=2)
breaks <- res$analysis[,"threshold"]
res <- calcBrainMask(MRIaggr.Pat1,param="T2_GRE_t0",type="threshold",
    th.breaks=breaks[breaks>50],th.select_optima=1,
    overwrite=TRUE,update.object=TRUE)

### display
multiplot(MRIaggr.Pat1,param="mask")
```
calcControlateral

Compute contraleral normalization values

Description

Associate each voxel to an hemisphere and compute the difference between the voxel values and their contralateral correspondent.

Usage

## S4 method for signature 'MRIaggr'
calcControlateral(object, param, num=NULL, type="mean", param.ref=NULL,
                  distband=1, lambda=1, trace=TRUE, update.object=FALSE, overwrite=FALSE)

Arguments

- **object**: an object of class `MRIaggr`. REQUIRED.
- **param**: the contrast parameters to normalize. *character vector*. REQUIRED.
- **num**: the slices to extract. *numeric vector* or `NULL`.
- **type**: the method used to compute the contralateral correspondent of each voxel. Can be "mean", "median" or "1NN_penalised".
calcControlateral

param.ref the parameter to use as a reference for the identification of the controlateral voxel. `character` or `NULL` if no reference parameter available.
distband the distance within which the controlateral values are considered. `positive numeric`.
lambda the importance of the penalization. `numeric`.
trace should the execution of the function be traced? `logical`.
update.object should the resulting controlateral parameters be stored in `object`? `logical`.
overwrite if contrast parameters with the same names are already stored in `object@data`, can they be overwritten? `logical`.

Details

ARGUMENTS:
Information about the `num` argument can be found in the details section of `initNum`.

FUNCTION:
To compute the controlateral correspondent of each voxel, the mean or median value of the controlateral observations present in the distband can be used. Otherwise, considering a reference parameter, the controlateral voxel that minimised the difference in contrast (denoted `I`) with the voxel of interest (denoted `x`) penalized by the distance is retained:

\[
\text{voxel}_{\text{contro}} = \arg\min_y |I(x) - I(y)| + \lambda \times \text{dist}(x,y) \]

\[
\sigma = \text{sd}(I)
\]

The `param.ref` and `lambda` are active only if `type` is "1NN_penalized". In this case `lambda` equal 0 means no penalization.

Value

An list containing:
- `[[\text{data}]]`: a `data.frame` containing the coordinates and the parameters normalized by the controlateral values.
- `[[\text{index_plot}]]`: two lists containing the observations used to compute the controlateral values, one for each hemisphere.

See Also

`calcHemisphere` to identify the hemispheres. `selectContrast` to select the controlateral normalized parameters or the hemisphere parameter.

Examples

```r
## load a MRIaggr object
data("MRIaggr.Pat1_red", package="MRIaggr")

## associate each voxel to its controlateral correspondent
## according `T1` parameter and compute the normalized parameters
res <- calcControlateral(MRIaggr.Pat1_red, param=c("DWI_t0","T2_FLAIR_t2"), num=NULL,
                        type="mean", param.ref="T1_t0", distband=1, lambda=1, trace=TRUE)
```

## Description

C++ function called by `calcControlateral` to seek the controlateral voxel and the controlateral values. For internal use.

## Usage

```
calcContro_cpp(contrast, coords_px, index_k, index_k_contro, d_lim, lambda, param_ref, var_ref, type_moy, type_med, type_NN, trace)
```

## Arguments

- `contrast` : the contrast value of each voxel on a given slice. *matrix*.
- `coords_px` : the coordinates of the observations. *matrix*.
- `index_k` : the index of the observations on the hemisphere of interest. *integer vector*.
- `index_k_contro` : the index of the observations on the controlateral hemisphere. *integer vector*.
- `d_lim` : the distance within which the controlateral values are considered. *numeric*.
- `lambda` : the importance of the penalization by the distance. *numeric*.
- `param_ref` : the parameter to be used as a reference to identify the controlateral voxel. *character*.
- `var_ref` : the variance of the reference parameter. *numeric*.
- `type_moy` : should the mean controlateral value be used? *logical*.
- `type_med` : should the median controlateral value be used? *logical*.
- `type_NN` : should the closest controlateral voxel according to the reference parameter be used? *logical*.
- `trace` : should the execution of the function be traced? *logical*. 
calcCriteriaGR

Assessment of clustering quality

Description
Compute several quality indexes of a two group clustering. For internal use.

Usage

calcCriteriaGR(contrast, groups, W=NULL, sigma=NULL, breaks, criterion_d1=FALSE, criterion_entropy=TRUE, criterion_Kalinsky=TRUE, criterion_Laboure=TRUE)

Arguments

- **contrast**: the contrast value of each observation. *numeric vector*. REQUIRED.
- **groups**: the indicator of group membership. *logical vector*. REQUIRED.
- **W**: the neighborhood matrix. dgCMatrix or NULL leading to not compute the d1 criterion.
- **sigma**: the sigma_max that have been used in the GR algorithm. *positive numeric vector*.
- **breaks**: the break points to use to categorize the contrast distribution. *numeric vector*.
- **criterion_d1**: should the d1 criterion be computed? *logical*. Require Wweight to be computed.
- **criterion_entropy**: should the entropy criterion be computed? *logical*.
- **criterion_Kalinsky**: should the Kalinsky criterion be computed? *logical*.
- **criterion_Laboure**: should the Laboure criterion be computed? *logical*.

References


calcdistmask

Euclidean distance to a spatial group

Description
Compute the euclidean distance to a spatial group.
Usage

```r
## S4 method for signature 'MRIaggr'
calcDistMask(object, mask, name_newparam=paste("dist",mask,sep="_"),
  spatial_res=c(1,1,1),as.logical=FALSE,Neighborhood="3D_N10",trace=TRUE,
  update.object=FALSE,overwrite=FALSE)
```

Arguments

- `object` an object of class `MRIaggr`. REQUIRED.
- `mask` the binary contrast parameter(s) defining the spatial groups from which the distance will be computed. character vector. REQUIRED.
- `name_newparam` the name of the new distance parameters. character vector.
- `spatial_res` a dilatation factor for the coordinates. positive numeric vector of size 3.
- `as.logical` should mask be convert to logical? logical.
- `Neighborhood` the type of neighborhood. character.
- `trace` should the execution of the function be traced? logical.
- `update.object` should the resulting distance parameters be stored in `object`? logical.
- `overwrite` if contrast parameters with the same names are already stored in `object@data`, can they be overwritten? logical.

Details

ARGUMENTS:
Information about the `num` argument can be found in the details section of `initNum`.

The `Neighborhood` argument can be a matrix or an array defining directly the neighborhood to use (i.e the weight of each neighbor) or a name indicating which type of neighborhood should be used (see the details section of `initNeighborhood`).

FUNCTION:
This function relies on the `nn2` function of the RANN package.

Value

An `data.frame` containing in row the observations and in columns the distance parameters.

Examples

```r
## load a MRIaggr object
data("MRIaggr.Pat1_red", package="MRIaggr")

## compute distance to initial and final mask
res <- calcDistMask(MRIaggr.Pat1_red,mask=c("MASK_DWI_t0","MASK_T2_FLAIR_t2"),
  update.object=TRUE,overwrite=TRUE)

multiplot(MRIaggr.Pat1_red,param="dist_MASK_DWI_t0",
  index1=list(coords="MASK_DWI_t0",outline=TRUE))

## compute distance to initial and final mask correcting anisotropy
```
calcDistTissues

Description

Compute the four first order statistics of the contrast parameters by cerebral structure.

Usage

## S4 method for signature 'MRIaggr'

calcDistTissues(object, param, class, num=NULL, hemisphere="both")

Arguments

- **object**: an object of class `MRIaggr`. REQUIRED.
- **param**: the contrast parameters to consider. `character vector`. REQUIRED.
- **class**: the parameters indicating the probabilistic membership of each observations to each cerebral structure. `character vector`.
- **num**: the slices to use. `numeric vector` or NULL.
- **hemisphere**: the hemisphere to use. `character`.

Details

ARGUMENTS :
Information about the `num` argument can be found in the details section of `initNum`.
Information about the `hemisphere` argument can be found in the details section of `selectContrast`.

Value

An `data.frame` containing in row the various cerebral structures and in columns the various moments for each contrast parameter.

See Also

calcTissueType to compute a probabilistic classification of the brain observations in WM/GM/CSF.
Examples

```r
## load a MRIaggr object
data("MRIaggr.Pat1_red", package="MRIaggr")

## Not run:
## perform segmentation (call mritc)
calcTissueType(MRIaggr.Pat1_red,param="T1_t0",update.object=TRUE,overwrite=TRUE)

## compute the distribution of DWI and T2 FLAIR for the CSF, WM, GM and lesion observations
res <- calcDistTissues(MRIaggr.Pat1_red,param=c("DWI_t0","T2_FLAIR_t2"),
class=c("CSF","WM","GM","MASK_DWI_t0")
)

## End(Not run)
```

calcFilter  

---

**Image filtration**

**Description**

Apply a filter to an image.

**Usage**

```r
## S4 method for signature 'array'
calcFilter(object,filter,norm=TRUE,w_contrast=FALSE,na.rm=FALSE)

## S4 method for signature 'MRIaggr'
calcFilter(object,param,filter,norm=TRUE,w_contrast=FALSE,
            na.rm=FALSE,name_newparam=NULL,trace=TRUE,update.object=FALSE,overwrite=FALSE)
```

**Arguments**

- `object`: an array or an object of class `MRIaggr`. **REQUIRED.**
- `param`: the contrast parameter to be filtered. **character vector. REQUIRED.**
- `filter`: the filter to use. Can be a `matrix` or an `array`, or a name indicating which filter should be used. **REQUIRED.**
- `norm`: should the filtered correspond to a weighted mean over site ? (or a weighted sum). **logical.**
- `w_contrast`: should the influence of each neighbor be ponderated by the difference in signal with the considered observation ? **logical.**
- `na.rm`: should observations with missing values in their neighborhood be set to NA ? Otherwise the ponderation is adjusted. **logical.**
- `name_newparam`: the name of the new parameters. **character vector.**
- `trace`: should the execution of the function be traced ? **logical.**
calcFilter

update.object  should the resulting filtered parameters be stored in object? logical.
overwrite   if contrast parameters with the same names are already stored in object$data, can they be overwritten? logical.

Details

ARGUMENTS:

Several types of pre-stored filters are availables and can be called by their name:

- filter for smoothing purpose (e.g. "3D_G5"): see the details section of initFilter.
- filter for neighborhood definition purpose (e.g. "3D_N10"): see the details section of initNeighborhood.

norm should be set to TRUE for gaussian, median or sobel filters to correct edge effects. This lead to weight the neighboring value in order to offset the incomplete neighborhood.

Value

An list containing:

- [[res]]: a data.frame containing the coordinates and the filtered parameters.
- [[filter]]: the name of the filter that has been used. character.

See Also

selectContrast to select the filtered parameter(s). initFilter or code initNeighborhood to select pre-stored filters.

Examples

### 1- array method ###
## load a nifti file
path <- system.file(file.path("nifti"), package = "MRIaggr")
nifti.Pat1_DWI_t0 <- readMRI(file=file.path(path,"DWI_t0"), format="nifti")

## before filtering
graphics::image(nifti.Pat1_DWI_t0[,1,1])

## after median filtering
nifti.Pat1_DWI_t0 <- calcFilter(nifti.Pat1_DWI_t0[,] , filter="2D_M3")$res
graphics::image(nifti.Pat1_DWI_t0[,] )

### 2- MRIaggr method ###
## load a MRIaggr object
data("MRIaggr.Pat1_red", package="MRIaggr")

## compute and affect filtered parameter to the MRIaggr object
# gaussian filter
calcFilter(MRIaggr.Pat1_red, param=c("T2_FLAIR_t2", "DWI_t0", "TTP_t0"),
    filter="2D_G3", w_contrast=FALSE, na.rm=FALSE, update.object=TRUE, overwrite=TRUE)
selectParameter(MRIaggr.Pat1_red)
# median filter
calcFilter(MRIaggr$Pat1_red,param=c("T2_FLAIR_t2","DWI_t0","TTP_t0"),
filter="2D_M3",na.rm=FALSE,update.object=TRUE,overwrite=TRUE)

## display
par(mfrow=c(2,2))
multiplot(MRIaggr$Pat1_red,param="T2_FLAIR_t2",
num=1,window=NULL,breaks=c(-100,seq(0,450),601))
multiplot(MRIaggr$Pat1_red,param="T2_FLAIR_t2_2D_G3",
num=1,legend=FALSE,window=NULL,breaks=c(-100,seq(0,450),601))
multiplot(MRIaggr$Pat1_red,param="T2_FLAIR_t2_2D_M3",
num=1,legend=FALSE,window=NULL,breaks=c(-100,seq(0,450),601))

## see the results of the different filters
# G : Gaussian filter
resG <- calcFilter(MRIaggr$Pat1_red,param="T2_FLAIR_t2",
filter="2D_G3")

# M : median filter
resM <- calcFilter(MRIaggr$Pat1_red,param="T2_FLAIR_t2",
filter="2D_M3")

# S : Sobel filter
resS <- calcFilter(MRIaggr$Pat1_red,param="T2_FLAIR_t2",
filter="2D_Sx")

# I
resI.T <- calcFilter(MRIaggr$Pat1_red,param="MASK_T2_FLAIR_t2",
filter="2D_I3",norm=TRUE)
resI.F <- calcFilter(MRIaggr$Pat1_red,param="MASK_T2_FLAIR_t2",
filter="2D_I3",norm=FALSE)

# N
resN.T <- calcFilter(MRIaggr$Pat1_red,param="MASK_T2_FLAIR_t2",
filter="3D_N10",norm=TRUE)
resN.F <- calcFilter(MRIaggr$Pat1_red,param="MASK_T2_FLAIR_t2",
filter="3D_N10",norm=FALSE)

## display
par(mfrow=c(2,2),mar=rep(2,4),mgp=c(2,0.75,0))
breaks <- seq(-50,500,1)
multiplot(MRIaggr$Pat1_red,param="T2_FLAIR_t2",num=3,
breaks=breaks,window=NULL,legend=FALSE,
main="no filtering",num.main=FALSE,main.legend="")
multiplot(resS$res[c("i","j","k")],contrast=resS$res[,4],
num=3,window=NULL,legend=FALSE,
palette="cm.colors",breaks=seq(-1000,1000),
main="sobelX filtering",num.main=FALSE)
multiplot(resG$res[c("i","j","k")],contrast=resG$res[,4],
num=3,window=NULL,legend=FALSE,breaks=breaks,
main="gaussian filtering",num.main=FALSE)
multiplot(resM$res[c("i","j","k")],contrast=resM$res[,4],
num=3,window=NULL,legend=FALSE,breaks=breaks,
**Description**

Call the `GRalgo` function to perform the Growing Region algorithm.

**Usage**

```r
calcGR(contrast, W, seed, sigma_max, range = c(-Inf, +Inf), range.seed = c(-Inf, +Inf), breaks = 100, scale = FALSE, iter_max = 100, sd.robust = FALSE, trace = TRUE, history_sigma = FALSE, history_step = FALSE, history_front = FALSE)
```

**Arguments**

- `contrast` the contrast value of each observation. *numeric vector*. REQUIRED.
- `W` the neighborhood matrix. *dgCMatrix*. REQUIRED.
- `seed` the index of the initial seeds or a binary indicator of the initial seeds. *positive integer vector* or *logical vector*. REQUIRED.
- `sigma_max` the maximum admissible value for the variability of the group contrast. *positive numeric*. REQUIRED.
- `range` the range of acceptable contrast values for the growing region group. *numeric vector* of size 2.
range.seed  
the range of acceptable contrast values for the seeds. numeric vector of size 2.

breaks  
the break points or the number of break points to use to categorize the contrast distribution. numeric vector or positive integer.

scale  
should the contrast be scaled? logical.

iter_max  
the maximum number of iterations for the expansion of the growing region. positive integer.

sd.robust  
should the median absolute deviation be used to estimate the variability of the group contrast, or the standard deviation? logical.

trace  
should the execution of the function be traced? logical.

history_sigma  
should the values of sigma be recorded? logical.

history_step  
should the number of observations included in the growing region set be recorded? logical.

history_front  
should the propagation front of the GR set be recorded? logical.

Details

FUNCTION:
This version of the Growing Region algorithm was been proposed by (Revol et al. 1997).

Value

An list containing:

- [[GR]]: the index of the observations in the growing region. integer vector.
- [[test.break]]: whether the GR algorithm was interrupted an during execution. logical.
- [[iter]]: the number of the last iteration of the algorithm. integer.
- [[test.id]]: whether the GR set has stabilised during the last iteration. logical.
- [[sigma]]: if history_sigma was set to TRUE, the value of the homogeneity criteria at the beginning and the end of each step (in columns) for all steps (in row). numeric matrix.
- [[history_GR]]: if history_step was set to TRUE, the step when each GR observation was included in the GR set. integer vector.
- [[breaks]]: if history_front was set to TRUE, the values used to categorize the contrast. numeric vector.

References


See Also
calcSigmaGR to find the optimal sigma value for a given clustering criterion.
calcGR

Examples

```r
## load a MRIaggr object
data(MRIaggr.Pat1_red, package="MRIaggr")

calcThresholdMRIaggr(MRIaggr.Pat1_red,param=c("TTP_t0","MTT_t0"),threshold=1:10,
name_newparam=c("TTP_th_t0","MTT.th_t0"),
update.object=TRUE,overwrite=TRUE)

## display raw parameter
multiplot(MRIaggr.Pat1_red,param="TTP.th_t0",num=3,as.logical=TRUE,
index1=list(coords="MASK_DWI_t0",outline=TRUE))

## extract raw parameter, coordinates and compute the neighborhood matrix
carto <- selectContrast(MRIaggr.Pat1_red,num=3,hemisphere="lesion",
param=c("TTP.th_t0","TTP_t0","MASK_DWI_t0"))
coords <- selectCoords(MRIaggr.Pat1_red,num=3,hemisphere="lesion")
W <- calcW(coords,distband=sqrt(2))

## the seed is taken to be the point with the largest TTP in the lesion mask
indexN <- which(carto$MASK_DWI_t0==1)
seed <- indexN[which.max(carto[indexN,"TTP_t0")

## Display step by step the GR algorithm with sigma = 1
for(iter in c(0,1,2,5,10)){
  resGR1 <- calcGR(contrast=carto$TTP.th_t0, W=W,
  seed=seed, sigma_max=1, iter_max=iter,trace=FALSE)

  multiplot(MRIaggr.Pat1_red,param="TTP.th_t0",num=3,hemisphere="lesion",legend=FALSE,
  breaks=seq(0,10,0.1),as.logical=TRUE,cex=2,
  main=paste("iteration",iter," - slice ",sep=""),
  index1=list(coords=coords[resGR1$GR1,],pch=20,cex=1),
  index2=list(coords=coords[seed,],pch=20,cex=1)
}

## Not run:
## GR with sigma = 2
resGR2 <- calcGR(contrast=carto$TTP.th_t0, W=W,
  seed=seed,sigma_max=2,iter_max=50,
  history_step=TRUE,history_front=TRUE)

## display
# display the GR over the raw contrast
multiplot(MRIaggr.Pat1_red,param="TTP.th_t0",num=3,hemisphere="lesion",legend=FALSE,
  breaks=seq(0,10,0.1),as.logical=TRUE,cex=2,
  index1=list(coords=coords[resGR2$GR2,],pch=20,cex=1)
}

# display the step of inclusion in GR group for each observation
multiplot(coords[resGR2$GR2,],
  resGR2$history_step,breaks=0:10,
  index1=list(coords=coords[seed,]),
```
palette=rainbow(10)

# display the front propagation
multiplot(coords[resGR2%GR,],
         resGR2%Mfront[,10],
         index1=list(coords=coords[seed,]))

## GR with sigma = 2 and range 6 to 10
resGR3 <- calcGR(contrast=carto$TTP.th_t0, W=W,
                  seed=seed, range=c(6,Inf),
                  sigma_max=2,
                 iter_max=20)

## display
multiplot(MRIaggr.Pat1_red,param="TTP.th_t0",num=3,hemisphere="lesion",legend=FALSE,
          breaks=seq(0,10,0.1),as.logical=TRUE,cex=2,
          index1=list(coords=coords[resGR3%GR,],pch=20,cex=1)
)

## End(Not run)

calcGroupsCoords  Compute spatial groups

Description

Compute the spatial groups using the coordinates of the observations.

Usage

calcGroupsCoords(coords,array=NULL,Neighborhood,max_groups=10000,trace=TRUE)

Arguments

coords  the spatial coordinates of the observations. data.frame. REQUIRED.
array  alternative specification of the spatial coordinates using an array where the non-
       NA values indicates the points of interest. array or NULL leading to consider the
       coords argument.
Neighborhood  the type of neighborhood. character.
max_groups  the maximum number of groups. positive integer.
trace  should the execution of the function be traced ? logical.

Details

ARGUMENTS:
the Neighborhood argument can be a matrix or an array defining directly the neighborhood to use
(i.e the weight of each neighbor) or a name indicating which type of neighborhood should be used
(see the details section of initNeighborhood).
calcGroupsCoords

Value

An list containing:

- `[[ls.group]]`: lists of the observations of each spatial group.
- `[[df.group]]`: a `data.frame` indicating the position and the group of each observation.
- `[[group_size]]`: the size of each spatial group. `integer vector`.

See Also

`initFilter` for various pre-stored filters.

Examples

```r
## load a MRIaggr object
data("MRIaggr.Pati_red", package="MRIaggr")

## select data
MASK_DWI_t0 <- selectContrast(MRIaggr.Pati_red, param="MASK_DWI_t0")
coords <- selectCoords(MRIaggr.Pati_red)

### 1- compute spatial groups using coordinates ###
res3DN18 <- calcGroupsCoords(coords=coords[[MASK_DWI_t0==1]], Neighborhood="3D_N18")
res3DN18$group_size

## display the lesion within the brain
multiplot(coords, contrast=MASK_DWI_t0, legend=FALSE,
         index1=list(coords=coords, outline=TRUE, cex=0.5),
         num.main=FALSE)

## display the lesion spatial groups
multiplot(res3DN18$df.group[,c("i","j","k")], contrast=res3DN18$df.group[,"group"],
          legend=FALSE, cex=0.5, index1=list(coords=coords, outline=TRUE))

## zoom
multiplot(res3DN18$df.group[,c("i","j","k")], contrast=res3DN18$df.group[,"group"],
          legend=FALSE, num=1, cex=0.1, index1=list(coords=coords, outline=TRUE))

### 2- compute spatial groups using an array ###
A.MASK_DWI_t0 <- df2array(MASK_DWI_t0, coords=coords)$contrast[[1]]
A.MASK_DWI_t0[A.MASK_DWI_t0==FALSE] <- NA

