Package ‘analytics’

June 14, 2017

Type Package

Title Regression Outlier Detection, Stationary Bootstrap, Testing Weak Stationarity, and Other Tools for Data Analysis

Version 2.0

Date 2017-06-14

Author Albert Dorador

Maintainer Albert Dorador <albert.dorador@estudiant.upc.edu>

Description Current version includes outlier detection in a fitted linear model, stationary bootstrap using a truncated geometric distribution, a comprehensive test for weak stationarity, column means by group, weighted biplots, and a heuristic to obtain a better initial configuration in non-metric MDS.

Depends R (>= 3.1.0)

Imports powerplus (>= 3.1), tcltk (>= 3.3.1), cluster (>= 2.0.4), MASS (>= 7.3-45), car (>= 2.1-4), robust (>= 0.4-18), trend (>= 0.2.0), TSA (>= 1.01), lmtest (>= 0.9-35), fractal (>= 2.0-1), urca (>= 1.3-0), np (>= 0.60-3)

License GPL-2

Encoding UTF-8

ByteCompile true

LazyData true

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2017-06-14 21:33:44 UTC

R topics documented:

`Minstress` *Better Starting Configuration For Non-Metric MDS*

Description

`Minstress` is a heuristic to find better non-metric MDS solutions, by finding better starting configurations, instead of just using a random one.

Usage

```
Minstress(x, p, s, k, iter = 5, pb = F, m = "euclidean")
```

Arguments

`x`	a data frame containing numeric values only
`p`	the size of the population of seeds (any positive integer)
`s`	the number of seeds we sample (any positive integer)
`k`	the number of dimensions wanted (any positive integer)
`iter`	a positive integer specifying the number of iterations.
`pb`	a Boolean variable declaring if one wants to display a pogress bar (default: False)
`m`	a string specifying the distance method (default: 'euclidean')

Details

This function performs several iterations, each using a different starting seed, and in turn each one of those iterations performs non-metric MDS many times (typically, thousands or more) in an attempt to find the best seed (which induces a particular initial configuration) of them all.

Value

A list informing about dimensionality, minimum STRESS level found, and best seed found. One can then use the best seed found to perform non-metric MDS with a better initial configuration (generally).

Author(s)

Albert Dorador

Examples

```
require(MASS)

swiss.x <- as.data.frame(swiss[, -1])
Minstress(swiss.x, 1e5, 50, 2, iter = 3)

# Comparing without using Minstress (for such a low value of s, difference is minimal)
swiss.x <- as.matrix(swiss[, -1])
swiss.dist <- dist(swiss.x)
swiss.mds <- isoMDS(swiss.dist)
```

`offliers` *Takes Outliers Off*

Description

`offliers` Finds the existing outliers after fitting a linear model, according to various criteria.

Usage

```
offliers(dataset, mod, CD = TRUE, DFB = FALSE, COVR = FALSE, pctg = 100,
  intersection = TRUE)
```

Arguments

`dataset`	an object containing the data used in the linear regression model, typically a data frame.
`mod`	an object of class "lm" (the model fitted by the user).
`CD`	a Boolean variable, indicating whether the Cook's Distance criterion is to be used.
`DFB`	a Boolean variable, indicating whether the DFBetas criterion is to be used.
`COVR`	a Boolean variable, indicating whether the COVRATIO criterion is to be used.
`pctg`	a real number between 0 and 100, indicating the maximum percentage of original observations to be removed.
`intersection`	a Boolean variable, indicating whether the intersection or the union of the outliers detected must be considered (in case more than one criterion has been selected).

Details

Criteria available: Cook's Distance, DFBetas, and COVRATIO. DFFits has not been included because it is conceptually equivalent to Cook's Distance; in fact there's a closed-form formula to convert one value to the other. See references. The user can select any combination of those, and take off the outliers in the intersection (default) or union.

