Package ‘HelpersMG’

February 21, 2020

Type Package
Title Tools for Environmental Analyses, Ecotoxicology and Various R Functions
Version 4.0
Date 2020-02-21
Author Marc Girondot
Maintainer Marc Girondot <marc.girondot@u-psud.fr>
Depends lme4, coda, R (>= 2.10)
Suggests RNetCDF, ncdf4, maps, XML, fields, shiny, Matrix, ppcor, pbmcapply, pbapply, parallel, visNetwork, igraph
Description Contains many functions useful for managing ‘NetCDF’ files (see <http://en.wikipedia.org/wiki/NetCDF>), get tide levels on any point of the globe, get moon phase and time for sun rise and fall, analyse and reconstruct periodic time series of temperature with irregular sinusoidal pattern, show scales and wind rose in plot with change of color of text, Metropolis-Hastings algorithm for Bayesian MCMC analysis, plot graphs or boxplot with error bars, search files in disk by there names or their content, read the contents of all files from a folder at one time.
License GPL-2
LazyData yes
LazyLoad yes
Encoding UTF-8
RoxygenNote 7.0.2
NeedsCompilation no
Repository CRAN
Date/Publication 2020-02-21 14:00:08 UTC

R topics documented:

HelpersMG-package .......................................................... 3
### R topics documented:

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>as.mcmc.mcmcComposite</td>
<td>5</td>
</tr>
<tr>
<td>as.parameters</td>
<td>7</td>
</tr>
<tr>
<td>as.quantiles</td>
<td>9</td>
</tr>
<tr>
<td>asc</td>
<td>10</td>
</tr>
<tr>
<td>barplot_errbar</td>
<td>11</td>
</tr>
<tr>
<td>cArrows</td>
<td>12</td>
</tr>
<tr>
<td>ChangeCoordinate</td>
<td>14</td>
</tr>
<tr>
<td>chr</td>
<td>15</td>
</tr>
<tr>
<td>clean.knitr</td>
<td>16</td>
</tr>
<tr>
<td>compare</td>
<td>16</td>
</tr>
<tr>
<td>compare_AIC</td>
<td>17</td>
</tr>
<tr>
<td>compare_AICc</td>
<td>19</td>
</tr>
<tr>
<td>compare_BIC</td>
<td>20</td>
</tr>
<tr>
<td>contingencyTable.compare</td>
<td>21</td>
</tr>
<tr>
<td>convert.tz</td>
<td>24</td>
</tr>
<tr>
<td>d</td>
<td>25</td>
</tr>
<tr>
<td>D1x</td>
<td>26</td>
</tr>
<tr>
<td>dSnbinom</td>
<td>28</td>
</tr>
<tr>
<td>duplicate.packages</td>
<td>30</td>
</tr>
<tr>
<td>ellipse</td>
<td>31</td>
</tr>
<tr>
<td>ExtractAIC.glm</td>
<td>33</td>
</tr>
<tr>
<td>flexit</td>
<td>35</td>
</tr>
<tr>
<td>FormatCompareAIC</td>
<td>36</td>
</tr>
<tr>
<td>growlnotify</td>
<td>37</td>
</tr>
<tr>
<td>IC_clean_data</td>
<td>38</td>
</tr>
<tr>
<td>IC_correlation_simplify</td>
<td>40</td>
</tr>
<tr>
<td>IC_threshold_matrix</td>
<td>41</td>
</tr>
<tr>
<td>index.periodic</td>
<td>44</td>
</tr>
<tr>
<td>ind_long_lat</td>
<td>45</td>
</tr>
<tr>
<td>inside.search</td>
<td>47</td>
</tr>
<tr>
<td>invlogit</td>
<td>48</td>
</tr>
<tr>
<td>LD50</td>
<td>49</td>
</tr>
<tr>
<td>LD50_MHmcmc</td>
<td>51</td>
</tr>
<tr>
<td>LD50_MHmcmc_p</td>
<td>54</td>
</tr>
<tr>
<td>list.packages</td>
<td>55</td>
</tr>
<tr>
<td>local.search</td>
<td>56</td>
</tr>
<tr>
<td>logit</td>
<td>57</td>
</tr>
<tr>
<td>logLik.compareAIC</td>
<td>58</td>
</tr>
<tr>
<td>logLik.LD50</td>
<td>59</td>
</tr>
<tr>
<td>merge.mcmcComposite</td>
<td>60</td>
</tr>
<tr>
<td>MHHalgoGen</td>
<td>61</td>
</tr>
<tr>
<td>minmax.periodic</td>
<td>65</td>
</tr>
<tr>
<td>modeled.hist</td>
<td>67</td>
</tr>
<tr>
<td>modifyVector</td>
<td>68</td>
</tr>
<tr>
<td>moon.info</td>
<td>69</td>
</tr>
<tr>
<td>MovingWindow</td>
<td>70</td>
</tr>
<tr>
<td>newcompassRose</td>
<td>71</td>
</tr>
<tr>
<td>newdbeta</td>
<td>72</td>
</tr>
</tbody>
</table>
**HelpersMG-package**

Tools for Environmental Analyses, Ecotoxicology and Various R Functions

**Description**

Contains functions useful for managing
'NetCDF' files (see http://en.wikipedia.org/wiki/NetCDF),
get tide levels on any point of the globe,
get moon phase and time for sun rise and fall,
analyse and reconstruct daily time series of temperature
with irregular sinusoidal pattern,
show scales and wind rose in plot with change of color of text,
Metropolis-Hastings algorithm for Bayesian MCMC analysis,
plot graphs or boxplot with error bars,
search files in disk by there names or their content,
read the contents of all files from a folder at one time,
calculate IC50 for ecotoxicological studies,
calculate the probability mass function of the sum of negative binomial
distributions.
The latest version of this package can always be installed using:
install.packages("http://max2.ese.u-psud.fr/epc/conservation/CRAN/HelpersMG.tar.gz", repos=NULL,
type="source")

Details

Helpers functions for several packages

<table>
<thead>
<tr>
<th>Package</th>
<th>HelpersMG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Package</td>
</tr>
<tr>
<td>Version</td>
<td>4.0 build 403</td>
</tr>
<tr>
<td>Date</td>
<td>2020-02-21</td>
</tr>
<tr>
<td>License</td>
<td>GPL (&gt;= 2)</td>
</tr>
<tr>
<td>LazyLoad</td>
<td>yes</td>
</tr>
</tbody>
</table>

Author(s)

Marc Girondot <marc.girondot@u-psud.fr>

Examples

```r
## Not run:
library(HelpersMG)
print('-----------------------------')
print('Examples for mcmcComposite objects')
print('-----------------------------')
require(coda)
x <- rnorm(30, 10, 2)
dnormx <- function(x, par) return(-sum(dnorm(x, mean=par['mean'], sd=par['sd'], log=TRUE)))
parameters_mcmc <- data.frame(Density=c('dnorm', 'dlnorm'),
Prior1=c(10, 0.5), Prior2=c(2, 0.5), SDProp=c(0.35, 0.2),
Min=c(-3, 0), Max=c(100, 10), Init=c(10, 2), stringsAsFactors = FALSE,
row.names=c('mean', 'sd'))
mcmc_run <- MHalgoGen(n.iter=100000, parameters=parameters_mcmc, data=x,
likelihood=dnormx, n.chains=1, n.adapt=100, thin=1, trace=1)
plot(mcmc_run, xlim=c(0, 20))
plot(mcmc_run, xlim=c(0, 10), parameters="sd")
mcmcforcoda <- as.mcmc(mcmc_run)
# Optimal rejection rate should be 0.234
rejectionRate(mcmcforcoda)
heidel.diag(mcmcforcoda)
raftery.diag(mcmcforcoda)
autocorr.diag(mcmcforcoda)
acf(mcmcforcoda[[1]],"mean", lag.max=20, bty="n", las=1)
acf(mcmcforcoda[[1]],"sd", lag.max=20, bty="n", las=1)
batchSE(mcmcforcoda, batchSize=100)
# The batch standard error procedure is usually thought to
```
# be not as accurate as the time series methods used in summary
summary(mcmcforcoda)$statistics[,"Time-series SE"]
summary(mcmc_run)
as.parameters(mcmc_run)
lastp <- as.parameters(mcmc_run, index="last")
parameters_mcmc[,"Init"] <- lastp
# The n.adapt set to 1 is used to not record the first set of parameters
# then it is not duplicated (as it is also the last one for
# the object mcmc_run)
mcmc_run2 <- MHalgoGen(n.iter=10000, parameters=parameters_mcmc, data=x,
likelihood=dnormx, n.chains=1, n.adapt=1, thin=1, trace=1)
mcmc_run3 <- merge(mcmc_run, mcmc_run2)

####### no adaptation, n.adapt must be 0
parameters_mcmc[,"Init"] <- c(mean(x), sd(x))
mcmc_run3 <- MHalgoGen(n.iter=10000, parameters=parameters_mcmc, data=x,
likelihood=dnormx, n.chains=1, n.adapt=0, thin=1, trace=1)

# Generate a timeserie of time
time.obs <- NULL
for (i in 0:9) time.obs <- c(time.obs, c(0, 6, 12, 18)+i*24)
# For these time, generate a timeseries of temperatures
temp.obs <- rep(NA, length(time.obs))
temp.obs[3+(0:9)*4] <- rnorm(10, 25, 3)
temp.obs[1+(0:9)*4] <- rnorm(10, 10, 3)
for (i in 1:length(time.obs)-1)
  if (is.na(temp.obs[i]))
    temp.obs[i] <- mean(c(temp.obs[i-1], temp.obs[i+1]))
  if (is.na(temp.obs[length(time.obs)]))
    temp.obs[length(time.obs)] <- temp.obs[length(time.obs)-1]/2
observed <- data.frame(time=time.obs, temperature=temp.obs)

# Search for the minimum and maximum values
r <- minmax.periodic(time.minmax.daily=c(Min=2, Max=15),
observed=observed, period=24)

# Estimate all the temperatures for these values
t <- temperature.periodic(minmax=r)

plot_errbar(x=t[,"time"], y=t[,"temperature"],
errbar.y=ifelse(is.na(t[,"sd"]), 0, 2*t[,"sd"]),
type="l", las=1, bty="n", errbar.y.polygon = TRUE,
xlab="hours", ylab="Temperatures", ylim=c(0, 35),
errbar.y.polygon.list = list(col="grey"))

plot_add(x=t[,"time"], y=t[,"temperature"], type="l")

## End(Not run)
Description

Take a mcmcComposite object and create a mcmc.list object to be used with coda package.

Usage

## S3 method for class 'mcmcComposite'
as.mcmc(x, ...)

Arguments

x             A mcmcComposite obtained as a result of MHalgoGen() function
...            Not used

Details

as.mcmc Extract mcmc object from the result of phenology_MHmcmc to be used with coda package

Value

A mcmc.list object

Author(s)

Marc Girondot

See Also

Other mcmcComposite functions: MHalgoGen(), as.parameters(), as.quantiles(), merge.mcmcComposite(), plot.mcmcComposite(), summary.mcmcComposite()

Examples

## Not run:
library(HelpersMG)
require(coda)
x <- rnorm(30, 10, 2)
dnormx <- function(data, x) {
  data <- unlist(data)
  return(-sum(dnorm(data, mean=x["mean"], sd=x["sd"], log=TRUE)))
}
parameters_mcmc <- data.frame(Density=c("dnorm", "dlnorm"),
Prior1=c(10, 0.5), Prior2=c(2, 0.5), SDProp=c(1, 1),
Min=c(-3, 0), Max=c(100, 10), Init=c(10, 2), stringsAsFactors = FALSE,
row.names=c("mean", "sd"))
mcmc_run <- MHalgoGen(n.iter=1000, parameters=parameters_mcmc, data=x,
likelihood=dnormx, n.chains=1, n.adapt=100, thin=1, trace=1)
plot(mcmc_run, xlim=c(0, 20))
plot(mcmc_run, xlim=c(0, 10), parameters="sd")
mcmcforcoda <- as.mcmc(mcmc_run)
#' heidel.diag(mcmcforcoda)
raftery.diag(mcmcforcoda)
as.parameters

Extract parameters from mcmcComposite object

Description

Take a mcmcComposite object and create a vector object with parameter value at specified iteration. If index="best", the function will return the parameters for the highest likelihood. It also indicates at which iteration the maximum likelihood has been observed. If index="last", the function will return the parameters for the last likelihood. index can also be a numeric value.

Usage

as.parameters(x, index = "best", chain = 1)

Arguments

x A mcmcComposite obtained as a result of MHalgoGen() function
index At which iteration the parameters must be taken
chain The number of the chain in which to get parameters

Value

A vector with parameters at maximum likelihood or index position
Author(s)

Marc Girondot

See Also

Other mcmcComposite functions: MHalgoGen(), as.mcmc.mcmcComposite(), as.quantiles(), merge.mcmcComposite(), plot.mcmcComposite(), summary.mcmcComposite()

Examples

## Not run:
library(HelpersMG)
require(coda)

x <- rnorm(30, 10, 2)
dnormx <- function(data, x) {
  data <- unlist(data)
  return(-sum(dnorm(data, mean=x['mean'], sd=x['sd'], log=TRUE)))
}

parameters_mcmc <- data.frame(Density=c("dnorm", "dlnorm"),
Prior1=c(10, 0.5), Prior2=c(2, 0.5), SDProp=c(1, 1),
Min=c(-3, 0), Max=c(100, 10), Init=c(10, 2), stringsAsFactors = FALSE,
row.names=c('mean', 'sd'))
mcmc_run <- MHalgoGen(n.iter=1000, parameters=parameters_mcmc, data=x,
likelihood=dnormx, n.chains=1, n.adapt=100, thin=1, trace=1)
plot(mcmc_run, xlim=c(0, 20))
plot(mcmc_run, xlim=c(0, 10), parameters="sd")
mcmcforcoda <- as.mcmc(mcmc_run)
  heidel.diag(mcmcforcoda)
  raftery.diag(mcmcforcoda)
  autocorr.diag(mcmcforcoda)

acf(mcmcforcoda[[1]], "mean", lag.max=20, bty="n", las=1)
acf(mcmcforcoda[[1]], "sd", lag.max=20, bty="n", las=1)
batchSE(mcmcforcoda, batchSize=100)

# The batch standard error procedure is usually thought to
# be not as accurate as the time series methods used in summary
summary(mcmcforcoda)$statistics[, "Time-series SE"]

summary(mcmc_run)
as.parameters(mcmc_run)
lastp <- as.parameters(mcmc_run, index="last")
parameters_mcmc[, "Init"] <- lastp

# The n.adapt set to 1 is used to not record the first set of parameters
# then it is not duplicated (as it is also the last one for
# the object mcmc_run)
mcmc_run2 <- MHalgoGen(n.iter=1000, parameters=parameters_mcmc, data=x,
likelihood=dnormx, n.chains=1, n.adapt=1, thin=1, trace=1)
mcmc_run3 <- merge(mcmc_run, mcmc_run2)

##### no adaptation, n.adapt must be 0
parameters_mcmc[, "Init"] <- c(mean(x), sd(x))
mcmc_run3 <- MHalgoGen(n.iter=1000, parameters=parameters_mcmc, data=x,
likelihood=dnormx, n.chains=1, n.adapt=0, thin=1, trace=1)

## End(Not run)
as.quantiles

Extract quantile distribution from mcmcComposite object

Description

Extract quantile distribution from mcmcComposite object

Usage

as.quantiles(
  x,
  chain = 1,
  fun = function(...) return(as.numeric(list(...))),
  probs = c(0.025, 0.975),
  xlim = NULL,
  nameparxlim = NULL,
  namepar = NULL
)

Arguments

x A mcmcComposite obtained as a result of MHalgoGen() function
chain The number of the chain in which to get parameters
fun The function to apply the parameters
probs The probability to get quantiles
xlim The values to apply in fun
nameparxlim The name of the parameter for xlim
namepar The name of parameters from mcmc object to be used in fun

Value

A data.frame with quantiles

Author(s)

Marc Girondot

See Also

Other mcmcComposite functions: MHalgoGen(), as.mcmc.mcmcComposite(), as.parameters(), merge.mcmcComposite(), plot.mcmcComposite(), summary.mcmcComposite()
Examples

```r
## Not run:
library(HelpersMG)
require(coda)
x <- rnorm(30, 10, 2)
dnormx <- function(data, x) {
  data <- unlist(data)
  return(-sum(dnorm(data, mean=x['mean'], sd=x['sd'], log=TRUE)))
}
parameters_mcmc <- data.frame(Density=c('dnorm', 'dlnorm'),
                               Prior1=c(10, 0.5), Prior2=c(2, 0.5), SDProp=c(1, 1),
                               Min=-3, Max=100, Init=c(10, 2), stringsAsFactors = FALSE,
                               row.names=c('mean', 'sd'))
mcmc_run <- MHalgoGen(n.iter=10000, parameters=parameters_mcmc, data=x,
                      likelihood=dnormx, n.chains=1, n.adapt=100, thin=1, trace=1)
k <- as.quantiles(x=mcmc_run, namepar="mean")
k <- as.quantiles(x=mcmc_run, namepar="mean",
                     xlim=c(1:5), nameparxlim="sd",
                     fun=function(...) return(sum(as.numeric(list(...)))))
## End(Not run)
```

---

**asc**

*Return the codes (in UTF-8) of a string*

**Description**

Return the codes (in UTF-8) of a string.

**Usage**

```r
asc(x)
```

**Arguments**

- `x` The string to be analyzed

**Details**

asc returns the codes (in UTF-8) of a string

**Value**

A vector with ITF-8 codes of a string

**Author(s)**

Based on this blog: http://datadebrief.blogspot.com/2011/03/ascii-code-table-in-r.html
See Also

Other Characters: \texttt{chr()}, \texttt{d()}, \texttt{tnirp}()

Examples

\begin{verbatim}
asc("abcd")
asc("ABCD")
\end{verbatim}

\begin{verbatim}
barplot_errbar
    \textit{Plot a barplot graph with error bar on y}
\end{verbatim}

Description

To plot data, just use it as a normal barplot but add the \texttt{errbar.y} values or \texttt{errbar.y.minus}, \texttt{errbar.y.plus} if bars for y axis are asymmetric. Use \texttt{y.plus} and \texttt{y.minus} to set absolut limits for error bars. Note that \texttt{y.plus} and \texttt{y.minus} have priority over \texttt{errbar.y}, \texttt{errbar.y.minus} and \texttt{errbar.y.plus}.

