Package ‘ror’

February 20, 2015

Version 1.2
Date 2013-02-22
Title Robust Ordinal Regression MCDA library
Author Tommi Tervonen
Maintainer Tommi Tervonen <tommi@smaa.fi>
Description An R package for computing both exact- and stochastic robust ordinal regression, and maximal vectors.
Depends R (>= 2.7.0), rJava (>= 0.8-0), ROI (>= 0.0.7),
ROI.plugin.glpk (>= 0.0-1), igraph (>= 0.6.4)
SystemRequirements Java (>= 1.5)
License GPL-3
NeedsCompilation no
Repository CRAN
Date/Publication 2013-02-25 18:24:01

R topics documented:

ror-package .......................................................... 2
maximalvectors .................................................. 3
maximalvectors.indices ................................. 4
rorsmaa ................................................................. 4
sample.vfs.gibbs ........................................ 5
sample.vfs.rejection ........................................... 7
utagms .......................................................... 8
utagms.strong.necessary ............................. 9

Index 11
Description

This package implements UTAGMS and RORSMAA MCDA methods for ranking multiple alternatives in terms of multiple criteria. The current version assumes ascending preferences, i.e., higher criterion evaluation means higher preferability (=better).

Author(s)

Tommi Tervonen
Maintainer: Tommi Tervonen <tommi@smaa.fi>

References


See Also

rorsmaa, utagms, sample.vfs.gibbs, sample.vfs.rejection

Examples

# Set Java VM memory use to 2g not to run out of heap space
options(java.parameters = "-Xmx2g")
library(ror)

## Function needed to generate pareto-optimal alternatives
randomPointFromHypersphere <- function(ncrit) {
  rns <- c()
  while(TRUE) {
    rns <- rnorm(ncrit)
    if (all(rns > 0)) {
      break
    }
  }
  mul <- 1 / sqrt(sum(rns * rns))
  return(rns * mul)
}

performances <- t(replicate(10, randomPointFromHypersphere(5))) # 10 alts, 5 crit
preferences <- matrix(c(1, 2, 4, 5, 7, 8, 1, 3), ncol=2, byrow=TRUE)

## Necessary relation
Maximal Vector Computation

Description
Maximal Vector Computation using the BEST algorithm. The current version assumes ascending preferences, i.e. higher criterion evaluation means higher preferability (=better).

Usage
maximalvectors(performances)

Arguments
performances m x n performance matrix with m alternatives and n criteria

Value
Matrix of performances of the non-dominated alternatives

See Also
ror-package, maximalvectors.indices

Examples
# Set Java VM memory use to 2g not to run out of heap space
options(java.parameters = "-Xmx2g")
library(ror)

## Test with pareto-optimal alternatives
performances <- matrix(c(0.1823587, 0.5232321, 0.7595968, 0.2964752, 0.1676054, 0.5488093, 0.1604821, 0.4699517, 0.4170541, 0.5357071, 0.1292226, 0.2366909, 0.7583132, 0.3765545, 0.4587448), ncol=5, byrow=TRUE)
nondominated <- maximalvectors(performances)
stopifnot(nrow(nondominated) == 3)
maximalvectors.indices

Maximal Vector index computation

Description
Maximal Vector Computation using the BEST algorithm. The current version assumes ascending preferences, i.e. higher criterion evaluation means higher preferability (=better).

Usage
maximalvectors.indices(performances)

Arguments
performances m x n performance matrix with m alternatives and n criteria

Value
Row indices of the non-dominated alternatives

See Also
ror-package,maximalvectors

Examples
# Set Java VM memory use to 2g not to run out of heap space
options(java.parameters = "-Xmx2g")
library(ror)

performances <- matrix(runif(n=50), nrow=10) # 10 alts, 5 crit
nonDominatedIdx <- maximalvectors.indices(performances)

rorsmaa

Robust Ordinal Regression SMAA sampler

Description
Implements stochastic simulation of the indices used in a SMAA-type decision analysis with UTA^GMS models. The current version assumes ascending preferences, i.e. higher criterion evaluation means higher preferability (=better).