## display
graphics::image(A.MASK_DWI_t0[,1:3])

## computation of the spatial groups
res3DN18.bis <- calcGroupsCoords(array=A.MASK_DWI_t0, Neighborhood="3D_N18")

res3DN18$group_size-res3DN18.bis$group_size # same result
```
calcGroupsCoords_cpp  Find spatial groups

Description

C++ function called by calcGroupsCoords to compute the spatial groups. For internal use.

Usage

calcGroupsCoords_cpp(coords_NNA, index_NNA, Neighborhood, coords_max, max_groups, trace)

Arguments

- coords_NNA: the spatial coordinates of the observations in C version (beginning at 0). matrix.
- index_NNA: the index of the coordinates in a array in C version (beginning at 0). numerical vector.
- Neighborhood: the type of neighborhood. character.
- coords_max: the maximum coordinate in each dimension. numerical vector.
- max_groups: the maximum number of groups. postive integer.
- trace: should the execution of the function be traced? logical.

calcGroupsMask  Compute spatial groups

Description

Compute the spatial groups defined by a binary parameter.

Usage

## S4 method for signature 'MRIaggr'
calcGroupsMask(object, mask, distband, spatial_res=c(1,1,1),
               as.logical=FALSE,W="ifany",trace=TRUE,update.object=FALSE,overwrite=TRUE)

Arguments

- object: an object of class MRIaggr. REQUIRED.
- mask: the binary contrast parameter that should be used to identifying the spatial groups. character. REQUIRED.
- distband: the neighborhood range. postive numeric. REQUIRED.
- spatial_res: a dilatation factor for the coordinates. positive numeric vector of size 3.
- as.logical: should mask be convert to logical? logical.
calcGroupsMask

W
the neighborhood matrix. dgCMatrix or "if any" leading to use the neighborhood matrix stored in the object if any and else compute this matrix.

trace
should the execution of the function be traced? logical.

update.object
should the resulting spatial groups be stored in object? logical.

overwrite
if spatial groups are already stored in object@ls_descStats, can they be overwritten? logical.

Details

FUNCTION:
Call the calcGroupsW function.

Value

An list containing for each parameter:

- [[group]]: a vector containing the group index for each observation.
- [[group_size]]: a vector with the size of each spatial group.
- [[group_number]]: the number of spatial groups. integer vector.
- [[group_max]]: the index of the largest group. integer vector.

See Also

selectDescStats to select the spatial groups.
calcW to compute the neighborhood matrix.

Examples

## load a MRIaggr object
data("MRIaggr.Pat1_red", package="MRIaggr")

## compute spatial groups
calcGroupsMask(MRIaggr.Pat1_red,mask=c("MASK_DWI_t0","MASK_T2_FLAIR_t2"),
distband=6,
spatial_res=c(1.875,1.875,6),
update.object=TRUE,overwrite=TRUE)

## extract spatial groups
selectDescStats(MRIaggr.Pat1_red,"GroupsLesion")
calcGroupsW

Compute spatial groups

Description

Compute the spatial groups using a neighborhood matrix.

Usage

calcGroupsW(W, subset=NULL, max_groups=10000)

Arguments

W the neighborhood matrix. \texttt{dgCMatrix}. \textbf{REQUIRED}.

subset the subset of observations to use. \texttt{positive integer vector} or \texttt{NULL} leading to use all observations.

max_groups the maximum number of groups. \texttt{positive integer}.

Value

An \texttt{list} containing:

- \texttt{[group]}: a \texttt{vector} containing the group index for each observation. Observations out of the subset are set to \texttt{NA}.
- \texttt{[subset]}: a \texttt{vector} containing the group index for each observation in subset.
- \texttt{[group.size]}: a \texttt{vector} with the size of each spatial group.
- \texttt{[group.number]}: the number of spatial groups. \texttt{integer vector}.
- \texttt{[group.max]}: the number of the largest group. \texttt{integer vector}.

Examples

```r
# load a MRIaggr object
data("MRIaggr.Pat1_red", package="MRIaggr")

# select data
MASK_DWI_t0 <- selectContrast(MRIaggr.Pat1_red, param="MASK_DWI_t0")
coords <- selectCoords(MRIaggr.Pat1_red)

# select compute W
W <- calcW(object=as.data.frame(coords[,MASK_DWI_t0==1,]), dist.band=sqrt(2), row.norm=TRUE)

# find spatial groups
res.Groups <- calcGroupsW(W)
res.Groups$group.size

# display
multiplot(coords[,MASK_DWI_t0==1,], contrast=res.Groups$group,)
```
calcGroupsW\_cpp

\[
\text{legend=}\text{FALSE}, \text{cex}=0.5, \\
\text{palette=}\text{rainbow(10)[[-1]], index1=list(coords=coords, outline=TRUE))}
\]

---

calcGroupsW\_cpp  \hspace{1cm} \textit{Find spatial groups}

**Description**

C++ function called by \texttt{calcGroupsW} to compute the spatial groups. For internal use.

**Usage**

\[
\text{calcGroupsW}\_cpp(W, \text{subset, max}\_\text{groups})
\]

**Arguments**

- \texttt{W} \hspace{1cm} the neighborhood matrix. \texttt{dgCMatrix}.
- \texttt{subset} \hspace{1cm} the subset of observations to use. \texttt{positive integer vector}.
- \texttt{max\_groups} \hspace{1cm} the maximum number of groups. \texttt{positive integer}.

---

calcHemisphere \hspace{1cm} \textit{Find the mid-saggital plan}

**Description**

Find a plane that distinguish the two cerebral hemispheres.

**Usage**

\[
\# S4 method for signature 'MRIaggr' \hspace{1cm} \\
\text{calcHemisphere(object, param, num=NULL, p=1, subset=NULL, penalty="symmetry", mask=NULL, as.logical=FALSE, i\_test=5, j\_test=5, angle\_test=5, unit\_angle="radian", n\_points=100, plot=TRUE, window=FALSE, filename=paste(object\_identifier, "," _calcHemisphere", sep=""), width=1000, height=700, path=NULL, unit="px", res=NA, trace=TRUE, update.object=FALSE, overwrite=FALSE)}
\]
Arguments

- **object**: an object of class `MRIaggr`. REQUIRED.
- **param**: the contrast parameter that should be used to distinguish the two hemispheres. `character`. REQUIRED.
- **num**: the slices to use. `numeric vector` or NULL.
- **p**: the type of distance for the penalization. `positive numeric`.
- **subset**: the subset of observations to use. `positive integer vector` or NULL leading to use all observations.
- **penalty**: the type of objective function. Can be "symmetry" or "asymmetry".
- **mask**: the binary contrast parameter(s) indicating the lesion. `character vector` or NULL if no mask is available.
- **as.logical**: should mask be convert to logical? `logical`.
- **i_test**: the abscissa or the number of abscissa to test. `numeric vector` or `positive integer`.
- **j_test**: the ordinates or the number of ordinates to test. `numeric vector` or `positive integer`.
- **angle_test**: the angle or the number of angle to test. `numeric vector` or `positive integer`.
- **unit_angle**: the unit in which the angle is given. Can be "radian" or "degree".
- **n.points**: the number of points that represent the mid-saggital plan to computed. `positive integer`.
- **plot**: should the results be plotted? `logical`.
- **window**: the type of device on which the plot will be displayed. `logical`, NULL or character.
- **filename**: the name of the file used to export the plot. `character`.
- **width**: the width of the device used to export the plot. `positive numeric`.
- **height**: the height of the device used to export the plot. `positive numeric`.
- **path**: the directory where the plot file will be created. `character`.
- **unit**: the units in which height and width are given. `character`.
- **res**: the nominal resolution in ppi which will be recorded in the bitmap file. `positive integer`.
- **trace**: should the execution of the function be traced? `logical`.
- **update.object**: should the resulting midplane be stored in `object`? `logical`.
- **overwrite**: if a midplane is already stored in `object@midplane`, can it be overwritten? `logical`.

Details

ARGUMENTS:
Information about the `num` argument can be found in the details section of `initNum`.
Setting `p` to 1 leads to use the absolute difference; setting `p` to 2 leads to use the euclidean distance.
Information about the `window`, `filename`, `width`, `height`, `path`, `unit` and `res` arguments can be found in the details section of `initWindow`. 
FUNCTION:
This function seeks the plane that minimize the difference between controlateral values of the two hemispheres or maximize the similarity between the two hemispheres. There are 3 degree of freedom: two for the position of the center and one for the angle. The separation between the hemisphere is assumed to be identical for all slices. From our experience, using an objective function based on symmetry gives better results compared to asymmetry.

Value
An list containing:
• `[[penalty]]`: an array containing the objective function for the various configurations.
• `[[nb]]`: an array containing the number of observations used to compute the penalty function for the various configurations.
• `[[moy]]`: an array containing the mean value of the objective function for the various configurations.
• `[[optimum]]`: the parameters of the optimal midplane. numeric vector
• `[[midplane]]`: the position of the midplane points. matrix.
• `[[data]]`: the position of the observations relatively to the mid-saggital plane. matrix.
• `[[cv]]`: Was the optimum reached inside the parameter space and not at a border? logical.

See Also
selectParameter to select the midplane.

Examples
```r
## load a MRIagr object
data("MRIagr.Pat1_red", package="MRIagr")

## Not run:
res <- calcHemisphere(MRIagr.Pat1_red,param="T2_GRE_t0",num=1,
                      i_test=2,j_test=2,angle_test=2,
                      trace=TRUE,update.object=TRUE,overwrite=TRUE)

## End(Not run)

## display the mid-saggital plan
multiplot(MRIagr.Pat1_red,param="T2_GRE_t0",num=3,legend=FALSE,
          midplane=TRUE,main="original coordinates - slice ")

## display
multiplot(selectContrast(MRIagr.Pat1_red,param=c("i_hemisphere","j_hemisphere","k")),
          contrast=selectContrast(MRIagr.Pat1_red,param="T2_GRE_t0"),num=3,
          index1=cbind(0,seq(-50,50),3),main="new coordinates - slice ",legend=FALSE)

## compute the mid-saggital plan and mark lesion
## Not run:
```
calcHemi_cpp  Mid-saggital plan search

Description

C++ function called by calcHemisphere to compute the objective function that evaluates the consistency of the mid-saggital plan. For internal use.

Usage

calcHemi_cpp(px_hemiL, px_hemir, sd_data, p, symetrie)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>px_hemiL</td>
<td>the coordinates and the contrast of the observations in the left hemisphere for a given slice. matrix.</td>
</tr>
<tr>
<td>px_hemir</td>
<td>the coordinates and the contrast of the observations in the right hemisphere for a given slice. matrix.</td>
</tr>
<tr>
<td>sd_data</td>
<td>the standard deviation of the parameter. numeric.</td>
</tr>
<tr>
<td>p</td>
<td>the penalization factor. positive numeric.</td>
</tr>
<tr>
<td>symetrie</td>
<td>the type of objective function. TRUE correspond to &quot;symmetry&quot; and FALSE to &quot;asymmetry&quot;.</td>
</tr>
</tbody>
</table>

calcNormalization  Compute normalization values

Description

Compute the normalization values for each contrast parameter.

Usage

## S4 method for signature 'MRIaggr'
calcNormalization(object, param, mu_type="mean", sigma_type="sd", rm.CSF=FALSE, rm.WM=FALSE, rm.GM=FALSE, trace=TRUE, update.object=FALSE, overwrite=FALSE)
calcNormalization

Arguments

- object: an object of class MRIaggr. REQUIRED.
- param: the contrast parameters to normalize. character vector. REQUIRED.
- mu_type: the type of centering. Can be "mean" or "median".
- sigma_type: the type of scaling. Can be "sd" or "mad".
- rm.CSF: should the cerebral spinal fluid observations be excluded? logical.
- rm.GM: should the grey matter observations be excluded? logical.
- rm.WM: should the white matter observations be excluded? logical.
- trace: should the execution of the function be traced? logical.
- update.object: should the resulting normalization values be stored in object? logical.
- overwrite: if normalization values are already stored in object@normalization, can they be overwritten? logical.

Details

FUNCTION:
If any of the rm.CSF, rm.WM or rm.GM is set to true, then the values of the parameters remaining to false (among CSF, WM and GM) are summed. Voxels with value under 0.5 are discarded.
Note that rm.CSF, rm.GM and rm.WM cannot be set simultaneously to TRUE.

Value
An list containing the normalization values, one element for each type of normalization.

See Also

selectNormalization to select the normalization values.
calcTissueType to compute a probabilistic classification of the brain observations in WM/GM/CSF.

Examples

```r
## load a MRIaggr object
data("MRIaggr. Pat1_red", package="MRIaggr")

## compute normalization values
res <- calcNormalization(MRIaggr. Pat1_red, param=c("DWI_t0","T2_FLAIR_t2"),
update.object=TRUE, overwrite=TRUE)

## display
par(mfrow=c(2,4),mar=rep(1.5,4),mgp=c(2.0.5,0))
multiplot(MRIaggr. Pat1_red, param="T2_FLAIR_t2", num=1:3,
legend=TRUE, window=NULL, main="raw - slice ")
multiplot(MRIaggr. Pat1_red, param="T2_FLAIR_t2", num=1:3,
norm_mu="controlateral", norm_sigma="controlateral",
legend=TRUE, window=NULL, main="normalized - slice ")

## extract normalization
selectNormalization(MRIaggr. Pat1_red, type="global", mu=TRUE, sigma=FALSE)
```
calcRadius_cpp  
*Compute geometric characteristics of a spatial group*

**Description**

C++ function called by `calcGR` to compute the barycenter of a spatial group. For internal use.

**Usage**

`calcRadius_cpp(coords, sample, threshold, subset_bary, trace)`

**Arguments**

- `coords`: the spatial coordinates of the observations. *matrix* with a number of rows equal to the length of `sample`.
- `sample`: the weight of each voxel in the computation of the barycenter. *positive numeric*.
- `threshold`: observations with a `sample` value below the value of `threshold` are discarded. *numeric*.
- `subset_bary`: an indicator of the observations that should be kept? *logical vector*.
- `trace`: should the radius of the spatial group be printed? *logical*.

calcRegionalContrast  
*Compute regional contrast parameters*

**Description**

Compute the regional contrast parameters.

**Usage**

```R
# S4 method for signature 'MRIaggr'
calcRegionalContrast(object, param, num=NULL, hemisphere="both", W="ifany", 
spatial_res=c(1,1,1), distband, distband_EDK, power_EDK=2, diagonal=FALSE, 
trace=TRUE, name_newparam=paste(param,"_regional",sep="_"), 
update.object=FALSE, overwrite=FALSE)
```

**Arguments**

- `object`: an object of class `MRIaggr`. REQUIRED.
- `param`: the contrast parameter(s) from which the regional parameter(s) will be computed. *character vector*. REQUIRED.
- `num`: the slices to use. *numeric vector* or NULL.
- `hemisphere`: the hemisphere to use. *character*. REQUIRED.
calcRegionalContrast

W the neighborhood matrix. *dgCMatrix* or "if any" leading to use the neighborhood matrix stored in the object if possible, else to compute this matrix.

spatial_res a dilatation factor for the coordinates. *positive numeric vector of size 3.*

distband the neighborhood range. *positive numeric.* Required only if W have to be computed.

distband_EDK the distband of the kernel. *positive numeric.* REQUIRED.

power_EDK the power of the kernel. *positive numeric.*

diagonal should the diagonal be added to the neighborhood matrix? *logical.*

trace should the execution of the function be traced? *logical.*

name_newparam the name of the new parameters. *character vector.*

update.object should the resulting regional parameters be stored in object? *logical.*

overwrite if contrast parameters with the same names are already stored in object can they be overwritten? *logical.*

Details

ARGUMENTS:
Information about the num argument can be found in the details section of *initNum.*

Information about the hemisphere arguments can be found in the details section of *selectContrast.*

Information about distband_EDK and power_EDK arguments can be found in the details section of EDK.

Value

A *data.frame* containing in columns the regional parameters.

See Also

*selectContrast* to select the regional parameter(s).

*calcW* to compute the neighboring matrix.

Examples

```r
## load a MRIaggr object
data("MRIaggr.Pat1_red", package="MRIaggr")

## compute regional values

## Not run:
res <- calcRegionalContrast(MRIaggr.Pat1_red,param=c("T2_FLAIR_t2","T1_t0"),
                           spatial_res=c(1.875,1.875,1.875),distband=6,distband_EDK=1.875,
                           update.object=TRUE,overwrite=TRUE)

## display
par(mfrow=c(2,4),mar=rep(1.5,4),mgp=c(2,0.5,0))
multiplot(MRIaggr.Pat1_red,param="T2_FLAIR_t2",num=1:3,
          window=NULL,main="raw - slice ")
```
calcROCthreshold

multiplot(MRIaggr.Pat1_red,param="T2 FLAIR_t2_regional",num=1:3,
      window=NULL,main="regional - slice ")

## End(Not run)

**Description**

Perform a ROC analysis on a continuous variable for identifying a binary outcome.

**Usage**

```r
## S4 method for signature 'MRIaggr'
calcROCthreshold(object, param, mask, as.logical=FALSE, digit=2,
                    plot="ROC Youden", digit.plot=3,
                    window=FALSE, filename=paste(object@identifier,"calcROCthreshold",plot,sep=" "),
                    width=1000, height=700, path=NULL, unit="px", res=NA,
                    trace=TRUE, update.object=FALSE, overwrite=FALSE)
```

**Arguments**

- **object**: an object of class `MRIaggr`. REQUIRED.
- **param**: the contrast parameter(s) that should be used to identify the observations inside the mask. character vector. REQUIRED.
- **mask**: the binary contrast parameter that will be used as the outcome in the ROC analysis. character vector. REQUIRED.
- **as.logical**: should mask be convert to logical? logical.
- **digit**: the number of decimal places to use when generating the thresholds. positive integer.
- **plot**: the type of the graphic to display? character or FALSE. See the details section.
- **digit.plot**: the number of decimal places to use for the legend. positive integer.
- **window**: the type of device on which the plot will be displayed. logical, NULL or character.
- **filename**: the name of the file used to export the plot. character.
- **width**: the width of the device used to export the plot. positive numeric.
- **height**: the height of the device used to export the plot. positive numeric.
- **path**: the directory where the plot file will be created. character.
- **unit**: the units in which height and width are given. character.
- **res**: the nominal resolution in ppi which will be recorded in the bitmap file. positive integer.
- **update.object**: should the resulting threshold analysis be stored in `object@ls_descStats`? logical.
- **overwrite**: if a threshold analysis is already stored in `object@ls_descStats`, can it be overwritten? logical.
- **trace**: should the execution of the function be traced? logical.
calcROCthreshold

Details

ARGUMENTS:
Information about the window, filename, width, height, path, unit and res arguments can be found in the details section of initWindow.

Possible values for plot are:

- "ROC_Youden" : display the ROC curve with the optimal threshold according to the Youden index.
- "ROC_prev" : display the ROC curve with the optimal threshold according to the utility function.
- "boxplot_Youden" : display a boxplot of the contrast parameter for each outcome with the optimal threshold according to the Youden index.
- "boxplot_prev" : display a boxplot of the contrast parameter for each outcome with the optimal threshold according to the utility function.
- FALSE : no graphic is displayed.

Value

An data.frame containing for each mask the AUC and AUPRC value, the optimal threshold and the corresponding sensitivity and specificity for the Youden criteria and a utility function taking into account the prevalence.

See Also

selectDescStats to select the mask caracteristics.

Examples

```r
## load a MRIaggr object
data("MRIaggr.Pat1_red", package="MRIaggr")

## ROC analysis
res <- calcROCthreshold(MRIaggr.Pat1_red,param=c("DWI_t0","T2_FLAIR_t2"),
                        mask=c("MASK_DWI_t0","MASK_T2_FLAIR_t2"),as.logical=TRUE)

res <- calcROCthreshold(MRIaggr.Pat1_red,param=c("DWI_t0","T2_FLAIR_t2"),
                        mask=c("MASK_DWI_t0","MASK_T2_FLAIR_t2"),as.logical=TRUE,
                        plot="boxplot_Youden",
                        update.object=TRUE,overwrite=TRUE)

selectDescStats(MRIaggr.Pat1_red,"Mask_threshold")
```
calcSigmaGR

**Automatic Growing Region algorithm**

**Description**

Evaluate the quality of the Growing Region partition for several homogeneity parameters.

**Usage**

```
calcSigmaGR (contrast, W, seed, sigma, range=c(-Inf,+Inf), range.seed=c(-Inf,+Inf),
breaks=100, scale=FALSE, iter_max=100, sd.robust=FALSE, criterion_d1=FALSE,
criterion_entropy=TRUE, criterion_Kalinsky=TRUE, criterion_Laboure=TRUE,
trace=TRUE, mar=rep(2,4), mgp=c(2,0.75,0), main="", window=FALSE,
filename="calcSigmaGR", width=1000, height=700, path=NULL, unit="px", res=NA)
```

**Arguments**

- `contrast` : the contrast value of each observation. *numeric vector*. REQUIRED.
- `W` : the neighborhood matrix. *dgCMatrix*. REQUIRED.
- `seed` : the index of the initial seeds or a binary indicator of the initial seeds. *positive integer vector* or *logical vector*. REQUIRED.
- `sigma` : the sequence of maximum admissible values for the group variability *positive numeric vector*. REQUIRED.
- `range` : the range of acceptable contrast values for the growing region group. *numeric vector of size 2*.
- `range.seed` : the range of acceptable contrast values for the seeds. *numeric vector of size 2*.
- `breaks` : the break points or the number of break points to use to categorize the contrast distribution. *numeric vector* or *positive integer*.
- `scale` : should the contrast be scaled? *logical*.
- `iter_max` : the maximum number of iterations for the expansion of the growing region. *positive integer*.
- `sd.robust` : should the median absolute deviation be used to estimate the variability of the group contrast (TRUE), or the standard deviation (FALSE)? *logical*.
- `criterion_d1` : should the d1 criterion be computed? *logical*. Require `W`eight to be computed.
- `criterion_entropy` : should the entropy criterion be computed? *logical*.
- `criterion_Kalinsky` : should the Kalinsky criterion be computed? *logical*.
- `criterion_Laboure` : should the Laboure criterion be computed? *logical*.
- `trace` : should the execution of the function be traced? *logical*.
- `mar` : the number of margin lines to be specified on the four sides of the plot. *positive numeric vector of size 4*.
calcSigmaGR

mgp

the margin line for the axis title, axis labels and axis line. *positive numeric vector of size 3.*

main

an overall title for the plot. *character.*

window

the type of device on which the plot will be displayed. *logical, NULL or character.*

filename

the name of the file used to export the plot. *character.*

width

the width of the device used to export the plot. *positive numeric.*

height

the height of the device used to export the plot. *positive numeric.*

path

the directory where the plot file will be created. *character.*

unit

the units in which height and width are given. *character.*

res

the nominal resolution in ppi which will be recorded in the bitmap file. *positive integer.*

Details

ARGUMENTS:

Information about the window, filename, width, height, path, unit and res arguments can be found in the details section of *initWindow.*

Information about the mar and mgp arguments can be found in *par.*

FUNCTION:

The Growing Region step has been proposed by (Revol et al. 1997) and (Revol et al. 2002).

Value

An list containing:

- \[\text{[[df.criterion]]}\]: the value of the clustering criteria (in columns) for each sigma value (in rows). *numeric matrix.*
- \[\text{[[list.GR]]}\]: the list of the optimal GR sets, one for each clustering criteria.
- \[\text{[[best]]}\]: the optimal value of each clustering criteria. *data.frame.*

References


Examples