Usage

\begin{verbatim}
barplot_errbar(
    ..., 
    errbar.y = NULL, 
    errbar.y.plus = NULL, 
    errbar.y.minus = NULL, 
    y.plus = NULL, 
    y.minus = NULL, 
    errbar.tick = 1/50, 
    errbar.lwd = par("lwd"), 
    errbar.lty = par("lty"), 
    errbar.col = par("fg"), 
    add = FALSE
)
\end{verbatim}

Arguments

\begin{verbatim}
... Parameters for barplot() such as main= or ylim=
errbar.y The length of error bars for y. Recycled if necessary.
errbar.y.plus The length of positive error bars for y. Recycled if necessary.
errbar.y.minus The length of negative error bars for y. Recycled if necessary.
y.plus The absolut position of the positive error bar for y. Recycled if necessary.
y.minus The absolut position of the negative error bar for y. Recycled if necessary.
errbar.tick Size of small ticks at the end of error bars defined as a proportion of total width or height graph size.
errbar.lwd Error bar line width, see par("lwd")
errbar.lty Error bar line type, see par("lty")
errbar.col Error bar line color, see par("col")
add If true, add the graph to the previous one.
\end{verbatim}
Details
barplot_errbar plot a barplot with error bar on y

Value
A numeric vector (or matrix, when beside = TRUE), say mp, giving the coordinates of all the bar midpoints drawn, useful for adding to the graph. If beside is true, use colMeans(mp) for the midpoints of each group of bars, see example.

Author(s)
Marc Girondot

See Also
plot_errorbar
Other plot and barplot functions: ScalePreviousPlot(), plot_add(), plot_errbar()

Examples
## Not run:
barplot_errbar(rnorm(10, 10, 3),
xlab="axe x", ylab="axe y", bty="n",
errbar.y.plus=rnorm(10, 1, 0.1), col=rainbow(10),
names.arg=paste("Group",1:10), cex.names=0.6)
y <- rnorm(10, 10, 3)
barplot_errbar(y,
   xlab="axe x", ylab="axe y", bty="n",
y.plus=y+2)
## End(Not run)

cArrows

\textit{Draw curved lines with arrowhead}

Description
Draw a curved line with arrowhead.

Usage
cArrows(
x1,  
y1,  
x2,  
y2,  
code = 2,  
size = 1,
\begin{verbatim}
width = 1.2/4/cin,
open = TRUE,
sh.adj = 0.1,
sh.lwd = 1,
sh.col = if (is.R()) par("fg") else 1,
sh.lty = 1,
h.col = sh.col,
h.col.bo = sh.col,
h.lwd = sh.lwd,
h.lty = sh.lty,
curved = FALSE,
beautiful.arrow = 2/3
)
\end{verbatim}

Arguments

- **x1** coordinates of points from which to draw.
- **y1** coordinates of points from which to draw.
- **x2** coordinates of points to which to draw.
- **y2** coordinates of points to which to draw.
- **code** integer code (1, 2, or 3), determining kind of arrows to be drawn.
- **size** size of the arrowhead.
- **width** width of the arrowhead.
- **open** shape of the arrowhead.
- **sh.adj** Shift the beginning of the line.
- **sh.lwd** width of the line.
- **sh.col** color of the line.
- **sh.lty** type of line.
- **h.col** color of the arrowhead.
- **h.col.bo** color of the arrowhead border.
- **h.lwd** width of the arrowhead.
- **h.lty** type of line for the arrowhead.
- **curved** 0 is a straigth line, positive of negative value make the line curved.
- **beautiful.arrow** if open is false, make the arrowhead more beautiful.

Details

cArrows draws curved lines with arrowhead

Value

A list wit lab.x and lab.y being the position where to draw label
Author(s)
Modified from iGraph

Examples

plot(c(1, 10), c(1, 10), type="n", bty="n")
cArrows(x1=2, y1=2, x2=6, y2=6, curved=1)
cArrows(x1=2, y1=2, x2=6, y2=6, curved=0)
cArrows(x1=2, y1=2, x2=6, y2=6, curved=1, sh.adj=1)
cArrows(x1=2, y1=2, x2=6, y2=6, curved=-1, open=FALSE)
cArrows(x1=9, y1=2, x2=6, y2=6, curved=-1, open=FALSE, sh.col="red")
cArrows(x1=9, y1=9, x2=6, y2=6, curved=-1, open=FALSE, h.col="red")
cArrows(x1=2, y1=9, x2=6, y2=6, curved=1, open=FALSE, h.col="red", h.col.bo="red")

ChangeCoordinate

Return a value in a changed coordinate

Description
Return a value in a changed coordinate system.

Usage
ChangeCoordinate(
  x = stop("At least one value to convert must be provided"),
  initial = stop("Set of two values must be provided as references"),
  transformed = stop("Set of two transformed values must be provided")
)

Arguments

x value to convert
initial Set of two values in the original system
transformed Set of the two values in the converted system

Details
ChangeCoordinate returns a value in a changed coordinate

Value
A value in the new system

Author(s)
Marc Girondot
**Examples**

```r
ChangeCoordinate(x=c(10, 20), initial=c(1, 100), transformed=c(0, 1))
```

---

**chr**

*Return the characters defined by the codes*

---

**Description**

Return a string with characters defined by the codes.

**Usage**

```r
chr(n)
```

**Arguments**

- `n` The code to be used to return a character

**Details**

`chr` returns the characters defined by the codes

**Value**

A string with characters defined by the codes

**Author(s)**

Based on this blog: http://datadebrief.blogspot.com/2011/03/ascii-code-table-in-r.html

**See Also**

Other Characters: `asc()`, `d()`, `tnirp()`

**Examples**

```r
chr(65:75)
chr(unlist(tapply(144:175, 144:175, function(x) {c(208, x)})))
```
clean.knitr  

Delete temporary files created during knitr compile

Description
Delete temporary files created during knitr compile in working directory.
This function works only in UNIX system (LINUX or MacOSX).

Usage

  clean.knitr()

Details

  clean.knitr deletes temporary files created during knitr compile

Value

  Nothing

Author(s)

  Marc Girondot

Examples

  ## Not run:
  clean.knitr()

  ## End(Not run)

compare  

Run a shiny application for basic functions of comparison

Description
Run a shiny application for basic functions of comparison.

Usage

  compare()

Details

  compare runs a shiny application for basic functions of comparison
**compare_AIC**

**Value**

Nothing

**Author(s)**

Marc Girondot

**References**


**See Also**

Other w-value functions: `contingencyTable.compare()`, `series.compare()`

**Examples**

```r
## Not run:
library(HelpersMG)
compare()
## End(Not run)
```

**Description**

This function is used to compare the AIC of several outputs obtained with the same data but with different set of parameters.

The parameters must be lists with $aic$ or $AIC$ or $value$ and $par$ elements or if AIC(element) is defined.

If $value$ and $par$ are present in the object, the AIC is calculated as $2 \times \text{factor.value} \times \text{value} + 2 \times \text{length(par)}$.

If $value$ is $-\log(\text{likelihood})$, then factor.value must be 1 and if $value$ is $\log(\text{likelihood})$, then factor.value must be -1.

If several objects are within the same list, their AIC are summed.

For example, `compare_AIC(g1=list(group), g2=list(separe1, separe2))` can be used to compare a single model onto two different sets of data against each set of data fitted with its own set of parameters.

Take a look at `ICtab` in package `bbmle` which is similar.

**Usage**

```r
compare_AIC(..., factor.value = 1, silent = FALSE)
```
Arguments

... Successive results to be compared as lists.

factor.value The value of the list object is multiplied by factor.value to calculate AIC.

silent If TRUE, nothing is displayed.

Details

compare_AIC compares the AIC of several outputs obtained with the same data.

Value

A list with DeltaAIC and Akaike weight for the models.

Author(s)

Marc Girondot

See Also

Other AIC: ExtractAIC.glm(), FormatCompareAIC(), compare_AICc(), compare_BIC()

Examples

```r
## Not run:
library("HelpersMG")
# Here two different models are fitted
x <- 1:30
y <- rnorm(30, 10, 2)+log(x)
plot(x, y)
d <- data.frame(x=x, y=y)
m1 <- lm(y ~ x, data=d)
m2 <- lm(y ~ log(x), data=d)
compare_AIC(linear=m1, log=m2)
# Here test if two datasets can be modeled with a single model
x2 <- 1:30
y2 <- rnorm(30, 15, 2)+log(x2)
plot(x, y, ylim=c(5, 25))
plot_add(x2, y2, col="red")
d2 <- data.frame(x=x2, y=y2)
m1_2 <- lm(y ~ x, data=d2)
x_grouped <- c(x, x2)
y_grouped <- c(y, y2)
d_grouped <- data.frame(x=x_grouped, y=y_grouped)
m1_grouped <- lm(y ~ x, data=d_grouped)
compare_AIC(separate=list(m1, m1_2), grouped=m1_grouped)
## End(Not run)
```
**compare_AICc**

Compares the AICc of several outputs

---

**Description**

This function is used to compare the AICc of several outputs obtained with the same data but with different set of parameters.

Each object must have associated `logLik()` method with `df` and `nobs` attributes.

AICc for object x will be calculated as $2 \times \text{factor.value} \times \logLik(x) + (2 \times \text{attributes(logLik(x))\$df} \times (\text{attributes(logLik(x))\$df} + 1)/(\text{attributes(logLik(x))\$nobs} - \text{attributes(logLik(x))\$df} - 1)$.

**Usage**

`compare_AICc(..., factor.value = -1, silent = FALSE)`

**Arguments**

- `...` Successive results to be compared as lists.
- `factor.value` The $value of the list object is multiplied by `factor.value` to calculate BIC.
- `silent` If TRUE, nothing is displayed.

**Details**

`compare_AICc` compares the AICc of several outputs obtained with the same data.

**Value**

A list with DeltaAICc and Akaike weight for the models.

**Author(s)**

Marc Girondot

**See Also**

Other AIC: `ExtractAIC.glm()`, `FormatCompareAIC()`, `compare_AIC()`, `compare_BIC()`

**Examples**

```r
## Not run:
library("HelpersMG")
# Here two different models are fitted
x <- 1:30
y <- rnorm(30, 10, 2)+log(x)
plot(x, y)
d <- data.frame(x=x, y=y)
m1 <- lm(y ~ x, data=d)
m2 <- lm(y ~ log(x), data=d)
```
compare_BIC(linear=m1, log=m2, factor.value=-1)

# Here test if two datasets can be modeled with a single model
x2 <- 1:30
y2 <- rnorm(30, 15, 2) + log(x2)
plot(x, y, ylim=c(5, 25))
plot_add(x2, y2, col="red")
d2 <- data.frame(x=x2, y=y2)
m1_2 <- lm(y ~ x, data=d2)
x_grouped <- c(x, x2)
y_grouped <- c(y, y2)
d_grouped <- data.frame(x=x_grouped, y=y_grouped)
m1_grouped <- lm(y ~ x, data=d_grouped)
compare_AICc(separate=list(m1, m1_2), grouped=m1_grouped, factor.value=-1)

# Or simply
compare_AICc(m1=list(AICc=100), m2=list(AICc=102))

## End(Not run)

---

**compare_BIC**  
*Compares the BIC of several outputs*

**Description**

This function is used to compare the BIC of several outputs obtained with the same data but with different set of parameters. Each object must have associated `logLik()` method with `df` and `nobs` attributes. BIC for object `x` will be calculated as:

\[
2 \times \text{factor.value} \times \sum \log \text{Lik}(x) + \sum \text{attributes}(\log \text{Lik}(x)) \times df \times \log(\text{attributes}(\log \text{Lik}(x)) \text{nobs})
\]

When several data (i..n) are included, the global BIC is calculated as:

\[
2 \times \text{factor.value} \times \sum \log \text{Lik}(x) \text{ for i..n} + \sum \text{attributes}(\log \text{Lik}(x)) \text{df} \text{ for i..n} \times \log(\text{attributes}(\log \text{Lik}(x)) \text{nobs}) \text{ for i..n})
\]

**Usage**

```r
compare_BIC(..., factor.value = -1, silent = FALSE)
```

**Arguments**

- `...` Successive results to be compared as lists.
- `factor.value` The $value of the list object is multiplied by factor.value to calculate BIC.
- `silent` If TRUE, nothing is displayed.

**Details**

`compare_BIC` compares the BIC of several outputs obtained with the same data.

**Value**

A list with DeltaBIC and Akaike weight for the models.
Author(s)
Marc Girondot

See Also
Other AIC: ExtractAIC.glm(), FormatCompareAIC().compare_AICc(), compare_AIC()

Examples

## Not run:
library("HelpersMG")
# Here two different models are fitted
x <- 1:30
y <- rnorm(30, 10, 2)+log(x)
plot(x, y)
d <- data.frame(x=x, y=y)
m1 <- lm(y ~ x, data=d)
m2 <- lm(y ~ log(x), data=d)
compare_BIC(linear=m1, log=m2, factor.value=-1)
# Here test if two datasets can be modeled with a single model
x2 <- 1:30
y2 <- rnorm(30, 15, 2)+log(x2)
plot(x, y, ylim=c(5, 25))
plot_add(x2, y2, col="red")
d2 <- data.frame(x=x2, y=y2)
m1_2 <- lm(y ~ x, data=d2)
x_grouped <- c(x, x2)
y_grouped <- c(y, y2)
d_grouped <- data.frame(x=x_grouped, y=y_grouped)
m1_grouped <- lm(y ~ x, data=d_grouped)
compare_BIC(separate=list(m1, m1_2), grouped=m1_grouped, factor.value=-1)
## End(Not run)
Arguments

- **table**: A matrix or a data.frame with series in rows and number of each category in column
- **criterion**: Which criterion is used for model selection
- **probs**: Series of probabilities used for conformity comparison

Details

contingencyTable.compare compares contingency table using Akaike weight.

Value

The probability that a single proportion model is sufficient to explain the data

Author(s)

Marc Girondot

References


See Also

Other w-value functions: `compare()`, `series.compare()`

Examples

```r
## Not run:
library("HelpersMG")

# Symmetry of Lepidochelys olivacea scutes
table <- t(data.frame(SriLanka=c(200, 157), AfricaAtl=c(19, 12),
            Guyana=c(8, 6), Suriname=c(162, 88),
            MexicoPac1984=c(42, 34), MexicoPac2014Dead=c(8, 9),
            MexicoPac2014Alive=c(13, 12),
            row.names =c("Symmetric", "Asymmetric")))
table
table
contingencyTable.compare(table)

table <- t(data.frame(SriLanka=c(200, 157), AfricaAtl=c(19, 12), Guyana=c(8, 6),
            Suriname=c(162, 88), MexicoPac1984=c(42, 34),
            MexicoPac2014Dead=c(8, 9),
            MexicoPac2014Alive=c(13, 12), Lepidochelys.kempii=c(99, 1),
            row.names =c("Symmetric", "Asymmetric")))
table
table
contingencyTable.compare(table)

# Conformity to a model
```
table <- matrix(c(33, 12, 25, 75), ncol = 2, byrow = TRUE)
probs <- c(0.5, 0.5)
contingencyTable.compare(table, probs=probs)

# Conformity to a model
table <- matrix(c(33, 12), ncol = 2, byrow = TRUE)
probs <- c(0.5, 0.5)
contingencyTable.compare(table, probs=probs)

# Conformity to a model
table <- matrix(c(33, 12, 8, 25, 75, 9), ncol = 3, byrow = TRUE)
probs <- c(0.8, 0.1, 0.1)
contingencyTable.compare(table, probs=probs)

# Comparison of chisq.test() and this function
table <- matrix(c(NA, NA, 25, 75), ncol = 2, byrow = TRUE)

pv <- NULL
aw <- NULL
par(new=FALSE)
n <- 100

for (GroupA in 0:n) {
  table[1, 1] <- GroupA
  table[1, 2] <- n-GroupA
  pv <- c(pv, chisq.test(table)$p.value)
  aw <- c(aw, contingencyTable.compare(table, criterion="BIC")[1])
}

x <- 0:n
y <- pv
y2 <- aw
plot(x=x, y=y, type="l", bty="n", las=1, xlab="Number of type P in Group B", ylab="Probability", main="", lwd=2)
lines(x=x, y=y2, type="l", col="red", lwd=2)

# w-value
(l1 <- x[which(aw>0.05)[1]])
(l2 <- rev(x)[which(rev(aw)>0.05)[1]])

aw[l1]
pv[l1]

aw[l2+2]
pv[l2+2]

# p-value
l1 <- which(pv>0.05)[1]
l2 <- max(which(pv>0.05))

aw[l1]
pv[l1]
convert.tz

Convert one Date-Time from one timezone to another

Description

Convert one Date-Time from one timezone to another. Available timezones can be shown using OlsonNames().

Usage

convert.tz(x, tz = Sys.timezone())

Arguments

x The date-time in POSIXlt or POSIXct format
tz The timezone

Details

convert.tz Convert one Date-Time from one timezone to another

Value

A POSIXlt or POSIXct date converted
**Author(s)**

Marc Girondot

**See Also**

Function `with_tz()` from lubridate package does the same. I keep it here only for compatibility with old scripts.

**Examples**

```r
> d <- as.POSIXlt("2010-01-01 17:34:20", tz="UTC")
> convert.tz(d, tz="America/Guatemala")
```

---

**d** — *Write an ASCII Representation of a vector object*

**Description**

Writes an ASCII text representation of an R object.

It can be used as a replacement of `dput()` for named vectors.

The controls "keepNA", "keepInteger" and "showAttributes" are utilized for named vectors.

**Usage**

```r
d(x, file = "", control = c("keepNA", "keepInteger", "showAttributes"), collapse = ", \n ")
```

**Arguments**

- **x** — A named vector object
- **file** — either a character string naming a file or a connection. "" indicates output to the console.
- **control** — character vector indicating deparsing options. See `.deparseOpts` for their description.
- **collapse** — Characters used to separate values.

**Details**

---

**Value**

A string
Author(s)
Marc Girondot

See Also
Other Characters: \texttt{asc()}, \texttt{chr()}, \texttt{tnirp()}

Examples

\begin{verbatim}
d(c(A=10, B=20))
dput(c(A=10, B=20))
\end{verbatim}

---

**DIx**

\textit{Return an index of quantitative asymmetry and complexity named Developmental Instability Index (DIx)}

**Description**

Return an index of quantitative asymmetry and complexity.

Higher is the value, higher is the complexity (number of objects) and diversity (difference between them).

The indice is based on the product of the average angular distance of Edwards (1971) for all permutations of measures for both sides with the geometric mean of the inverse of Shannon entropy \( H \) for both sides. Let \( p_1 \) and \( p_2 \) two vectors of relative measures of objects with \( \text{sum}(p_1) = 1 \) and \( \text{sum}(p_2) = 1 \) and \( n_1 \) being the number of objects in \( p_1 \) and \( n_2 \) being the number of objects in \( p_2 \). Edwards distance for all permutations of \( p_1 \) and \( p_2 \) objects are computed and the average value \( E \) is calculated.

The maximum possible Shannon index for identical \( n_1 \) is \( \text{max} = \text{sum}(1/n_1) \cdot \text{log}(1/n_1) \).

The Shannon index is \( v_1 = \text{sum}(p_1 \cdot \text{log}(p_1)) \).

If version == 2, the complementary of Shannon index for these \( n_1 \) objects is used: \( c_1 = 2 \cdot \text{max} - v_1 \)

If version == 1, the Shannon index is used directly.