Value

a list containing, according to each criterion selected, which observations have been identified as outliers, how many they are, what percentage of the total number of observations they represent, and, if more than one criterion has been selected, the final outliers, quantity and percentage.

Author(s)

Albert Dorador

References

Cook & Weisberg 1982, "Residuals and Influence in Regression". http://conservancy.umn.edu/handle/11299/37076

Examples

```
require(graphics)
## Annette Dobson (1990) "An Introduction to Generalized Linear Models".
## Page 9: Plant Weight Data.
ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)
trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)
group <- gl(2, 10, 20, labels = c("Ctl","Trt"))
weight <- c(ctl, trt)
lm.D9 <- lm(weight ~ group)
db <- data.frame(weight, group)

offliers(db,lm.D9)
offliers(db,lm.D9, CD = FALSE, DFB = TRUE, COVR = TRUE)
offliers(db,lm.D9, CD = TRUE, DFB = TRUE, COVR = TRUE, intersection = FALSE)
offliers(db,lm.D9, CD = TRUE, DFB = TRUE, COVR = TRUE, pctg = 10, intersection = FALSE)
```

`rowmean` *Give Column Means of a Matrix-like Object, Based on a Grouping Variable*

Description

Compute column sums across rows of a numeric matrix-like object for each level of a grouping variable.

Usage

```
rowmean(M, group = rownames(M), reord = FALSE, na_rm = FALSE,
  big = TRUE, ...)
```

Arguments

`M`	a matrix, data frame or vector of numeric data. Missing values are allowed. A numeric vector will be treated as a column vector.
`group`	a vector or factor giving the grouping, with one element per row of M. Default: rownames of M.
`reord`	if TRUE, then the result will be in order of sort(unique(group)), if FALSE (the default), it will be in the order that groups were encountered.
`na_rm`	logical (TRUE or FALSE). Should NA (including NaN) values be discarded?
`big`	is your object big and integer overflow is likely? If TRUE, then M is multiplied by 1.0 to ensure values are of type double (perhaps taking more RAM).
`...`	other arguments to be passed to or from methods.

Details

This function is a wrapper for base function `rowsum` which allows one to compute the mean instead of the sum, while handling integer overflow.

Note: although data frames ara allowed, keep in mind that data frames do not allow duplicate row names.

To compute the mean over all the rows of a matrix (i.e. a single group) use colMeans, which should be even faster.

Value

A matrix-like object containing the means by group. There will be one row per unique value of group. If object supplied in fact (explicitly) had just one group, base function `colMeans` is called for maximum efficiency and a numeric vector containing the mean of each column is returned.

Author(s)

Albert Dorador

See Also

`rowsum` `colMeans`

Examples

```
A <- matrix(1:8, ncol = 2)
rownames(A) <- c("A", "B", "A", "B")
rowmean(A)

B <- matrix(1:40, ncol = 2)
gr <- rep(1:5, 4)
B.mean <- rowmean(B, group = gr)
sum(B.mean[, 1])*4 == sum(B[, 1]) #basic sanity check
sum(B.mean[, 2])*4 == sum(B[, 2]) #basic sanity check
```

```
dfB <- as.data.frame(B)
gr <- rep(1:5, 4)
dfB.mean <- rowmean(dfB, group = gr)

numbers <- rnorm(1e7, mean = 3)
C <- matrix(numbers, ncol = 5)
gr <- rep(1:20, 1e5)
rowmean(C, group = gr) # Handles Big Data fast

vec <- 1:10
gr <- rep(1:2, 5)
rowmean(vec, gr)

onegroup = matrix(1:40, ncol = 2)
gr = rep(1,20)
rowmean(onegroup, gr)[1] == mean(onegroup[,1])
rowmean(onegroup, gr)[2] == mean(onegroup[,2])
```

`tgsboot` *Bootstrap for Stationary Data*

Description

Generate bootstrap samples for stationary data, using a truncated geometric distribution to more accurately determine the value of the `p` parameter involved in the algorithm.