```r
## Not run:
## load a MRIaggr object
data(MRIaggr.Pat1_red, package="MRIaggr")

## select data
carto <- selectContrast(MRIaggr.Pat1_red,num=3,param=c("TTP_t0","MASK_DWI_t0"),hemisphere="lesion") coords <- selectCoords(MRIaggr.Pat1_red,num=3,hemisphere="lesion")
```
calcSmoothMask <- calcW(coords, distband = sqrt(2))

indexN <- which(carto$MASK_DWI_t0 == 1)
seed <- indexN[which.max(carto[indexN, "TP_t0")]]

## find optimal sigma
resGR_sigma <- calcSigmaGR(contrast = carto$TP_t0, W = W, seed = seed,
    sigma = seq(0.8, 2, 0.1), iter_max = 50)

## End(Not run)

calcSmoothMask Spatial regularization

Description

Perform a spatial regularization of a binary mask.

Usage

## S4 method for signature 'MRIaggr'
calcSmoothMask(object, mask = "mask", as.logical = FALSE,
    size_2Dgroup = 50, Neighborhood_2D = "3D_N8", rm.2Dhole = FALSE,
    size_3Dgroup = "unique", Neighborhood_3D = "3D_N10", rm.3Dhole = TRUE,
    erosion_th = 0.75, Vmask_min = 0.25, Vbackground_max = 0.75,
    Neighborhood_V = "3D_N10", trace = TRUE, update.object = FALSE, overwrite = FALSE)

Arguments

object an object of class MRIaggr. REQUIRED.
mask the binary contrast parameter that should be smoothed. character.
as.logical should mask be convert to logical? logical.
size_2Dgroup the minimum size of the 2D groups. positive integer or "unique".
Neighborhood_2D the type of 2D neighborhood. character.
rm.2Dhole should the 2D wholes inside the mask be removed? logical.
size_3Dgroup the minimum size of the 3D groups. positive integer or "unique".
Neighborhood_3D the type of 3D neighborhood. character.
rm.3Dhole should the 3D wholes inside the mask be removed? logical.
erosion_th the threshold below which the observations will be removed by the erosion. numeric between 0 and 1.
Vmask_min mask observations with a proportion of neighbors belonging to the mask lower than Vmask_min are affected to the background. numeric between 0 and 1.
**calcSmoothMask**

Vbackground\_max

background observations with a proportion of neighbors belonging to the mask higher than Vbackground\_max are affected to the mask. numeric between 0 and 1.

Neighborhood\_V

the type of neighborhood to use for the spatial regularization. character.

trace

should the execution of the function be traced? logical.

update.object

should the resulting regularized mask be stored in object? logical.

overwrite

if a mask is already stored in object\@data, can it be overwritten? logical.

Details

ARGUMENTS:

the Neighborhood\_2D or Neighborhood\_3D arguments can be a matrix or an array defining directly the neighborhood to use (i.e the weight of each neighbor) or a name indicating which type of neighborhood should be used (see the details section of initNeighborhood).

FUNCTION:

This function applies 6 smoothing steps:

- exclusion of the small 2D groups from the mask (to skip set size\_2Dgroup to FALSE). Note that size\_2Dgroup="unique" lead to keep the largest 2D group of each slice.
- filling of the small 2D holes in the mask (to skip set rm\_2Dhole to FALSE).
- exclusion of the small 3D groups from the mask (to skip set size\_3Dgroup to FALSE). Note that size\_3Dgroup="unique" lead to keep only the largest 3D group.
- erosion that first temporarily remove observations from the mask that have less than erosion\_th percent of their neighborhood in the mask. Then it computes the new 3D groups and remove permanently all the new 3D groups from the mask. To skip set erosion\_th to FALSE.
- filling of the small 3D holes in the mask (to skip set rm\_3Dhole to FALSE).
- spatial regularization that homogenize the local neighborhood (to skip set both Vmask\_min and Vbackground\_max to FALSE).

Value

An data.frame containing the mask and the coordinates in columns.

See Also

selectContrast to select the smoothed mask. calcBrainMask to compute an indicator of the brain observations.

Examples

```r
## load data and build MRIaggr
path <- system.file(file.path("nifti"),package = "MRIaggr")
ls.array <- list(readMRI(file=file.path(path,"T2\_GRE\_t0"),format="nifti"))
MRIaggr.Pat1 <- constMRIaggr(ls.array,identifier="Pat1",param="T2\_GRE\_t0")

## create the cerebral mask
```
calcTableHypoReperf

Compute reperfusion and hypoperfusion tables

Description

Compute the reperfusion and hypoperfusion values.

Usage

```r
res <- calcBrainMask(MRIaggr.Pat1,param="T2_GRE_t0",type="kmeans",
  kmeans.n_groups=2:4,
  update.object=TRUE,overwrite=TRUE)

## smooth the cerebral mask
res <- calcSmoothMask(MRIaggr.Pat1,update.object=TRUE,overwrite=TRUE)

## display
multiplot(MRIaggr.Pat1,param="mask",legend=FALSE)
```

Arguments

- `object` an object of class `MRIaggr`. REQUIRED.
- `param` the perfusion parameter(s). character vector. REQUIRED.
- `time` two time points. character vector of size 2. REQUIRED.
- `threshold` the value of the hypoperfusion thresholds. numeric vector.
- `sep` the separator between the parameter names and the time points. character.
- `mask` the binary contrast parameter indentifying the lesion at time[1]. character or NULL if no mask is available.
  - `as.logical` should mask be converted to logical ? logical.
- `norm_mu` the type of centering to apply on the parameter values. character.
- `norm_sigma` the type of scaling to apply on the parameter values. character.
- `trace` should the execution of the function be traced ? logical.
- `param.update` which type of parameter should be stored in the object ? Any of "shift" "reperf" "reperf_pc" "deperf" "deperf_pc".
- `update.object` should the resulting values be stored in object ? logical.
- `overwrite` if reperfusion or hypoperfusion values are already stored in object@table_reperfusion or object@table_hypofusion, can they be overwritten ? logical.
Details

ARGUMENTS:
Information about the norm_mu and norm_sigma arguments can be found in the details section of selectContrast.

FUNCTION:
If mask is set to NULL, no mismatch volume will not be computed.

Value

An list containing:

- `[[voxel]]`: a data.frame containing the coordinates and the reperfusion values.
- `[[volume_hypo]]`: the number of hypoperfused observations for the various thresholds.
- `[[volume_reperf]]`: the number of reperfused observations for the various thresholds.

See Also

calcThresholdMRIaggr to process contrast parameters.
selectTable to select the reperfusion/hypoperfusion tables.
calcW to compute the neighboring matrix.

Examples

```r
## load a MRIaggr object
data('MRIaggr.Pat1_red', package="MRIaggr")

### 1- directly ###
res <- calcTableHypoReperf(MRIaggr.Pat1_red, param=c("TTP","MTT"), time=c("t0","t1"),
                          mask="MASK_DWI_t0", as.logical=TRUE,
                          update.object=TRUE, overwrite=TRUE)

carto_TTP_t0 <- selectContrast(MRIaggr.Pat1_red, param="TTP_t0")
carto_TTP_t1 <- selectContrast(MRIaggr.Pat1_red, param="TTP_t1")

## hypoperfusion
sum( (carto_TTP_t0>=4) )
selectTable(MRIaggr.Pat1_red,"hypoperfusion")[4,"Vhypo.TTP_t0"]

## mismatch
testN <- (selectContrast(MRIaggr.Pat1_red,param="MASK_DWI_t0")==0)
sum( (carto_TTP_t0>=4)*testN )
selectTable(MRIaggr.Pat1_red,"hypoperfusion")[4,"Vmismatch.TTP"]

sum( (carto_TTP_t0>=4)*testN )/sum( testN==FALSE )
selectTable(MRIaggr.Pat1_red,"hypoperfusion")[4,"PCmismatch.TTP"]

## reperfusion
sum((carto_TTP_t0>=4)*(carto_TTP_t1<4))
```
selectTable(MRIaggr.Pat1_red,"reperfusion")["4","Vreperf.TTP"]

sum((carto_TTP_t0>4)*(carto_TTP_t1<4))/sum( (carto_TTP_t0>=4 )
selectTable(MRIaggr.Pat1_red,"reperfusion")["4","PCreperf.TTP"]

## W reperfusion
carto_TTPth_t0 <- carto_TTP_t0
carto_TTPth_t0[carto_TTPth_t0>10] <- 10
carto_TTPth_t0[carto_TTPth_t0<0] <- 0

carto_TTPth_t1 <- carto_TTP_t1
carto_TTPth_t1[carto_TTP_t1>10] <- 10
carto_TTPth_t1[carto_TTPth_t1<0] <- 0

weight <- (carto_TTPth_t0-carto_TTPth_t1)/carto_TTPth_t0
weight[ ((carto_TTPth_t0==0)+(carto_TTPth_t0<4)+(carto_TTPth_t1>=4) > 0 ] <- 0

sum((carto_TTP_t0>4)*(carto_TTP_t1<4)*weight)
selectTable(MRIaggr.Pat1_red,"reperfusion")["4","Vdeperf.TTP"]

sum((carto_TTP_t0<4)*(carto_TTP_t1<4))/sum( (carto_TTP_t0>4 )
selectTable(MRIaggr.Pat1_red,"reperfusion")["4","PCdeperf.TTP"]

## deperfusion
sum((carto_TTPth_t0<carto_TTPth_t1))/sum( (carto_TTPth_t0<4 )
selectTable(MRIaggr.Pat1_red,"reperfusion")["4","Vshift_reperf.TTP"]

sum((carto_TTPth_t0-carto_TTPth_t1)/sum( (carto_TTPth_t0<4 )
selectTable(MRIaggr.Pat1_red,"reperfusion")["4","PCshift_reperf.TTP"]

#### 2- via calcThresholdMRIaggr ####
## perform segmentation (call mritc)
## Not run:
calcTissueType(MRIaggr.Pat1_red,param="T1_t0",update.object=TRUE,overwrite=TRUE)

calcThresholdMRIaggr(MRIaggr.Pat1_red,param=c("TTP_t0","MTT_t0","TTP_t1","MTT_t1"),
threshold=1:10,name_newparam=c("TTP.GR_t0","MTT.GR_t0","TTP.GR_t1","MTT.GR_t1"),
rm.CSF=TRUE,hemisphere="lesion",
GRalgo=TRUE,seed=c("MASK_T2_FLAIR_t2","MASK_DWI_t0"),W=NULL,W.distband=sqrt(2),
update.object=TRUE,overwrite=TRUE)

res <- calcTableHypoReperf(MRIaggr.Pat1_red,param=c("TTP.GR","MTT.GR"),time=c("t0","t1"),
mask="MASK_DWI_t0",as.logical=TRUE,
update.object=TRUE,overwrite=TRUE)
calcTableLesion 53

## display
selectTable(MRIaggr.Pat1_red,"hypoperfusion")["4","Vhypo.TTP.GR_t0"]

par(mfrow=c(2,4),mar=rep(1.5,4),mgp=c(2,0.5,0))
multiplot(MRIaggr.Pat1_red,param="TTP_t0",num=1:3,
       palette=rainbow(10),window=NULL,main="raw - slice ",breaks=(0:10)-10^(-10))
multiplot(MRIaggr.Pat1_red,param="TTP.GR_t0",num=1:3,
       palette=rainbow(10),window=NULL,main="GR - slice ",breaks=(0:10)-10^(-10))

## End(Not run)

calcTableLesion | Vertical distribution of the lesion

### Description

Compute the number of lesion observations at each slice.

### Usage

```r
## S4 method for signature 'MRIaggr'
calcTableLesion(object,maskN,mask=NULL,as.logical=FALSE,
                 trace=TRUE,update.object=FALSE,overwrite=FALSE)
```

### Arguments

- **object**: an object of class `MRIaggr`. REQUIRED.
- **maskN**: the binary contrast parameter indicating the lesion. character. REQUIRED.
- **mask**: the binary contrast parameter indicating the brain. character or NULL if no mask is available.
- **as.logical**: should mask and maskN values be converted to logical ? logical.
- **trace**: should the execution of the function be traced ? logical.
- **update.object**: should the resulting lesion table be stored in object ? logical.
- **overwrite**: if a lesion table is already stored in object@table_lesion, can it be overwritten ? logical.

### Value

A `data.frame` containing the number of observation within the mask at each slice.
Examples

```r
## load a MRIaggr object
data("MRIaggr.Pat1_red", package="MRIaggr")

## compute table
res <- calcTableLesion(MRIaggr.Pat1_red, maskN=c("MASK_DWI_t0","MASK_T2 flair_t2"),
                        as.logical=TRUE, update.object=TRUE, overwrite=TRUE)

## extract table
res <- selectTable(MRIaggr.Pat1_red,"lesion")
```

---

calcThreshold  

*Image thresholding*

Description

Threshold a contrast parameter at one or several values.

Usage

```r
## S4 method for signature 'MRIaggr'
calcThresholdMRIaggr(object,param,hemisphere="both",rm.CSF=FALSE,
                      threshold=1:10,decreasing=FALSE,GRalgo=FALSE,W=NULL,seed=NULL,
                      as.logical=FALSE,W.distband=NULL,W.spatial_res=rep(1,3),
                      name_newparam=paste(param,"Th",sep="_"),trace=TRUE,
                      update.object=FALSE,overwrite=FALSE)
```

```r
calcThreshold(contrast,param,hemisphere=NULL,rm.CSF=FALSE,threshold=1:10,
              decreasing=FALSE,GRalgo=FALSE,W=NULL,seed=NULL,as.logical=FALSE,
              trace=TRUE)
```

Arguments

- **object**  
  an object of class `MRIaggr`. REQUIRED.

- **contrast**  
  the dataset containing the contrast parameter to be thresholded. matrix. REQUIRED.

- **param**  
  the contrast parameters to be thresholded. character vector. REQUIRED.

- **hemisphere**  
  the hemisphere to consider. character or NULL.

- **rm.CSF**  
  should the cerebral spinal fluid observations be excluded? logical or character.

- **threshold**  
  the thresholds to be used for the discretization of the contrast parameter. numeric vector.

- **decreasing**  
  should the increasing thresholding (FALSE) or decreasing thresholding (TRUE) be used. logical.

- **GRalgo**  
  should a Growing Region algorithm be used to clean the thresholded parameter? logical.
calcThreshold

W
the neighborhood matrix. dgCMatrix.

seed
the index of the seeds for the growing region algorithm. positive integer vector.

as.logical
should seed be converted to logical ? logical.

W.distband
only distances smaller than delta are recorded (for the computation of W). positive numeric.

W.spatial_res
da dilatation factor for the coordinates (for the computation of W). positive numeric vector of size 3.

name_newparam
the name of the new parameters. character.

trace
should the execution of the function be traced ? logical.

update.object
should the resulting thresholded parameters be stored in object ? logical.

overwrite
contrast parameters with the same names are already stored in object@data, can it be overwritten ? logical.

Details

ARGUMENTS:
[data.frame method] the hemisphere argument must be one of the levels present in the column named “hemisphere” in data. NULL leading to use all observations.
[MRIaggr method] Information about the hemisphere arguments can be found in the details section of selectContrast.

FUNCTION:
[data.frame method] By default the indicator of CSF will be extract from the column named CSF. If it is contained in another column the user must specify rm.CSF with the name of this column.
[MRIaggr method] Setting rm.CSF to TRUE require to have the corresponding parameter (by default CSF) stored in the object. It can be done using calcTissueType.

References


See Also

selectContrast to select contrast parameters in the MRIaggr object.

Examples

```r
## load a MRIaggr object
data(MRIaggr.Pat1_red, package="MRIaggr")
## Not run:
calcTissueType(MRIaggr.Pat1_red,param="T1_t0",update.object=TRUE,overwrite=TRUE)
## End(Not run)

#### 1- MRIaggr method
## raw parameter
multiplot(MRIaggr.Pat1_red,param="TTP_t0",legend=FALSE,main="TTP_t0 - slice ",
  palette=rainbow(10),breaks=seq(0,10)-10*(-10))
```
## thresholded parameter
calcThresholdMRIaggr(MRIaggr.Pat1_red.param=c("TTP_t0","MTT_t0"), threshold=1:10, 
name_newparam=c("TTP.th_th0","MTT.th_th0"), 
update.object=TRUE, overwrite=TRUE)

multiplot(MRIaggr.Pat1_red.param="TTP.th_th0", main="TTP.th_th0 - slice", 
legend=FALSE, palette=rainbow(10), breaks=(0:10)-10\(-10)\)

## Not run:
## 1st correction
calcThresholdMRIaggr(MRIaggr.Pat1_red.param=c("TTP_t0","MTT_t0"), threshold=1:10, 
rm.CSF=TRUE, hemisphere="lesion", name_newparam=c("TTP.red_t0","MTT.red_t0"), 
update.object=TRUE, overwrite=TRUE)

multiplot(MRIaggr.Pat1_red.param="TTP.red_t0", main="TTP.red_t0 - slice", 
legend=FALSE, palette=rainbow(10), breaks=(0:10)-10\(-10)\)

## 2nd correction
calcThresholdMRIaggr(MRIaggr.Pat1_red.param=c("TTP_t0","MTT_t0"), threshold=1:10, 
rm.CSF=TRUE, hemisphere="lesion", name_newparam=c("TTP.red_t0","MTT.red_t0"), 
GRalgo=TRUE, seed=c("MASK_T2_FLAIR_t2","MASK_DWI_t0"), W=NULL, W.distband=sqrt(2), 
update.object=TRUE, overwrite=TRUE)

multiplot(MRIaggr.Pat1_red.param="TTP.red_t0", main="TTP.GR_t0 - slice", 
legend=FALSE, palette=rainbow(10), breaks=(0:10)-10\(-10)\)

### 2- data.frame function ###
## raw parameter
multiplot(MRIaggr.Pat1_red.param="TTP_t0", legend=FALSE, main="TTP_t0 - slice ", 
palette=rainbow(10), breaks=seq(0,10)-10\(-10)\)

## thresholded parameter
data <- selectContrast(MRIaggr.Pat1_red.param=c("TTP_t0","MTT_t0","hemisphere","CSF","WM","GM"))
hypo_th1_10 <- calcThreshold(data, param=c("TTP_t0","MTT_t0"), threshold=1:10)

multiplot(selectCoords(MRIaggr.Pat1_red), main="TTP_t0_th - slice ", 
hypo_th1_10[,1], legend=FALSE, palette=rainbow(10), breaks=(0:10)-10\(-10)\)

## 1st correction
data$CSF <- as.numeric(apply(data[,c("CSF","WM","GM")],1, which.max)==1)

hypoC_th1_10 <- calcThreshold(data, param=c("TTP_t0","MTT_t0"), threshold=1:10, 
hemisphere="left", rm.CSF=TRUE)

multiplot(selectCoords(MRIaggr.Pat1_red), main="TTP_t0_thC - slice ", 
hypoC_th1_10[,1], legend=FALSE, palette=rainbow(10), breaks=(0:10)-10\(-10)\)

## 2nd correction
maskN <- c("MASK_T2_FLAIR_t2","MASK_DWI_t0")
data[,maskN] <- selectContrast(MRIaggr.Pat1_red, param=maskN)
W <- calcW(MRIaggr.Pat1_red, distband=sqrt(2*1.875*2+0.001), row.norm=TRUE, upper=NULL,
calcTissueType

spatial_res=c(1.875, 1.875, 6))

max(rowSums(W>0))

hypoCC_Th1_10 <- calcThreshold(data, param=c("TPP_t0","MTT_t0"), threshold=1:10,
   hemisphere="left", rm.CSF=TRUE,
   Gralgo=TRUE, seed=c("MASK_T2\_FLAIR_t2","MASK_DWI_t0"), W=W)

multiplot(selectCoords(MRIaggr.Pat1_red), main="TPP_t0\_thCC - slice",
   hypoCC_Th1_10[,1], legend=FALSE, palette=rainbow(10), breaks=(0:10)-10^(0:10))

## End(Not run)

calcTissueType

### Probabilistic tissue type segmentation

**Description**

Perform a probabilistic segmentation of the voxel in Cerebro Spinal Fluid, White Matter and Grey Matter classes.

**Usage**

```
## S4 method for signature 'MRIaggr'
calcTissueType(object, param, niter=100, nnei=6,
   beta=if(sub=TRUE)\{0.3\}else\{0.7\}, sub=TRUE, digit=0, trace=TRUE,
   name_newparam=c("CSF","GM","WM"), update.object=FALSE, overwrite=FALSE)
```

**Arguments**

- `object`: an object of class `MRIaggr`. **REQUIRED.**
- `param`: the contrast parameter that should be used to distinguish the WM, the GM and the CSF. **character. REQUIRED.**
- `niter`: the number of iterations used by `mritc.bayes`. **positive integer.**
- `nnei`: the number of neighbors. **positive integer.**
- `beta`: the parameter 'inverse temperature' of the Potts model. **numeric.**
- `sub`: if TRUE, use the higher resolution model; otherwise, use the whole voxel method. **logical.**
- `digit`: the number of decimal places to use for the initialization. **positive integer.**
- `trace`: indicate the level of output as the algorithm runs. **logical.**
- `name_newparam`: the name of the new parameters containing the probabilistic segmentation. **character vector of size 3.**
- `update.object`: should the resulting tissue types be stored in `object`? **logical.**
- `overwrite`: if tissue types are already stored in `object@data`, can they be overwritten? **logical.**
Details

ARGUMENTS:
Information about the nnei and sub arguments can be found in `makeMRIspatial`.
Information about the niter and beta arguments can be found in `mritic`.

FUNCTION:
This function uses the `mritic.bayes` function of the `mritic` package to compute the probabilistic segmentation. T1 sequence is the recommended sequence to identify the various tissue types but T2 gradient echo may also be used.

The initialization function `initOtsu` was found much more slower when the contrast parameter values have a large number of digits. The `digit` argument enables to round the contrast values only for the computation of the initialization values.

Value

An `list` containing:

- `[[prob]]`: the group membership of each voxel for each class. `matrix`.
- `[[mu]]`: the mean value of each class. `numeric vector`.
- `[[sigma]]`: the standard deviation of each class. `numeric vector`.

Examples

```r
## load a MRIaggr object
data("MRIaggr.Pat1_red", package="MRIaggr")