The geometry mean between both sides defined the measure of diversity within each side: \( S = \sqrt{c_1 \cdot c_2} \)

The Developmental Instability Index is then \( S \cdot E \)

**Usage**

\begin{verbatim}
DIx(l1, l2, details = FALSE, version = 1)
\end{verbatim}

**Arguments**

- **l1**: Set of measures at one side of an organism
- **l2**: Set of measures at the other side of an organism
- **details**: If TRUE, will show the details of computing
- **version**: Can be 1 or 2; see description
DIx

Details

DIx returns an index of quantitative asymmetry and complexity

Value

A numeric value

Author(s)

Marc Girondot

References


Examples

```r
## Not run:
l1 <- c(0.1, 0.1, 0.05, 0.2, 0.3, 0.25)
l2 <- c(0.2, 0.3, 0.5)
DIx(l1, l2)

l1 <- c(0.1, 0.1, 0.05, 0.2, 0.3, 0.25)
l2 <- c(0.1, 0.1, 0.05, 0.2, 0.3, 0.25)
DIx(l1, l2)

l1 <- c(0.2, 0.3, 0.5)
l2 <- c(0.2, 0.3, 0.5)
DIx(l1, l2)

l1 <- c(0.2, 0.2, 0.2, 0.2, 0.2)
l2 <- c(0.3333, 0.3333, 0.3333)
DIx(l1, l2)

l1 <- c(0.2, 0.2, 0.2, 0.2, 0.2)
l2 <- c(0.2, 0.2, 0.2, 0.2)
DIx(l1, l2)

l1 <- c(0.3333, 0.3333, 0.3333)
l2 <- c(0.3333, 0.3333, 0.3333)
DIx(l1, l2)

## End(Not run)
```
Density for the sum of random variable with negative binomial distributions.

```r
dSnbinom(
x = stop("You must provide a x value"),
size = NULL,
prob = NULL,
mu = NULL,
log = FALSE,
tol = 1e-06,
infinite = 1000
)
```

Arguments

- `x`: vector of (non-negative integer) quantiles.
- `size`: target for number of successful trials, or dispersion parameter (the shape parameter of the gamma mixing distribution). Must be strictly positive, need not be integer.
- `prob`: probability of success in each trial. $0 < \text{prob} \leq 1$.
- `mu`: alternative parametrization via mean.
- `log`: logical; if TRUE, probabilities p are given as log(p).
- `tol`: Tolerance for recurrence
- `infinite`: Maximum level of recursion

Details

dSnbinom returns the density for the sum of random variable with negative binomial distributions

Value

dSnbinom gives the density

Author(s)

Marc Girondot
dSnbinom

References

See Also
Other Distribution of sum of random variable with negative binomial distributions: pSnbinom(), qSnbinom(), rSnbinom()

Examples

```
## Not run:
library(HelpersMG)
alpha <- c(1, 2, 5, 1, 2)
p <- c(0.1, 0.12, 0.13, 0.14, 0.14)
dSnbinom(20, size=alpha, prob=p)
dSnbinom(20, size=alpha, prob=p, log=TRUE)
dSnbinom(20, size=2, mu=c(0.01, 0.02, 0.03))
dSnbinom(20, size=2, mu=c(0.01, 0.02, 0.03), log=TRUE)
# Test with a single distribution
dSnbinom(20, size=1, mu=20)
# when only one distribution is available, it is the same as dnbinom()
dnbinom(20, size=1, mu=20)
# If a parameter is supplied as only one value, it is supposed to be constant
dSnbinom(20, size=1, mu=c(14, 15, 10))
# The function is vectorized:
plot(0:200, dSnbinom(0:200, size=alpha, prob=p), bty="n", type="h", xlab="x", ylab="Density")
# Comparison with simulated distribution using rep replicates
alpha <- c(2.1, 2.05, 2)
mu <- c(10, 30, 20)
rep <- 100000
distEmpirique <- rSnbinom(rep, size=alpha, mu=mu)
tabledistEmpirique <- rep(0, 301)
names(tabledistEmpirique) <- as.character(0:300)
tabledistEmpirique[names(table(distEmpirique))] <- table(distEmpirique)/rep
plot(0:300, dSnbinom(0:300, size=alpha, mu=mu), type="h", bty="n",
    xlab="x", ylab="Density", ylim=c(0,0.02))
plot_add(0:300, tabledistEmpirique, type="l", col="red")
legend(x=200, y=0.02, legend=c("Empirical", "Theoretical"),
       text.col=c("red", "black"), bty="n")
# Example with the approximation mu=mean(mu)
plot(0:300, dSnbinom(0:300, size=alpha, mu=mu), type="h", bty="n",
    xlab="x", ylab="Density", ylim=c(0,0.02))
plot_add(0:300, tabledistEmpirique, type="l", col="red")
legend(x=200, y=0.02, legend=c("Empirical", "Theoretical"),
       text.col=c("red", "black"), bty="n")

# Example to fit the distribution
data <- rnbinom(1000, size=1, mu=10)
hist(data)
```
ag <- rep(1:100, 10)
r <- aggregate(data, by=list(ag), FUN=sum)
hist(r[,2])
parx <- c(size=1, mu=10)

dSnbinomx <- function(x, par) {
  -sum(dSnbinom(x=x[,2], mu=rep(par["mu"], 10), size=par["size"], log=TRUE))
}

fit_mu_size <- optim(par = parx, fn=dSnbinomx, x=r, method="BFGS", control=c(trace=TRUE))
fit_mu_size$par

## End(Not run)

duplicate.packages  
List the duplicated packages with their locations

Description

A data.frame with the duplicated packages and their locations and version.
The columns Lib1 and Version1 should have the oldest version of the packages. Then you can try:
li <- duplicate.packages() if (nrow(li) != 0) for (i in 1:nrow(li)) remove.packages(rownames(li)[i],
lib=li[i, "Lib1"])

Usage

duplicate.packages()

Details

duplicate.packages lists the duplicated packages with their locations

Value

A data.frame with 4 elements for each duplicated packages:
- versions: the version of the packages
- libraries: the locations

Author(s)

Marc Girondot
Examples

## Not run:
library(HelpersMG)
duplicate.packages()

## End(Not run)

---

**ellipse**

*Plot an ellipse*

Description

Plot a ellipse dined by the center and the radius. The options for binomial confidence are:
- alpha is 1 - confidence interval
- method must be one of these "wilson", "exact", "asymptotic"
col parameter can be a list of colors. See examples

Usage

```r
ellipse(
  center.x = 0,
  center.y = 0,
  radius.x = 1,
  radius.y = 1,
  radius.x.lower = NULL,
  radius.x.upper = NULL,
  radius.y.lower = NULL,
  radius.y.upper = NULL,
  alpha = 0,
  binconf.x = NULL,
  binconf.y = NULL,
  control.binconf = list(alpha = 0.05, method = "wilson"),
  length = 100,
  ...
)
```

Arguments

- `center.x` Center of the ellipse on x axis
- `center.y` Center of the ellipse on y axis
- `radius.x` Radius along the x axis
- `radius.y` Radius along the y axis
- `radius.x.lower` Radius along the x axis, at left of center
- `radius.x.upper` Radius along the x axis, at right of center
- `radius.y.lower` Radius along the y axis, at bottom of center
radius.y.upper  Radius along the y axis, at top of center
alpha  Rotation in radians
binconf.x  A data.frame or a matrix with two columns, x and n or with three columns, PointEst, Lower, and Upper
binconf.y  A data.frame or a matrix with two columns, x and n or with three columns, PointEst, Lower, and Upper
control.binconf  A list with options for binomial confidence
length  Number of points to draw the ellipse
...  Graphical parameters

Details

ellipse plots an ellipse

Value

Nothing

Author(s)

marc.girondot@u-psud.fr

Examples

plot(0:1, 0:1, xlim=c(0, 1), ylim=c(0,1), lty=2, type="l", las=1, bty="n", xlab="Variable x", ylab="variable y")

ellipse(center.x = c(0.2, 0.3, 0.25), center.y = c(0.7, 0.6, 0.55),
radius.x = c(0.1, 0.1, 0.1), radius.y = c(0.15, 0.2, 0.4),
border=NA, col=rgb(red = 0.1, green = 0.1, blue = 0.1, alpha = 0.1))

ellipse(center.x = 0.5, center.y = 0.5,
radius.x.lower = 0.1, radius.x.upper = 0.3,
radius.y = 0.2,
border=NA, col=rgb(red = 0.1, green = 0.1, blue = 0.1, alpha = 0.1))

ellipse(center.x = 0.6, center.y = 0.3,
radius.x.lower = 0.3, radius.x.upper = 0.3,
radius.y.lower = 0.2, radius.y.upper = 0.4,
border=NA, col=rgb(red = 0.1, green = 0.1, blue = 0.1, alpha = 0.1))

plot(0:1, 0:1, xlim=c(0, 1), ylim=c(0,1), lty=2, type="l", bty="n", asp=1,
xlab="Variable x", ylab="variable y", axes=FALSE)
axis(1, at=c(0, 0.25, 0.5, 0.75, 1))
axis(2, at=c(0, 0.25, 0.5, 0.75, 1), las=1)

ellipse(center.x = 0.5, center.y = 0.5, radius.x = 0.2, radius.y = 0.4,
border=NA, col=rgb(red = 0.1, green = 0.1, blue = 0.1, alpha = 0.1))
ellipse(center.x = 0.5, center.y = 0.5, radius.x = 0.2, radius.y = 0.4,
border=NA, col=rgb(red = 0.1, green = 0.1, blue = 0.1, alpha = 0.1), alpha = pi/4)

plot(0:1, 0:1, xlim=c(0, 1), ylim=c(0, 1), lty=2, type="l", las=1, bty="n",
     xlab="Variable x", ylab="variable y")

for (k in 0:8)
  ellipse(center.x=0.5, center.y=0.5, radius.x=0.1, radius.y=0.4,
          alpha=seq(from=0, to=pi/4, length=9)[k],
          border=rainbow(9)[k])

# Exemple with confidence of proportions
males <- c(10, 25, 3, 4)
N <- c(12, 52, 17, 10)
males2 <- c(12, 20, 3, 6)
N2 <- c(15, 50, 20, 12)

plot(0:1, 0:1, xlim=c(0, 1), ylim=c(0,1), lty=2, type="l", las=1, bty="n",
     xlab="Variable x", ylab="variable y")

ellipse(binconf.x = data.frame(x=males, n=N), binconf.y = data.frame(x=males2, n=N2),
        border=NA, col=rgb(red = 0.1, green = 0.5, blue = 0.1, alpha = 0.1))

plot(0:1, 0:1, xlim=c(0, 1), ylim=c(0,1), lty=2, type="l", las=1, bty="n",
     xlab="Variable x", ylab="variable y")

ellipse(binconf.x = data.frame(x=males, n=N),
        binconf.y = data.frame(PointEst=c(0.1, 0.2, 0.3, 0.5),
                                Lower=c(0.02, 0.12, 0.25, 0.30),
                                Upper=c(0.18, 0.29, 0.35, 0.67)),
        border=NA, col=rgb(red = 0.1, green = 0.5, blue = 0.1, alpha = 0.1))

# Examples with a gradient
plot(0:1, 0:1, xlim=c(0, 1), ylim=c(0,1), lty=2, type="l", las=1, bty="n",
     xlab="Variable x", ylab="variable y")

ellipse(center.x = 0.6, center.y = 0.3,
        radius.x.lower = 0.3, radius.x.upper = 0.3,
        radius.y.lower = 0.2, radius.y.upper = 0.4,
        border=NA, col=grey.colors(100, alpha = 0.1))

plot(0:1, 0:1, xlim=c(0, 1), ylim=c(0,1), lty=2, type="l", las=1, bty="n",
     xlab="Variable x", ylab="variable y")

ellipse(binconf.x = data.frame(x=males, n=N), binconf.y = data.frame(x=males2, n=N2),
        border=NA, col=grey.colors(100, alpha = 0.1))

---

ExtractAIC.glm  Return AIC, AICc or BIC from a glm object
Description

For glm fits the family’s aic() function is used to compute the AIC. The choice between different criteria is done by setting a global option AIC. It can be checked using show.option=TRUE. Indeed, it is not possible to use the ... parameter due to a bug in some functions of MASS package. If you want to use this function as a replacement for setpAIC(), do extractAIC.glm <- ExtractAIC.glm before.

Usage

ExtractAIC.glm(fit, scale = 0, k = 2, ...)

Arguments

fit fitted model, the result of a fitter glm.
scale unused for glm.
k numeric specifying the ‘weight’ of the equivalent degrees of freedom (= edf) part in the AIC formula.
... further arguments (currently unused because addterm.glm and dropterm.glm using this function do not transmit them).

Details

ExtractAIC.glm returns AIC, AICc or BIC from a glm object

Value

A numeric named vector of length 2, with first and second elements giving edf the ‘equivalent degrees of freedom’ for the fitted model fit.

x the Information Criterion for fit.

Author(s)

Modified from stats:::extract.AIC.glm

See Also

Other AIC: FormatCompareAIC(), compare_AICc(), compare_AIC(), compare_BIC()

Examples

extractAIC.glm <- ExtractAIC.glm
n <- 100
x <- rnorm(n, 20, 2)
A <- rnorm(n, 20, 5)
g <- glm(x ~ A)
extractAIC(g, show.option=TRUE)
options(AIC="AIC")
extractAIC(g)
options(AIC="BIC")
Return the flexit

Description

Return a vector with the probabilities. The flexit equation is not still published:

if dose < P then
  (1 + (2^{K1} - 1) \times \exp(4 \times S1 \times (P - x)))^{1/K1} - 1/K1

if dose > P then
  1 - ((1 + (2^{K2} - 1) \times \exp(4 \times S2 \times (x - P)))^{1/K2} - 1/K2)

with:

S1 = S / ((4/K1) \times (2^{1 - K1}))^{1/K1 + 1} \times (2^{K1} - 1))
S2 = S / ((4/K2) \times (2^{1 - K2}))^{1/K2 + 1} \times (2^{K2} - 1))

Usage

flexit(x, par = NULL, P = NULL, S = NULL, K1 = NULL, K2 = NULL, zero = 1e-09, error0 = 0, error1 = 1)

Arguments

x The values at which the flexit model must be calculated
par The vector with P, S, K1, and K2 values
P P value
S S value
K1 K1 value
K2 K2 value
zero Value to replace zero
error0 Value to return if an error is observed toward 0
error1 Value to return if an error is observed toward 1
Details

Return the flexit value

Value

A vector with the probabilities

Author(s)

Marc Girondot

See Also

Other logit: invlogit(), logit()

Examples

n <- flexit(x=1:100, par=c(P=50, S=0.001, K1=0.01, K2=0.02))
n <- flexit(x=1:100, P=50, S=0.001, K1=0.01, K2=0.02)

FormatCompareAIC

Format data to be used with compare_AIC()

Description

Format data to be used with compare_AIC(), compare_AICc() and compare_BIC().
Note that logLik is supposed to not be -logLik.

Usage

FormatCompareAIC(logLik, nobs, df)

Arguments

logLik The log likelihood
nobs Number of observations
df Number of parameters

Details

FormatCompareAIC formats data to be used with compare_AIC()

Value

An object to be used with compare_AIC()
**Author(s)**

Marc Girondot

**See Also**

Other AIC: \texttt{ExtractAIC.glm()}, \texttt{compare_AICc()}, \texttt{compare_AIC()}, \texttt{compare_BIC()}

**Examples**

```r
## Not run:
ED <- FormatCompareAIC(logLik=-140, nobs=100, df=3)
L <- FormatCompareAIC(logLik=-145, nobs=100, df=4)
compare_AIC(L=L, ED=ED)
compare_AICc(L=L, ED=ED)
compare_BIC(L=L, ED=ED)
## End(Not run)
```

---

**growlnotify**

Send growl notification for MacOS X system.

**Description**

This function is used to send a notification to MacOS user.

**Usage**

```r
growlnotify(textinfo ="")
```

**Arguments**

- `textinfo` Text to display in the growlnotify window

**Details**

growlnotify send growl notification for MacOS X systems.

**Value**

None

**Author(s)**

Marc Girondot
IC_clean_data

Examples

## Not run:
# If growlnotify is used on a non-mac system, it just quits.
growlnotify("It works if you are on a Mac with GrowlNotify installed!")

## End(Not run)

IC_clean_data

Clean the dataframe before to be used with IC_threshold_matrix

Description

This function must be used if missing values are present in the dataset. It ensures that all correlations and partial correlations can be calculated. The columns of the dataframe are removed one per one until all can be calculated without error. It is possible to say that one or more columns must be retained because they are of particular importance in the analysis. The use and method parameters are used by cor() function. The function uses by default a parallel computing in Unix or MacOSX systems. If progress is TRUE and the package pbmcapply is present, a progress bar is displayed. If debug is TRUE, some informations are shown during the process. https://fr.wikipedia.org/wiki/Iconographie_des_corrélations

Usage

IC_clean_data(
  data = stop("A data.frame object is required"),
  use = c("pairwise.complete.obs", "everything", "all.obs", "complete.obs", "na.or.complete"),
  method = c("pearson", "kendall", "spearman"),
  variable.retain = NULL,
  test.partial.correlation = TRUE,
  progress = TRUE,
  debug = FALSE
)

Arguments

data The data.frame to be cleaned
use an optional character string giving a method for computing covariances in the presence of missing values. This must be (an abbreviation of) one of the strings "everything", "allObs", "complete.obs", "na.or.complete", or "pairwise.complete.obs".
method a character string indicating which correlation coefficient (or covariance) is to be computed. One of "pearson" (default), "kendall", or "spearman": can be abbreviated.
variable.retain a vector with the name of columns to keep
test.partial.correlation should the partial correlations be tested?
**progress**

Show a progress bar

**debug**

if TRUE, information about progression of cleaning are shown

**Details**

IC_clean_data checks and corrects the dataframe to be used with IC_threshold_matrix

**Value**

A dataframe

**Author(s)**

Marc Girondot

**References**


**See Also**

Other Iconography of correlations: IC_correlation_simplify(), IC_threshold_matrix(), plot.IconoCorel()

**Examples**

```r
## Not run:
library("HelpersMG")
es <- matrix(c("e1", "52", "12", "12", "5",
"e2", "59", "12.5", "9", "5",
"e3", "55", "13", "15", "9",
"e4", "58", "14.5", "5", "5",
"e5", "66", "15.5", "11", "13.5",
"e6", "62", "16", "15", "18",
"e7", "63", "17", "12", "18",
"e8", "69", "18", "9", "18"), ncol=5, byrow = TRUE)
colnames(es) <- c("Élève", "Poids", "Âge", "Assiduité", "Note")
es <- as.data.frame(es, stringsAsFactor=FALSE)
es[, 2] <- as.numeric(as.character(es[, 2]))
es[, 3] <- as.numeric(as.character(es[, 3]))
es[, 4] <- as.numeric(as.character(es[, 4]))
es[, 5] <- as.numeric(as.character(es[, 5]))
es

df <- IC_clean_data(es, debug = TRUE)
cor_matrix <- IC_threshold_matrix(data=df, threshold = NULL, progress=FALSE)
cor_threshold <- IC_threshold_matrix(data=df, threshold = 0.3)
par(mar=c(1,1,1,1))
set.seed(4)
plot(cor_threshold)
cor_threshold_Note <- IC_correlation_simplify(matrix=cor_threshold, variable="Note")
```
IC_correlation_simplify

Simplify the correlation matrix

Description

This function can be used to simplify the network of correlations. If no vector of variables is given, the variables not linked to any other variable are removed. If a vector of variables is given, only link to these variables are retained. https://fr.wikipedia.org/wiki/Iconographie_des_corrélations

Usage

IC_correlation_simplify(matrix, variable = NULL)

Arguments

matrix The correlation matrix to simplify
variable a vector with the name of columns to keep

Details

IC_correlation_simplify simplifies the correlation matrix

Value

A list

Author(s)

Marc Girondot

References


See Also

Other Iconography of correlations: IC_clean_data(), IC_threshold_matrix(), plot.IconoCorel()
**IC_threshold_matrix**

**Calculate correlation matrix**

**Description**

This function calculates the matrix of correlations thresholded using partial correlation.