Usage
rorsmaa(performances, preferences)
sample.vfs.gibbs

Arguments

performances  m x n performance matrix with m alternatives and n criteria
preferences  k x 2 matrix of preferences statements (row indices of alternatives in performance matrix). Each row r is a preference statements meaning that preferences[k,1] is weakly preferred to preferences[k,2]

See Also
utagms,ror-package

Examples

library(ror)

### Function needed to generate pareto-optimal alternatives
randomPointFromHypersphere <- function(ncrit) {
  rns <- c()
  while(TRUE) {
    rns <- rnorm(ncrit)
    if (all(rns > 0)) {
      break
    }
  }
  mul <- 1 / sqrt(sum(rns * rns))
  return(rns * mul)
}

performances <- t(replicate(10, randomPointFromHypersphere(5))) # 10 alts, 5 crit
preferences <- matrix(c(1, 2, 4, 5, 7, 8, 1, 3), ncol=2, byrow=TRUE)

### RORSMAA gives the POIs and RAIs
ror <- rorsmaa(performances, preferences)
print(ror$poi)
print(ror$rai)
cat(ror$misses, "misses while generating 10k value functions\n")

### Not run:
### Plot the results
plot(ror$rai)
plot(ror$poi)

### End(Not run)
Description

Rejection Gibbs sampling of general monotone value functions. The current version assumes ascending preferences, i.e. higher criterion evaluation means higher preferability (=better).

Usage

sample.vfs.gibbs(performances, preferences, nr=10000, thinning=1, updInterval=1000)

Arguments

- performances: m x n performance matrix with m alternatives and n criteria
- preferences: k x 2 matrix of preferences statements (row indices of alternatives in performance matrix). Each row r is a preference statements meaning that preferences[k,1] is preferred to preferences[k,2]
- nr: The number of value functions to sample
- thinning: The thinning factor to use
- updInterval: Update interval for printing out current iteration. 0 = no information will be printed out

Value

Named tuple, where ‘vfs’ is a list where element [i] are the value functions for the i’th criterion, one function per row and ‘misses’ contains the amount of misses during value function sampling.

See Also

rorsmaa, utagms, ror-package, sample.vfs.rejection

Examples

# Set Java VM memory use to 2g not to run out of heap space
options(java.parameters = "-Xmx2g")
library(ror)

## Function needed to generate pareto-optimal alternatives
randomPointFromHypersphere <- function(ncrit) {
  rns <- c()
  while(TRUE) {
    rns <- rnorm(ncrit)
    if (all(rns > 0)) {
      break
    }
  }
  mul <- 1 / sqrt(sum(rns * rns))
  return(rns * mul)
}

performances <- t(replicate(10, randomPointFromHypersphere(5))) # 10 alts, 5 crit
preferences <- matrix(c(1, 2, 4, 5, 7, 8, 1, 3), ncol=2, byrow=TRUE)
sample.vfs.rejection

Description
Pure rejection sampling of general monotone value functions. The current version assumes ascending preferences, i.e. higher criterion evaluation means higher preferability (=better).

Usage
sample.vfs.rejection(performances, preferences, nr=10000, updInterval=1000)

Arguments
performances m x n performance matrix with m alternatives and n criteria
preferences k x 2 matrix of preferences statements (row indices of alternatives in performance matrix). Each row r is a preference statements meaning that preferences[k,1] is preferred to preferences[k,2]
nr The number of value functions to sample
updInterval Update interval for printing out current iteration. 0 = no information will be printed out

Value
Named tuple, where `vfs` is a list where element [[i]] are the value functions for the i'th criterion, one function per row and 'misses' contains the amount of misses during value function sampling.

See Also
rorsmaa,utagms,ror-package,sample.vfs.gibbs

Examples
# Set Java VM memory use to 2g not to run out of heap space
options( java.parameters = "-Xmx2g" )
library(ror)

## Function needed to generate pareto-optimal alternatives
randomPointFromHypersphere <- function(ncrit) {
  rns <- c()
  while(TRUE) {
    rns <- rnorm(ncrit)
    if (all(rns > 0)) {
      break
    }
  }

  value <- sum(rns)
  return(c(value, rns))
}
utagms

UTA^GMS MCDA solver

Description

Implements UTA^GMS robust ordinal regression: computes either the necessary- or the possible relation. Assumes ascending preferences, i.e. higher criterion evaluation means higher preferability (=better).

Usage

utagms(performances, strongPrefs=NULL, weakPrefs=NULL, indifPrefs=NULL, necessary=TRUE, strictVF=FALSE)

Arguments

performances   m x n performance matrix with m alternatives and n criteria.
strongPrefs    k x 2 matrix of strong preferences statements (row indices of alternatives in the performance matrix). Each row r is a preference statements meaning that preferences[k,1] is strongly preferred to preferences[k,2].
weakPrefs      k x 2 matrix of weak preferences statements (row indices of alternatives in the performance matrix). Each row r is a preference statements meaning that preferences[k,1] is weakly preferred to preferences[k,2].
indifPrefs     k x 2 matrix of indifference preferences statements (row indices of alternatives in the performance matrix). Each row r is a preference statements meaning that preferences[k,1] is equally preferred to preferences[k,2].
necessary      Whether to compute the necessary relation (TRUE) or the possible one (FALSE).
strictVF       Whether to use strictly increasing (TRUE) or monotonously increasing (FALSE) value functions.