## Not run:
## perform segmentation (call mritic)
calcTissueType(MRIaggr.Pat1_red, param="T1_t0", update.object=TRUE, overwrite=TRUE)

calcTissueType(MRIaggr.Pat1_red, param="T1_t0", update.object=TRUE, overwrite=TRUE)

calcW

## display
multiplot(MRIaggr.Pat1_red, num=1,
          param=c("CSF","WM","GM"), legend=FALSE,
          palette="rgb")

## End(Not run)
```

Description

Computate a neighborhood matrix using spatial coordinates.
Usage

```r
calcW(object, distband, method = "euclidean", upper = NULL,
   format = "dgCMatrix", row.norm = FALSE, spatial_res = rep(1, ncol(object)))
```

Arguments

- **object**: A `data.frame` containing the coordinates of the observations or an object of class `MRIaggr`. REQUIRED.
- **distband**: only distances smaller than delta are recorded. positive numeric. REQUIRED.
- **method**: the distance measure to be used. character. This must be one of "euclidean", "maximum", "minkowski" or "greatcircle".
- **spatial_res**: a dilatation factor for the coordinates. positive numeric vector of size 3.
- **num**: the slices to use. numeric vector or NULL.
- **hemisphere**: the hemisphere to use. character.
- **subset**: the subset of observations to use. positive integer vector or NULL leading to use all observations.
- **upper**: should the entire matrix (NULL) or only the upper-triangular (TRUE) or only the lower-triangular (FALSE) values be calculated?
- **format**: the format of the output. Could be "spam" or "dgCMatrix".
- **row.norm**: should the resulting matrix be row-normalized? TRUE/FALSE.
- **trace**: should the execution of the function be traced? logical.
- **update.object**: should the resulting neighborhood matrix be stored in object? logical.
- **overwrite**: if a neighborhood matrix is already stored in object@ls_descStats, can it be overwritten? logical.

Details

ARGUMENTS:
Information about the num argument can be found in the details section of `initNum`
Information about the distband argument can be found in `nearest.dist`
Information about the hemisphere argument can be found in the details section of `selectContrast`.
The row.norm argument is ignored if format is set to "spam".

FUNCTION:
This function relies on the nearest.dist function of the spam package.
Each of the num, hemisphere and subset argument define a subset of the total set of observations. It is the intersection of all these three subsets that is extracted.
Value

A `spam` or `dgCMatrix` object.

Examples

```r
## load a MRIaggr object
data("MRIaggr.Pat1_red", package="MRIaggr")

### 1- data.frame method ###
coords <- selectCoords(MRIaggr.Pat1_red,num=1:3,hemisphere="lesion")

# full W
W <- calcW(object=coords,distband=sqrt(2))
W[1:10,1:10]
table(rowSums(W))

# full W normalized by row
W <- calcW(object=coords,distband=sqrt(2),row.norm=TRUE)
W[1:10,1:10]
table(rowSums(W))

# upper W
W <- calcW(object=coords,distband=sqrt(2),upper=TRUE)
W[1:10,1:10]

### 2- data.frame method ###

# compute W (regular lattice)
W <- calcW(MRIaggr.Pat1_red,distband=sqrt(2),upper=NULL,num=1:3,hemisphere="lesion")
table(rowSums(W>=0))

# compute W (irregular lattice)
W <- calcW(MRIaggr.Pat1_red,distband=sqrt(2*1.875*2),upper=NULL,num=1:3,hemisphere="lesion",
            spatial_res=c(1.875,1.875,6))
table(rowSums(W>=0))
```

---

**Carto3D-class**

class "Carto3D"

Description

Patient-specific storage of univariate contrast data.

Arguments

- **identifier**: the patient identifier. `character`.
- **parameter**: the name of the contrast parameter. `character`.
- **contrast**: the contrast and the spatial coordinates of each voxel. `data.frame` with four columns named "i" "j" "k" and the name of the contrast parameter.
voxelDim the spatial dimensions of the lattice containing the observations. *data.frame* with 1 row and 3 columns named "i" "j" "k".

default_value the reference values of the contrast parameters (e.g. the background value). *character*.

**S4 methods**

- **selectIdentifier** return the slot identifier.
- **selectParameter** return the slot parameter.
- **selectContrast** return the slot carto3D.
- **selectVoxelDim** return the slot voxelDim.
- **selectDefault_value** return the slot default_value.
- **multiplot** display the contrast data.
- **constMRIaggr** aggregate several Carto3D objects into a single objet MRIaggr.
- **initNum** check the validity of the num argument.

**See Also**

- MRIaggr class that aggregates several carto3D objects. Carto3D2MRIaggr to convert carto3D objects into MRIaggr objects.

---

**Carto3D2MRIaggr**  
*Carto3D to MRIaggr converter*

**Description**

Construct a MRIaggr object by aggregating Carto3D objects.

**Usage**

```r
Carto3D2MRIaggr(ls.Cart03D,rm.Cart03D=FALSE, tol=10^-10, num=NULL, trace=TRUE)
```

**Arguments**

- **ls.Cart03D** a list of Carto3D objects. REQUIRED.
- **rm.Cart03D** should the object on which the ls.Cart03D argument points be removed form the global environment? logical.
- **tol** numeric precision for the consistency check. positive numeric.
- **num** the slices to extract. numeric vector or NULL.
- **trace** should the execution of the function be traced? logical.
Details

ARGUMENTS:
Information about the num argument can be found in the details section of initNum.

Value

A MRIaggr object.

Examples

```r
## load nifti files
path <- system.file(file.path("nifti"), package = "MRIaggr")

Pat1.TTP.t0.nifti <- readMRI(file=file.path(path,"TTP_t0"),format="nifti")
Pat1.DWI.t0.nifti <- readMRI(file=file.path(path,"DWI_t0"),format="nifti")
Pat1.MASK_DWI.t0.nifti <- readMRI(file=file.path(path,"MASK_DWI_t0"),format="nifti")
Pat1.MASK_T2_FLAIR.t2.nifti <- readMRI(file=file.path(path,"MASK_T2_FLAIR_t2"),format="nifti")

## convert them to Carto3D
Pat1.TTP.t0.Carto3D <- constCarto3D(Pat1.TTP.t0.nifti,
                                     identifier="Pat1",param="TTP_t0",default_value=NA)
Pat1.DWI.t0.Carto3D <- constCarto3D(Pat1.DWI.t0.nifti,
                                     identifier="Pat1",param="DWI_t0",default_value=NA)
Pat1.MASK_DWI.t0.Carto3D <- constCarto3D(Pat1.MASK_DWI.t0.nifti,
                                     identifier="Pat1",param="MASK_DWI_t0",default_value=NA)
Pat1.MASK_T2_FLAIR.t2.Carto3D <- constCarto3D(Pat1.MASK_T2_FLAIR.t2.nifti,
                                     identifier="Pat1",param="MASK_T2_FLAIR_t2",default_value=NA)

## convert Carto3D to MRIaggr
MRIaggr.Pat1 <- Carto3D2MRIaggr(list(Pat1.TTP.t0.Carto3D,
                                          Pat1.DWI.t0.Carto3D,
                                          Pat1.MASK_DWI.t0.Carto3D,
                                          Pat1.MASK_T2_FLAIR.t2.Carto3D))
```

constCarto3D  

Array constructor for Carto3D objects

Description

Creates a Carto3D object from an array.

Usage

```r
constCarto3D(array,identifier,param,default_value=NULL,
pos_default_value=c(1,1,1),rm.array=FALSE)
```
null
Arguments

object: an object of class \texttt{MRIaggr}. REQUIRED.

factor: the compression factor. \textit{positive integer}. REQUIRED.

param: the contrast parameters to load in the new MRIaggr object. \textit{character vector} or \texttt{NULL}.

mask: the binary contrast parameter(s). \textit{character vector}.

threshold: the value above which the local mean of the binary parameters is assigned to 1 (and otherwise to 0). \textit{numeric between 0 and 1}.

trace: should the execution of the function be traced? \textit{logical}.

Details

ARGUMENTS:
Information about the \texttt{param} argument can be found in the details section of \texttt{initParameter}.

FUNCTION:
The function uses a local mean to compress the initial parameters maps to a lower resolution.

Value

A \texttt{MRIaggr} object.

Examples

```r
# load a MRIaggr object
data("MRIaggr.Pat1_red", package="MRIaggr")

# compress the MRIaggr object
MRIaggr.compressed <- constCompressMRIaggr(MRIaggr.Pat1_red, factor=2,
                                            param=c("DWI_t0","T2_FLAIR_t2","MASK_T2_FLAIR_t2"),
                                            mask="MASK_T2_FLAIR_t2")

# display
par(mfrow=c(2,4),mar=rep(1.75,4),mgp=c(2,0.75,0))
multiplot(MRIaggr.Pat1_red,param="DWI_t0",window=NULL,breaks=seq(0,350,1),
          midplane=TRUE,main="before - slice ")
multiplot(MRIaggr.compressed,param="DWI_t0",window=NULL,breaks=seq(0,350,1),
           midplane=TRUE,main="after - slice ")
multiplot(MRIaggr.Pat1_red,param="MASK_T2_FLAIR_t2",main="before - slice ")
multiplot(MRIaggr.compressed,param="MASK_T2_FLAIR_t2",main="after - slice ")
```
**Description**

Construct a `MRIaggr` object from a list of array, each array corresponding to a different contrast parameters.

**Usage**

```r
cstMRIaggr(ls.array, identifier, param, default_value = NULL,
    pos_default_value = c(1,1,1), tol = 10^{-10}, trace = TRUE, rm.ls.array = FALSE)
```

**Arguments**

- `ls.array`: the value of the contrast parameter(s) for each observation. list of array. REQUIRED.
- `identifier`: the identifier of the patient to which belong the contrast parameters. character. REQUIRED.
- `param`: the contrast parameter(s). character vector or NULL. REQUIRED.
- `default_value`: the reference values of the contrast parameters (e.g. the background value). character or NULL leading to search the reference value in `array[pos_default_value]`.
- `pos_default_value`: the coordinates of the observations that contains the reference value. numeric vector.
- `tol`: numeric precision for the consistency check. positive numeric.
- `trace`: should the execution of the function be traced? logical.
- `rm.ls.array`: should the object on which the `ls.array` argument points be removed form the global environment? logical.

**Details**

ARGUMENTS:
All the array in `ls.array` in must have the same dimensions.

Information about the `param` argument can be found in the details section of `initParameter`.

`pos_default_value` is active only if `default_value` is set to NULL.

**Value**

A `MRIaggr` object.
Examples

### 1- 1st method ###

```r
# load nifti files
path <- system.file(file.path("nifti"), package = "MRIaggr")

nifti.Pat1_TTP_t0 <- readMRI(file=file.path(path, "TTP_t0"), format="nifti")
nifti.Pat1_DWI_t0 <- readMRI(file=file.path(path, "DWI_t0"), format="nifti")
nifti.Pat1_MASK_DWI_t0 <- readMRI(file=file.path(path, "MASK_DWI_t0"), format="nifti")
nifti.Pat1_MASK_T2_FLAIR_t2 <- readMRI(file=file.path(path, "MASK_T2_FLAIR_t2"), format="nifti")

# convert them to MRIaggr
MRIaggr.Pat1 <- constMRIaggr(list(nifti.Pat1_TTP_t0, nifti.Pat1_DWI_t0,
                                 nifti.Pat1_MASK_DWI_t0, nifti.Pat1_MASK_T2_FLAIR_t2),
                                 identifier="Pat1", param=c("TTP_t0", "DWI_t0", "MASK_DWI_t0", "MASK_T2_FLAIR_t2"))
```

### 2- 2nd method ###

```r
# load nifti files
param <- c("DWI_t0.nii", "MASK_DWI_t0.nii", "MTT_t0.nii", "TTP_t0.nii", "T1_t0.nii", "T2_GRE_t0.nii",
            "MTT_t1.nii", "TTP_t1.nii", "T2_FLAIR_t2.nii", "MASK_T2_FLAIR_t2.nii")

ls.array <- list()
for(iter_param in 1:length(param)){
  ls.array[[iter_param]] <- readMRI(file=file.path(path, param[iter_param]), format="nifti")
}

# convert them to MRIaggr
param <- gsub(".nii", ",", param)

MRIaggr.Pat1 <- constMRIaggr(ls.array, identifier="Pat1", param=param)
```

### additional examples

#### Not run:

```r
# load an analyse file (example of oro.nifti::readANALYZE)
path <- system.file("anlz", package="oro.nifti")
analyse.avg <- readMRI(file.path(path, "avg15T1"), format="analyze")
analyse.MRIaggr <- constMRIaggr(analyse.avg, param="avg", identifier="Pat1")
```

#### load a nifti file (example of oro.nifti::readNIFTI)

```r
nifti.ffd <- readMRI(file.path(system.file("nifti", package="oro.nifti"), "filtered_func_data"),
                     format="nifti")
nifti.MRIaggr <- constMRIaggr(nifti.ffd[, , 32, drop=FALSE], param="ffd", identifier="Pat1")
```

#### load a dicom file (examples of oro.dicom::readDICOMFile)

```r
dicom.Abdo <- readMRI(system.file("dcm/Abdo.dcm", package="oro.dicom"), format="dicom")
dicom.MRIaggr <- constMRIaggr(dicom.Abdo, param="Abdo", identifier="Pat1")
```

### End(Not run)
Description

Construct a MRIaggr object restricted to a subset of observations.

Usage

```r
## S4 method for signature 'MRIaggr'
constReduceMRIaggr(object, mask, as.logical=FALSE, keep.index=TRUE)
```

Arguments

- `object`: an object of class MRIaggr. REQUIRED.
- `mask`: the binary contrast parameter or a vector indicating the observations to be kept. character or logical vector with length equal to the number of observations in object. REQUIRED.
- `as.logical`: should mask be converted to logical? logical.
- `keep.index`: should the previous index parameter be saved in the ls_descStats slot? logical.

Value

A MRIaggr object.

See Also

calcBrainMask to compute an indicator of the brain observations.

Examples

```r
## load nifti files and convert them to MRIaggr
path <- system.file(file.path("nifti"), package = "MRIaggr")
ls.array <- list(readMRI(file=file.path(path,"T2_GRE_t0"),format="nifti"))
MRIaggr.Pat1 <- constMRIaggr(ls.array,identifier="Pat1",param="T2_GRE_t0")

## create the cerebral mask

## Not run:
res <- calcBrainMask(MRIaggr.Pat1,param="T2_GRE_t0",type="kmeans",
                      kmeans.n_groups=2:4,
                      update.object=TRUE,overwrite=TRUE)
res <- calcSmoothMask(MRIaggr.Pat1,update.object=TRUE,overwrite=TRUE)

## End(Not run)
## display
multiplot(MRIaggr.Pat1,param="mask",legend=FALSE)

## construct the reduced object
MRIaggr.Pat1_red <- constReduceMRIaggr(MRIaggr.Pat1,mask="mask")

## display
constsweave

Constructor for Sweave report

Description

Construct a Sweave report that includes plots generated by the `multiplot` function.

Usage

```r
constsweave(dir=NULL, identifier=NULL, param=NULL, table=NULL, extra_text=NULL,
             subsection=NULL, index_subsection=NULL, subsubsection=NULL,
             index_subsubsection=NULL, legend=NULL, trace=TRUE,
             width=list(0.9, 0.9, 0.9), trim=list(c(0, 0, 0, 0),
             c(0, 0, 0, 0), c(0, 160, 0, 0)), width.legend=0.35,
             trim.legend=c(0, 0, 0, 0), title="", date="", author="")
```

Arguments

dir the path to the root(s) directory(ies) containing the image files. `character` or `character vector`. REQUIRED.

identifier the identifiers of the patients for which the graphics should be displayed. `character vector` or NULL leading to use all patients.

param the names of directories containing the images. `character vector` or NULL leading to use all directories.

table a list of data.frame to display in the table format. `list of data.frame` or NULL if there is no table to display.

extra_text additionnal text to display. `list of character vector` or NULL if there is no extra text to display.

subsection the names of subsections for the latex document. `character vector` or NULL leading to use `param` for naming the subsections.

index_subsection the position of the images in the subsections. `numeric vector` or NULL leading to use a subsection for each image.

subsubsection A list containing the names of the subsubsections for the latex document. `list of character vector` or NULL leading to no subsubsection.

index_subsubsection A list of positions of the images in the subsubsections. `list of numeric vector` or NULL.

legend the legend of each image. `character vector` or NULL leading to use `param` for the legend.
trace should the execution of the function be traced? *logical*.

width the width of each image relative to the linewidth. *list of positive numeric*.

trim the length in mm with which the imported images will be cropped (left, bottom, right top). *list of numeric vector of size 4*.

width.legend the width of the legend image relative to the linewidth. *numeric between 0 and 1*.

trim.legend the length in mm with which the legend of the images will be cropped. *numeric vector of size 4*.

title the title of the latex document. *character*.

date the date on the latex document. *character*.

author the author of the latex document. *character*.

Details

ARGUMENTS:
table must contains in its first column the patient identifiers. This column will not be display.

The width argument must be a *list* containing the width used to display each image file. For instance if width=list(0.9,0.9,0.9) the first image file (slices 1 to 9) will be display with a width of 0.9*linewidth, as well as the second and third image file.

index_subsubsection has to be modified if subsubsections are defined by the user.

FUNCTION:
Patient identifiers are read from the file names: it is the character string preceding the first underscore (“_”).

The function generate latex code that reads the images using the *includegraphicx* latex function. In particular arguments width, trim, width.legend and trim.legend are used by this function to adjust the display of the images.

Value

A *list* containing:

- *[[text.preamble]]*: a *character* containing the preamble of a latex document.
- *[[text.begin]]*: a *character* for beginning the latex report.
- *[[ls.text]]*: a *list of character*, each containing the content of the report for each patient.
- *[[text.end]]*: a *character* for ending the latex report.
- *[[ls.plot]]*: a *list* containing how the plots has been managed.
- *[[ls.legend]]*: a *list* containing how the legends has been managed.
- *[[ls.newplot]]*: a *list* containing how the figures has been managed.
Examples

```r
## Not run:
## load a mriaggr object
data("MRIaggr.Pat1_red", package="MRIaggr")

## directories for storage
if(!"Display" %in% list.files()) == FALSE){dir.create("Display")}
if(!"DWI" %in% list.files("Display")) == FALSE){dir.create("Display/DWI")}
if(!"DWI_lesion" %in% list.files("Display")) == FALSE){dir.create("Display/DWI_lesion")}
if(!"T2" %in% list.files("Display")) == FALSE){dir.create("Display/T2")}

## plot generation
multiplot(MRIaggr.Pat1_red,param="DWI_t0", window="png",path="Display/DWI/")
multiplot(MRIaggr.Pat1_red,param="DWI_t0", index1=list(coords="MASK_T2_FLAIR_t2",outline=TRUE),as.logical=TRUE,
window="png",path="Display/DWI_lesion/")
multiplot(MRIaggr.Pat1_red,param="T2_FLAIR_t2", window="png",path="Display/T2/")

MRIaggr.Pat1_red@identifier <- "Pat2"

multiplot(MRIaggr.Pat1_red,param="DWI_t0", window="png",path="Display/DWI/")
multiplot(MRIaggr.Pat1_red,param="DWI_t0", index1=list(coords="MASK_T2_FLAIR_t2",outline=TRUE),as.logical=TRUE,
window="png",path="Display/DWI_lesion/")
multiplot(MRIaggr.Pat1_red,param="T2_FLAIR_t2", window="png",path="Display/T2/")

## Sweave generation

tablePat <- list(cbind(Id=MRIaggr.Pat1_red@identifier, selectclinic(MRIaggr.Pat1_red,param=c("Age","Gender"))),
                 cbind(Id=MRIaggr.Pat1_red@identifier, selectclinic(MRIaggr.Pat1_red,param=c("FinalStroke_volume","AcuteStroke_volume")))
)

res <- constSweave(dir="Display",
                   table=tablePat)
cat(res$ls.text[[1]],sep="")

## Sweave doc

```
**df2array**

---

**data.frame to array converter**

**Description**

Convert observations stored in the data.frame format into the array format.

**Usage**

```
df2array(contrast, coords, format="any", default_value=NA, range.coords=NULL)
```

**Arguments**

- **contrast** the dataset containing the observations in rows and the contrast parameters in columns. vector or data.frame. REQUIRED.
coords the spatial coordinates of the observations. matrix with a number of rows equal to the number of rows of data. REQUIRED.

format the format of the output. Can be "any","matrix","data.frame" or "list".

default_value the element used to fill the missing observations. numeric.

range.coords the maximum coordinate in each dimension to be considered. numeric vector with length equal to the number of columns of coords.

Details

FUNCTION:
If contrast contains several parameters, they are treated one at a time and the result is returned in the form of a list. If range.coords is NULL then the maxima coordinates are those of the coords argument. If only one parameter is specified and the format is set to "any" then a vector is returned.

Value

a list containing:

- [[contrast]]: list containing the new contrast in the new format.
- [[coords]]: a data.frame containing the coordinates of each observation.
- [[unique_coords.group]]: two list containing the possibles coordinates in each dimension.

Examples

```r
### 1- with simulated data ###
# simulate
set.seed(10)
n <- 4
Y <- rnorm(n^2)

# conversion
res1 <- df2array(contrast=Y,coords=expand.grid(1:n+0.5,1:n+0.5))
res2 <- df2array(contrast=Y,coords=expand.grid(1:n,1:n),format="matrix")
res3 <- df2array(contrast=Y,coords=expand.grid(2*(1:n),2*(1:n)))
res4 <- df2array(contrast=cbind(Y,Y,Y),coords=expand.grid(2*(1:n),2*(1:n)),
range.coords=c(10,10))

# display
par(mfrow=c(2,2),mar=rep(2,4),mgp=c(1.5,0.5,0))
fields::image.plot(unique(res1$coords[,1]),unique(res1$coords[,2]),res1$contrast[[1]],
xlab="",ylab="")
fields::image.plot(unique(res2$coords[,1]),unique(res2$coords[,2]),res2$contrast,
xlab="",ylab="")
fields::image.plot(res3$contrast[[1]])
fields::image.plot(res4$contrast[[2]])
```

### 2- with MRIaggr data ###
# load a MRIaggr object
data("MRIaggr.Pat1_red",package="MRIaggr")
carto <- selectContrast(MRIaggr.Pat1_red,param="DWI_t0")
coords <- selectCoords(MRIaggr.Pat1_red)

## conversion 1
array.DWI_t0 <- df2array(carto, coords=coords, default_value=1000)$contrast[[1]]

# display
fields::image.plot(array.DWI_t0[,1])

## conversion 2
array.DWI_t0 <- df2array(carto, coords=coords, default_value=1000,
                         range.coords = c(256,256,6))$contrast[[1]]

# display
fields::image.plot(array.DWI_t0[,1])

---

**EDK**

---

**Gaussian kernel**

**Description**

Apply a gaussian kernel to observations. For internal use.

**Usage**

```r
EDK(x, distband, power=2)
```

**Arguments**

- `x` the data on which the kernel will be applied. *numeric* or *numeric vector*.
- `distband` the distband of the kernel. *numeric*.
- `power` the power of the kernel. *numeric*.