If the threshold is not given, the object that is produced can be used later for thresholding.

For model OAT: a correlation is retained if it is higher that the threshold and if all partial correlations of the two variables and any third one are all lower than the threshold.

For model AAT: a correlation is retained if it is higher than the threshold and the partial correlation is lower than the threshold. In this case, no missing value is accepted.

The use and method parameters are used by cor() function. The function uses by default a parallel computing in Unix or MacOSX systems. If progress is TRUE and the package pbmcapply is present, a progress bar is displayed. If debug is TRUE, some informations are shown during the process but parallel computing is not used.

Usage

IC_threshold_matrix(
  data = stop("A dataframe or an IconoCorel object is required"),
  threshold = NULL,
  use = c("pairwise.complete.obs", "everything", "all.obs", "complete.obs",
          "na.or.complete"),
  method = c("pearson", "kendall", "spearman"),
  model = c("OAT", "ATT"),
  progress = TRUE,
  debug = FALSE
)

Arguments

data A dataframe or an IconoCorel object from a previous run of IC_threshold_matrix
threshold threshold for partial and full correlations
use an optional character string giving a method for computing covariances in the presence of missing values. This must be (an abbreviation of) one of the strings "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs".
method a character string indicating which correlation coefficient (or covariance) is to be computed. One of "pearson" (default), "kendall", or "spearman": can be abbreviated.
model a character string indicating if linear model uses all variables at a time (AAT) or one at a time (OAT).
progress show a progress bar
debug display information about progression of computing

Details

IC_threshold_matrix calculates correlation matrix thresolated by partial correlation

Value

A list

Author(s)

Marc Girondot

References


See Also

Other Iconography of correlations: IC_clean_data(), IC_correlation_simplify(), plot.IconoCorel()
## Not run:
library("HelpersMG")
es <- matrix(c("e1", "52", "12", "5",
              "e2", "59", "12.5", "9", "5",
              "e3", "55", "13", "15", "9",
              "e4", "58", "14.5", "5", "5",
              "e5", "66", "15.5", "11", "13.5",
              "e6", "62", "16", "15", "18",
              "e7", "63", "17", "12", "18",
              "e8", "69", "18", "9", "18"), ncol=5, byrow = TRUE)
colnames(es) <- c("Élève", "Poids", "Âge", "Assiduité", "Note")
es <- as.data.frame(es, stringsasFactor=FALSE)
es[, 2] <- as.numeric(as.character(es[, 2]))
es[, 3] <- as.numeric(as.character(es[, 3]))
es[, 4] <- as.numeric(as.character(es[, 4]))
es[, 5] <- as.numeric(as.character(es[, 5]))
es
df <- IC_clean_data(es, debug = TRUE)
cor_matrix <- IC_threshold_matrix(data=df, threshold = NULL, progress=FALSE)
cor_threshold <- IC_threshold_matrix(data=cor_matrix, threshold = 0.3)
par(mar=c(1,1,1,1))
set.seed(4)
plot(cor_threshold)
cor_threshold_Note <- IC_correlation_simplify(matrix=cor_threshold, variable="Note")
plot(cor_threshold_Note)

# Using the model All at a time

cor_threshold_AAT <- IC_threshold_matrix(data=df, threshold = 0.3, model="AAT")
par(mar=c(1,1,1,1))
set.seed(4)
plot(cor_threshold_AAT, show.legend.strength="bottomleft")
```r
dta0 <- dta[, 2:ncol(dta)]
ic0 <- IC_threshold_matrix(data = dta0)
cor_threshold <- IC_threshold_matrix(data=ic0, threshold = 0.3)
par(mar=c(1,1,1,1))
set.seed(4)
library("igraph")
plot(cor_threshold, vertex.color="red", show.legend.strength = FALSE)
plot(IC_correlation_simplify(matrix=cor_threshold),
     show.legend.strength = FALSE, show.legend.direction = FALSE)
```

```r
# End(Not run)
```

### index.periodic

Estimate indices in periodic timeseries based on anchored minimum and maximum.

**Description**

Estimate indices in periodic timeseries based on anchored minimum and maximum. The data.frame minmax can be generated manually. It should have three columns (time, index, SD), with all the successive minimum and maximum indices. It can be used with sun.info() to get the time of minimum and maximum air temperature or with getTide() to reconstruct the sea level.

**Usage**

```r
index.periodic(minmax, time = NULL, replicates = 100, progressbar = FALSE)
```

**Arguments**

- `minmax`: A data.frame returned by minmax.periodic
- `time`: The time at which produced the estimate
- `replicates`: Number of replicates to estimate SD
- `progressbar`: Does a progression bar must be shown

**Details**

`index.periodic` estimate indices in periodic timeseries based on anchored minimum and maximum

**Value**

A data.frame with a column time and a column index

**Author(s)**

Marc Girondot <marc.girondot@u-psud.fr>
### Examples

```r
## Not run:
# Generate a timeserie of time
time.obs <- NULL
for (i in 0:9) time.obs <- c(time.obs, c(0, 6, 12, 18)+i*24)
# For these time, generate a timeseries of temperatures
temp.obs <- rep(NA, length(time.obs))
temp.obs[3+(0:9)*4] <- rnorm(10, 25, 3)
temp.obs[1+(0:9)*4] <- rnorm(10, 10, 3)
for (i in 1:(length(time.obs)-1))
  if (is.na(temp.obs[i]))
    temp.obs[i] <- mean(c(temp.obs[i-1], temp.obs[i+1]))
  if (is.na(temp.obs[length(time.obs)]))
    temp.obs[length(time.obs)] <- temp.obs[length(time.obs)-1]/2
observed <- data.frame(time=time.obs, temperature=temp.obs)
# Search for the minimum and maximum values
r <- minmax.periodic(time.minmax.daily=c(Min=2, Max=15),
                      observed=observed, period=24, colname.index="temperature")
# Estimate all the temperatures for these values
t <- index.periodic(minmax=r)
plot.errbar(x=t[, "time"], y=t[, "index"],
            errbar.y=ifelse(is.na(t[, "sd"]), 0, 2*t[, "sd"]),
            type="l", las=1, bty="n", errbar.y.polygon = TRUE,
            xlab="hours", ylab="Temperatures", ylim=c(0, 35),
            errbar.y.polygon.list = list(col="grey"))
plot.add(x=t[, "time"], y=t[, "index"], type="l")
plot.add(observed$time, observed$temperature, pch=19, cex=0.5)
## End(Not run)
```
ind_long_lat

long = NULL,
lat = NULL,
indice.long = NULL,
indice.lat = NULL,
name.lon = "lon",
name.lat = "lat"
)

Arguments
ncdf An object read from package ncdf4, ncdf or RNetCDF
long Longitude in decimal format
lat Latitude in decimal format
indice.long Index of longitude
indice.lat Index of latitude
name.lon Name of argument for longitude, default is lon
name.lat Name of argument for latitude, default is lat

Details
ind_long_lat is used to manage ncdf information

Value
Or the index in ncdf object from lat/longitude or inverse

Author(s)
Marc Girondot

Examples
## Not run:
url <- paste0(url, "sst.day.mean.2012.v2.nc")
dest <- paste(Sys.getenv("HOME"), "/sst.day.mean.2012.v2.nc", sep="")
download.file(url, dest)
library("ncdf4")
dta2012 <- nc_open(dest)
indices <- ind_long_lat(ncdf=dta2012, lat=5.89, long=-20.56)
coordinates <- ind_long_lat(ncdf=dta2012, indice.lat=20, indice.long=30)
# library("RNetCDF")
# dta2012 <- open.nc(dest)
# indices <- ind_long_lat(ncdf=dta2012, lat=5.89, long=-20.56)
# coordinates <- ind_long_lat(ncdf=dta2012, indice.lat=20, indice.long=30)
# ncdf library is depreciated in CRAN
# library("ncdf")
# dta2012 <- open.ncdf(dest)
# indices <- ind_long_lat(ncdf=dta2012, lat=5.89, long=-20.56)
inside.search

# coordinates <- ind_long_lat(ncdf=dta2012, indice.lat=20, indice.long=30)

## End(Not run)

inside.search Search a string within files of a folder

Description

Search for a string inside the files of a folder and return where the string is found. The pattern for files that must be included uses regex for filtering.

Usage

inside.search(
  path = ".",
  pattern = "*\.R$",
  showallfilenames = FALSE,
  ...
  fixed = TRUE,
  ignore.case = FALSE,
  text = stop("A text to be searched for is necessary") 
)

Arguments

path Path of the folder to search in
pattern Pattern for file names to search in
showallfilenames logical. Show all the filenames search for in
... Options for readLines(), example warn = FALSE
fixed logical. If TRUE, pattern is a string to be matched as is. Overrides all conflicting arguments (see gsub)
ignore.case logical. if FALSE, the pattern matching for text is case sensitive and if TRUE, case is ignored during matching.
text Text to search in files

Details

inside.search Search a string within files of a folder

Value

Return an invisible vector with filenames in which the pattern occurs
invlogit

Description

Return the inverse logit.

Usage

invlogit(n)

Arguments

n

The value to inverse to get the probability

Details

invlogit returns the inverse logit

Value

A value

Author(s)

Marc Girondot

See Also

Other logit: flexit(), logit()

Examples

n <- logit(0.5)
invlogit(n)
Description

Estimate the parameters that best describe LD50

Logistic and logit models are the same but with different parametrization:

\[
\begin{align*}
\text{logistic} &= \frac{1}{1 + \exp((1/S)\times(P-d))} \\
\text{logit} &= \frac{1}{1 + \exp(P + d\times S)}
\end{align*}
\]

See these publications for the description of equations:


The flexit equation is not still published:

\[
\begin{align*}
\text{if } \text{dose} < P & \text{ then } (1 + (2^{K1} - 1) \times \exp(4 \times S1 \times (P - x)))^{1/1/K1} \\
\text{if } \text{dose} > P & \text{ then } 1 - ((1 + (2^{K2} - 1) \times \exp(4 \times S2 \times (x - P)))^{1/1/K2})
\end{align*}
\]

with:

\[
\begin{align*}
S1 &= S / ((4/K1)^{(2^K1 - 1)^{1/1/K1 + 1}}) * (2^K1 - 1)) \\
S2 &= S / ((4/K2)^{(2^K2 - 1)^{1/1/K2 + 1}}) * (2^K2 - 1))
\end{align*}
\]

Usage

```
LD50(
  df = NULL,
  alive = NULL,
  dead = NULL,
  N = NULL,
  doses = NULL,
  l = 0.05,
  parameters.initial = NULL,
  fixed.parameters = NULL,
  SE = NULL,
  equation = "logistic",
  replicates = 1000,
  range.CI = 0.95,
  limit.low.TRD.minimum = 5,
  limit.high.TRD.maximum = 1000,
  print = TRUE,
  doses.plot = seq(from = 0, to = 1000, by = 0.1)
)
```
**Arguments**

- **df**: A dataframe with at least two columns named alive, dead or N and doses columns
- **alive**: A vector with alive individuals at the end of experiment
- **dead**: A vector with dead individuals at the end of experiment
- **N**: A vector with total numbers of tested individuals
- **doses**: The doses
- **l**: The limit to define TRD (see Girondot, 1999)
- **parameters.initial**: Initial values for P, S or K search as a vector, ex. c(P=29, S=-0.3)
- **fixed.parameters**: Parameters that will not be changed during fit
- **SE**: Standard errors for parameters
- **equation**: Could be "logistic", "logit", "probit", "Hill", "Richards", "Hulin", "flexit" or "Double-Richards"
- **replicates**: Number of replicates to estimate confidence intervals
- **range.CI**: The range of confidence interval for estimation, default=0.95
- **limit.low.TRD.minimum**: Minimum lower limit for TRD
- **limit.high.TRD.maximum**: Maximum higher limit for TRD
- **print**: Do the results must be printed at screen? TRUE (default) or FALSE
- **doses.plot**: Sequences of doses that will be used for plotting. If NULL, does not estimate them

**Details**

LD50 estimates the parameters that best describe LD50

**Value**

A list with the LD50, Transitional Range of Doses and their SE

**Author(s)**

Marc Girondot <marc.girondot@u-psud.fr>

**See Also**

Other LD50 functions: `LD50_MHmcmc_p()`, `LD50_MHmcmc()`, `logLik.LD50()`, `plot.LD50()`, `predict.LD50()`
**Examples**

```r
## Not run:
library("HelpersMG")
data <- data.frame(Doses=c(80, 120, 150, 150, 180, 200),
Alive=c(10, 12, 8, 6, 2, 1),
Dead=c(0, 1, 5, 6, 9, 15))
LD50_logistic <- LD50(data, equation="logistic")
predict(LD50_logistic, doses=c(140, 170))
plot(LD50_logistic, xlim=c(0, 300), at=seq(from=0, to=300, by=50))
LD50_probit <- LD50(data, equation="probit")
predict(LD50_probit, doses=c(140, 170))
plot(LD50_probit)
LD50_logit <- LD50(data, equation="logit")
predict(LD50_logit, doses=c(140, 170))
plot(LD50_logit)
LD50_hill <- LD50(data, equation="hill")
predict(LD50_hill, doses=c(140, 170))
plot(LD50_hill)
LD50_Richards <- LD50(data, equation="Richards")
predict(LD50_Richards, doses=c(140, 170))
plot(LD50_Richards)
LD50_Hulin <- LD50(data, equation="Hulin")
predict(LD50_Hulin, doses=c(140, 170))
plot(LD50_Hulin)
LD50_DoubleRichards <- LD50(data, equation="Double-Richards")
predict(LD50_DoubleRichards, doses=c(140, 170))
plot(LD50_DoubleRichards)
LD50_flexit <- LD50(data, equation="flexit")
predict(LD50_flexit, doses=c(140, 170))
plot(LD50_flexit)
## End(Not run)
```

**LD50_MHmcmc**

*Metropolis-Hastings algorithm for LD50*

**Description**

Run the Metropolis-Hastings algorithm for tsd.
Deeply modified from a MCMC script by Olivier Martin (INRA, Paris-Grignon).
The number of iterations is n.iter+n.adapt+1 because the initial likelihood is also displayed.
I recommend that thin=1 because the method to estimate SE uses resampling.
If initial point is maximum likelihood, n.adapt = 0 is a good solution.
To get the SE from result_mcmc <- tsd_MHmcmc(result=try), use:
result_mcmc$BatchSE or result_mcmc$TimeSeriesSE
The batch standard error procedure is usually thought to be not as accurate as the time series methods.
Based on Jones, Haran, Caffo and Neath (2005), the batch size should be equal to sqrt(n.iter).

The parameters intermediate and filename are used to save intermediate results every 'intermediate' iterations (for example 1000). Results are saved in a file of name filename. The parameter previous is used to indicate the list that has been save using the parameters intermediate and filename. It permits to continue a mcmc search. These options are used to prevent the consequences of computer crash or if the run is very very long and processes at time limited.

Usage

LD50_MHmcmc(
  result = stop("A result of LD50() fit must be provided"),
  n.iter = 10000,
  parametersMCMC = NULL,
  n.chains = 1,
  n.adapt = 0,
  thin = 1,
  trace = FALSE,
  batchSize = sqrt(n.iter),
  adaptive = FALSE,
  adaptive.lag = 500,
  adaptive.fun = function(x) { ifelse(x > 0.234, 1.3, 0.7) },
  intermediate = NULL,
  filename = "intermediate.Rdata",
  previous = NULL
)

Arguments

result An object obtained after a SearchR fit
n.iter Number of iterations for each step
parametersMCMC A set of parameters used as initial point for searching with information on priors
n.chains Number of replicates
n.adapt Number of iterations before to store outputs
thin Number of iterations between each stored output
trace True or False, shows progress
batchSize Number of observations to include in each batch fo SE estimation
adaptive Should an adaptive process for SDProp be used
adaptive.lag Lag to analyze the SDProp value in an adaptive content
adaptive.fun Function used to change the SDProp
intermediate Period for saving intermediate result, NULL for no save
filename If intermediate is not NULL, save intermediate result in this file
previous Previous result to be continued. Can be the filename in which intermediate results are saved.
LD50_MHmcmc

Details
LD50_MHmcmc runs the Metropolis-Hastings algorithm for LD50 (Bayesian MCMC)

Value
A list with resultMCMC being mcmc.list object, resultLnL being likelihoods and parametersMCMC being the parameters used

Author(s)
Marc Girondot

See Also
Other LD50 functions: **LD50_MHmcmc_p()**, **LD50()**, **logLik.LD50()**, **plot.LD50()**, **predict.LD50()**

Examples
```r
## Not run:
library("HelpersMG")
data <- data.frame(Doses=c(80, 120, 150, 150, 180, 200),
                  Alive=c(10, 12, 8, 6, 2, 1),
                  Dead=c(0, 1, 5, 6, 9, 15))
LD50_logistic <- LD50(data, equation="logistic")
pMCMC <- LD50_MHmcmc_p(LD50_logistic, accept=TRUE)
# Take care, it can be very long
result_mcmc_LD50 <- LD50_MHmcmc(result=LD50_logistic,
                                 parametersMCMC=pMCMC, n.iter=10000, n.chains = 1,
                                 n.adapt = 0, thin=1, trace=1000, adaptive=TRUE, )
# summary() permits to get rapidly the standard errors for parameters
summary(result_mcmc_LD50)
plot(x=result_mcmc_LD50, parameters="S", scale.prior=TRUE, las=1)
plot(result_mcmc_LD50, parameters="S", scale.prior=TRUE, las=1, xlim=c(-20, 20))
plot(result_mcmc_LD50, parameters="P", scale.prior=TRUE, las=1)
1-rejectionRate(as.mcmc(result_mcmc_LD50))
raftery.diag(as.mcmc(result_mcmc_LD50))
heidel.diag(as.mcmc(result_mcmc_LD50))

#### Example with Uniforms priors

pMCMC <- structure(list(Density = c("dunif", "dunif"),
                        Prior1 = c(77.6216005852911, -31.0438095277258),
                        Prior2 = c(310.486402341165, 31.0438095277258),
                        SDProp = c(2, 0.5),
                        Min = c(77.6216005852911, -31.0438095277258),
                        Max = c(310.486402341165, 31.0438095277258),
                        Init = c(155.243201170582, -15.5219047638629),
                        row.names = c("P", "S"), class = "data.frame")
result_mcmc_LD50 <- LD50_MHmcmc(result=LD50_logistic,
                                 parametersMCMC=pMCMC, n.iter=10000, n.chains = 1,
                                 n.adapt = 0, thin=1, trace=1000, adaptive=TRUE, )
# summary() permits to get rapidly the standard errors for parameters
```
LD50_MHmcmc_p

Generates set of parameters to be used with LD50_MHmcmc()

Description
Interactive script used to generate set of parameters to be used with LD50_MHmcmc().