Usage

```
tgsboot(tseries, nb = 1, b.info = FALSE)
```

Arguments

`tseries` a numeric vector or time series giving the original data.

`nb` the number of bootstrap series to compute.

`b.info` if TRUE, the value of the `b` parameter found is returned as well. The default is FALSE.

Details

The value of the `b` parameter involved in the stationary bootstrap algorithm is determined using the heuristic laid out in Politis & White (2004). Then, the value of the `p` parameter is found by numerically solving a polynomial of order N+1 in variable `q`, where $q = 1 - p$ and N is the length of the data supplied. The previous polynomial is derived using the expectation of a truncated geometric distribution (for the stochastic block length), shown in Olatayo (2014).

The general structure of the algorithm is similar to the one laid out in James & Yang (2010).

Value

If b.info is FALSE, a matrix or time series with nb columns and length(tseries) rows containing the bootstrap data. Each column contains one bootstrap sample. If b.info is TRUE, a list with two fields: one containing the bootstrap data, and another containing the `b` value found.

Author(s)

Albert Dorador

References

Politis, D.N. and Romano, J.P. (1994), ’The stationary bootstrap’, Journal of the American Statistical Association 89(428), 1303-1313.

Politis, D.N. and White, H. (2004), ’Automatic block-length selection for the dependent bootstrap’, Econometric Reviews 23(1), 53-70.

Olatayo, T.O. (2014), ’Truncated geometric bootstrap method for timeseries stationary process’, Applied Mathematics 5, 2057-2061.

James, J. and Yang, L. (2010), ’Stop-losses, maximum drawdown-at-risk and replicating financial time series with the stationary bootstrap’, Quantitative Finance 10(1), 1-12.

Examples

```
set.seed(123)
x = rnorm(1e4)
boot = tgsboot(x)
boot = tgsboot(x, b.info = TRUE)
boot = tgsboot(x, nb = 2)
boot = tgsboot(x, nb = 2, b.info = TRUE)
```

`Wbiplot` *Weighted Biplot*

Description

`Wbiplot` produces a biplot with any weight distribution between Row and Column markers. This way the full spectrum from perfect row resolution (Row-metric preserving biplot) to perfect column resolution (Column-metric preserving biplot) is available.

Usage

```
Wbiplot(df, numer1, denom1 = 1, numer2, denom2 = 1, cx = 0.5)
```

Arguments

`df`	a dataframe with numeric values only
`numer1`	numerator of first exponent (can be a decimal)
`denom1`	denominator of first exponent (default: 1)
`numer2`	numerator of second exponent (can be a decimal)
`denom2`	denominator of second exponent (default: 1)
`cx`	graphical magnification factor (default: 0.5)

Details

This function makes use of function `Matpow` from package **powerplus** to be able to raise any valid matrix (see `Matpow` documentation) to any real power between 0 and 1 included.

Value

A biplot of a dataframe with the specified weights. Weights can either be supplied as two fractions, or as two decimal numbers.

Author(s)

Albert Dorador

See Also

`Matpow`

Examples

```
require(graphics)

# Exemple 1: Row metric preserving
Wbiplot(USArrests, numer1 = 1, numer2 = 0, cx = 0.6)

# Exemple 2: Column metric preserving
Wbiplot(USArrests, numer1 = 0, numer2 = 1, cx = 0.6)

# Comparison with function \code{biplot} from package \pkg{stats}
biplot(princomp(USArrests), cex = 0.6)

# Example 3: Custom, 50-50
Wbiplot(USArrests, numer1 = 0.5, numer2 = 0.5)

# Example 4: Custom, 20-80
Wbiplot(USArrests, numer1 = 0.2, numer2 = 0.8)
```

`weakly.stationary` *Testing for Weak Stationarity in a Time Series*

Description

Performs a series of statistical tests aimed at detecting non-stationarity.