**Details**

**FUNCTION:**

This function corresponds to the following kernel:

\[
\frac{1}{\sqrt{2 \pi \cdot \text{distband}^2}} \cdot \exp\left(-\frac{x}{\text{distband}^\text{power}}\right)
\]

Setting `power` to 2 lead to use a gaussian kernel.

**Value**

A *numeric*. 
filtrage2Dmed_cpp  2D median filtering

Description
C++ function call by calcFilter that performs two dimensional median filtering. For internal use.

Usage
filtrage2Dmed_cpp(M_data, M_operateur, index_data, na_rm)

Arguments
- M_data: matrix to which the filter will be applied.
- M_operateur: the filter to be applied.
- index_data: index of the non NA data.
- na_rm: should the observations with missing values in their neighborhood be removed? Otherwise the ponderation is adjusted.

filtrage2D_cpp  2D filtering

Description
C++ function call by calcFilter that performs two dimensional filtering. For internal use.

Usage
filtrage2D_cpp(M_data, M_operateur, index_data, w_contrast, na_rm)

Arguments
- M_data: matrix to which the filter will be applied.
- M_operateur: the filter to be applied.
- index_data: index of the non NA data.
- w_contrast: should the influence of each neighbor be ponderated by the difference in signal with the considered observation?
- na_rm: should the observations with missing values in their neighborhood be removed? Otherwise the ponderation is adjusted.
filtrage3Dmed_cpp 3D median filtering

Description

C++ function call by calcFilter that performs three dimensional median filtering. For internal use.

Usage

filtrage3Dmed_cpp (Vec_data, p_data, Vec_operateur, p_operateur, index_data, na_rm)

Arguments

Vec_data vector of data to which the filter will be applied.
p_data spatial dimensions of the data.
Vec_operateur vector representing the filter to be applied.
p_operateur spatial dimensions of the filter.
index_data index of the non NA data.
na_rm should the observations with missing values in their neighborhood be removed? Otherwise the ponderation is adjusted.

filtrage3D_cpp 3D filtering

Description

C++ function call by calcFilter that performs three dimensional filtering. For internal use.

Usage

filtrage3D_cpp (Vec_data, p_data, Vec_operateur, p_operateur, index_data, na_rm, w_contrast)

Arguments

Vec_data vector of data to which the filter will be applied.
p_data spatial dimensions of the data.
Vec_operateur vector representing the filter to be applied.
p_operateur spatial dimensions of the filter.
index_data index of the non NA data.
w_contrast should the influence of each neighbor be ponderated by the difference in signal with the considered observation?
na_rm should the observations with missing values in their neighborhood be removed? Otherwise the ponderation is adjusted.
GRalgo  

_Growing Region algorithm_

Description

Perform the Growing Region algorithm proposed by (Revol et al., 1997). For internal use.

Usage

GRalgo(contrast, W, seed, sigma_max, range, breaks, step, operator, iter_max,
history_sigma, history_step, history_front)

Arguments

- **contrast**  
  the contrast value of each observation. *numeric vector.*

- **W**  
  the neighborhood matrix. *dgCMatrix.*

- **seed**  
  the index of the initial seeds or a binary indicator of the initial seeds. *positive integer vector* or *logical vector.*

- **sigma_max**  
  the maximum admissible value for the variability of the group contrast. *positive numeric.*

- **range**  
  the range of acceptable contrast values for the growing region group. *numeric vector of size 2.*

- **breaks**  
  the break points to use to categorize the contrast distribution. *numeric vector.*

- **step**  
  the step between two consecutive breaks. *numeric.*

- **iter_max**  
  the maximum number of iterations for the expansion of the growing region. *positive integer.*

- **operator**  
  should the median absolute deviation be used to estimate the variability of the group contrast ("mad") or the standard deviation ("sd").

- **history_sigma**  
  should the values of sigma be recorded? *logical.*

- **history_step**  
  should the number of observations included in the GR set be recorded? *logical.*

- **history_front**  
  should the propagation front of the GR set be recorded? *logical.*

References

heatmapMRIaggr

Correlation between contrast parameters

Description
Display a correlation map of the contrast parameters.

Usage

```r
## S4 method for signature 'MRIaggr'
heatmapMRIaggr(object, param, num=NULL, hemisphere="both",
scale=TRUE, method="pearson", points.values=TRUE, type="image",
digit=3, breaks=NULL, window=FALSE, col=cm.colors(256), main=NULL,
mgp=c(2,0.5,0), mar=c(4,4,1,6), las=1, cex.axis=1,
filename=paste(object@identifier,"heatmapMRIaggr", sep="_"),
width=1000, height=700, path=NULL, unit="px", res=NA)
```

Arguments

- **object**: an object of class `MRIaggr`. REQUIRED.
- **param**: the contrast parameters used to compute the correlations. character vector. REQUIRED.
- **num**: the slices to use. numeric vector or NULL.
- **hemisphere**: the hemisphere to use. character. See the details section of `selectContrast`.
- **scale**: should the contrast parameters be scaled? logical.
- **method**: the correlation coefficient which is to be computed. Can be "pearson", "kendall" or "spearman".
- **points.values**: should the correlation values be printed on the plot? logical.
- **type**: the type of plot to display. Any of "image" or "image.plot" or FALSE meaning no plot.
- **digit**: the number of decimal places to use for the labels. positive integer.
- **breaks**: the break points to use to generate the color intervals. numeric vector or NULL leading to automatic breakpoints generation.
- **window**: the type of device on which the plot will be displayed. logical, NULL or character.
- **col**: the colors with which the correlations will be displayed. character vector.
- **main**: an overall title for the plot. character.
- **mgp**: the margin line for the axis title, axis labels and axis line. positive numeric vector of size 3.
- **mar**: the number of margin lines to be specified on the four sides of the plot. positive numeric vector of size 4.
- **las**: the style of the axis labels. Any of 0, 1, 2 or 3.
cex.axis the magnification to be used for axis annotation relative to the current setting of cex.

filename the name of the file used to export the plot. character.

width the width of the device used to export the plot. positive numeric.

height the height of the device used to export the plot. positive numeric.

path the directory where the plot file will be created. character.

unit the units in which height and width are given. character.

res the nominal resolution in ppi which will be recorded in the bitmap file. positive integer.

Details

ARGUMENTS:
Information about the num argument can be found in the details section of initNum.
Information about the hemisphere argument can be found in the details section of selectContrast.
If breaks is not NULL, it must be of length length(col)+1.
Information about the window, filename, width, height, path, unit and res arguments can be found in the details section of initWindow.
Information about the mar, las and mgp arguments can be found in par.

Value

None.

Examples

```r
## load a MRIaggr object
data("MRIaggr_Pat1_red", package="MRIaggr")

## pearson
heatmapMRIaggr(MRIaggr_Pat1_red, param=c("MASK_T2_FLAIR_t2", "DWI_t0", "TTP_t0", "MTT_t0"),
               las=1, type="image", cex=0.75,
               breaks=seq(-1,1,length.out=51),
               col=cm.colors(50))

## spearman
heatmapMRIaggr(MRIaggr_Pat1_red, param=c("MASK_T2_FLAIR_t2", "DWI_t0", "TTP_t0", "MTT_t0"),
               las=1, type="image", cex=0.75, method="spearman",
               breaks=seq(-1,1,length.out=51),
               col=cm.colors(50))

## spearman with legend
heatmapMRIaggr(MRIaggr_Pat1_red, param=c("MASK_T2_FLAIR_t2", "DWI_t0", "TTP_t0", "MTT_t0"),
               las=1, type="image.plot", cex=0.75, method="spearman",
               breaks=seq(-1,1,length.out=51),
               col=cm.colors(50))
```
Description

Check and initialize display arguments. For internal use.

Usage

initCol(contrast, coords, param=NULL, pch, col, palette,
        breaks, legend, type.breaks, method)

Arguments

- **contrast**: the contrast value of each observation. *matrix.*
- **coords**: the spatial coordinates of the observations. *data.frame.*
- **param**: the contrast parameter to display. *character.*
- **pch**: the symbol with which the observations will be displayed. *positive integer.*
- **col**: the color with which the observations will be displayed. *character vector.*
- **palette**: the colors or the palette to use when associating colors to intensities. *character vector or character.*
- **breaks**: the break points or the number of breakpoints to use to generate the color intervals. *numeric vector or positive integer.*
- **legend**: how should the legend be displayed? *logical* or *NULL.*
- **type.breaks**: should the break points be equally spaced according to the range of data values ("range"), centered ("range_center") or correspond to the quantile values ("quantile")?
- **method**: the name of the function that called the initializer. *character.*

Details

ARGUMENTS:

palette must have a *character vector* of length length(breaks)-1 containing colors or a single character corresponding to a palette name: "rainbow", "grey.colors", "heat.colors", "terrain.colors", "topo.colors" or "cm.colors".

Possible values for legend are:

- **TRUE** : the legend is displayed in the current device.
- **NULL** : the legend is displayed in a new device.
- **FALSE** : the legend is not displayed.

FUNCTION:

breaks and palette are active only if col is NULL.
initDisplayWindow  

Device management

Description

Display the device on which the plot will be displayed. For internal use.

Usage

initDisplayWindow(window, filename, path, width, height, scale, res, mfrow, bg, pty, mar, mgp)

Arguments

- **window**: the type of device on which the plot will be displayed. *logical*, *NULL* or *character*.
- **filename**: the name of the file used to export the plot. *character*.
- **path**: the directory where the plot file will be created. *character*.
- **width**: the width of the device used to export the plot in inches. *positive numeric*.
- **height**: the height of the device used to export the plot. *positive numeric*.
- **scale**: the scaling factor to convert height and height to standard unit. *numeric*.
- **res**: the nominal resolution in ppi which will be recorded in the bitmap file. *positive integer*.
- **mfrow**: the division of the device in plot region. *numeric vector of size 2*.
- **bg**: the color used for the background. *character*.
- **pty**: the type of plot region to be used. Can be "s" or "m".
- **mar**: the number of margin lines to be specified on the four sides of the plot. *positive numeric vector of size 4*.
- **mgp**: the margin line for the axis title, axis labels and axis line. *positive numeric vector of size 3*.

Details

ARGUMENTS:

Information about the window, filename, width, height, path, unit and res arguments can be found in the details section of **initWindow**.

Information about the bg, pty, mar and mgp arguments can be found in **par**.
initFilter

Initialization of a filter

Description

Return a filter corresponding to the specified name.

Usage

initFilter(filter, method)

Arguments

filter the filter to be initialized. character.
method the name of the function that called the initializer. character.

Details

ARGUMENTS:
filter refers to classical filters that will be constructed by the function:
The first two characters refer to the dimension of the filter: "2D" or "3D".
The third character must be "_".
The fourth character refers to the type of filter among: "M", "G", "S" or "I" :

- "M" : median filter (by default a weight of 1 is attributed to each site)
- "G" : gaussian filter (sites are weight with a gaussian kernel)
- "S" : sobel filter (gradient in the x, y or z direction)
- "I" : influence filter (sites with distance to the central site inferior or equal to \( \sqrt{\text{p}} \) have weight 1, otherwise 0)

The last one or two characters indicates the size p of the filter except for the sobel filter where it indicates the direction of the gradient ("x", "y" or "z")

Value

A list were:

- [[filter]]: contains the filter. matrix of dimension \( p \times p \) or array of dimension \( p \times p \times p \).
- [[filter_split]]: contains the decomposition of the filter name. character vector.
Examples

```r
initFilter("2D_G3",method="calcFilter")$filter
initFilter("2D_G5",method="calcFilter")$filter
initFilter("3D_G3",method="calcFilter")$filter

initFilter("2D_M9",method="calcFilter")$filter
initFilter("3D_M3",method="calcFilter")$filter

initFilter("2D_Sx",method="calcFilter")$filter
initFilter("2D_Sy",method="calcFilter")$filter
initFilter("3D_Sx",method="calcFilter")$filter
initFilter("3D_Sz",method="calcFilter")$filter

initFilter("2D_I3",method="calcFilter")$filter
initFilter("2D_I5",method="calcFilter")$filter
initFilter("2D_I7",method="calcFilter")$filter
initFilter("2D_I9",method="calcFilter")$filter
initFilter("3D_I5",method="calcFilter")$filter
```

---

**initGR**

*Growing Region initialization*

**Description**

Initialize arguments of the Growing Region algorithm. For internal use.

**Usage**

```r
initGR(contrast,W,seed,range,range.seed,breaks,scale,trace,method)
```

**Arguments**

- `contrast` the contrast value of each observation. *numeric vector.*
- `W` the neighborhood matrix. *dgCMatrix.*
- `seed` the index of the initial seeds or a binary indicator of the initial seeds. *positive integer vector or logical vector.*
- `range` the range of acceptable contrast values for the growing region group. *numeric vector of size 2.*
- `range.seed` the range of acceptable contrast values for the seeds. *numeric vector of size 2.*
- `breaks` the break points or the number of break points to use to categorize the contrast distribution. *numeric vector or positive integer.*
- `scale` should the contrast be scaled? *logical.*
- `trace` should the execution of the function be traced? *logical.*
- `method` the name of the function that called the initializer. *character.*
initIndex

Index initialization

Description

Check and initialize index. For internal use.

Usage

initIndex(object, index, num, hemisphere="both", as.logical, indexNum=NULL, cex.default, pch.default, col.default, filter_default, method)

Arguments

- **object**: an object of class `MRIaggr` or `NULL`.
- **index**: the coordinates of additional points to display. `data.frame` or `list`.
- **num**: the slices to display. `numeric vector` or `NULL`.
- **hemisphere**: the hemisphere to display. `character`.
- **as.logical**: if a parameter is specified for the index arguments, should it be converted to logical? `logical`.
- **indexNum**: the number associated to the index (for display). `numeric`.
- **cex.default**: the default expansion factor used to plot the observations. `numeric`.
- **pch.default**: the default label used to plot the observations. `numeric`.
- **col.default**: the default color used to plot the observations. `character vector`.
- **filter_default**: the default filter used to define the neighborhood. `character`.
- **method**: the name of the function that called the initializer. `character`.

Details

ARGUMENTS:

Information about the `num` argument can be found in the details section of `initNum`.

Information about the `hemisphere` argument can be found in the details section of `selectContrast`.

Possible values for index are:

- **NULL**: the argument is skipped, no additional points are displayed.
- **character**: the name of a logical parameter contained in the object (if called by a `MRIaggr` method)
- **data.frame** with 3 columns named "i","j","k" containing the coordinates of the points to display.
initNeighborhood

* list with an element named "data" containing the coordinates of the points to display (see previous point) or the name of a logical parameter (if called by a MRIaggr method). It can also contain "cex", "pch" and "col" component to specify the how the points should be displayed.
To only display the outline of the spatial group instead of the spatial group itself, the list must contains an element named "outline" that have for value TRUE.
The type of neighborhood used to determine the outline can be specified with a "filter" element in the list. The default neighborhood is "2D_N4".

Description
Return the neighborhood configuration corresponding to the specified name.

Usage
initNeighborhood(Neighborhood, method)

Arguments
- Neighborhood the name of neighborhood configuration.
  Any of "2D_N4", "2D_N8", "3D_N6", "3D_N10", "3D_N18", "3D_N26".
- method the name of the function that called the initializer. character.

Details
ARGUMENTS:
- Neighborhood refers to classical neighborhood configurations:
  The first two characters refer to the dimension d of the filter: "2D" or "3D".
  The third character must be "_".
  The fourth character refers to the type of filter and must be "N".
  The last one or two characters indicates the number of neighbors (denoted n) in each neighborhood.

Value
A n*d matrix with in line the coordinates of the neighbors relative to the current observation.

Examples
# 2D neighborhood
initNeighborhood("2D_N4", method="calcFilter") # rock neighborhood
initNeighborhood("2D_N8", method="calcFilter") # queen neighborhood

# 3D neighborhood
initNeighborhood("3D_N6", method="calcFilter") # rock neighborhood
initNeighborhood("3D_N10", method="calcFilter") # queen neighborhood
initNum

Initialization of the slice numbers

Description

Check and initialize the num argument of a Carto3D or a MRIaggr method. For internal use.

Usage

```r
## S4 method for signature 'Carto3D'
initNum(object,num,test=TRUE,init=TRUE)

## S4 method for signature 'MRIaggr'
initNum(object,num,test=TRUE,init=TRUE,slice_var="k",method)
```

Arguments

- **object**: an object of class Carto3D or MRIaggr.
- **num**: the slice numbers to check or initialize. numeric vector or NULL. See the details section.
- **test**: should the slice numbers be checked? logical.
- **init**: should the slice numbers be initialized if num equals NULL? logical.
- **slice_var**: the type of slice to extract. "i" for sagittal, "j" for coronal and "k" for transverse. character.
- **method**: the name of the function that called the initializer. character.

Details

ARGUMENTS:
Setting num to NULL leads to load all available slices.
initParameter

**Description**

Check and initialize the `param` argument in MRIaggr methods. For internal use.

**Usage**

```r
## S4 method for signature 'MRIaggr'
initParameter(object, param, test=TRUE, init=FALSE,
               accept.coords=TRUE, accept.mask=TRUE, accept.index=TRUE,
               arg_name="param", long_name="parameters", method)
```

**Arguments**

- `object` an object of class `MRIaggr`.
- `param` the contrast parameters to check or initialize. *character* vector or `NULL`.
- `test` should the parameters be checked? *logical*.
- `init` should the parameters be initialized if `param` equals `NULL`? *logical*.
- `accept.coords` should coordinates be accepted as parameters? *logical*.
- `accept.mask` should mask be accepted as a parameter? *logical*.
- `accept.index` should index be accepted as a parameter? *logical*.
- `arg_name` a short name for the error message? *character*.
- `long_name` the complete name for the error message? *character*.
- `method` the name of the function that called the initializer. *character*.

**Details**

ARGUMENTS:
Setting `param` to `NULL` leads to load all available parameters including coordinates, mask and index if `accept.coords`, `accept.mask` or `accept.index` are respectively set to `TRUE`.

initWindow

**Description**

Check and initialize display arguments. For internal use.

**Usage**

```r
initWindow(window, filename, path, width, height, unit, res,
           n.plot, mfrow, xlim, ylim, method)
```
Arguments

- **window**: the type of device on which the plot will be displayed. *logical*, *NULL* or *character*.
- **filename**: the name of the file used to export the plot. *character*.
- **path**: the directory where the plot file will be created. *character*.
- **width**: the width of the device used to export the plot. *positive numeric*.
- **height**: the height of the device used to export the plot. *positive numeric*.
- **unit**: the units in which height and width are given. *character*.
- **res**: the nominal resolution in ppi which will be recorded in the bitmap file. *positive integer*.
- **n.plot**: the number of images to display. *integer*.
- **mfrow**: the division of the device in plot region. *numeric vector of size 2*.
- **xlim**: the x limits of the plot. *numeric vector of size 2*.
- **ylim**: the y limits of the plot. *numeric vector of size 2*.
- **method**: the name of the function that called the initializer. *character*.

Details

ARGUMENTS:
Possible values for windows are:

- **NULL**: the plot is displayed on the current device with no reshape.
- **FALSE**: the plot is displayed on the current device with the appropriate reshape.
- **TRUE**: a new graphical device is open.
- "eps": the plot is displayed in an image file using *postscript*.
- "svg": the plot is displayed in an image file using *svg*.
- "png": the plot is displayed in an image file using *png*.
- "pdf": the plot is displayed in an image file using *pdf*.

If path is set to *NULL*, the image file is exported in the current working directory.

unit can be any of "px", "in", "cm" or "mm".

Information about the filename, width, height, path, unit and res arguments can be found in *postscript*, *svg* and *png*.

FUNCTION:
The arguments filename, width, height, path, unit and res are only active if window is "eps", "svg" or "png".
**legendMRI**

*Display a legend of a contrast map*

**Description**

Display a legend from break values. For internal use.

**Usage**

```r
legendMRI(breaks, palette, mar, cex, cex.main, main, quantiles, digit)
```

**Arguments**

- `breaks`: the break points to use to generate the color intervals. *numeric vector*.
- `palette`: the colors or the palette to use when associating colors to observation intensities. *character vector*.
- `mar`: the number of margin lines to be specified on the four sides of the legend. *positive numeric vector of size 4*.
- `cex.main`: the expansion factor for the legend title. *positive numeric*.
- `cex`: the expansion factor for the legend labels. *positive numeric*.
- `main`: an overall title for the legend. *character*.
- `quantiles`: the quantiles values to display on the legend. *numeric vector of size 5 or NULL*.
- `digit`: the number of decimal places to use for the legend label. *positive integer*.

**Details**

ARGUMENTS:
Information about the `mar` argument can be found in `par`.

Information about the `palette` argument can be found in the details section of `initCol`.

---

**MRIaggr-class**

*Class "MRIaggr"*

**Description**

Patient-specific storage of multivariate contrast data and clinical data.
Arguments

- **identifier**: the patient identifier. *character.*
- **contrast**: the value of the contrast parameters for each observation. *data.frame.*
- **clinic**: the clinical data of the patient. *data.frame.*
- **voxelDim**: the spatial dimensions of the lattice containing the observations. *data.frame.*
- **default_value**: the reference values of the contrast parameters. *data.frame.*
- **history**: the list of the calc or const methods that have been already applied on the MRIaggr object. *data.frame.*
- **normalization**: the normalization values for the contrast parameters. *list.*
- **hemispheres**: the presence or absence of lesion in each cerebral hemisphere. *data.frame.*
- **midplane**: the position of the mid-sagittal plan. *data.frame.*
- **table_lesion**: the vertical distribution of the lesion volumes. *data.frame.*
- **table_reperfusion**: the volumic reperfusion data. *data.frame.*
- **table_hypoperfusion**: the volumic hypoperfusion data. *data.frame.*
- **ls_statDesc**: a slot to store additional data. *list.*

S4 methods

Getters:

- `selectContrast`
- `selectCoords`
- `selectClinic`
- `selectDefault_value`
- `selectDescStats`
- `selectHemispheres`
- `selectHistory`
- `selectIdentifier`
- `selectMidplane`
- `selectN`
- `selectNormalization`
- `selectParameter`
- `selectTable`
- `selectVoxelDim`
- `selectVoxelSize`

Setters:

- `affectContrast<-`
affectClinic<-  
affectDescStats<-  
affectHemisphere<-  
affectNormalization<-  
affectTable<-  
supprContrast<-  
supprDescStats<-  

Calculators :
  calcBrainMask  
  calcControlateral  
  calcDistMask  
  calcDistTissues  
  calcFilter  
  calcGroupsMask  
  calcHemisphere  
  calcROCThreshold  
  calcNormalization  
  calcRegionalContrast  
  calcSmoothMask  
  calcTableHypoReperf  
  calcTableLesion  
  calcThresholdMRIaggr  
  calcTissueType  
  calcW  

Displayers :
  boxplotMask  
  heatmapMRIaggr  
  multiplot  
  plotDistClass  
  pointsHemisphere  
  plotLesion3D  
  plotTableLesion  
  outlineMRIaggr  
  summary, MRIaggr-method  

Constructors :
  constCompressMRIaggr
constReduceMRIaggr

Initializers:

initNum
initParameter

See Also

constMRIaggr to build a MRIaggr object from a list of array.
readMRI to read imaging files.