Usage
LD50_MHmcmc_p(
    result = stop("An output from LD50() must be provided"),
    accept = FALSE
)

Arguments
result An object obtained after a LD50 fit
accept If TRUE, the script does not wait user information

Details
LD50_MHmcmc_p generates set of parameters to be used with LD50_MHmcmc()

Value
A matrix with the parameters

Author(s)
Marc Girondot

See Also
Other LD50 functions: LD50_MHmcmc(), LD50(), loglik.LD50(), plot.LD50(), predict.LD50()
Examples

## Not run:
library("HelpersMG")
data <- data.frame(Doses=c(80, 120, 150, 150, 180, 200),
                   Alive=c(10, 12, 8, 6, 2, 1),
                   Dead=c(0, 1, 5, 6, 9, 15))
LD50_logistic <- LD50(data, equation="logistic")
pmcmc <- LD50_MHmcmc_p(LD50_logistic, accept=TRUE)

## End(Not run)

---

list.packages

List the installed packages with their locations

Description

List the installed packages with their locations and version.

Usage

list.packages()

Details

list.packages lists the installed packages with their locations

Value

A list with the installed packages and their version.

Author(s)

Marc Girondot

Examples

## Not run:
library(HelpersMG)
list.packages()

## End(Not run)
local.search

Return path of file searched for in local disk based on its file name

Description

Return path of file searched for in local disk based on its file name. It has been tested only with Windows XP and MacOSX. In MacOSX, you must have created the locate database first. Use OnyX utilities for this purpose.

Usage

local.search(
  pattern,
  directory = "",
  folder = "$HOME",
  intern = TRUE,
  ignore.stdout = FALSE,
  ignore.stderr = TRUE
)

Arguments

pattern The name of file to be searched for. Can use wildcards *
directory The path of directory to be explored in for Windows
folder The path of folder to be explored in for Unix based systems
intern A logical (not NA) which indicates whether to capture the output of the command as an R character vector (see system()).
ignore.stdout a logical (not NA) indicating whether messages written to 'stdout' should be ignored (see system()).
ignore.stderr a logical (not NA) indicating whether messages written to 'stderr' should be ignored (see system()).

Details

local.search() returns path of file searched in local disk based on its file name

Value

A vector with paths

Author(s)

Marc Girondot
logit

Description
Return the logit.

Usage
logit(p)

Arguments
p The probability

Details
logit returns the logit

Value
A value

Author(s)
Marc Girondot

See Also
Other logit: flexit(), invlogit()

Examples
n <- logit(0.5)
invlogit(n)

Examples
## Not run:
RnwFiles <- local.search("*.Rnw")
nc.files <- local.search("*.nc", folder=paste0("",getwd(),""))
## End(Not run)
logLik.compareAIC  Return Log Likelihood generated by FormatCompareAIC

Description

Return Log Likelihood generated by FormatCompareAIC

Usage

## S3 method for class 'compareAIC'
logLik(object, ...)

Arguments

object A result generated by FormatCompareAIC
... Not used

Details

logLik.compareAIC Return Log Likelihood of a fit

Value

The Log Likelihood value for the fitted model with data

Author(s)

Marc Girondot

Examples

## Not run:
ED <- FormatCompareAIC(logLik=-140, nobs=100, df=3)
logLik(ED)

## End(Not run)
logLik.LD50

Return Log Likelihood of a fit generated by LD50

Description

Return Log Likelihood of a fit generated by LD50

Usage

## S3 method for class 'LD50'
logLik(object, ...)

Arguments

object A result file generated by fitRMU
...

Not used

Details

logLik.LD50 Return Log Likelihood of a fit for LD50

Value

The Log Likelihood value for the fitted model with data

Author(s)

Marc Girondot

See Also

Other LD50 functions: LD50_MHmcmc_p(), LD50_MHmcmc(), LD50(), plot.LD50(), predict.LD50()

Examples

## Not run:
data <- data.frame(Doses=c(80, 120, 150, 150, 180, 200),
Alive=c(10, 12, 8, 6, 2, 1),
Dead=c(0, 1, 5, 6, 9, 15))
LD50_logistic <- LD50(data, equation="logistic")
logLik(LD50_logistic)
AIC(LD50_logistic)

## End(Not run)
Description

Merge two mcmcComposite results and produced a new one mcmcComposite object. Note that the initial value for the second run must use the last value of the first one as shown in example.

Usage

```r
## S3 method for class 'mcmcComposite'
merge(x, y, ...)
```

Arguments

- `x`: A mcmcComposite obtained as a result of `MHalgoGen()` function
- `y`: A mcmcComposite obtained as a result of `MHalgoGen()` function
- `...`: not used

Details

merge.mcmcComposite Merge two mcmcComposite results

Value

A mcmcComposite result

Author(s)

Marc Girondot

See Also

Other mcmcComposite functions: `MHalgoGen()`, `as.mcmc.mcmcComposite()`, `as.parameters()`, `as.quantiles()`, `plot.mcmcComposite()`, `summary.mcmcComposite()`

Examples

```r
## Not run:
library(HelpersMG)
require(coda)
x <- rnorm(30, 10, 2)
dnormx <- function(data, x) {
  data <- unlist(data)
  return(-sum(dnorm(data, mean=x['mean'], sd=x['sd'], log=TRUE)))
}
parameters_mcmc <- data.frame(Density=c('dnorm', 'dlnorm'),
```
MHalgoGen

Monte-Carlo Markov-chain with Metropolis-Hastings algorithm

Description

The parameters must be stored in a data.frame with named rows for each parameter with the following columns:

- Density. The density function name, example dnorm, dlnorm, dunif
- Prior1. The first parameter to send to the Density function
- Prior2. The second parameter to send to the Density function
- SDProp. The standard error from new proposition value of this parameter
- Min. The minimum value for this parameter
• Max. The maximum value for this parameter
• Init. The initial value for this parameter

This script has been deeply modified from a MCMC script provided by Olivier Martin (INRA, Paris-Grignon).
The likelihood function must use a parameter named parameters_name for the named parameters.
For adaptive mcmc, see:
S. Brooks, A. Gelman, G. Jones, and X.-L. Meng, editors. MCMC Handbook. Chapman and
Hall/CRC.

Usage

MHalgoGen(
  likelihood = stop("A likelihood function must be supplied"),
  parameters_name = "x",
  parameters = stop("Priors must be supplied"),
  ...,
  n.iter = 10000,
  n.chains = 1,
  n.adapt = 100,
  thin = 30,
  trace = FALSE,
  traceML = FALSE,
  adaptive = FALSE,
  adaptive.lag = 500,
  adaptive.fun = function(x) { ifelse(x > 0.234, 1.3, 0.7) },
  intermediate = NULL,
  filename = "intermediate.Rdata",
  previous = NULL
)

Arguments

likelihood The function that returns -ln likelihood using data and parameters
parameters_name The name of the parameters in the likelihood function, default is "x"
parameters A data.frame with priors; see description and examples
... Parameters to be transmitted to likelihood function
n.iter Number of iterations for each chain
n.chains Number of chains
n.adapt Number of iteration to stabilize likelihood
thin Interval for thinning likelihoods
trace Or FALSE or period to show progress
traceML TRUE or FALSE to show ML
adaptive Should an adaptive process for SDProp be used
MHalgoGen

adaptive.lag    Lag to analyze the SDProp value in an adaptive context
adaptive.fun    Function used to change the SDProp
intermediate    Or NULL of period to save intermediate result
filename        Name of file in which intermediate results are saved
previous        The content of the file in which intermediate results are saved

Details

MHalgoGen is a function to use mcmc with Metropolis-Hastings algorithm

Value

A mcmcComposite object with all characteristics of the model and mcmc run

Author(s)

Marc Girondot

See Also

Other mcmcComposite functions: as.mcmc.mcmcComposite(), as.parameters(), as.quantiles(), merge.mcmcComposite(), plot.mcmcComposite(), summary.mcmcComposite()

Examples

## Not run:
library(HelpersMG)
require(coda)
val <- rnorm(30, 10, 2)
dnormx <- function(data, x) {
data <- unlist(data)
return(-sum(dnorm(data, mean=x['mean'], sd=x['sd'], log=TRUE)))
}
parameters_mcmc <- data.frame(Density=c('dnorm', 'dlnorm'), Prior1=c(10, 0.5), Prior2=c(2, 0.5), SDProp=c(0.35, 0.2), Min=c(-3, 0), Max=c(100, 10), Init=c(10, 2), stringsAsFactors = FALSE, row.names=c('mean', 'sd'))
# Use of trace and traceML parameters
# trace=1 : Only one likelihood is printed
mcmc_run <- MHalgoGen(n.iter=50000, parameters=parameters_mcmc, data=val, likelihood=dnormx, n.chains=1, n.adapt=100, thin=1, trace=1)
# trace=10 : 10 likelihoods are printed
mcmc_run <- MHalgoGen(n.iter=50000, parameters=parameters_mcmc, data=val, likelihood=dnormx, n.chains=1, n.adapt=100, thin=1, trace=10)
# trace=TRUE : all likelihoods are printed
mcmc_run <- MHalgoGen(n.iter=50000, parameters=parameters_mcmc, data=val, likelihood=dnormx, n.chains=1, n.adapt=100, thin=1, trace=TRUE)
# trace=FALSE : No likelihood is printed
mcmc_run <- MHalgoGen(n.iter=50000, parameters=parameters_mcmc, data=val, likelihood=dnormx, n.chains=1, n.adapt=100, thin=1, trace=FALSE)
# traceML=TRUE : values when likelihood is better are shown

mcmc_run <- MHalgoGen(n.iter=100, parameters=parameters_mcmc, data=val, likelihood=dnormx, n.chains=1, n.adapt=100, thin=1, trace=TRUE, traceML=TRUE)
mcmc_run <- MHalgoGen(n.iter=100, parameters=parameters_mcmc, data=val, likelihood=dnormx, n.chains=1, n.adapt=100, thin=1, trace=FALSE, traceML=TRUE)

plot(mcmc_run, xlim=c(0, 20))
plot(mcmc_run, xlim=c(0, 10), parameters="sd")
library(graphics)
library(fields)
# show a scatter plot of the result
x <- mcmc_run$resultMCMC[, 1]
y <- mcmc_run$resultMCMC[, 2]
marpre <- par(mar=c(4, 4, 2, 6)+0.4)
smoothScatter(x, y)
# show a scale
n <- matrix(0, ncol=128, nrow=128)
xrange <- range(x)
yrange <- range(y)
for (i in 1:length(x)) {
posx <- 1+floor(127*(x[i]-xrange[1])/(xrange[2]-xrange[1]))
posy <- 1+floor(127*(y[i]-yrange[1])/(yrange[2]-yrange[1]))
n[posx, posy] <- n[posx, posy]+1}
image.plot(legend.only=TRUE, zlim= c(0, max(n)), nlevel=128, col=colorRampPalette(c("white", blues9))(128))
# Compare with a heatmap
x <- seq(from=8, to=12, by=0.2)
y <- seq(from=1, to=4, by=0.2)
df <- expand.grid(mean=x, sd=y)
df <- cbind(df, L=rep(0, length(nrow(df))))
for (i in 1:nrow(df)) df[i, "L"] <- -sum(dnorm(val, df[i, 1], df[i, 2], log = TRUE))
hm <- matrix(df[, "L"], nrow=length(x))
par(mar = marpre)
image.plot(x=x, y=y, z=hm, las=1)
# Diagnostic function from coda library
mcmcforcuda <- as.mcmc(mcmc_run)
#' heidel.diag(mcmcforcuda)
raftery.diag(mcmcforcuda)
autocorr.diag(mcmcforcuda)
acf(mcmcforcuda[, "mean"], lag.max=20, bty="n", las=1)
acf(mcmcforcuda[, "sd"], lag.max=20, bty="n", las=1)
batchSE(mcmcforcuda, batchSize=100)
# The batch standard error procedure is usually thought to
# be not as accurate as the time series methods used in summary
summary(mcmcforcuda)$statistics[, "Time-series SE"]

summary(mcmc_run)
as.parameters(mcmc_run)
lastp <- as.parameters(mcmc_run, index="last")
parameters_mcmc[,"Init"] <- lastp
# The n.adapt set to 1 is used to not record the first set of parameters
# then it is not duplicated (as it is also the last one for
# the object mcmc_run)
mcmc_run2 <- MHalgoGen(n.iter=1000, parameters=parameters_mcmc, x=x,
minmax.periodic

Search for minimum and maximum indices in periodic timeseries

Description

Search for minimum and maximum for periodic timeseries when only intermediate values are known.

For each couple of value with an increasing or decreasing segment of the sinusoid function, it is possible to estimate a minimum and maximum values using analytical algebra.

Then the average and standard deviations of all minima and maxima are evaluated.

It should be noted that any extremum can be estimated at least twice, one by increasing segment and one by decreasing segment. Both are used here to produce SD.

time.minmax.daily should be used when the time at which maximum and minimum indices are regular and time.minmax permits to define this time day by day.

Usage

minmax.periodic(
  time.minmax.daily = NULL,
  time.minmax = NULL,
  progressbar = FALSE,
  observed = stop("data.frame with observed indices"),
  period = 24,
  colname.time = "time",
  colname.index = "index",
  colname.SD = "SD",
  plot = FALSE
)
minmax.periodic

Arguments

time.minmax.daily  A named vector with Min and Max being the time in the day with minimum and maximum indices (temperature or level)
time.minmax       A named vector daily with time in the day at which minimum and maximum indices are observed
progressbar        Tell if a progression bar must be shown
observed           A dataframe with at least two columns: time and temperatures. A third column SD can indicate the know error in index
period             The unit of day period (24 for hours, 24*60 for minutes)
colname.time       The name of the column for time in observed
colname.index      The name of the column for indices in observed
colname.SD         The name of the column for SD in observed
plot               If TRUE, show a plot with the different estimates

Details

minmax.periodic search for minimum and maximum indices (temperatures or levels) in periodic timeseries

Value

A data.frame with a column time, a column index and a column SD

Author(s)

Marc Girondot

See Also

Other Periodic patterns of indices: index.periodic(), moon.info(), sun.info(), tide.info()

Examples

```r
# Not run:
library(" HelpersMG")
# Generate a timeserie of time
time.obs <- NULL
for (i in 0:9) time.obs <- c(time.obs, c(0, 6, 12, 18)+i*24)
# For these time, generate a timeseries of temperatures
temp.obs <- rep(NA, length(time.obs))
temp.obs[3*(0:9)*4] <- rnorm(10, 25, 3)
temp.obs[1+(0:9)*4] <- rnorm(10, 10, 3)
for (i in 1:(length(time.obs)-1))
  if (is.na(temp.obs[i]))
    temp.obs[i] <- mean(c(temp.obs[i-1], temp.obs[i+1]))
  if (is.na(temp.obs[length(time.obs)]))
    temp.obs[length(time.obs)] <- temp.obs[length(time.obs)-1]/2
```
observed <- data.frame(time=time.obs, temperature=temp.obs)
# Search for the minimum and maximum values
r <- minmax.periodic(time.minmax.daily=c(Min=2, Max=15),
observed=observed, period=24, colname.index="temperature")

# Estimate all the temperatures for these values
t <- index.periodic(minmax=r)

plot_errbar(x=t[,"time"], y=t[,"index"],
errbar.y=ifelse(is.na(t[,"sd"]), 0, 2*t[,"sd"]),
type="l", las=1, bty="n", errbar.y.polygon = TRUE,
xlab="hours", ylab="Temperatures", ylim=c(0, 35),
errbar.y.polygon.list = list(col="grey"))

plot_add(x=t[,"time"], y=t[,"index"], type="l")

plot_add(observed$time, observed$temperature, pch=19, cex=0.5)

## End(Not run)

modeled.hist

Return the theoretical value for the histogram bar

Description

Return the theoretical value for the histogram bar based on a model of distribution.

Usage

modeled.hist(breaks, FUN, ..., sum = 1)

Arguments

breaks Vector with the breaks; it can be obtained directly from hist()
FUN Function to be used to integrate the density, ex. pnorm
... Parameters to be used by FUN
sum Total numbers in the histogram; 1 for emperical frequencies

Details

modeled.hist returns the theoretical value for the histogram bar based on a model of distribution.

Value

A list with x (the center of the bar) and y components

Author(s)

Marc Girondot
modifyVector

## Not run:

n <- rnorm(100, mean=10, sd=2)
breaks <- 0:20
hist(n, breaks=breaks)

s <- modeled.hist(breaks=breaks, FUN=pnorm, mean=10, sd=2, sum=100)

points(s$x, s$y, pch=19)
lines(s$x, s$y)

n <- rlnorm(100, meanlog=2, sdlog=0.4)
b <- hist(n, ylim=c(0, 70))

s <- modeled.hist(breaks=b$breaks, FUN=plnorm, meanlog=2, sdlog=0.4, sum=100)

points(s$x, s$y, pch=19)
lines(s$x, s$y)

## End(Not run)

modifyVector

Modifies Elements of a Vector

Description

Modifies a vector by changing a subset of elements to match a second vector.

Usage

modifyVector(x, val, add = TRUE)

Arguments

- **x**: A named vector.
- **val**: A named vector with components to replace corresponding components in `x`.
- **add**: If FALSE, only existing elements of `x` are returned.

Details

modifyVector modifies elements of a vector

Value

A modified version of `x`, with the elements of `val` replacing the elements of `x`

Author(s)

Marc Girondot
Examples

library("HelpersMG")
e <- c(M=10, L=20, J=30)
modifyVector(e, c(U=10, M=30))
modifyVector(e, c(U=10, M=30), add=FALSE)

Description

The script gives an index (base 100) that represents moon phase.
If the return value (from 0 to 100) is between:
0 and 1.6931595 or 98.3068405 and 100, it is full moon,
23.3068405 and 26.6931595, last quarter,
48.3068405 and 51.6931595, new moon,
73.3068405 and 76.6931595, first quarter
When phase is set to TRUE, a character representing the moon phase is returned.

Usage

moon.info(date = Sys.Date(), phase = FALSE)

Arguments

date A date in class Date. By default, it will use today date
phase If TRUE, a vector of characters with NM, FQ, FL, LQ will be returned

Details

moon.info calculates the moon phase based on a date.