Usage

```
weakly.stationary(tseries, signific_gen = 0.05, signific_pp.df = 0.05,
  MK = FALSE, BP = TRUE, PSR = TRUE, weak.dep = FALSE,
  mode = "neutral")
```

Arguments

`tseries` a 1-D or 2-D array. In the latter case, the time series to be evaluated must be placed in the 2nd dimension (columns). If that’s not your case, transpose it.

`signific_gen` significance level for all tests except Phillips-Perron and Augmented Dickey-Fuller.

`signific_pp.df` significance level for the Phillips-Perron and Augmented Dickey-Fuller tests.

`MK` if TRUE, the Mann-Kendall test for constant mean is executed, instead of a faster basic test. Default is FALSE.

`BP` if TRUE, the Breusch-Pagan test for constant (residual) variance is executed on the residuals of an auxiliary linear model that includes a time variable, instead of the McLeod-Li test. The default is TRUE.

`PSR` if TRUE, the Priestley-Subba Rao test for nonstationarity across time is executed. The default is TRUE.

`weak.dep` if TRUE, then the Phillips-Perron, Augmented Dickey-Fuller, and KPSS tests for weak stationarity (assuming an AR(p)) are performed.

`mode` one of "neutral", "strict", "loose". Case insenstive. The default is "neutral".

Details

This function offers a great deal of customization: diverse significance levels, multiple tests specialized in certain aspects of (weak) stationarity, as well as handy predefined sets of parameters providing a more or less strict diagnostic: "neutral", "strict" and "loose" modes. By including this possibility, the technical burden on the user is made lighter. Mode "strict" includes two tests for constant mean (basic & Mann-Kendall), two tests for constant variance (McLeod-Li & Breusch-Pagan tests), the Priestley-Subba Rao (PSR) test for nonstationarity across time, and three tests for weak dependence (Phillips-Perron, Augmented Dickey-Fuller, and KPSS tests), which test weak stationarity if and only if the underlying data generating process is assumed to be an AR(p). Mode "loose" just performs the basic test for constant mean (a linear model that includes a trend whose statistical significance is determined using robust regression if the Durbin Watson test detects serial correlation in the residuals), and the Breusch-Pagan test (on the previous auxiliary linear model’s residuals) for constant variance. Mode "neutral" (the default) provides all the default parameter options. Significance levels also differ across modes. This function differentiates two significance levels: general (`signific_gen`) and specific to the Phillips-Perron and Augmented Dickey-Fuller tests (`signific_pp.df`). In mode "strict", `signific_gen` is 0.1, and `signific_pp.df` is 0.01. In mode "loose", `signific_gen` is 0.01, and `signific_pp.df` is irrelevant. In mode "neutral", both significance levels are set to 0.05.

Value

if a 1-D array is supplied, then a Boolean is returned indicating whether the time series supplied is weakly stationary (TRUE) or not (FALSE). If a 2-D array is supplied, then a vector of Booleans is returned indicating whether each individual time series supplied is weakly stationary (TRUE) or not (FALSE).

Author(s)

Albert Dorador

Examples

```
x1 <- rnorm(1e2)
weakly.stationary(tseries = x1)
weakly.stationary(tseries = x1, signific_gen = 0.025)
weakly.stationary(tseries = x1, signific_pp.df = 0.1)
weakly.stationary(tseries = x1, PSR = FALSE)
weakly.stationary(tseries = x1, weak.dep = TRUE)
weakly.stationary(tseries = x1, mode = "strict")
weakly.stationary(tseries = x1, mode = "loose")

require(stats)
set.seed(123)
x2 <- arima.sim(n = 1e2, list(ar = 0.4))
weakly.stationary(tseries = x2)
weakly.stationary(tseries = x2, signific_gen = 0.01)
weakly.stationary(tseries = x2, PSR = FALSE)
weakly.stationary(tseries = x2, weak.dep = TRUE)
weakly.stationary(tseries = x2, mode = "strict")
weakly.stationary(tseries = x2, mode = "loose")
```

Index