MRIaggr.Pat1_red Example of processed MRIaggr object

Description

Example of processed MRIaggr object using the functionalities of the MRIaggr package.

Usage

MRIaggr.Pat1_red

Format

A MRIaggr object containing in addition to the contrast data:

- the position of the mid-saggital plan.
- the normalization values for the contrast parameters.
- the spatial groups of lesion.
- the regional and controlateral normalised contrast values.
- the CSF/WM/GM probabilistic membership.
- the neighborhood matrix.
- the lesion volumes per slice
- the hypoperfusion and reperfusion volumes for various time thresholds.

Source

I-know study: www.i-know-stroke.eu
multiplot

Spatial display using coordinates

Description

Make a spatial display of a `data.frame`, a Carto3D or a MRIaggr object.

Usage

```r
# S4 method for signature 'data.frame'
multiplot(object, contrast = NULL, num = NULL,
          index1 = NULL, index2 = NULL, index3 = NULL, breaks = 50,
          type = NULL, breaks = "range",
          palette = "terrain.colors", col = NULL, pch = NULL, cex = 1,
          NA = "lightyellow", pch = NA, NA = NULL, axes = TRUE,
          window = FALSE, legend = TRUE, mfrow = NULL,
          mar = rep(1.5, 4), mgp = c(2, 0.5, 0), pty = NULL, asp = 1,
          bg = "lightblue", xlab = "", ylab = "", main = "slice ",
          num.main = TRUE, cex.main = 1.5,
          quantiles.legend = TRUE, digit.legend = 3,
          cex.legend = 1.5,
          main.legend = c(2, 7, 2, 2), main.legend = names(contrast),
          filename = "multiplot", width = 1000, height = 700,
          path = NULL, unit = "px", res = NA)

# S4 method for signature 'Carto3D'
multiplot(object, num = NULL,
          breaks = 50, type = NULL, breaks = "range",
          palette = "terrain.colors", col = NULL, pch = NULL, cex = 1,
          NA = "lightyellow", pch = NA, NA = NULL, axes = TRUE,
          window = FALSE, legend = TRUE, mfrow = NULL,
          mar = rep(1.5, 4), mgp = c(2, 0.5, 0), pty = NULL, asp = 1,
          bg = "lightblue", xlab = "", ylab = "", main = NULL,
          num.main = TRUE, cex.main = 1.5,
          quantiles.legend = TRUE, digit.legend = 3,
          cex.legend = 1.5,
          main.legend = c(2, 7, 2, 2),
          filename = "multiplot", width = 1000, height = 700,
          path = NULL, unit = "px", res = NA)

# S4 method for signature 'MRIaggr'
multiplot(object, param, num = NULL,
          index1 = NULL, index2 = NULL, index3 = NULL, midplane = FALSE,
          slice_var = "k",
          hemisphere = "both", norm_mu = FALSE, norm_sigma = FALSE,
          as.logical = FALSE, breaks = 50,
          type = NULL, breaks = "range",
          palette = "terrain.colors", col = NULL, pch = NULL, cex = 1,
          NA = "lightyellow", pch = NA, NA = NULL, axes = TRUE,
          window = FALSE, legend = TRUE, mfrow = NULL,
          mar = rep(1.5, 4), mgp = c(2, 0.5, 0), pty = NULL,
          asp = 1, bg = "lightblue", xlab = "", ylab = "",
          main = NULL, num.main = TRUE, cex.main = 1.5,
          quantiles.legend = TRUE, digit.legend = 3,
          cex.legend = 1.5,
          main.legend = c(2, 7, 2, 2),
          filename = "multiplot", width = 1000, height = 700,
          path = NULL, unit = "px", res = NA)
```

Arguments

- **object**: an object of class MRIaggr or Carto3D or a 3 column data.frame containing the coordinates of the observations in columns. **REQUIRED**.
- **param**: the contrast parameter to display. `character`. **REQUIRED**.
- **contrast**: the intensities to display. `numerical vector` or NULL leading to use the same color for all observations.
num the slices to display. numeric vector or NULL.
index1, index2, index3 the coordinates of additional points to display. data.frame or list or NULL.
midplane should the mid-sagittal plan be displayed? logical.
slice_var the type of view to use. "i" for sagittal view, "j" for coronal view and "k" for transverse view. character.
hemisphere the hemisphere to display. character.
norm_mu the type of centering to apply on the parameter values. character.
norm_sigma the type of scaling to apply on the parameter values. character.
as.logical if a parameter is specified for the index arguments, should it be converted to logical? logical.
breaks the break points or the number of breakpoints to use to generate the color intervals. numeric vector or positive integer.
type.breaks should the break points be equally spaced according to the range of data values ("range"), centered ("range_center") or correspond to the quantile values ("quantile")?
palette the colors or the palette to use when associating colors to observation intensities. character vector or character.
col the color with which the observations will be displayed. character vector or NULL leading to determine the colors using the palette and breaks arguments.
pch the symbol with which the observations will be displayed. positive integer or NULL leading to use the image function instead of plot.
cex the expansion factor used to plot the observations. positive numeric.
col.NA the color to use to plot the NAs. character.
pch.NA the label to use to plot the NAs. positive integer.
col.midplane the color in which the mid-sagittal plane should appear. character.
xlim the x limits of the plot. numeric vector of size 2 or NULL leading to automatic setting of the x limits.
ylim the y limits of the plot. numeric vector of size 2 or NULL leading to automatic setting of the y limits.
axes should the axes be plotted? logical.
window the type of device on which the plot will be displayed. logical, NULL or character.
legend how should the legend be displayed? logical or NULL.
mfrow the division of the device in plot region. numeric vector of size 2 or NULL leading automatic adjustment.
mar the number of margin lines to be specified on the four sides of the plot. positive numeric vector of size 4.
mgp the margin line for the axis title, axis labels and axis line. positive numeric vector of size 3.
pty the type of plot region to be used. Can be "s" or "m".
asp the aspect ratio y/x. numeric.
bg the color used for the background. character.

xlab a title for the x axis. character.

ylab a title for the y axis. character.

main an overall title for the plot. character.

num.main should the slice number be written over each plot. logical.

cex.main the expansion factor for the main title. numeric.

quantiles.legend should the quantiles of the data be displayed on the legend? logical.

digit.legend the number of decimal places to use for the legend labels. integer.

cex.legend the expansion factor of the legend. positive numeric.

mar.legend the number of margin lines to be specified on the four sides of the legend. numeric vector of size 4.

main.legend a main title for the legend. character.

filename the name of the file used to export the plot. character.

width the width of the device used to export the plot. positive numeric.

height the height of the device used to export the plot. positive numeric.

path the directory where the plot file will be created. character.

unit the units in which height and width are given. character.

res the nominal resolution in ppi which will be recorded in the bitmap file. positive integer.

Details

ARGUMENTS:

Information about the num argument can be found in the details section of initNum.

Information about the index1, index2 and index3 arguments can be found in the details section of initIndex (argument index).

Information about the hemisphere, norm_mu and norm_mu arguments can be found in the details section of selectContrast.

Information about the window, filename, width, height, path, unit and res arguments can be found in the details section of initWindow.

Information about the mfrow, mar, mar.legend, mgp pty, asp and bg arguments can be found in par.

Information about the palette and legend arguments can be found in the details section of initCol.

Value

None.

See Also

plotLesion3D for a 3D plot of the lesion.
slices3d of the misc3d package for a more interactive 3D plot.
Examples

```r
### 1 - data.frame ###
# simulate
n <- 10
Y <- rnorm(n*2)

# display
multiplot(object=data.frame(expand.grid(1:n,1:n),1),
          contrast=Y,window=FALSE)

# load a MRIaggr object
data(MRIaggr.Pat1_red, package="MRIaggr")

# select data
data <- selectContrast(MRIaggr.Pat1_red,param=c("DWI_t0","TTP_t0","MTT_t0","MASK_T2_FLAIR_t2"),
                          hemisphere="lesion",coords=TRUE)

# fit model
glm.1 <- glm(MASK_T2_FLAIR_t2 ~ DWI_t0 + TTP_t0 + MTT_t0,data=data,family=binomial(link="logit"))

# display fitted values
multiplot(object=data[,c("i","j","k")],
          contrast=predict(glm.1,type="response"),window=FALSE)

# display residuals
multiplot(object=data[,c("i","j","k")],num=3,
          contrast=predict(glm.1,type="response"),window=FALSE,
          index1=list(coords=data$data$MASK_T2_FLAIR_t2,c("i","j","k")),outline=TRUE)

### 2 - carto3D ###
# load nifti files and convert them to carto3D
path.nifti_files <- system.file("nifti",package = "MRIaggr")
nifti.Pat1_TTP_t0 <- readMRI(file=file.path(path.nifti_files,"TTP_t0"),format="nifti")
Carto3D.Pat1_TTP_t0 <- constCarto3D(nifti.Pat1_TTP_t0,identifier="Pat1",param="TTP_t0")

# display
multiplot(Carto3D.Pat1_TTP_t0)

### Not run:
multiplot(Carto3D.Pat1_TTP_t0,num=1:2)
multiplot(Carto3D.Pat1_TTP_t0,num=1:2,axes=FALSE)
multiplot(Carto3D.Pat1_TTP_t0,num=1:2,axes=FALSE,legend=FALSE)
multiplot(Carto3D.Pat1_TTP_t0,num=1:2,axes=FALSE,legend=FALSE,main="",num.main=FALSE)
multiplot(Carto3D.Pat1_TTP_t0,num=1:2,axes=FALSE,main="",num.main=FALSE,
          palette="gray.colors",breaks=seq(0,100))

# End(Not run)

### 3 - MRIaggr ###
# load a MRIaggr object
data(MRIaggr.Pat1_red, package="MRIaggr")
```
```r
# display 3 slices
multiplot(MRIaggr.Pat1_red,param="DWI_t0",
  num=1:3)

## Not run:
# display 3 slices with no axes and white background
multiplot(MRIaggr.Pat1_red,param="DWI_t0",
  num=1:3,axes=FALSE,bg="white")

# remove the legend
multiplot(MRIaggr.Pat1_red,param="DWI_t0",
  num=1:3,legend=FALSE)

## End(Not run)

## display an set of points
# using a binary parameter stored in the object
multiplot(MRIaggr.Pat1_red,param="DWI_t0",
  num=1:3,index1=list(coords="MASK_DWI_t0"))

## Not run:
# customize the display of the points
multiplot(MRIaggr.Pat1_red,param="DWI_t0",
  num=1:3,index1=list(coords="MASK_DWI_t0",col="pink",pch=14))

# display only the edges of the set
multiplot(MRIaggr.Pat1_red,param="DWI_t0",num=3, legend=FALSE, index1=list(coords="MASK_DWI_t0",outline=TRUE))

## specify the index of points using coordinates
coordsIndex <- data.frame(i=c(40,60),j=c(80,100),k=c(3,3))

multiplot(MRIaggr.Pat1_red,param="DWI_t0",num=3,legend=FALSE, index2=list(coords=coordsIndex,col="black",pch=15,cex=4))

# various possibilities for the display
multiplot(MRIaggr.Pat1_red,num=1:3,param="DWI_t0",
  legend=FALSE,window=FALSE)
multiplot(MRIaggr.Pat1_red,num=1:3,param="DWI_t0",
  legend=TRUE,window=FALSE)
multiplot(MRIaggr.Pat1_red,num=1:3,param="DWI_t0",
  legend=NULL,window=FALSE)

## End(Not run)
```
Outline a region on a slice

Description

Tool for graphical definition of a spatial region on an image.

Usage

`outline(n=50, sequential=FALSE, min_dist=1, col=c("blue", "red", "grey"), pch=20, cex=c(0.75, 1, 0.75))`

Arguments

- `n`: maximum number of points to define the outline. integer.
- `sequential`: should the region edge be updated on the graphical device after each point? logical.
- `min_dist`: if the distance between the new point and the initial point is inferior to `min_dist`, then the definition of the region ends. numeric. Only active if `sequential` is TRUE.
- `col`: the colors in which the user-defined edge points, the interpolated edge points and the interior points should be plotted. character vector of size 3.
- `pch`: the symbol with which the observations will be displayed. positive integer.
- `cex`: the expansion factor used to plot the edge points, the interpolated edge points and the interior points. positive numeric vector of size 3.

Details

FUNCTION:
This function uses the `locator` function to obtain the coordinates of the cursor. It enable a point by point definition of a region where a linear interpolation is used between user-defined points to define the edge of the region.

In the non sequential mode, the definition of the points stop if the number of points exceed `n` or using Echap. In the sequential mode, the definition of the points stop if the number of points exceed `n` or if the new point is close enough to the initial point.

After defining the edge, the region is filled.

Value

A list of two elements:

- `[[edge]]`: a `data.frame` containing the position ("i" "j") of the edge of the region, the edge number ("edge"), whether its user-specified point or interpolated point ("points") and whether it was removed for the filling procedure ("valid").
- `[[surface]]`: a `data.frame` containing the position ("i" "j") of the points belonging to the region.
Examples

```r
## Not run:
## load a MRIaggr object
data("MRIaggr.Pat1_red", package="MRIaggr")
num <- 3

## display 1
multiplot(MRIaggr.Pat1_red,param="T2 FLAIR_t2",
  num=num,legend=FALSE,
  window=FALSE)

## outline on display 1
res <- outline(sequential=TRUE, min_dist=3)

## display the results
multiplot(MRIaggr.Pat1_red,param="T2 FLAIR_t2",
  num=num,legend=FALSE,
  index1=data.frame(k=num,res$edge[,c("i","j")]),
  index2=data.frame(k=num,res$surface[,c("i","j")]),
  window=FALSE)
carto <- selectContrast(MRIaggr.Pat1_red,param=c("MASK_T2 FLAIR_t2","index"),num=num,coords=TRUE)
carto <- merge(carto,cbind(res$surface,outline=TRUE),all = TRUE)
carto[is.na(carto$outline),"outline"] <- FALSE
head(carto)

## display the results next to MASK_T2 FLAIR_t2
multiplot(MRIaggr.Pat1_red,param="T2 FLAIR_t2",
  num=num,legend=FALSE,
  index1=carto[,c("i","j","k")],
  index2=carto[,c("i","j","k")],
  window=FALSE)

## End(Not run)
```

**outlineMRIaggr**  
*Outline a region on a slice*

**Description**

Tool for graphical definition of a spatial region on an image.

**Usage**

```r
## S4 method for signature 'MRIaggr'
outlineMRIaggr(object,param,index1=NULL,num=NULL,hemisphere="both",
  as.logical = FALSE,xlim=NULL,ylim=NULL,legend = FALSE,
  palette="terrain.colors",col=NULL,breaks=25,fill=TRUE,n=50,
  sequential=FALSE,min_dist=1,operator_index1="none",
```
Arguments

object  an object of class `MRIaggr`. REQUIRED.
param  the contrast parameter map on which the outline will be drawn. character. REQUIRED.
index1  the coordinates of additional points to display. data.frame or list or NULL.
num  the slices on which the outline will be drawn. numeric vector or NULL.
hemisphere  the hemisphere to consider. character.
as.logical  if a parameter is specified for the index argument, should it be converted to logical? logical.
xlim  the y limits of the plot. numeric vector of size 2 or NULL leading to automatic setting of the x limits.
ylim  the y limits of the plot. numeric vector of size 2 or NULL leading to automatic setting of the y limits.
legend  how should the legend be displayed? logical or NULL.
palette  the colors or the palette to use when associating colors to observation intensities. character vector or character.
col  the color to use to plot the observations. character vector or NULL leading to automatic generation of the colors using the breaks and palette arguments.
breaks  the break points or the number of breakpoints to use to generate the color intervals. numeric vector or positive integer.
fill  should the spatial region be filled? Otherwise only the edge is used. logical.
n  maximum number of points to define the outline. integer.
sequential  should the region edge be updated on the graphical device after each point? logical.
min_dist  if the distance between the new point and the initial point is inferior to `min_dist`, then the definition of the region ends. numeric. Only active if `sequential` is TRUE.
operator_index1  the operator to apply between the index1 observations and the outlined observations. Can be "none" "intersection" "difference" or "union".
col.outline  the colors in which the user-defined edge points, the interpolated edge points and the interior points should be plotted. character vector[3].
pch  the symbol with which the observations will be displayed. positive integer.
cex  the expansion factor used to plot the edge points, the interpolated edge points and the interior points. positive numeric vector of size 3.
trace  should the execution of the function be traced? logical.
name_newparam  the name of the new parameter. character.
update.object  should the new parameter be stored in `object`? logical.
overwrite  if a contrast parameter with the same names is already stored in `object@data`, can it be overwritten? logical.
Details

ARGUMENTS:
Information about the num argument can be found in the details section of initNum.
Information about the index1 argument can be found in the details section of initIndex (argument index).
Information about the hemisphere argument can be found in the details section of selectContrast.
Information about the palette and legend arguments can be found in the details section of initCol.

FUNCTION:
This function uses the locator function to obtain the coordinates of the cursor. See outline for more details. It will display slice by slice the parameter map and on each slice a region should be drawn.
Press Echap and 1 to restart the draw on the same slice.
Press Echap and 0 to skip a slice.

Value
A matrix containing in columns:
- [,c("i","j","k")]: the coordinates of the observations.
- [,"index"]: the index of the observation in object.
- [,"edge"]: whether each observation is on an edge.
- [,"surface"]: whether each observation is in the interior.
- [,"userMask"]: whether each observation belong to index1 (if no not defined, always FALSE).

Examples
```r
## Not run:
## load a MRIaggr object
data("MRIaggr.Pat1_red", package="MRIaggr")

## outline the area of interest
res <- outlineMRIaggr(MRIaggr.Pat1_red,param="DWI_t0",
num=1:3,sequential=TRUE,overwrite=TRUE,update.object=TRUE)
multiplot(MRIaggr.Pat1_red,param="userMask",
num=1:3)

## outline an edge of interest
res <- outlineMRIaggr(MRIaggr.Pat1_red,param="DWI_t0",index1="MASK_DWI_t0",
fill=FALSE,num=1:3,sequential=TRUE,overwrite=TRUE,update.object=TRUE)
multiplot(MRIaggr.Pat1_red,param="userMask",
num=1:3)

## define an new area as the union of the outlined area and the initial lesion mask
res <- outlineMRIaggr(MRIaggr.Pat1_red,param="DWI_t0",index1="MASK_DWI_t0",
...)```
plotDistClass

Plot the distribution of the contrast parameter

Description
Distribution of a contrast parameter according to the tissue class.

Usage
```
## S4 method for signature 'MRIaggr'
plotDistClass(object, param, class, num=NULL, hemisphere="both",
              norm_mu=FALSE, norm_sigma=FALSE, bw.adjust=1, kernel="gaussian",
              from=NULL, to=NULL, ylim=NULL, window=FALSE, col=1:6, main=NULL, mgp=c(2,0.5,0),
              type="l", pch=20, lwd=1, x.legend="topright", y.legend=NULL, cex.legend=0.8,
              filename=paste(object@identifier,"plotDistClass",sep="_"))
```

Arguments
- **object**: an object of class `MRIaggr`. REQUIRED.
- **param**: the contrast parameter to display. `character`. REQUIRED.
- **class**: the parameters indicating the probabilistic membership to the tissue classes. `character vector`.
- **num**: the slices to use. `numeric vector` or NULL.
- **hemisphere**: the hemisphere to use. `character`.
- **norm_mu**: the type of centering to apply on the parameter values. `character`.
norm_sigma the type of scaling to apply on the parameter values. character.

bw.adjust the smoothing bandwidth to use. numeric. See density for more details.

kernel the smoothing kernel to use. character. See density for more details.

from.to the left and right-most points of the grid at which the density is to be estimated. numeric or NULL leading to automatic adjustment. See density for more details.

ylim the y limits of the plot. numeric vector of size 2 or NULL leading to automatic setting of the y limits.

window the type of device on which the plot will be displayed. logical, NULL or character.

col the colors with which the distributions will be displayed. character vector.

main an overall title for the plot. character.

mgp the margin line for the axis title, axis labels and axis line. positive numeric vector of size 3.

type the type of plot to display. character. See plot.default for more details.

pch the symbol with which the distribution will be displayed. positive integer.

lwd the line width. positive numeric.

x.legend the x coordinates of the legend. numeric or character.

y.legend the y coordinates of the legend. numeric or character.

cex.legend the expansion factor of the legend. positive numeric.

filename the name of the file used to export the plot. character.

width the width of the device used to export the plot. positive numeric.

height the height of the device used to export the plot. positive numeric.

path the directory where the plot file will be created. character.

unit the units in which height and width are given. character.

res the nominal resolution in ppi which will be recorded in the bitmap file. positive integer.

Details

ARGUMENTS:
Information about the num argument can be found in the details section of initNum.

Information about the hemisphere, norm.mu and norm.mu arguments can be found in the details section of selectContrast.

Information about the window, filename, width, height, path, unit and res arguments can be found in the details section of initWindow.

Information about the lwd argument can be found in par.

Information about the x.legend, y.legend, cex.legend arguments can be found in legend (cex.legend is the cex argument of legend).

FUNCTION:
This method relies on the density function.
**plotLesion3D**

### Value
None.

### Examples

```r
## load a MRIaggr object
data("MRIaggr.Pat1_red", package="MRIaggr")

## display
plotDistClass(MRIaggr.Pat1_red,param="DWI_t0", class=c("MASK_T2 FLAIR_t2"))

# specify the smoothing bandwidth
plotDistClass(MRIaggr.Pat1_red,param="DWI_t0",bw.adjust=1, class=c("MASK_T2 FLAIR_t2"))

# specify the scale
plotDistClass(MRIaggr.Pat1_red,param="DWI_t0",bw.adjust=1, from=200,to=300,class=c("MASK_T2 FLAIR_t2"))

# use several classes
## Not run:
calcTissueType(MRIaggr.Pat1_red,param="T1_t0",update.object=TRUE,overwrite=TRUE)

plotDistClass(MRIaggr.Pat1_red,param="TTP_t0",bw.adjust=2, class=c("CSF","WM","GM","MASK_T2 FLAIR_t2"))

plotDistClass(MRIaggr.Pat1_red,param="DWI_t0",bw.adjust=2, class=c("CSF","WM","GM","MASK_T2 FLAIR_t2"))

## End(Not run)
```

---

**plotLesion3D**

3D plot of the lesion

---

### Description

Make a 3D plot of the lesion. (experimental version)

### Usage