Value

Return a value describing the moon phase:
0 and 100 are full moon, 50 is new moon, 25 last quarter and 75 first quart

Author(s)

Marc Girondot <marc.girondot@u-psud.fr>

See Also

Other Periodic patterns of indices: index.periodic(), minmax.periodic(), sun.info(), tide.info()
### Examples

```r
## Not run:
library("HelpersMG")
moon.info(as.Date("2001-12-31"))
moon.info(as.Date("14/04/2010", "%d/%m/%Y"))
moon.info(as.Date("22/06/07", "%d/%m/%Y"))
moon.info(seq(from=as.Date("2012-03-01"),
to=as.Date("2012-04-15"), by="days"))
moon.info(seq(from=as.Date("2012-03-01"),
to=as.Date("2012-04-15"), by="days"), phase=TRUE)

## End(Not run)
```

---

**MovingWindow**

*Return a moving average of a vector.*

#### Description

Return a moving average of a vector. The hole parameter can be none, bothL, bothR, both, begin, end.

#### Usage

```
MovingWindow(x, window, hole = "begin", fill = TRUE, FUN = mean)
```

#### Arguments

- `x`: The vector to analyze
- `window`: The window size
- `hole`: Should the returned vector have the same length than `x`
- `fill`: TRUE or FALSE, should the vector return NA
- `FUN`: Function to apply to the window

#### Details

MovingWindow returns a moving average of a vector.

#### Value

A vector

#### Author(s)

Marc Girondot
newcompassRose

Examples

MovingWindow(1:10, window = 4, fill = TRUE, hole="bothL")
MovingWindow(1:10, window = 4, fill = TRUE, hole="bothR")
MovingWindow(1:10, window = 4, fill = TRUE, hole="both")
MovingWindow(1:10, window = 4, fill = TRUE, hole="none")
MovingWindow(1:10, window = 4, fill = TRUE, hole="begin")
MovingWindow(1:10, window = 4, fill = TRUE, hole="end")
MovingWindow(1:10, window = 4, fill = TRUE, hole="end", FUN=sd)

newcompassRose  Display a compass rose

Description

Displays a basic compass rose, usually to orient a map.
newcompassRose displays a conventional compass rose at the position requested.
The size of the compass rose is determined by the character expansion, as the central "rose" is calculated relative to the character size.
Rotation is in degrees counterclockwise.

Usage

newcompassRose(
  x,
  y,
  rot = 0,
  cex = 1,
  col = "black",
  col.arrows.light = "white",
  col.arrows.dark = "black"
)

Arguments

x  The position of the center of the compass rose in user units.
y  The position of the center of the compass rose in user units.
rot  Rotation for the compass rose in degrees. See Details.
cex  The character expansion to use in the display.
col  The color of text
col.arrows.light  The color of lighter lines
col.arrows.dark  The color of darker lines

Details

newcompassRose  Display a compass rose
Value

none

Author(s)
modified from Jim Lemon; See compassRose sp

Examples

```r
## Not run:
library(HelpersMG)
require("maps")
map("world", "China")
newcompassRose(x=110, y=35, col.arrows.light="grey")

## End(Not run)
```

newdbeta Density for the Beta distributions.

Description

Density for the Beta distribution with parameters mu and v or shape1 and shape2 (and optional non-centrality parameter ncp).

Usage

```r
ewdbeta(x, mu = NULL, v = NULL, shape1, shape2, ncp = 0, log = FALSE)
```

Arguments

- `x`: vector of quantiles.
- `mu`: mean of the Beta distribution.
- `v`: variance of the Beta distribution.
- `shape1`: non-negative parameters of the Beta distribution.
- `shape2`: non-negative parameters of the Beta distribution.
- `ncp`: non-centrality parameter.
- `log`: logical; if TRUE, probabilities p are given as log(p).

Details

newdbeta returns the density for the Beta distributions

The Beta distribution with parameters shape1 = a and shape2 = b has density $\frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}x^{a-1}(1-x)^{b-1}$ for $a > 0$, $b > 0$ and $0 \leq x \leq 1$ where the boundary values at $x=0$ or $x=1$ are defined as by continuity (as limits).

The mean is $a/(a+b)$ and the variance is $ab/((a+b)^2 (a+b+1))$. These moments and all distributional properties can be defined as limits.
addScale {newdbeta} gives the density for the Beta distributions

Author(s)
Marc Girondot

Examples

```R
pi <- rbeta(100, shape1=0.48, shape2=0.12)
hist(pi, freq=FALSE, breaks=seq(from=0, to=1, by=0.1), ylim=c(0, 8), las=1)
library("HelpersMG")
mx <- ScalePreviousPlot()$ylim["end"]/
max(newdbeta(seq(from=0.01, to=0.99, by=0.01), mu = 0.8, v=0.1))
curve(newdbeta(x, mu = 0.8, v=0.1)*mx, add=TRUE, col="red")
```
Details

newmap.scale Add Scale to Existing Unprojected Map

Value

The exact calculated scale is returned.

Author(s)

See map.scale maps

Examples

```r
## Not run:
library("maps")
library("HelpersMG")
map("world", "China")
newmap.scale(col.line = "red", col="blue")
## End(Not run)
```

Description

This function plots the data as a network. It returns an invisible object that can be used with visNetwork from package visNetwork. [https://fr.wikipedia.org/wiki/Iconographie_des_corrélations](https://fr.wikipedia.org/wiki/Iconographie_des_corrélations)

Usage

```r
## S3 method for class 'IconoCorel'
plot(
  x,
  ..., 
  show.legend.direction = "bottomright",
  show.legend.strength = "topleft",
  title = "Correlation iconography",
  vertex.label.color = "black",
  vertex.label = NULL,
  vertex.color = "white",
  plot = TRUE
)
```
Arguments

- **x** The correlation matrix to show
- ... other options of plot.igraph()
- **show.legend.direction** the position of the legend of direction; FALSE to not show it
- **show.legend.strength** the position of the legend with intensity of correlation; FALSE to not show it
- **title** the title of the plot
- **vertex.label.color** a vector with the colors of labels
- **vertex.label** a vector with the labels
- **vertex.color** a vector of colors
- **plot** if TRUE, the plot is shown

Details

plot.IconoCorel checks and corrects the dataframe to be used with IC_threshold_matrix

Value

A igraph object

Author(s)

Marc Girondot

References


See Also

Other Iconography of correlations: IC_clean_data(), IC_correlation_simplify(), IC_threshold_matrix()

Examples

```r
## Not run:
library("HelpersMG")
es <- matrix(c("e1", "52", "12", "12", "5",
    "e2", "59", "12.5", "9", "5",
    "e3", "55", "13", "15", "9",
    "e4", "58", "14.5", "5", "5",
    "e5", "66", "15.5", "11", "13.5",
    "e6", "62", "16", "15", "18",
    "e7", "63", "17", "12", "18",
    "e8", "69", "18", "9", "18"), ncol=5, byrow = TRUE)
colnames(es) <- c("Élève", "Poids", "Âge", "Assiduité", "Note")
```
es <- as.data.frame(es, stringsAsFactor=FALSE)
es[, 2] <- as.numeric(as.character(es[, 2]))
es[, 3] <- as.numeric(as.character(es[, 3]))
es[, 4] <- as.numeric(as.character(es[, 4]))
es[, 5] <- as.numeric(as.character(es[, 5]))

es
df <- IC_clean_data(es, debug = TRUE)
cor_matrix <- IC_threshold_matrix(data=df, threshold = NULL, progress=FALSE)
cor_threshold <- IC_threshold_matrix(data=df, threshold = 0.3)
ap(mar=c(7,1,1,1))
set.seed(4)
library("igraph")
library("visNetwork")
kk <- plot(cor_threshold, vertex.color="red")
# it can be shown also with the visNetwork package
visIgraph(kk)
cor_threshold_Note <- IC_correlation_simplify(matrix=cor_threshold, variable="Note")
plot(cor_threshold_Note)

## End(Not run)

plot.LD50

Plot results of LD50() that best describe LD50

Description

Plot the estimates that best describe lethality of doses.

Usage

## S3 method for class 'LD50'
plot(
x,
...
las.x = 1,
las.y = 1,
lab.PT = "LD50",
at = NULL,
lab.TRD = paste0("Transitional range of doses l=" , l * 100, "%"),
col.TRD = "gray",
col.TRD.CI = rgb(0.8, 0.8, 0.8, 0.5),
col.PT.CI = rgb(0.8, 0.8, 0.8, 0.5),
show.CI = TRUE
)
plot.LD50

Arguments

x A result file generated by IC50()
...
las.x las parameter for x axis
las.y las parameter for y axis
lab.PT Label to describe pivotal dose
at Position of ticks in x-axis
lab.TRD Label to describe transitional range of dose
col.TRD The color of TRD
col.TRD.CI The color of CI of TRD based on range.CI
col.PT.CI The color of CI of PT based on range.CI
show.CI Do the CI for the curve should be shown

Details

plot.LD50 plot result of IC50() that best describe IC50

Value

Nothing

Author(s)

Marc Girondot

See Also

Other LD50 functions: LD50_MHmcmc_p(), LD50_MHmcmc(), LD50(), logLik.LD50(), predict.LD50()

Examples

## Not run:
data <- data.frame(Doses=c(80, 120, 150, 150, 180, 200),
                   Alive=c(10, 12, 8, 6, 2, 1),
                   Dead=c(0, 1, 5, 6, 9, 15))
LD50_logistic <- LD50(data, equation="logistic")
predict(LD50_logistic, doses=c(140, 170))
plot(LD50_logistic, xlim=c(0, 300))

## End(Not run)
plot.mcmcComposite  
Plot the result of a mcmcComposite object

Description
Plot the results within a mcmcComposite object.
If scale.prior is TRUE, another scale is shown at right.
legend can take these values:
FALSE, TRUE, topleft, topright, bottomleft, bottomright, c(x=, y=)

Usage
## S3 method for class 'mcmcComposite'
plot(
  x,
  ...,  
  chain = 1,  
  parameters = 1,  
  transform = NULL,  
  scale.prior = TRUE,  
  legend = "topright",  
  ylab = "Posterior density",  
  las = 1,  
  show.prior = TRUE,  
  col.prior = "red",  
  lty.prior = 1,  
  lwd.prior = 1,  
  col.posterior = "white",  
  lty.posterior = 1,  
  lwd.posterior = 1,  
  ylab.prior = "Prior density"  
)

Arguments
x  A mcmcComposite object
...  Graphical parameters to be sent to hist()
chain  The chain to use
parameters  Name of parameters or "all"
transform  Function to be used to transform the variable
scale.prior  If TRUE, the prior is scaled at the same size as posterior
legend  If FALSE, the legend is not shown; see description
ylab  y-label for posterior
las  las parameter (orientation of y-axis graduation)
plot.mcmcComposite

- `show.prior`: Whether the prior should be shown?
- `col.prior`: Color for prior curve
- `lty.prior`: Type of line for prior curve
- `lwd.prior`: Width of line for prior curve
- `col.posterior`: Color for posterior histogram
- `lty.posterior`: Type of line for posterior histogram
- `lwd.posterior`: Width of line for posterior histogram
- `ylab.prior`: y-label for prior

**Details**

`plot.mcmcComposite` plots the result of an MCMC search.

**Value**

None

**Author(s)**

Marc Girondot

**See Also**

Other `mcmcComposite` functions: `MHalgoGen()`, `as.mcmc.mcmcComposite()`, `as.parameters()`, `as.quantiles()`, `merge.mcmcComposite()`, `summary.mcmcComposite()`

**Examples**

```r
# Not run:
library(HelpersMG)
require(coda)
x <- rnorm(30, 10, 2)
dnormx <- function(data, x) {
  data <- unlist(data)
  return(-sum(dnorm(data, mean=x['mean'], sd=x['sd'], log=TRUE)))
}
parameters_mcmc <- data.frame(Density=c('dnorm', 'dlnorm'),
  Prior1=c(10, 0.5), Prior2=c(2, 0.5), SDProp=c(1, 1),
  Min=c(-3, 0), Max=c(100, 10), Init=c(10, 2), stringsAsFactors = FALSE,
  row.names=c('mean', 'sd'))
mcmc_run <- MHalgoGen(n.iter=50000, parameters=parameters_mcmc, data=x,
  adaptive = TRUE,
  likelihood=dnormx, n.chains=1, n.adapt=100, thin=1, trace=1)
plot(mcmc_run, xlim=c(0, 20))
plot(mcmc_run, xlim=c(0, 10), parameters="sd")
mcmcforcoda <- as.mcmc(mcmc_run)
# heidel.diag(mcmcforcoda)
raftery.diag(mcmcforcoda)
autocorr.diag(mcmcforcoda)
```
acf(mcmcforcoda[[1]],"mean", lag.max=20, bty="n", las=1)
acf(mcmcforcoda[[1]],"sd", lag.max=20, bty="n", las=1)
batchSE(mcmcforcoda, batchSize=100)

# The batch standard error procedure is usually thought to
# be not as accurate as the time series methods used in summary
summary(mcmcforcoda)$statistics[,"Time-series SE"]
summary(mcmc_run)

as.parameters(mcmc_run)
lastp <- as.parameters(mcmc_run, index="last")
parameters_mcmc[,"Init"] <- lastp
# The n.adapt set to 1 is used to not record the first set of parameters
# then it is not duplicated (as it is also the last one for
# the object mcmc_run)
mcmc_run2 <- MHalgoGen(n.iter=50000, parameters=parameters_mcmc, data=x,
  adaptive = TRUE,
  likelihood=dnorm, n.chains=1, n.adapt=1, thin=1, trace=1)
mcmc_run3 <- merge(mcmc_run, mcmc_run2)

# The n.adapt set to 1 is used to not record the first set of parameters
# then it is not duplicated (as it is also the last one for
# the object mcmc_run)
mcmc_run2 <- MHalgoGen(n.iter=50000, parameters=parameters_mcmc, data=x,
  adaptive = TRUE,
  likelihood=dnorm, n.chains=1, n.adapt=1, thin=1, trace=1)

x.1<-rnorm(6000, 2.4, 0.6)
x.2<-rlnorm(10000, 1.3,0.1)
X<-c(x.1, x.2)

hist(X,100,freq=FALSE, ylim=c(0,1.5))
Lnormlnorm <- function(par, val) {
  p <- invlogit(par["p"])
  return(-sum(log(p*dnorm(val, par["m1"], abs(par["s1"]), log = FALSE)+
    (1-p)*dlnorm(val, par["m2"], abs(par["s2"]), log = FALSE))))
}

# Mean 1
m1=2.3; s1=0.5
# Mean 2
m2=1.3; s2=0.1
# proportion of category 1 - logit transform
p=0
par<-c(m1=m1, s1=s1, m2=m2, s2=s2, p=p)
result2<-optim(par, Lnormlnorm, method="BFGS", val=X,
  hessian=FALSE, control=list(trace=1))

lines(seq(from=0, to=5, length=100),
  dnorm(seq(from=0, to=5, length=100),
    result2$par["m1"], abs(result2$par["s1"])), col="red")
plot.mcmcComposite

```r
lines(seq(from=0, to=5, length=100),
     dlnorm(seq(from=0, to=5, length=100),
            result2$par['m2'], abs(result2$par['s2'])), col="green")

p <- invlogit(result2$par['p'])
paste("Proportion of Gaussian data", p)

lines(seq(from=0, to=5, length=100),
     p*dnorm(seq(from=0, to=5, length=100),
            result2$par['m1'], result2$par['s1']) +
     (1-p)*dlnorm(seq(from=0, to=5, length=100),
                 result2$par['m2'], result2$par['s2']), col="blue")

parameters_mcmc <- data.frame(Density=c('dunif', 'dunif', 'dunif', 'dunif', 'dunif'),
                                Prior1=c(0, 0.001, 0, 0.001, -3),
                                Prior2=c(10, 10, 10, 10, 3),
                                SDProp=c(1, 1, 1, 1, 1),
                                Min=c(0, 0.001, 0, 0.001, -3),
                                Max=c(10, 10, 10, 10, 3),
                                Init=result2$par, stringsAsFactors = FALSE,
                                row.names=c('m1', 's1', 'm2', 's2', 'p'))

mcmc_run <- MHalgoGen(n.iter=50000, parameters=parameters_mcmc, val=X,
                       parameters_name = "par",
                       adaptive = TRUE,
                       likelihood=Lnormlnorm, n.chains=1,
                       n.adapt=100, thin=1, trace=100)

plot(mcmc_run, parameters="m1", breaks=seq(from=0, to =10, by=0.1),
     legend=c(x=6, y=0.10))
plot(mcmc_run, parameters="p", transform=invlogit, xlim=c(0,1),
     breaks=seq(from=0, to=1, by=0.01), legend=c(x=0.6, y=0.10))
plot(mcmc_run, parameters="p", xlim=c(-3,3),
     breaks=seq(from=-3, to =3, by=0.05), legend=c(x=1, y = 0.10))

parameters_mcmc <- data.frame(Density=c('dunif', 'dunif', 'dunif', 'dunif', 'dnorm'),
                                Prior1=c(0, 0.001, 0, 0.001, 0.5),
                                Prior2=c(10, 10, 10, 10, 1),
                                SDProp=c(1, 1, 1, 1, 1),
                                Min=c(0, 0.001, 0, 0.001, -3),
                                Max=c(10, 10, 10, 10, 3),
                                Init=result2$par, stringsAsFactors = FALSE,
                                row.names=c('m1', 's1', 'm2', 's2', 'p'))

mcmc_run_pnorm <- MHalgoGen(n.iter=50000, parameters=parameters_mcmc, val=X,
                           parameters_name = "par",
                           adaptive = TRUE,
                           likelihood=Lnormlnorm, n.chains=1,
                           n.adapt=100, thin=1, trace=100)

plot(mcmc_run_pnorm, parameters="m1", breaks=seq(from=0, to =10, by=0.1),
     legend=c(x=6, y=0.10))
plot(mcmc_run_pnorm, parameters="p", transform=invlogit, xlim=c(0,1),
     breaks=seq(from=0, to=1, by=0.01), legend=c(x=0.6, y=0.10))
```
plot.mcmcComposite

plot(x=mcmc_run_pnorm, parameters="p", xlim=c(-3,3),
     breaks=seq(from=-3, to =3, by=0.05), legend=c(x=1, y= 0.10))

# Note that it is more logic to use beta distribution for p as a
# proportion. However p value must be checked to be used in optim
# The use of logit transform can be a problem because it can stuck
# the p value to 1 or 0 during fit.