See Also

rorsmaa,ror-package
Examples

```r
library(ror)

## example with S alternatives and S criteria
performances <- matrix(c(1.0, 1.0, 1.0, 2.0, 1.0, 1.1, 2.0, 0.5, 3.0), ncol=3, byrow=TRUE)
## a3 > a2 (strongly preferred)
strongPrefs <- matrix(c(3, 2), ncol=2, byrow=TRUE)

## Necessary relation
necrel <- utagms(performances, strongPrefs, necessary=TRUE, strictVF=TRUE)

## Possible relation with strictly increasing value functions
posrel <- utagms(performances, strongPrefs, necessary=FALSE, strictVF=FALSE)

## Sanity check, the necessary relation should be
## T F F
## T T F
## T T T
stopifnot(necrel == matrix(c(TRUE, FALSE, FALSE, TRUE, TRUE, FALSE, TRUE, TRUE, TRUE), ncol=3, byrow=TRUE))

## Sanity check, the possible relation should be
## T T F
## T T F
## T T T
stopifnot(posrel == matrix(c(TRUE, TRUE, FALSE, TRUE, TRUE, FALSE, TRUE, TRUE, TRUE), ncol=3, byrow=TRUE))

## Test with a2 ~ a3
necrel <- utagms(performances, strongPrefs=NULL, indifPrefs=strongPrefs, necessary=TRUE, strictVF=TRUE)

## Sanity check, the necessary relation should represent (a2 ~ a3)

## Not run:

## Plot the relation
plot(necrel)
```

```
## a3 > a2 and a2 > a3
strongPrefs <- matrix(c(3, 2, 2, 3), ncol=2, byrow=TRUE)

## Error as the model is infeasible
necrel <- utagms(performances, strongPrefs, necessary=TRUE, strictVF=TRUE)

## End(Not run)
```

Description

Implements UTA^GMS robust ordinal regression: computes the strong necessary relation. Assumes ascending preferences, i.e. higher criterion evaluation means higher preferability (=better).
Usage

utagms.strong.necessary(performances, strongPrefs=NULL, weakPrefs=NULL, indifPrefs=NULL, strictVF=FALSE)

Arguments

performances m x n performance matrix with m alternatives and n criteria.
strongPrefs k x 2 matrix of strong preferences statements (row indices of alternatives in the performance matrix). Each row r is a preference statements meaning that preferences[k,1] is strongly preferred to preferences[k,2].
weakPrefs k x 2 matrix of weak preferences statements (row indices of alternatives in the performance matrix). Each row r is a preference statements meaning that preferences[k,1] is weakly preferred to preferences[k,2].
indifPrefs k x 2 matrix of indifference preferences statements (row indices of alternatives in the performance matrix). Each row r is a preference statements meaning that preferences[k,1] is equally preferred to preferences[k,2].
strictVF Whether to use sctrictly increasing (TRUE) or monotonously increasing (FALSE) value functions.

See Also

rorsmaa,ror-package

Examples

library(ror)

## Example with 3 alternatives and 3 criteria
performances <- matrix(c(1.0, 1.0, 1.0, 2.0, 1.0, 1.1, 2.0, 0.5, 3.0), ncol=3, byrow=TRUE)
## a3 > a2 (strongly preferred)
strongPrefs <- matrix(c(3, 2), ncol=2, byrow=TRUE)

## Strong necessary relation
strongnec <- utagms.strong.necessary(performances, strongPrefs)

## Sanity check, the relation have a3 >^N a2
stopifnot(strongnec[3,2] == TRUE)

## Not run:
## Plot the relation
plot(strongnec)

## End(Not run)
Index

*Topic robust

- maximalvectors, 3
- maximalvectors.indices, 4
- ror-package, 2
- rorsmaa, 4
- sample.vfs.gibbs, 5
- sample.vfs.rejection, 7
- utagms, 8
- utagms.strong.necessary, 9

maximalvectors, 3
maximalvectors.indices, 4
ror (ror-package), 2
ror-package, 2
rorsmaa, 4
sample.vfs.gibbs, 5
sample.vfs.rejection, 7
utagms, 8
utagms.strong.necessary, 9