```r
## S4 method for signature 'MRIaggr'
plotLesion3D(object,mask,edge=FALSE,neighborhood="3D_N6", as.logical=FALSE,spatial_res=c(1,1,1),xlim=NULL,ylim=NULL,zlim=NULL, type.plot="shapelists3d",px_max=10000, radius=1, type="s",col="red",col.edge="black")
```
Arguments

- **object**: an object of class `MRIaggr`. **REQUIRED**.
- **mask**: the binary contrast parameter indicating the lesion. `character`. **REQUIRED**.
- **edge**: should the edges of the lesion be plotted instead of the core? `logical`.
- **Neighborhood**: the type of neighborhood used to define the edges. `character`.
- **as.logical**: should mask be converted to logical? `logical`.
- **spatial_res**: a dilatation factor for the coordinates. `positive numeric vector of size 3`.
- **xlim**: the x limits of the plot. `numeric vector of size 2` or NULL leading to automatic setting of the x limits.
- **ylim**: the y limits of the plot. `numeric vector of size 2` or NULL leading to automatic setting of the y limits.
- **zlim**: the z limits of the plot. `numeric vector of size 2` or NULL leading to automatic setting of the z limits.
- **type.plot**: the type of plot to be displayed. Can be "plot3d" or "shapelist3d".
- **px_max**: the maximum number of points that can be plotted. `integer`.
- **radius**: the radius of spheres. `numeric`. See `plot3d` for more details.
- **type**: the type of item to plot. `character`. See `plot3d` for more details.
- **col**: the color of the core of the lesion. `character`.
- **col.edge**: the color of the edge of the lesion. `character`.

Details

ARGUMENTS:
the **Neighborhood** argument can be a `matrix` or an `array` defining directly the neighborhood to use (i.e. the weight of each neighbor) or a name indicating which type of neighborhood should be used (see the details section of `initNeighborhood`).

FUNCTION:
This functions uses the `plot3d` or `shapelist3d` and thus require the `rgl` package to work. This package is not automatically loaded by the `MRIaggr` package.

Value

None.

Examples

```r
## load a MRIaggr object
data("MRIaggr.Pat1_red", package="MRIaggr")

## Not run:
if(require(rgl)){
  # global view
  plotLesion3D(MRIaggr.Pat1_red, mask="MASK_T2_FLAIR_t2", spatial_res=c(1.875,1.875,6),
               as.logical=TRUE)
}
```
plotMRI

Display a contrast parameter by coordinates

Description
Display a single slice of a contrast parameter. For internal use.

Usage
plotMRI(contrast, coords, breaks, palette, col, asp, 
xlim, ylim, pch, cex, axes, col.NA, pch.NA, xlab, ylab, main, cex.main)

Arguments
- **contrast**: the intensities to display. *numeric vector*.
- **coords**: the spatial coordinates of the observations. *data.frame*.
- **breaks**: the break points to use to generate the color intervals. *numeric vector*.
- **palette**: the colors or the palette to use when associating colors to observation intensities. *character vector*.
- **col**: the colors with which the observations will be displayed. *character vector*.
- **asp**: the aspect ratio y/x. *numeric*.
- **xlim**: the x limits of the plot. *numeric vector of size 2*.
- **ylim**: the y limits of the plot. *numeric vector of size 2*.
- **pch**: the symbol with which the observations will be displayed. *positive integer*.
- **cex**: the expansion factor for the observation labels. *positive numeric*.
- **axes**: should the axes be plotted? *logical*.
- **col.NA**: the color to use to plot the NAs. *character*.
- **pch.NA**: the label to use to plot the NAs. *positive integer*.
- **xlab**: a title for the x axis. *character*.
- **ylab**: a title for the y axis. *character*.
- **main**: an overall title for the plot. *character*.
- **cex.main**: the expansion factor for the main title. *numeric*.

Details
ARGUMENTS:
Information about the palette argument can be found in the details section of *initCol*. 
plotTableLesion

Lesion volume displayed by slices

Description

Display the lesion volume by slices using the table_lesion slot.

Usage

```r
## S4 method for signature 'MRIaggr'
plotTableLesion(object, mask, num=NULL, type="matplot",
    window=FALSE, col=1:5, lty=1:5, lwd=1, mgp=c(2,0.5,0), mar=rep(3,4),
    main=paste("lesion - ", object@identifier, sep=" "),
    cex.legend=1, cex.main=1, cex.axis=1, cex.lab=1,
    filename=paste(object@identifier,"plotTableLesion", sep=" "),
    width=1000, height=700, path=NULL, unit="px", res=NA)
```

Arguments

- `object`: an object of class `MRIaggr`. REQUIRED.
- `mask`: the binary contrast parameter indicating the lesion. character. REQUIRED.
- `num`: the slices to display. numeric vector or NULL.
- `type`: the type of plot to display. Can be "matplot" or "evolution".
- `window`: the type of device on which the plot will be displayed. logical, NULL or character.
- `col`: the colors with which the volumes will be displayed. character vector or numeric vector.
- `lty`: the line type used to represent the volume. numeric vector.
- `lwd`: the line width. positive numeric.
- `main`: an overall title for the plot. character.
- `mgp`: the margin line for the axis title, axis labels and axis line. positive numeric vector of size 3.
- `mar`: the number of margin lines to be specified on the four sides of the plot. positive numeric vector of size 4.
- `cex.main`: the expansion factor for the main title. numeric.
- `cex.legend`: the expansion factor of the legend. positive numeric.
- `cex.axis`: the magnification to be used for axis annotation relative to the current setting of cex. positive numeric.
- `cex.lab`: the magnification to be used for x and y labels relative to the current setting of cex. positive numeric.
- `filename`: the name of the file used to export the plot. character.
- `width`: the width of the device used to export the plot. positive numeric.
- `height`: the height of the device used to export the plot. positive numeric.
**pointsHemisphere**

Add the position of the mid-saggital plan

---

**Description**

Add to an existing plot the mid-saggital plan. For internal use.

**Usage**

```r
## S4 method for signature 'MRIaggr'
pointsHemisphere(object, col="red", lwd=2, lty=1)
```
Arguments

- object: an object of class MRIaggr.
- col: the color with which the mid-saggital plan will be plotted. character.
- lwd: the line width. positive numeric.
- lty: the type of line used to represent the mid-saggital plan. numeric vector.

Details

ARGUMENTS:
Information about the lwd and lty arguments can be found in par.

pointsOutline  Compute the outline of a spatial group

Description

Return a the outline of a spatial group for a given neighborhood. For internal use.

Usage

pointsOutline(coords, array=NULL, filter="2D_N4")

Arguments

- coords: the spatial coordinates of the observations. data.frame.
- array: alternative specification of the spatial coordinates using an array where the non-NA values indicates the points of interest. array or NULL leading to use the coords argument.
- filter: the type of filter, see calcFilter for more details. character.

Details

FUNCTION:
This function apply a filter to the spatial group and consider as the outline all the observations that do not reach the maximum value (i.e. do not have a complete neighborhood).

Value

A data.frame with the spatial coordinates of the outline.
**readMRI**  
*Read imaging file*

**Description**

Read an imaging file and convert it into an array.

**Usage**

```r
readMRI(file, format, what="integer", na.value = 0, dim = NULL,
          SPM=FALSE, reorient = FALSE, flipud=FALSE)
```

**Arguments**

- **file**
  
  the file name of the imaging file. *character*. REQUIRED.

- **format**
  
  the format of the image file. Can be "raw.gz", "analyze", "nifti" or "dicom". REQUIRED.

- **what**
  
  an object whose mode will give the mode of the vector to be read, or a character vector of length one describing the mode: one of "numeric", "double", "integer", "int", "logical", "complex", "character", "raw". Only active if `format` equals `raw.gz`.

- **na.value**
  
  the value with which NA values are replaced. *numeric* or NA.

- **dim**
  
  the number of bytes per element in the byte stream. The default, NA_integer_, uses the natural size. Size changing is not supported for raw and complex vectors. Only active if `format` equals `raw.gz`.

- **SPM**
  
  is a logical variable (default = FALSE) that forces the voxel data values to be rescaled using the `funused1 ANALYZE` header field. This is an undocumented convention of ANALYZE files processed using the Statistical Parametric Mapping (SPM) software. Only active if `format` equals `analyze`.

- **reorient**
  
  is a logical variable (default = TRUE) that enforces Qform/Sform transformations. Only active if `format` equals `nifti`.

- **flipud**
  
  is a logical variable for vertical flipping of the image (default is TRUE). Only active if `format` equals `dicom`.

**Details**

ARGUMENTS: file argument corresponds to:

- the `con` argument of the base::`readBin` function.
- the `fname` argument of the `oro.nifti::readANALYZE` function. It should be a pathname to .img or .hdr files without the suffix.
- the `fname` argument of the `oro.nifti::readNIftI` function.
- the `fname` argument of the `oro.dicom::readDICOMFile` function. It should be the file name with suffix.
The what and dim correspond to the what and size argument of the `base::readBin` function. The SPM corresponds to the SPM argument of the `oro.nifti::readANALYZE` function. The reorient corresponds to the reorient argument of the `oro.nifti::readNIfTI` function. The flipud corresponds to the flipud argument of the `oro.dicom::readDICOMFile` function.

**FUNCTION:**
This function is a slightly modified version of the `readMRI` function from the mritc package.

**Value**
An array.

**Examples**

```r
## load a nifti file
path <- system.file(file.path("nifti"), package = "MRIaggr")
nifti.Pat1_TTP_t0 <- readMRI(file = file.path(path,"TTP_t0"), format = "nifti")
dim(nifti.Pat1_TTP_t0)

## Not run:
## load an analyse file (example of oro.nifti::readANALYZE)
path <- system.file("anlz", package = "oro.nifti")
analyse.avg <- readMRI(file.path(path,"avg152T1"), format = "analyze")
graphics::image(analyse.avg[, 45, 1])

## load a nifti file (example of oro.nifti::readNIfTI)
nifti.ffd <- readMRI(file.path(system.file("nifti", package = "oro.nifti"), "filtered_func_data"),
                     format = "nifti")
graphics::image(nifti.ffd[, 10, 32])

## load a dicom file (examples of oro.dicom::readDICOMFile)
dicom.Abdo <- readMRI(system.file("dcm/Abdo.dcm", package = "oro.dicom"), format = "dicom")
graphics::image(dicom.Abdo, col = grey(0:64/64), axes = FALSE, xlab = "", ylab = "",
                 main = "Abdo.dcm")

## End(Not run)
```

---

**selectClinic**

**Extract clinical data**

**Description**
Extract the clinical data from a `MRIaggr` object.

**Usage**

```r
## S4 method for signature 'MRIaggr'
selectClinic(object, param=NULL)
```
selectContrast

Arguments

object an object of class MRIaggr. REQUIRED.
param the clinical parameters to extract. character vector or NULL leading to extract all
the clinical parameters.

Value

A one line data.frame containing the clinical data in columns.

See Also

affectClinic<- to affect values in the clinic slot.

Examples

## load a MRIaggr object
data("MRIaggr.Pat1_red", package="MRIaggr")

## select all clinical data
res <- selectClinic(MRIaggr.Pat1_red)

## select only the gender
res <- selectClinic(MRIaggr.Pat1_red, param="Gender")

selectContrast Extract contrast parameters

Description

Extract the contrast parameters from a Carto3D or a MRIaggr object.

Usage

## S4 method for signature 'Carto3D'
selectContrast(object, num=NULL, na.rm=FALSE, coords=TRUE,
format="any")

## S4 method for signature 'MRIaggr'
selectContrast(object, param=NULL, num=NULL, format="any",
slice_var="k", coords=FALSE, hemisphere="both", norm_mu=FALSE, norm_sigma=FALSE,
na.rm=FALSE, subset=NULL)
selectContrast

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>an object of class Carto3D or MRIaggr. REQUIRED.</td>
</tr>
<tr>
<td>param</td>
<td>the contrast parameters to extract. character vector or NULL.</td>
</tr>
<tr>
<td>num</td>
<td>the slices to extract. numeric vector or NULL.</td>
</tr>
<tr>
<td>format</td>
<td>the format of the output. Can be &quot;matrix&quot;, &quot;data.frame&quot; or &quot;any&quot;.</td>
</tr>
<tr>
<td>slice_var</td>
<td>the type of slice to extract. &quot;i&quot; for sagittal, &quot;j&quot; for coronal and &quot;k&quot; for transverse. character.</td>
</tr>
<tr>
<td>coords</td>
<td>the coordinates that should be extracted. logical or any of &quot;i&quot; &quot;j&quot; &quot;k&quot;.</td>
</tr>
<tr>
<td>hemisphere</td>
<td>the hemisphere to extract. character. See the details section.</td>
</tr>
<tr>
<td>norm_mu</td>
<td>the type of centering to extract on the parameter values. character. See the details section.</td>
</tr>
<tr>
<td>norm_sigma</td>
<td>the type of scaling to apply on the parameter values. character. See the details section.</td>
</tr>
<tr>
<td>na.rm</td>
<td>should observations with missing values be removed? logical.</td>
</tr>
<tr>
<td>subset</td>
<td>the subset of observations to extract. positive integer vector or NULL leading to use all observations</td>
</tr>
</tbody>
</table>

Details

ARGUMENTS:
Information about the param argument can be found in the details section of initParameter.
Information about the num argument can be found in the details section of initNum.
Possible values for the hemisphere argument are:

- "both": select all the observations.
- "left": select the observations from the left hemisphere.
- "right": select the observations from the right hemisphere.
- "lesion": select the observations belonging to the hemisphere(s) that contain(s) the lesion (if any).
- "controlateral": select the observations belonging to the hemisphere(s) that do not contain(s) the lesion (if any).

To select observations from a given hemisphere (all values except "both"), the parameter hemisphere must have been affected to the object using, for instance, calcHemisphere. In addition for "lesion" and "controlateral" values, the slot @hemispheres has to be filled using, for instance, calcHemisphere.
Possible values for the centering argument (norm_mu) and the scaling argument (norm_sigma) are:

- "FALSE": no normalization
- "global": the centering or scaling value is computed using all the observations.
- "global_1slice": the centering or scaling value is computed using all the observations that belong to the slice of the observation to normalize.
- "global_3slices": the centering or scaling value is computed using all the observations that belong to the slice of the observation to normalize, the slice above (if any) and the slice below (if any).
• "controlateral": the centering or scaling value is computed using the observations from the controlateral hemisphere.

• "controlateral_1slice": the centering or scaling value is computed using the observations from the controlateral hemisphere that belong to the slice of the observation to normalize.

• "controlateral_3slices": the centering or scaling value is computed using the observations from the controlateral hemisphere that belong to the slice of the observation to normalize, the slice above (if any) and the slice below (if any).

• "default_value": the default value of the parameter stored in the slot @default_value is used for the centering (for norm_mu only).

If coords is set to TRUE the dataset containing the contrast parameters values will also contains all the coordinates. If coords is set to FALSE, it will not contain any coordinates.

Argument subset can be a character value that refers to a logical parameter in the object defining the subset of observation to extract.

FUNCTION:
Each of the num, hemisphere and subset argument define a subset of the total set of observations.
It is the intersection of all these three subsets that is extracted.

When a normalisation is requested to center (resp. scale) the data, the normalisation value is extracted for each parameter in the element of the slot normalization that match the argument norm_mu (resp. norm_sigma). The parameters values are first centered by substraction with the value returned by norm_mu. Then they are scaled by division with the value returned by norm_sigma.

Value
A data.frame or a matrix containing the parameter in columns and the observations in rows. If only one parameter is requested and the format is set to "any" then a vector containing the parameter values is returned.

See Also
calcControlateral, calcRegionalContrast, calcFilter and calcTissueType to retreat and affect the modified contrast parameters. affectContrast<- to affect new contrast parameters. calcNormalization to compute and affect the normalisation values. affectNormalization<- to affect the normalization values when obtained from an external source. calcHemisphere and calcControlateral to compute and affect the hemispheres. affectHemisphere<- and affectContrast<- to affect hemispheres obtained from an external source.

Examples
### 1- Carto3D method ###
# load nifti files and convert them to Carto3D
path.nifti_files <- system.file("nifti",package = "MRIaggr")
nifti.Pat1_TTP_t0 <- readMRI(file=file.path(path.nifti_files,"TTP_t0"),format="nifti")
Carto3D.Pat1_TTP_t0 <- constCarto3D(nifti.Pat1_TTP_t0,identifier="Pat1",param="TTP_t0")

# select all observations
```r
carto1 <- selectContrast(Carto3D.Pat1_TTP_t0)
dim(carto1)

## select observations from slices 1 to 3 and return the result into a data.frame
carto2 <- selectContrast(Carto3D.Pat1_TTP_t0,num=1:3,coords=FALSE,format="data.frame")
dim(carto2)

## select observations from slices 1 to 3 and return the result into a vector
carto3 <- selectContrast(Carto3D.Pat1_TTP_t0,num=1:3,coords=FALSE)

length(carto3)

### 2- MRIaggr method ###
## load a MRIaggr object
data("MRIaggr.Pat1_red", package="MRIaggr")

## select all parameters and all observations
carto <- selectContrast(MRIaggr.Pat1_red)
dim(carto)
head(carto)

## select a subset of parameters
carto <- selectContrast(MRIaggr.Pat1_red,param=c("DWI_t0","T2_FLAIR_t2"))
dim(carto)
head(carto)

## select a subset of parameters on slices 1 to 3
carto <- selectContrast(MRIaggr.Pat1_red,num=1:3,param=c("DWI_t0","T2_FLAIR_t2"))
dim(carto)
head(carto)

## select a subset of parameters on slices 1 to 3 and normalized the center
## the values using the controlateral

carto <- selectContrast(MRIaggr.Pat1_red,num=1:3,param=c("DWI_t0","T2_FLAIR_t2"),
norm_mu="controlateral")
dim(carto)
head(carto)

## select only observations which are lesioned at admission (i.e. MASK_DWI_t0=TRUE)
carto <- selectContrast(MRIaggr.Pat1_red,subset="MASK_DWI_t0",
                        param=c("DWI_t0","T2_FLAIR_t2","MASK_DWI_t0"))
dim(carto)
head(carlo)

## select only observations which are lesioned at admission (i.e. MASK_DWI_t0=TRUE) with coordinates
carto <- selectContrast(MRIaggr.Pat1_red,subset="MASK_DWI_t0",
                        param=c("DWI_t0","T2_FLAIR_t2","MASK_DWI_t0"),coords=TRUE)
dim(carto)
head(carto)

## select only observations for which i=55
carto <- selectContrast(MRIaggr.Pat1_red,slice_var="i",num=55,coords=TRUE)
dim(carto)
head(carto)
```

selectCoords

Extract spatial coordinates

Description

Extract the coordinates from a Carto3D or from a MRIaggr object.

Usage

```r
## S4 method for signature 'Carto3D'
selectCoords(object, coords=c("i", "j", "k"), num=NULL, format="any")

## S4 method for signature 'MRIaggr'
selectCoords(object, coords=c("i", "j", "k"), spatial_res=c(1,1,1),
              num=NULL, hemisphere="both", subset=NULL, slice_var="k", format="data.frame")
```

Arguments

- `object` an object of class Carto3D or MRIaggr. REQUIRED.
- `coords` the coordinates that should be extracted. Any of "i" "j" "k" or "index".
- `spatial_res` a dilatation factor for the coordinates. positive numeric vector of size 3.
- `num` the slices to extract. numeric vector or NULL.
- `hemisphere` the hemisphere to extract. character.
- `subset` the subset of observations to extract. positive integer vector or NULL leading to use all observations
- `slice_var` the type of slice to extract. "i" for sagittal, "j" for coronal and "k" for transverse. character.
- `format` the format of the output. Can be "matrix", "data.frame" or "any".

Details

ARGUMENTS:
Information about the num argument can be found in the details section of initNum.
Information about the hemisphere argument can be found in the details section of selectContrast.

FUNCTION:
Each of the num, hemisphere and subset argument define a subset of the total set of observations.
It is the intersection of all these three subsets that is extracted.

Value

A data.frame or a matrix containing the coordinates in columns and the observations in rows.
If only one coordinate is requested and the format is set to "any" then a vector containing the coordinate values is returned.
See Also

calchemisphere to identify the hemispheres.
affecthemisphere and affectcontrast to affect hemispheres obtained from an external source.

Examples

    ### 1- Carto3D method ###
    # load nifti files and convert them to Carto3D
    path.nifti_files <- system.file("nifti", package = "MRIaggr")
    nifti.Pat1_TTP_t0 <- readMRI(file = file.path(path.nifti_files, "TTP_t0"), format = "nifti")
    Carto3D.Pat1_TTP_t0 <- constCarto3D(nifti.Pat1_TTP_t0, identifier = "Pat1", param = "TTP_t0")

    ## selection all coordinates
    coords1 <- selectCoords(Carto3D.Pat1_TTP_t0)
    dim(coords1)

    ## selection coordinates i and j from slices 1 to 3
    coords2 <- selectCoords(Carto3D.Pat1_TTP_t0, num = 1:3, coords = c("i", "j"))
    dim(coords2)

    ### 2- MRIaggr method ###
    # load a MRIaggr object
    data("MRIaggr.Pat1_red", package = "MRIaggr")

    ## selection all coordinates for all observations
    coords <- selectCoords(MRIaggr.Pat1_red)
    dim(coords)
    head(coords)

    ## selection coordinate i for slices 1 and 3
    coords <- selectCoords(MRIaggr.Pat1_red, coord = "i", num = c(1, 3))
    dim(coords)
    head(coords)

    ## selection coordinate i for observations in the hemisphere containing the lesion
    coords <- selectCoords(MRIaggr.Pat1_red, hemisphere = "lesion", num = c(1, 3))
    dim(coords)
    head(coords)

    ## selection coordinate i for observations in the right hemisphere
    coords <- selectCoords(MRIaggr.Pat1_red, hemisphere = "right", num = c(1, 3))
    dim(coords)
    head(coords)

    ## selection all coordinates and rescale them
    coords <- selectCoords(MRIaggr.Pat1_red, spatial_res = c(1.875, 1.875, 6))
    dim(coords)
    head(coords)

    ## selection coordinate i and j and return a matrix
    coords <- selectCoords(MRIaggr.Pat1_red, format = "matrix")
**selectDefault_value**

is(coords)
head(coords)

---

**selectDefault_value**  
*Extract reference values*

### Description

Extract the reference values of the contrast parameters from a **Carto3D** or from a **MRIaggr** object.

### Usage

```r
## S4 method for signature 'Carto3D'
selectDefault_value(object)

## S4 method for signature 'MRIaggr'
selectDefault_value(object, param=NULL, as.numeric=FALSE)
```

### Arguments

- **object**: an object of class **Carto3D** or **MRIaggr**. REQUIRED.
- **param**: the contrast parameters for which the reference values should be returned. *character vector* or NULL leading to extract reference values for all available contrast parameters.
- **as.numeric**: should the default values be converted from character to numeric. *logical*.

### Value

A **data.frame** with one line an several columns containing the default value of each parameter.

### Examples