Lnormlnorm <- function(par, val) {
  p <- par["p"]
  return(-sum(log(p*dnorm(val, par["m1"], abs(par["s1"])), log = FALSE)+
              (1-p)*dlnorm(val, par["m2"], abs(par["s2"])), log = FALSE)))
}

# Example of beta distribution
# Mean is alpha/(alpha+beta)
# Variance is (alpha+beta)/(((alpha+beta)^2*(alpha+beta+1))
alpha = 5
beta = 9
plot(x = seq(0.0001, 1, by = .0001),
     y = dbeta(seq(0.0001, 1, by = .0001), alpha, beta),
     type = "l", ylab="Density", xlab="p", bty="n")
points(x=alpha/(alpha+beta), y=0, pch=4)
segments(x0=alpha/(alpha+beta)-sqrt((alpha*beta)/((alpha+beta)^2*(alpha+beta+1))),
         x1=alpha/(alpha+beta)+sqrt((alpha*beta)/((alpha+beta)^2*(alpha+beta+1))),
         y0=0, y1=0)

# Use of optim with L-BFGS-B to limit p between 0 and 1 and s > 0
# Mean 1
m1=2.3; s1=0.5
# Mean 2
m2=1.3; s2=0.1
# proportion of category 1 - logit transform
p=0.5

par <- c(m1=m1, s1=s1, m2=m2, s2=s2, p=p)
result2 <- optim(par, Lnormlnorm, method="L-BFGS-B", val=X,
                 lower = c(-Inf, 0, -Inf, 0, 0),
                 upper = c(Inf, Inf, Inf, Inf, 1),
                 hessian=FALSE, control=list(trace=1))

parameters_mcmc <- data.frame(Density=c("dunif", "dunif", "dunif", "dunif", "dbeta"),
                                Prior1=c(0, 0.001, 0, 0.001, 5),
                                Prior2=c(10, 10, 10, 10, 9),
                                SDProp=c(1, 1, 1, 1, 1),
                                Min=c(0, 0.001, 0, 0.001, 0),
                                Max=c(10, 10, 10, 10, 1),
                                Init=c("m1" = 2.4,
                                       "s1" = 0.6,
mcmc_run_pbeta <- MHalgoGen(n.iter=50000, parameters=parameters_mcmc, val=X,
parameters_name = "par",
adaptive = TRUE,
likelihood=Lnormlnorm, n.chains=1,
n.adapt=100, thin=1, trace=100)
plot(mcmc_run_pbeta, parameters="m1", breaks=seq(from=0, to =10, by=0.1),
legend=c(x=6, y=0.10))
plot(mcmc_run_pbeta, parameters="p", xlim=c(0,1),
breaks=seq(from=0, to=1, by=0.01), legend=c(x=0.6, y=2))

### End(Not run)

---

### plot_add

Add a plot to a previous one

---

**Description**

To plot data, just add use it as a normal plot. It will plot the new data without axes, or labels for axes.

This function is complementary to matlines() and matpoints() from package graphics.

**Usage**

plot_add(...)

**Arguments**

... Parameters for plot()

**Details**

plot_add adds a plot to a previous one

**Value**

Nothing

**Author(s)**

Marc Girondot
See Also

Other plot and barplot functions: `ScalePreviousPlot()`, `barplot_errbar()`, `plot_errbar()`

Examples

```r
## Not run:
plot(x=1:100, y=sin(1:100), type="l", bty="n", xlim=c(1,200), xlab="x", ylab="y")
plot_add(x=1:200, y=cos(1:200), type="l", bty="n", col="red")
## End(Not run)
```

---

**plot_errbar**

*Plot a xy graph with error bar on x and/or y*

**Description**

To plot data, just use it as a normal plot but add the `errbar.x` and `errbar.y` values or `errbar.x.minus`, `errbar.x.plus` if bars for x axis are asymmetric and `errbar.y.minus`, `errbar.y.plus` if bars for y axis are asymmetric. Use `x.plus`, `x.minus`, `y.plus` and `y.minus` to set absolute limits for error bars. Note that `x.plus` and `x.minus` have priority over `errbar.x`, `errbar.x.minus` and `errbar.x.plus` and that `y.plus` and `y.minus` have priority over `errbar.y`, `errbar.y.minus` and `errbar.y.plus`.

The parameter `errbar.y.polygon=TRUE` permits to define error as an envelop for y axis.

**Usage**

```r
plot_errbar( 
  ..., 
  errbar.x = NULL, 
  errbar.y = NULL, 
  errbar.x.plus = NULL, 
  errbar.x.minus = NULL, 
  errbar.y.plus = NULL, 
  errbar.y.minus = NULL, 
  x.plus = NULL, 
  x.minus = NULL, 
  y.plus = NULL, 
  y.minus = NULL, 
  errbar.tick = 1/50, 
  errbar.lwd = par("lwd"), 
  errbar.lty = par("lty"), 
  errbar.col = par("fg"), 
  errbar.y.polygon = FALSE, 
  errbar.y.polygon.list = list(NULL), 
  add = FALSE 
)
```
**plot_errbar**

**Arguments**

- **...** Parameters for plot() such as main= or ylim=
- **errbar.x** The length of error bars for x. Recycled if necessary.
- **errbar.y** The length of error bars for y. Recycled if necessary.
- **errbar.x.plus** The length of positive error bars for x. Recycled if necessary.
- **errbar.x.minus** The length of negative error bars for x. Recycled if necessary.
- **errbar.y.plus** The length of positive error bars for y. Recycled if necessary.
- **errbar.y.minus** The length of negative error bars for y. Recycled if necessary.
- **x.plus** The absolute position of the positive error bar for x. Recycled if necessary.
- **x.minus** The absolute position of the negative error bar for x. Recycled if necessary.
- **y.plus** The absolute position of the positive error bar for y. Recycled if necessary.
- **y.minus** The absolute position of the negative error bar for y. Recycled if necessary.
- **errbar.tick** Size of small ticks at the end of error bars defined as a proportion of total width or height graph size.
- **errbar.lwd** Error bar line width, see par("lwd")
- **errbar.lty** Error bar line type, see par("lwd")
- **errbar.col** Error bar line color, see par("col")
- **errbar.y.polygon** If true, the errors are shown as a filled polygon.
- **errbar.y.polygon.list** List of parameters to be used for polygon.
- **add** If true, add the graph to the previous one.

**Details**

plot_errbar plot a xy graph with error bar on x and/or y

**Value**

Nothing

**Author(s)**

Marc Girondot

**See Also**

barplot_errorbar

Other plot and barplot functions: ScalePreviousPlot(), barplot_errbar(), plot_add()
Examples

```r
## Not run:
plot_errbar(1:100, rnorm(100, 1, 2),
  xlab="axe x", ylab="axe y", bty="n", xlim=c(1,100),
  errbar.x=2, errbar.y=rnorm(100, 1, 0.1))
x <- 1:100
plot_errbar(x=x, rnorm(100, 1, 2),
  xlab="axe x", ylab="axe y", bty="n", xlim=c(1,100),
  x.minus=x-2, x.plus=x+2)
x <- 1:100
plot_errbar(x=x, rnorm(100, 1, 2),
  xlab="axe x", ylab="axe y", bty="n",
  pch=21, bg="white",
  x.minus=x-10, x.plus=x+10)
x <- (1:200)/10
y <- sin(x)
plot_errbar(x=x, y=y, xlab="axe x", ylab="axe y", bty="n", xlim=c(1,20),
  y.minus=y-1, y.plus=y+1, ylim=c(-3, 3), type="l",
  errbar.y.polygon=TRUE,
  errbar.y.polygon.list=list(border=NA, col=rgb(0, 0, 0, 0.5)))
## End(Not run)
```

---

**predict.LD50**

Estimate survival according to doses

**Description**

Estimate survival according to doses.
The returned data.frame has the following components:
doses, SE, survival, CI.minus.sexratio, CI.plus.sexratio, range.CI

**Usage**

```r
## S3 method for class 'LD50'
predict(
  object,
  doses = NULL,
  SE = NULL,
  range.CI = 0.95,
  replicates = 1000,
  progressbar = FALSE,
  ...
)
```
Arguments

- **object**: A result file generated by LD50
- **doses**: A vector of temperatures
- **SE**: The standard error for doses, optional
- **range.CI**: The range of confidence interval for estimation, default=0.95
- **replicates**: Number of replicates to estimate CI
- **progressbar**: Logical. Does a progression bar must be shown
- **...**: Not used

Details

predict.LD50 Estimate survival according to doses

Value

A data.frame with informations about survival

Author(s)

Marc Girondot

See Also

Other LD50 functions: LD50_MHmcmc_p(), LD50_MHmcmc(), LD50(), logLik.LD50(), plot.LD50()

Examples

```r
## Not run:
data <- data.frame(Doses=c(80, 120, 150, 150, 180, 200),
Alive=c(10, 12, 8, 6, 2, 1),
Dead=c(0, 1, 5, 6, 9, 15))
LD50_logistic <- LD50(data, equation="logistic")
predict(LD50_logistic, doses=c(140, 170))
plot(LD50_logistic)
## End(Not run)
```

pSnbinom

Distribution function for the sum of random variable with negative binomial distributions.

Description

Distribution function for the sum of random variable with negative binomial distributions.
Usage

pSnbinom(
  q = stop("At least one quantile must be provided"),
  size = NULL,
  prob = NULL,
  mu = NULL,
  lower.tail = TRUE,
  log.p = FALSE,
  tol = 1e-06
)

Arguments

q     vector of quantiles.
size  target for number of successful trials, or dispersion parameter (the shape parameter of the gamma mixing distribution). Must be strictly positive, need not be integer.
prob  probability of success in each trial. 0 < prob <= 1.
mu    alternative parametrization via mean.
lower.tail  logical; if TRUE (default), probabilities are P[X <= x], otherwise, P[X > x].
log.p  logical; if TRUE, probabilities p are given as log(p).
tol   Tolerance for recurrence

Details

pSnbinom returns the distribution function for the sum of random variable with negative binomial distributions

Value

pSnbinom returns distribution function

Author(s)

Marc Girondot

See Also

Other Distribution of sum of random variable with negative binomial distributions: dSnbinom(), qSnbinom(), rSnbinom()

Examples

```r
## Not run:
alpha <- c(2.1, 2.05, 2)
mu <- c(10, 30, 20)
p <- pSnbinom(q=10, size=alpha, mu=mu, lower.tail = TRUE)
```
qSnbinom  

Quantile function for the sum of random variable with negative binomial distributions.

Description

Quantile function for the sum of random variable with negative binomial distributions.

Usage

qSnbinom(
  p = stop("At least one probability must be provided"),
  size = stop("size parameter is mandatory"),
  prob = NULL,
  mu = NULL,
  lower.tail = TRUE,
  log.p = FALSE,
  tol = 1e-06
)

Arguments

  p  vector of probabilities.
  size target for number of successful trials, or dispersion parameter (the shape parameter of the gamma mixing distribution). Must be strictly positive, need not be integer.
  prob probability of success in each trial. 0 < prob <= 1.
  mu alternative parametrization via mean.
  lower.tail logical; if TRUE (default), probabilities are P[X <= x], otherwise, P[X > x].
  log.p logical; if TRUE, probabilities p are given as log(p).
  tol Tolerance for recurrence

Details

qSnbinom returns the quantile function for the sum of random variable with negative binomial distributions

Value

qSnbinom returns quantile function

Author(s)

Marc Girondot
qvlmer

Quasi Variances for lmer Model Coefficients

Description

Computes a set of quasi variances (and corresponding quasi standard errors) for estimated model coefficients relating to the levels of a categorical (i.e., factor) explanatory variable. For details of the method see Firth (2000), Firth (2003) or Firth and de Menezes (2004). Quasi variances generalize and improve the accuracy of “floating absolute risk” (Easton et al., 1991). This device for economical model summary was first suggested by Ridout (1989).

Modified from qvcac.lm() of packages qvcalc by David Firth, d.firth@warwick.ac.uk

Usage

qvlmer(object, factorname = NULL, coef.indices = NULL, dispersion = NULL, ...)
Author(s)

marc.girondot@u-psud.fr

References


Examples

```r
## Not run:
x <- rnorm(100)
y <- rnorm(100)
G <- as.factor(sample(c("A", "B", "C", "D"), 100, replace = TRUE))
R <- as.factor(rep(1:25, 4))
library(lme4)
m <- lmer(y ~ x + G + (1 | R))
qvlmer(m, factorname="G")
## End(Not run)
```

r2norm

Random generation for Gaussian distributions different at left and right

Description

Random generation for Gaussian distributions different at left and right

Usage

`r2norm(n, mean = 0, sd_low = 1, sd_high = 1)`
RandomFromHessianOrMCMC

Arguments

- **n**: number of observations.
- **mean**: vector of means.
- **sd_low**: vector of standard deviations below the mean.
- **sd_high**: vector of standard deviations above the mean.

Details

`r2norm` returns random numbers for Gaussian distributions different at left and right.

Value

`r2norm` returns random numbers.

Author(s)

Marc Girondot

Examples

```r
## Not run:
n <- r2norm(1000, mean=25, sd_low=2, sd_high=10)
hist(n)
## End(Not run)
```

Description

A data.frame with one column for each parameter.

Usage

```r
RandomFromHessianOrMCMC(
  Hessian = NULL,
  mcmc = NULL,
  chain = 1,
  fitted.parameters = NULL,
  fixed.parameters = NULL,
  probs = c(0.025, 0.5, 0.975),
  replicates = 10000,
  fn = NULL,
  ...
)
```
RandomFromHessianOrMCMC

Arguments

- **Hessian**: An Hessian matrix
- **mcmc**: A result from MHallogen()
- **chain**: MCMC chain to be used
- **fitted.parameters**: The fitted parameters
- **fixed.parameters**: The fixed parameters
- **probs**: Probability for quantiles
- **replicates**: Number of replicates to generate
- **fn**: The function to apply to each replicate
- ... Parameters send to fn function

Details

RandomFromHessianOrMCMC returns random numbers based on Hessian matrix or MCMC

Value

Returns a list with two data.frames named df_random and df_fn

Author(s)

Marc Girondot

Examples

```R
## Not run:
library(HelpersMG)
val <- rnorm(100, mean=20, sd=5)+(1:100)/10
# Return -ln L of values in val in Gaussian distribution with mean and sd in par
fitnorm <- function(par, data) {
  -sum(dnorm(data, par["mean"], abs(par["sd"])), log = TRUE))
}
# Initial values for search
p<-c(mean=20, sd=5)
# fit the model
result <- optim(par=p, fn=fitnorm, data=val, method="BFGS", hessian=TRUE)
# Using Hessian
df <- RandomFromHessianOrMCMC(Hessian=result$hessian, fitted.parameters=result$par)$df_random
hist(df[, 1], main="mean")
hist(df[, 2], main="sd")
plot(df[, 1], df[, 2], xlab="mean", ylab="sd", las=1, bty="n")

# Using MCMC
parameters_mcmc <- data.frame(Density=c('dnorm', 'dlnorm'),
Prior1=c(10, 0.5), Prior2=c(2, 0.5), SDProp=c(0.35, 0.2),
Min=c(-3, 0), Max=c(100, 10), Init=c(10, 2), stringsAsFactors = FALSE,
```
# Use of trace and traceML parameters
# trace=1 : Only one likelihood is printed
mcmc_run <- MHalgoGen(n.iter=50000, parameters=parameters_mcmc, data=val,
parameters_name = "par",
likelihood=fitnorm, n.chains=1, n.adapt=100, thin=1, trace=1)
df <- RandomFromHessianOrMCMC(mcmc=mcmc_run, fitted.parameters=NULL)$df_random
hist(df[, 1], main="mean")
hist(df[, 2], main="sd")
plot(df[, 1], df[, 2], xlab="mean", ylab="sd", las=1, bty="n")

# Using a function
fitnorm <- function(par, data, x) {
  y=par["a"]*x+par["b"]
  -sum(dnorm(data, y, abs(par["sd"])), log = TRUE)
}
p<-c(a=0.1, b=20, sd=5)
# fit the model
x <- 1:100
result <- optim(par=p, fn=fitnorm, data=val, x=x, method="BFGS", hessian=TRUE)
# Using Hessian
df <- RandomFromHessianOrMCMC(Hessian=result$hessian, fitted.parameters=result$par,
  fn=function(par) (par["a"]*x+par["b"]))
plot(1:100, val)
lines(1:100, df$quantile["50%", ])
lines(1:100, df$quantile["2.5%", ], lty=2)
lines(1:100, df$quantile["97.5%", ], lty=2)

## End(Not run)

---

**read_folder**

*Read files present in a folder and creates a list with the content of these files*

### Description

To create a list, the syntax is:
```r
datalist <- read_folder(folder=".", read=read.delim, header=FALSE)
```
It returns an error if the folder does not exist.
The names of the elements of the list are the filenames.
The parameter file can be used to predefine a list of file. If file is NULL, all the files of the
folder/directory are used.

### Usage

```r
read_folder(
  folder = try(file.choose(), silent = TRUE),
  file = NULL,
  wildcard = ".*",
)```

read = read.delim,
...
)

Arguments

folder Where to search for files; can be or a file path or a folder path
file list of files
wildcard Define which files are to be read (examples: "*.", "*.xls", "essai*.txt"). It can be also a vector with all filenames.
read Function used to read file. Ex: read.delim or read.xls from gdata package
...
Parameters send to the read function

Details

read_folder reads all files present in a folder

Value

Return a list with the data in the files of the folder (directory for windows users)

Author(s)

Marc Girondot

Examples

## Not run:
library(HelpersMG)
# Read all the .csv files from the current folder/directory
ccontentaslist <- read_folder(folder=".", wildcard="*.csv", read=read.csv2)
# Read all the files from the current folder/directory
ccontentaslist <- read_folder(folder=".", wildcard=".*", read=read.csv2)
# Read two files from the current folder/directory
cfiles <- c("filename1.csv", "filename2.csv")
ccontentaslist <- read_folder(folder=".", wildcard=files, read=read.csv2)
## End(Not run)

---

RectangleRegression  Return parameters of rectangle regression

Description

Fit a line using least rectangle method.
Usage

RectangleRegression(
  x1,
  x2,
  replicate = 1000,
  x1new = seq(from = min(x1), to = max(x1), length.out = 100)
)

Arguments

x1 The first series of data
x2 The second series of data
replicate Number of replicates for bootstrap
x1new Values for x1 to generate x2

Details

RectangleRegression performs rectangle regression

Value

A list with parameters of rectangle regression

Author(s)

Marc Girondot

Examples

x1 <- runif(100, min=10, max=20)
x2 <- runif(100, min=10, max=20)+x1
rectreg <- RectangleRegression(x1, x2)

plot(x=x1, y=x2, bty="n", las=1, xlim=c(10, 20), ylim=c(20, 40))
abline(a=rectreg$par["Intercept"], b=rectreg$par["Slope"], lwd=2)
par(xpd=FALSE)
lines(rectreg$x2new["x1new", ], rectreg$x2new["50%", ], lty=2)
lines(rectreg$x2new["x1new", ], rectreg$x2new["2.5%", ], lty=2)
lines(rectreg$x2new["x1new", ], rectreg$x2new["97.5%", ], lty=2)
abline(a=rectreg$Intercept[1], b=rectreg$Slope[3], col="red")
abline(a=rectreg$Intercept[3], b=rectreg$Slope[1], col="red")
rSnbinom

Random generation for the sum of random variable with negative binomial distributions.

Description

Random numbers for the sum of random variable with negative binomial distributions.

Usage

rSnbinom(n = 1, size = NULL, prob = NULL, mu = NULL)

Arguments

- **n**: number of observations.
- **size**: target for number of successful trials, or dispersion parameter (the shape parameter of the gamma mixing distribution). Must be strictly positive, need not be integer.
- **prob**: probability of success in each trial. 0 < prob <= 1.
- **mu**: alternative parametrization via mean.

Details

rSnbinom returns random numbers for the sum of random variable with negative binomial distributions

Value

rSnbinom returns random number

Author(s)

Marc Girondot

See Also

Other Distribution of sum of random variable with negative binomial distributions: dSnbinom(), pSnbinom(), qSnbinom()

Examples