```r
#### 1- Carto3D method ####
## load nifti files and convert them to Carto3D
path.nifti_files <- system.file("nifti", package = "MRIaggr")
nifti.Patient1_TTP_t0 <- readMRI(file=file.path(path.nifti_files, "TTP_t0"), format="nifti")
Carto3D.Patient1_TTP_t0 <- constCarto3D(nifti.Patient1_TTP_t0, identifier="Patient1", param="TTP_T0")

## selection
selectDefault_value(Carto3D.Patient1_TTP_t0)

#### 2- MRIaggr method ####
## load a MRIaggr object
data("MRIaggr.Patient1_red", package="MRIaggr")

## select default values for all parameters
res <- selectDefault_value(MRIaggr.Patient1_red)
```
## selectDescStats

### Description

Extract elements in the `ls_descStats` slot of a `MRIaggr` object.

### Usage

```r
# S4 method for signature 'MRIaggr'
selectDescStats(object, name=NULL, subset_W=NULL, hemisphere="both", num=NULL, slice_var="k")
```

### Arguments

- **object**: an object of class `MRIaggr`. REQUIRED.
- **name**: the name of the element to select. character or NULL leading to select all available elements.
- **subset_W**: the subset of observations to select. integer vector or NULL leading to select all the observations.
- **hemisphere**: the hemisphere to use. character.
- **num**: the slices to use. numeric vector or NULL.
- **slice_var**: the type of slice to extract. "i" for sagittal, "j" for coronal and "k" for transverse. character.

### Details

**ARGUMENTS:**
Information about the `num` argument can be found in the details section of `initNum`.

Information about the `hemisphere` argument can be found in the details section of `selectContrast`.

**FUNCTION:**
Arguments `index.W`, `hemisphere`, `num`, `format.W`, `style.W` are active only if `name` equal "W_euclidienn".
If `index.W` is not NULL then arguments `hemisphere` and `num` are ignored.
Each of the `num`, `hemisphere` and `subset` argument define a subset of the total set of observations.
It is the intersection of all these three subsets is extracted.
selectHemispheres

See Also

affectDescStats<- to affect elements in the ls_descStats slot.
calcHemisphere to identify the hemispheres.
affectHemisphere<- and affectContrast<- to affect hemispheres obtained from an external source.

Examples

## load a MRIaggr object
data("MRIaggr.Pat1_red", package="MRIaggr")
calcGroupsMask(MRIaggr.Pat1_red,mask=c("MASK_DWI_t0","MASK_T2_FLAIR_t2"),
distband=6,
spatial_res=c(1.875,1.875,6),
update.object=TRUE,overwrite=TRUE)

## select all elements in the slot @ls_descStats
ls_descStats <- selectDescStats(MRIaggr.Pat1_red)
names(ls_descStats)

## get the name of all elements present in the slot @ls_descStats
selectParameter(MRIaggr.Pat1_red,type="ls_descStats")

## select a specific element
res <- selectDescStats(MRIaggr.Pat1_red,name="GroupsLesion")

## compute and affect a neighborhood matrix
calcW(MRIaggr.Pat1_red,distband=3,update.object=TRUE,overwrite=TRUE,
spatial_res=c(1.875,1.875,6))

## select the neighborhood matrix for a subset of observations
res <- selectDescStats(MRIaggr.Pat1_red,name="W_euclidean",hemisphere="lesion",num=3)
dim(res)
table(rowSums(res>=0))
res <- selectDescStats(MRIaggr.Pat1_red,name="W_euclidean",subset_W = 1:10)
res

selectHemispheres Extract the position of the lesion in the hemispheres

Description

Extract the position of the lesion in the hemispheres from a MRIaggr object.

Usage

## S4 method for signature 'MRIaggr'
selectHemispheres(object,hemisphere="both")
Arguments

object an object of class Carto3D or MRIaggr. REQUIRED.

hemisphere the hemisphere of interest. character. See the details section.

Details

Possible values for the hemisphere argument are:

- "both": indicates the presence of the lesion in both hemispheres.
- "left": indicates the presence of the lesion in the left hemisphere.
- "right": indicates the presence of the lesion in the right hemisphere.
- "lesion": indicates which hemispheres contains the lesion.
- "controlateral": indicates which hemispheres are healthy.

Value

A character vector or a data.frame.

Examples

data("MRIaggr.Pat1_red", package="MRIaggr")

## selection
selectHemispheres(MRIaggr.Pat1_red)
selectHemispheres(MRIaggr.Pat1_red,"right")
selectHemispheres(MRIaggr.Pat1_red,"left")
selectHemispheres(MRIaggr.Pat1_red,"lesion")
selectHemispheres(MRIaggr.Pat1_red,"controlateral")

selectHistory Extract the history of the object

Description

Extract the history of a MRIaggr object.

Usage

## S4 method for signature 'MRIaggr'
selectHistory(object)

Arguments

object an object of class MRIaggr. REQUIRED.
**selectIdentifier**

**Value**

A list of each calc or const method that was applied to the object. For each method, it contains the call, the date of call and potential extra elements.

**Examples**

```r
## load a MRIaggr object
data("MRIaggr.Pat1_red", package="MRIaggr")

## selection
selectHistory(MRIaggr.Pat1_red)
```

**Description**

Extract the patient identifier from a Carto3D or from a MRIaggr object.

**Usage**

```r
## S4 method for signature 'Carto3D'
selectIdentifier(object)

## S4 method for signature 'MRIaggr'
selectIdentifier(object)
```

**Arguments**

- `object` an object of class Carto3D or MRIaggr. REQUIRED.

**Value**

A character.

**Examples**

```r
#### 1- Carto3D method ####
## load nifti files and convert them to Carto3D
path.nifti_files <- system.file("nifti",package = "MRIaggr")
nifti.Pat1_TTP_t0 <- readMRI(file=file.path(path.nifti_files,"TTP_t0"),format="nifti")
Carto3D.Pat1_TTP_t0 <- constCarto3D(nifti.Pat1_TTP_t0,identifier="Pat1",param="TTP_t0")

## selection
selectIdentifier(Carto3D.Pat1_TTP_t0)

#### 2- MRIaggr method ####
## load a MRIaggr object
data("MRIaggr.Pat1_red", package="MRIaggr")
```
## selectN

selectN(object, num=NULL, hemisphere="both", subset=NULL)

### Description

Extract the number of observations contained in a Carto3D or in a MRIaggr object.

### Usage

```r
## S4 method for signature 'MRIaggr'
selectN(object, num=NULL, hemisphere="both", subset=NULL)
```

---

## selectMidplane

Select the position of the mid-saggital plan

### Description

Extract the position of the mid-saggital plan from a MRIaggr object.

### Usage

```r
## S4 method for signature 'MRIaggr'
selectMidplane(object)
```

### Arguments

- `object`: an object of class MRIaggr. REQUIRED.

### Value

A two column matrix.

### Examples

```r
data("MRIaggr.Pat1_red", package="MRIaggr")

## S4 method for signature 'MRIaggr'
selectMidplane(MRIaggr.Pat1_red)
```
selectNormalization

Arguments

object an object of class MRIaggr. REQUIRED.
num the slices to consider. numeric vector or NULL.
hemisphere the hemisphere to consider. character.
subset the subset of observations to consider. positive integer vector or NULL leading to consider all observations.

Details

ARGUMENTS:
Information about the num argument can be found in the details section of initNum.
Information about the hemisphere argument can be found in the details section of selectContrast.

FUNCTION:
Each of the num, hemisphere and subset argument define a subset of the total set of observations. It is the length of the intersection of all these three subsets that is measured.

Value

An integer.

Examples

```r
## load a MRIaggr object
data("MRIaggr.Pat1_red", package="MRIaggr")

## total number of observations
res <- selectN(MRIaggr.Pat1_red)

## number of observations for the hemisphere that contains the lesion
res <- selectN(MRIaggr.Pat1_red,hemisphere="lesion")

## number of observations in the first 1000 observations t
## that are in the hemisphere containing the lesion
res <- selectN(MRIaggr.Pat1_red,subset=1:1000,hemisphere="lesion")
```

selectNormalization Extract the normalization values

Description

Extract the normalization values from a MRIaggr object.

Usage

```r
## S4 method for signature 'MRIaggr'
selectNormalization(object,type=NULL,mu=TRUE,sigma=TRUE,
                    hemisphere="both",num=NULL,param=NULL)
```
selectNormalization

Arguments

object an object of class MRIaggr. REQUIRED.
type the type of normalization. Must be one of "global", "slice", "3slices" or NULL leading to select all types of normalization.
mu should the centering values for the normalization be returned. logical. Active only if type is "slice" or "3slices".
sigma should the scaling values for the normalization be returned. logical. Active only if type is "slice" or "3slices".
num the slices to extract. numeric vector or NULL.
hemisphere the hemisphere to extract. character.
param the contrast parameters for which the normalization values should be extracted. character vector or NULL indicating all available contrast parameters.

Details

ARGUMENTS:
Information about the num argument can be found in the details section of initNum.
Information about the hemisphere argument can be found in the details section of selectContrast.

Value

A data.frame or a list.

See Also

calcNormalization to compute and affect the normalisation values.
affectNormalization to affect the normalization values when obtained from an external source.
calcHemisphere to identify the hemispheres.
affectHemisphere and affectContrast to affect hemispheres obtained from an external source.

Examples

```r
## load a MRIaggr object
data("MRIaggr.Pat1_red", package="MRIaggr")

## select all normalization values
res <- selectNormalization(MRIaggr.Pat1_red)
names(res)

## select specific normalization normalization values
# computed on the whole brain
res <- selectNormalization(MRIaggr.Pat1_red, type="global", mu=TRUE, sigma=FALSE, hemisphere="both")
# idem but only for DWI_t0
res <- selectNormalization(MRIaggr.Pat1_red, type="global", mu=TRUE, sigma=FALSE, param="DWI_t0")

# computed by slice
res <- selectNormalization(MRIaggr.Pat1_red, type="slice", mu=TRUE, sigma=FALSE, hemisphere="both")
```
selectParameter

# idem for slice 1
res <- selectNormalization(MRIaggr.Pat1_red, type="slice", mu=TRUE, sigma=FALSE, num=1)

# computed on 3 consecutive slices
res <- selectNormalization(MRIaggr.Pat1_red, type="3slices", mu=FALSE, sigma=TRUE,
                           hemisphere="left", num=2, param="T2_FLAIR_t2")

selectParameter Extract parameters

Description

Extract parameters from a Carto3D or from a MRIaggr object.

Usage

## S4 method for signature 'Carto3D'
selectParameter(object)

## S4 method for signature 'MRIaggr'
selectParameter(object, type="contrast", mask=TRUE)

Arguments

object an object of class MRIaggr. REQUIRED.
type the type of parameter to select. character. See the details section.
mask should the mask be considered as an available contrast parameter? logical.

Details

ARGUMENTS:
Possible values for type are:

- "clinic": return the name of the clinical parameters.
- "contrast": return the name of the contrast parameters.
- "ls_descStats": return the name of the elements in @ls_descStats.

See Also

affectContrast<- to affect new parameters.
Examples

#### 1- Carto3D method ####

```r
## load nifti files and convert them to Carto3D
path.nifti_files <- system.file("nifti", package = "MRIaggr")
nifti.Pat1.TTP_t0 <- readMRI(file = file.path(path.nifti_files, "TTP_t0"), format = "nifti")
Carto3D.Pat1.TTP_t0 <- constCarto3D(nifti.Pat1.TTP_t0, identifier = "Pat1", param = "TTP_t0")
```

#### selection ####

```r
selectParameter(Carto3D.Pat1.TTP_t0)
```

#### 2- MRIaggr method ####

```r
## load a MRIaggr object
data("MRIaggr.Pat1_red", package = "MRIaggr")

## extract all imaging parameters
res <- selectParameter(MRIaggr.Pat1_red)

## extract the names of the parameters in the slot @ls_descStats
res <- selectParameter(MRIaggr.Pat1_red, type = "ls_descStats")
```

---

**selectTable**

Extract volumic information

---

**Description**

Extract the volumic information (lesion, hyperfusion or reperfusion) from a **MRIaggr** object.

**Usage**

```r
## S4 method for signature 'MRIaggr'
selectTable(object, type, size = FALSE)
```

**Arguments**

- **object**: an object of class **MRIaggr**. REQUIRED.
- **type**: the table to extract. Can be "lesion", "reperfusion" or "hypoperfusion". REQUIRED.
- **size**: should the values in the table correspond to a number of voxels (FALSE) or a volume (TRUE).

**Value**

A **data.frame** containing volumetric measurements.
selectVoxelDim

See Also
calcTableHypReperf to compute and affect the hypoperfusion and reperfusion tables.
calcTableLesion to compute and affect the lesion table.
affectTable<-. to affect data in the table slots.

Examples

```r
## load a MRIaggr object
data("MRIaggr.Pat1_red", package="MRIaggr")

## selection
res <- selectTable(MRIaggr.Pat1_red,type="lesion")
res <- selectTable(MRIaggr.Pat1_red,type="reperfusion")
res <- selectTable(MRIaggr.Pat1_red,type="hypoperfusion")
```

Description

Extract the spatial dimensions of the contrast data from a Carto3D or from a MRIaggr object.

Usage

```r
## S4 method for signature 'Carto3D'
selectVoxelDim(object)

## S4 method for signature 'MRIaggr'
selectVoxelDim(object)
```

Arguments

- `object`: an object of class Carto3D or MRIaggr. REQUIRED.

Value

A data.frame with one line and three columns named "i", "j", "k" indicating the corresponding dimension.

Examples

```r
## I- Carto3D method
## load nifti files and convert them to Carto3D
path.nifti_files <- system.file("nifti", package = "MRIaggr")
nifti.Pat1_TTP_t0 <- readMRI(file=file.path(path.nifti_files,"TTP_t0"),format="nifti")
Carto3D.Pat1_TTP_t0 <- constCarto3D(nifti.Pat1_TTP_t0,identifier="Pat1",param="TTP_t0")

## selection
```
selectVoxelDim(Carto3D_Pat1_TTP_t0)

### 2- MRIaggr method ###
## load a MRIaggr object
data("MRIaggr_Pat1_red", package="MRIaggr")

## selection
selectVoxelDim(MRIaggr_Pat1_red)

---

**selectVoxelSize**

*Extract the dimensions of a voxel*

### Description

Extract the spatial dimensions of a voxel contained in a `MRIaggr` object.

### Usage

```r
## S4 method for signature 'MRIaggr'
selectVoxelSize(object, unit=TRUE)
```

### Arguments

- `object`: an object of class `MRIaggr`. REQUIRED.
- `unit`: should the unit be returned? `logical`.

### Value

A `data.frame` with one line and three or four columns containing the voxel size for each dimension and the size unit (if requested).

### Examples

```r
data("MRIaggr_Pat1_red", package="MRIaggr")

## selection
selectVoxelSize(MRIaggr_Pat1_red)
```
Summary Method for Class "MRIaggr"

Description

Summarize some information of an object of class "MRIaggr".

Usage

```r
## S4 method for signature 'MRIaggr'
summary(object, param=FALSE, clinic=FALSE, descStats=FALSE, history=FALSE)
```

Arguments

- `object`: an object of class `MRIaggr`. REQUIRED.
- `param`: should detailed information be printed for the contrast parameters? logical.
- `clinic`: should detailed information be printed for the clinical attribute? logical.
- `descStats`: should detailed information be printed for the `ls_descStats` attribute? logical.
- `history`: should the `calc` and `const` methods that have been applied to the object be listed? logical.

Value

None.

Examples

```r
## load a MRIaggr object
data("MRIaggr.Pat1_red", package="MRIaggr")
summary(MRIaggr.Pat1_red)
```

Remove a contrast parameter

Description

Remove a contrast parameter from a MRIaggr object.

Usage

```r
## S4 replacement method for signature 'MRIaggr'
supprContrast(object, trace=TRUE) <- value
```
Arguments

- object: an object of class MRIaggr. REQUIRED.
- value: the name of the parameter(s) that should be removed. character vector. REQUIRED.
- trace: should the execution of the function be traced? logical.

Details

ARGUMENTS:
value can be a numeric vector indicating the position of the parameters to remove in the data slot.

Value

None.

See Also

affectContrast<- to affect a contrast parameter.
selectParameter to display the contrast parameters.

Examples

```r
## load a MRIaggr object
data("MRIaggr.Pat1_red", package="MRIaggr")

## available contrast parameters
selectParameter(MRIaggr.Pat1_red)

## delete two contrast parameters
supprContrast(MRIaggr.Pat1_red) <- c("MTT_t0","MASK_DWI_t0")

## remaining contrast parameters
selectParameter(MRIaggr.Pat1_red)
```

supprDescStats  Remove an element of ls_descStats

Description

Remove an element from the ls_descStat attribute of a MRIaggr object.

Usage

```r
## S4 replacement method for signature 'MRIaggr'
supprDescStats(object, trace=TRUE) <- value
```
supprDescStats

Arguments

object        an object of class MRIaggr. REQUIRED.
value         the name of the element(s) that should be removed. character vector. REQUIRED.
trace         should the execution of the function be traced ? logical.

Value

None.

See Also

affectDescStats<- to affect an element to the ls_descStat attribute.
selectParameter to display the elements of the ls_descStat attribute.

Examples

    ## load a MRIaggr object
data("MRIaggr.Pat1_red", package="MRIaggr")

    ## existing elements in @ls_descStats
selectParameter(MRIaggr.Pat1_red,type="ls_descStats")

    ## delete one element in @ls_descStats
supprDescStats(MRIaggr.Pat1_red) <- "index_sauve"

    ## remaining elements in @ls_descStats
selectParameter(MRIaggr.Pat1_red,"ls_descStats")
Index

*Topic classes
  Carto3D-class, 60
  MRIaggr-class, 88

*Topic datasets
  MRIaggr.Pat1_red, 91

*Topic function, internal
  initCol, 79
  initDisplayWindow, 80
  initIndex, 83
  initWindow, 86
  legendMRI, 88
  plotMRI, 105

*Topic functions, Sweave
  constSweave, 68

*Topic functions, internal
  calcContro_cpp, 22
  calcCriteriaGR, 23
  calcGroupsCoords_cpp, 34
  calcGroupsW_cpp, 37
  calcHemi_cpp, 40
  calcRadius_cpp, 42
  filtrage2D_cpp, 74
  filtrage2Dmed_cpp, 74
  filtrage3D_cpp, 75
  filtrage3Dmed_cpp, 75
  GRalgo, 76
  initGR, 82
  pointsOutline, 108

*Topic functions
  array2df, 13
  calcAUPRC, 16
  calcGR, 29
  calcGroupsCoords, 32
  calcGroupsW, 36
  calcSigmaGR, 46
  Carto3D2MRIaggr, 61
  constCarto3D, 62
  constMRIaggr, 65
  df2array, 71
  EDK, 73
  initFilter, 81
  initNeighborhood, 84
  outline, 97
  readMRI, 109

*Topic methods, internal
  initNum, 85
  initParameter, 86
  pointsHemisphere, 107

*Topic methods, summary
  summary-methods, 129

*Topic methods
  affectClinic, 4
  affectContrast, 5
  affectDescStats, 7
  affectHemisphere, 8
  affectNormalization, 10
  affectTable, 11
  boxplotMask, 14
  calcBrainMask, 17
  calcControlateral, 20
  calcDistMask, 23
  calcDistTissues, 25
  calcFilter, 26
  calcGroupsMask, 34
  calcHemisphere, 37
  calcNormalization, 40
  calcRegionalContrast, 42
  calcROCthreshold, 44
  calcSmoothMask, 48
  calcTableHypoReperf, 50
  calcTableLesion, 53
  calcTissueType, 57
  calcW, 58
  constCompressMRIaggr, 63
  constReduceMRIaggr, 66
  heatmapMRIaggr, 77
  multiplot, 92
  outlineMRIaggr, 98
INDEX

outline, 97, 100
outlineMRIaggr, 90, 98
outlineMRIaggr, MRIaggr-method
   (outlineMRIaggr), 98
par, 15, 47, 78, 80, 88, 102, 107, 108
pdf, 87
plot.default, 102
plotDistClass, 90, 101
plotDistClass, MRIaggr-method
   (plotDistClass), 101
plotLesion3D, 90, 94, 103
plotLesion3D, MRIaggr-method
   (plotLesion3D), 103
plotMRI, 105
plotTableLesion, 12, 90, 106
plotTableLesion, MRIaggr-method
   (plotTableLesion), 106
png, 87
pointsHemisphere, 90, 107
pointsHemisphere, MRIaggr-method
   (pointsHemisphere), 107
pointsOutline, 108
postscript, 87
readMRI, 91, 109

readClinic, 5, 89, 110
readClinic, MRIaggr-method
   (readClinic), 110
readContrast, 6, 15, 19, 21, 25, 27, 43, 49,
   51, 55, 59, 77, 78, 83, 89, 94, 100,
   102, 111, 115, 118, 123, 124
readContrast, Carto3D-method
   (readContrast), 111
readContrast, MRIaggr-method
   (readContrast), 111
readCoords, 89, 115
readCoords, Carto3D-method
   (readCoords), 115
readCoords, MRIaggr-method
   (readCoords), 115
readDefault_value, 89, 117
readDefault_value, Carto3D-method
   (readDefault_value), 117
readDefault_value, MRIaggr-method
   (readDefault_value), 117
readDescStats, 7, 33, 45, 89, 118

readDescStats, MRIaggr-method
   (readDescStats), 118
readHistories, 89, 119
readHistories, MRIaggr-method
   (readHistories), 119
readHistory, 89, 120
readHistory, MRIaggr-method
   (readHistory), 120
readIdentifier, 89, 121
readIdentifier, Carto3D-method
   (readIdentifier), 121
readIdentifier, MRIaggr-method
   (readIdentifier), 121
readMidplane, 89, 122
readMidplane, MRIaggr-method
   (readMidplane), 122
readN, 89, 122
readN, MRIaggr-method (readN), 122
readNormalization, 11, 41, 89, 123
readNormalization, MRIaggr-method
   (readNormalization), 123
readParameter, 9, 39, 89, 125, 130, 131
readParameter, Carto3D-method
   (readParameter), 125
readParameter, MRIaggr-method
   (readParameter), 125
readTable, 12, 51, 89, 126
readTable, MRIaggr-method
   (readTable), 126
readVoxelDim, 89, 127
readVoxelDim, Carto3D-method
   (readVoxelDim), 127
readVoxelDim, MRIaggr-method
   (readVoxelDim), 127
readVoxelSize, 89, 128
readVoxelSize, MRIaggr-method
   (readVoxelSize), 128
summary, MRIaggr-method, 90
summary, MRIaggr-method
   (summary-methods), 129
summary-methods, 129
suppContrast, 129
suppContrast<-, 90
suppContrast<-(suppContrast), 129
suppContrast<-, MRIaggr-method
   (suppContrast), 129
suppDescStats, 130
suppDescStats<-, 90

suppDescStats<-, MRIaggr-method
   (suppDescStats), 130
supprDescStats<- (supprDescStats), 130
supprDescStats<-, MRIaggr-method
  (supprDescStats), 130
svg, 87