```r
## Not run:
alpha <- c(2.1, 2.05, 2)
mu <- c(10, 30, 20)
rep <- 100000
distEmpirique <- rSnbinom(n=rep, size=alpha, mu=mu)
tabledistEmpirique <- rep(0, 301)
names(tabledistEmpirique) <- as.character(0:300)
```
tableDistEmpirique <- table(distEmpirique)/rep

plot(0:300, dSnbinom(0:300, size=alpha, mu=mu), type="h", bty="n", xlab="x", ylab="Density", ylim=c(0, 0.02)
plot_add(0:300, tableDistEmpirique, type="l", col="red")
legend(x=200, y=0.02, legend=c("Empirical", "Theoretical"), text.col=c("red", "black"), bty="n")

## End(Not run)

ScalePreviousPlot

Return the scale of the previous plot

Description
Return a list with the limits of the previous plot, the center, the range, and the position of label on this axe.

Usage
ScalePreviousPlot()

Details
ScalePreviousPlot returns the scale of the previous plot

Value
A list with xlim and ylim

Author(s)
Marc Girondot

See Also
Other plot and barplot functions: barplot_errbar(), plot_add(), plot_errbar()

Examples
## Not run:
par(xaxs="i", yaxs="i")
plot(x=1:100, y=sin(1:100), type="l", bty="n", xlim=c(1, 200), xlab="x", ylab="y")
xlim = ScalePreviousPlot()$xlim[1:2]
ylim = ScalePreviousPlot()$ylim[1:2]
par(xaxs="r", yaxs="i")
plot(x=1:100, y=sin(1:100), type="l", bty="n", xlim=c(1, 200), xlab="x", ylab="y")
xlim = ScalePreviousPlot()$xlim[1:2]
ylim = ScalePreviousPlot()$ylim[1:2]
# Here is an example of the use of the label output
SEfromHessian

Standard error of parameters based on Hessian matrix

Description

Standard error of parameters based on Hessian matrix.
The strategy is as follow:
First it tries to inverse the Hessian matrix. If it fails, it uses the near positive definite matrix of the Hessian.
So now the inverse of the Hessian matrix can be computed.
The diagonal of the inverse of the Hessian matrix is calculated. If all values are positive, the SEs are the square root of the inverse of the Hessian.
If not all values are positive, it will estimate the pseudo-variance matrix based on GILL & King (2004). It necessitates a Cholesky matrix.
If from some reason it fails (for example all SE are 0 in output), then the strategy of Rebonato and Jackel will be used to generate the Cholesky matrix.

Usage

SEfromHessian(a, hessian = FALSE)

Arguments

a An Hessian matrix
hessian If TRUE, return a list with the hessian and SE

Details

SEfromHessian returns standard error of parameters based on Hessian matrix

Value

SEfromHessian returns a vector with standard errors

Author(s)

Marc Girondot
References


Examples

```r
## Not run:
val=rnorm(100, mean=20, sd=5)
# Return -ln L of values in val in Gaussian distribution with mean and sd in par
fitnorm<-function(par, val) {
  -sum(dnorm(val, par["mean"], par["sd"], log = TRUE))
}
# Initial values for search
p<-c(mean=20, sd=5)
# fit the model
result <- optim(par=p, fn=fitnorm, val=val, method="BFGS", hessian=TRUE)
SE <- SEfromHessian(result$hessian)
library(MASS)
fitdistr(val, densfun = "normal")
## End(Not run)
```

series.compare Data series comparison using Akaike weight

Description

This function is used as a replacement of t.test() to not use p-value.

Usage

```r
series.compare(..., criterion = c("BIC", "AIC", "AICc"), var.equal = TRUE)
```

Arguments

| ... | Series of data (at least two or data are in a table with series in different rows) |
| criterion | Which criterion is used for model selection. can be AIC, AICc or BIC |
| var.equal | Should the variances of all series being equal? Default TRUE |

Details

series.compare compares series of data using Akaike weight.

Value

The probability that a single proportion model is sufficient to explain the data
Author(s)
Marc Girondot

References

See Also
Other w-value functions: `compare()`, `contingencyTable.compare()

Examples

```r
## Not run:
library("HelpersMG")
A <- rnorm(100, 10, 2)
B <- rnorm(100, 11.1, 2)
series.compare(A, B, criterion = "BIC", var.equal=TRUE)
B <- B[1:10]
series.compare(A, B, criterion = "BIC", var.equal=TRUE)
A <- rnorm(100, 10, 2)
B <- rnorm(100, 10.1, 2)
C <- rnorm(100, 10.5, 2)
series.compare(A, B, C, criterion = "BIC", var.equal=TRUE)
B <- B[1:10]
series.compare(A, B, criterion = "BIC", var.equal=TRUE)
t.test(A, B, var.equal=TRUE)
# Example with a data.frame
series.compare(t(data.frame(A=c(10, 27, 19, 20, NA), B=c(10, 20, NA, NA, NA))))
# Test in the context of big data
A <- rnorm(10000, 10, 2)
B <- rnorm(10000, 10.1, 2)
series.compare(A, B, criterion = "BIC", var.equal=TRUE)
t.test(A, B, var.equal=TRUE)

w <- NULL
p <- NULL
for (i in 1:1000) {
  A <- rnorm(50000, 10, 2)
  B <- rnorm(50000, 10.01, 2)
  w <- c(w, unname(series.compare(A, B, criterion = "BIC", var.equal=TRUE)[1]))
  p <- c(p, t.test(A, B, var.equal=TRUE)$p.value)
}
layout(mat = 1:2)
par(mar=c(4, 4, 1, 1)+0.4)
hist(p, main="", xlab="", las=1, breaks = (0:20)/20, freq=FALSE, xlab = expression(italic("p")**"-value"))
```
```r
hist(w, main="", xlab=expression(italic(”w”)*”-value"),
freq=FALSE, xlab = expression(italic("w")*"-value"))

x <- seq(from=8, to=13, by=0.1)
pv <- NULL
aw <- NULL
A <- rnorm(100, mean=10, sd=2)
B <- A-2
for (meanB in x) {
  pv <- c(pv, t.test(A, B, var.equal = FALSE)$p.value)
  aw <- c(aw, series.compare(A, B, criterion="BIC", var.equal = FALSE)[1])
  B <- B + 0.1
}

par(mar=c(4, 4, 2, 1)+0.4)
y <- pv
plot(x=x, y=y, type="l", lwd=2,
     bty="n", las=1, xlab="Mean B value (SD = 4)", ylab="Probability", ylim=c(0,1),
     main="")
y2 <- aw
lines(x=x, y=y2, type="l", col="red", lwd=2)

l1 <- which(aw>0.05)[1]
l2 <- max(which(aw>0.05))
aw[l1]
pv[l1]
aw[l2]
pv[l2]

l1 <- which(pv>0.05)[1]
l2 <- max(which(pv>0.05))
aw[l1]
pv[l1]
aw[l2]
pv[l2]

par(xpd=TRUE)
segments(x0=10-1.96*2/10, x1=10+1.96*2/10, y0=1.1, y1=1.1, lwd=2)
segments(x0=10, x1=10, y0=1.15, y1=1.05, lwd=2)
par(xpd=TRUE)
text(x=10.5, y=1.1, labels = "Mean A = 10, SD = 2", pos=4)

v1 <- c(expression(italic("p")*"-value"), expression("based on "*italic("t")*"-test"))
v2 <- c(expression(italic("w")*"-value for A"), expression("and B identical models"))
legend("topright", legend=c(v1, v2),
       y.intersp = 1,
```
similar

Test if two vectors contains the same elements independently of their order

Description

Return TRUE only if all elements of x are present and only once in y.

Usage

similar(x, y, test.names = FALSE)

Arguments

x
A vector with numeric or character elements

y
A vector with numeric or character elements

test.names
Logical. If TRUE, the names of the vector elements must be also identical and unique

Value

A logical TRUE or FALSE

Author(s)

Marc Girondot

Examples

## Not run:
A <- c("A", "B", "C", "D")
B <- c("A", "B", "C", "D")
similar(A, B)
similar(B, A)
A <- c(x="A", y="B", z="C", k="D")
B <- c(x="A", y="B", z="C", l="D")
similar(B, A)
similar(A, B, test.names=TRUE)
A <- c(x="A", y="B", z="C", k="D")
B <- c(x="A", z="C", k="D", y="B")
similar(B, A)
similar(A, B, test.names=TRUE)

## End(Not run)

---

**summary.mcmcComposite**  
*Summarize the result of a mcmcComposite object*

**Description**

Summary for the result of a mcmcComposite object.

**Usage**

```r
## S3 method for class 'mcmcComposite'  
summary(object, chain = NULL, ...)
```

**Arguments**

- `object`: A mcmcComposite object
- `chain`: The chain to use
- `...`: Not used

**Details**

`summary.mcmcComposite` get info on the result of a mcmcComposite object

**Value**

A summary of the result

**Author(s)**

Marc Girondot

**See Also**

Other mcmcComposite functions: `MHalgoGen()`, `as.mcmc.mcmcComposite()`, `as.parameters()`, `as.quantiles()`, `merge.mcmcComposite()`, `plot.mcmcComposite()`
Examples

```r
## Not run:
library(HelpersMG)
require(coda)
x <- rnorm(30, 10, 2)
dnormx <- function(data, x) {
  data <- unlist(data)
  return(-sum(dnorm(data, mean=x['mean'], sd=x['sd'], log=TRUE)))
}
parameters_mcmc <- data.frame(Density=c('dnorm', 'dlnorm'),
                               Prior1=c(10, 0.5), Prior2=c(2, 0.5), SDProp=c(1, 1),
                               Min=c(-3, 0), Max=c(100, 10), Init=c(10, 2), stringsAsFactors = FALSE,
                               row.names=c('mean', 'sd'))
mcmc_run <-MHalgoGen(n.iter=1000, parameters=parameters_mcmc, data=x,
                      likelihood=dnormx, n.chains=1, n.adapt=100, thin=1, trace=1)
plot(mcmc_run, xlim=c(0, 20))
plot(mcmc_run, xlim=c(0, 10), parameters="sd")
mcmcforcoda <- as.mcmc(mcmc_run)
  `r heidel.diag(mcmcforcoda)
  `r raftery.diag(mcmcforcoda)
  autocorr.diag(mcmcforcoda)
  acf(mcmcforcoda[[1]],"mean", lag.max=20, bty="n", las=1)
  acf(mcmcforcoda[[1]],"sd", lag.max=20, bty="n", las=1)
  batchSE(mcmcforcoda, batchSize=100)
  # The batch standard error procedure is usually thought to
  # be not as accurate as the time series methods used in summary
  summary(mcmcforcoda)$statistics,"Time-series SE"
summary(mcmc_run)
as.parameters(mcmc_run)
lastp <- as.parameters(mcmc_run, index="last")
parameters_mcmc[, "Init"] <- lastp
  # The n.adapt set to 1 is used to not record the first set of parameters
  # then it is not duplicated (as it is also the last one for
  # the object mcmc_run)
mcmc_run2 <-MHalgoGen(n.iter=1000, parameters=parameters_mcmc, data=x,
                      likelihood=dnormx, n.chains=1, n.adapt=1, thin=1, trace=1)
mcmc_run3 <- merge(mcmc_run, mcmc_run2)
  # no adaptation, n.adapt must be 0
parameters_mcmc[, "Init"] <- c(mean(x), sd(x))
mcmc_run3 <-MHalgoGen(n.iter=1000, parameters=parameters_mcmc, data=x,
                      likelihood=dnormx, n.chains=1, n.adapt=0, thin=1, trace=1)
```

```r
## End(Not run)
```

sun.info

Estimate the time of sunrise and sunset according to longitude, latitude
and date
Description
Estimate the sun fates according to latitude and date. Can be compared with the function sunrise.set() of package StreamMetabolism.

Usage
sun.info(date, latitude, longitude)

Arguments
date A vector with the time at which sun fates are needed
latitude The latitude at which estimate the sun fates
longitude The longitude at which estimate the sun fates

Details
sun.info estimate the time of sunrise and sunset according to longitude, latitude and date

Value
A data.frame with information about daily sun

Author(s)
Marc Girondot <marc.girondot@u-psud.fr>

References

See Also
Other Periodic patterns of indices: index.periodic(), minmax.periodic(), moon.info(), tide.info()

Examples
## Not run:
# Generate a timeserie of time
date <- seq(from=as.Date("2000-01-01"), to=as.Date("2000-12-31"), by="1 day")
plot(date, sun.info(date, latitude=23, longitude=0)$day.length, bty="n",
     las=1, type="l", xlab="Ordinal days", ylab="Day length in hours")
plot(date, sun.info(date, latitude=23, longitude=0)$sunrise, bty="n",
     las=1, type="l", xlab="Ordinal days", ylab="Sun rise in hours")
## End(Not run)
symbol.Female

Plot a female symbol in the plotting region

Description

Plot a female symbol in the plotting region.

Usage

symbol.Female(centerx, centery, rayonx, lwd = 2, col = "black")

Arguments

centerx
  The x position of the center of the circle
centery
  The y position of the center of the circle
rayonx
  The size of the rayon in the scale of the x axis
lwd
  The width of the line of the symbol
col
  The color of the symbol

Details

symbol.Female plot a female symbol in the plotting region

Value

Nothing

Author(s)

Marc Girondot

See Also

Other Symbol: symbol.Male()

Examples

## Not run:
plot(x=1:2, y=c(10,20), type="n", bty="n", xlab="", ylab="")

rayonx <- 0.01
centerx <- 1.2
centery <- 15

symbol.Male(centerx=centerx, centery = centery, rayonx=rayonx)
symbol.Female(centerx=centerx+0.5, centery = centery, rayonx=rayonx)

rayonx <- 0.03
symbol.Male

Plot a male symbol in the plotting region

Description

Plot a male symbol in the plotting region.

Usage

symbol.Male(centerx, centery, rayonx, lwd = 2, col = "black")

Arguments

centerx  The x position of the center of the circle
centery  The y position of the center of the circle
rayonx   The size of the rayon in the scale of the x axis
lwd      The width of the line of the symbol
col      The color of the symbol

Details

symbol.Male plot a male symbol in the plotting region

Value

Nothing

Author(s)

Marc Girondot

See Also

Other Symbol: symbol.Female()
symmetricize

## Not run:
```r
plot(x=1:2, y=c(10,20), type="n", bty="n", xlab="", ylab="")

rayonx <- 0.01
centerx <- 1.2
centery <- 15

symbol.Male(centerx=centerx, centery = centery, rayonx=rayonx)
symbol.Female(centerx=centerx+0.5, centery = centery, rayonx=rayonx)

rayonx <- 0.03
centerx <- 1.2
centery <- 18

symbol.Male(centerx=centerx, centery = centery, rayonx=rayonx, lwd=3)
symbol.Female(centerx=centerx+0.5, centery = centery, rayonx=rayonx, lwd=3, col="red")

rayonx <- 0.05
centerx <- 1.4
centery <- 13

symbol.Male(centerx=centerx, centery = centery, rayonx=rayonx, lwd=4, col="blue")
symbol.Female(centerx=centerx+0.5, centery = centery, rayonx=rayonx, lwd=4, col="red")
```

## End(Not run)

---

symmetricize  
Make a matrix symmetric

### Description

This function was part of the package ENA. This package is no more available and it cannot be installed from archive because some dependencies are no more available.

### Usage

```r
symmetricize(
  matrix,
  method = c("max", "min", "avg", "ld", "ud"),
  adjacencyList = FALSE
)
```

### Arguments

- **matrix**: The matrix to make symmetric
- **method**: The method to use to make the matrix symmetric. Default is to take the maximum.
• "max" For each position, \( m_{i,j} \), use the maximum of \((m_{i,j}, m_{j,i})\)
• "min" For each position, \( m_{i,j} \), use the minimum of \((m_{i,j}, m_{j,i})\)
• "avg" For each position, \( m_{i,j} \), use the mean: \((m_{i,j} + m_{j,i})/2\)
• "ld" Copy the lower triangular portion of the matrix to the upper triangular portion.
• "ud" Copy the upper triangular portion of the matrix to the lower triangular portion.

adjacencyList Logical. If false, returns the symmetric matrix (the same format as the input). If true, returns an adjacency list representing the upper triangular portion of the adjacency matrix with addressing based on the row.names of the matrix provided.

Details

Make the matrix symmetric by making all "mirrored" positions consistent. A variety of methods are provided to make the matrix symmetrical.

Value

The symmetric matrix

Author(s)

Jeffrey D. Allen <Jeffrey.Allen@UTSouthwestern.edu>

Examples

#Create a sample 3x3 matrix
mat <- matrix(1:9, ncol=3)

#Copy the upper diagonal portion to the lower
symmetricize(mat, "ud")

#Take the average of each symmetric location
symmetricize(mat, "avg")

---

tide.info

Annual tide calendar for one particular location

Description

The script extracts tide information from http://tides.mobilegeographics.com/ into a data.frame. The presence of XLM package is required for this function.
Usage

tide.info(
  file = NULL,
  year = as.POSIXlt(Sys.time())$year + 1900,
  location = 0,
  latitude = NA,
  longitude = NA,
  tz = ""
)

Arguments

  file       An html file from the site http://tides.mobilegeographics.com/
  year       Year to get the calendar
  location   Code based on http://tides.mobilegeographics.com/
  latitude   The latitude of the tide information
  longitude  The longitude of the tide information
  tz         Timezone

Details

tide.info gets the annual tide calendar for one particular location.

Value

  Return a data.frame with tide calendar:
  Level is the tide level, Tide is the High or Low Tide information and Date.Time is the date/time in POSIXlt format.

Author(s)

  Marc Girondot <marc.girondot@u-psud.fr>

See Also

  Other Periodic patterns of indices: index.periodic(), minmax.periodic(), moon.info(), sun.info()

Examples

  ## Not run:
  library("HelpersMG")
  lat <- 5.74
  long <- -54
  Awala2004 <- tide.info(year=2004, longitude=long, latitude=lat, tz="America/Cayenne")
  with(Awala2004, plot(Date.Time, Level, bty="n", las=1, type="l",
    xlab=paste("Year", as.POSIXlt(Date.Time[1])$year+1900),
    ylab="Tide level in m"))

  ## End(Not run)
tnirp  
*Read an ASCII text representation of a named or not vector object*

**Description**
Read an ASCII text representation of a named or not vector object.
Note that `paste0(rev(c("p", "r", "i", "n", "t")), collapse="") = "tnirp"`

**Usage**

```r
tnirp(x, named = TRUE)
```

**Arguments**
- **x**: A string or a vector of strings with value and possibly names.
- **named**: TRUE if names are included.

**Details**

`tnirp` reads an ASCII text representation of a named or not vector object.

**Value**

A vector

**Author(s)**

Marc Girondot

**See Also**

Other Characters: `asc()`, `chr()`, `d()`

**Examples**

```r
A <- structure(runif(26), .Names=letters)
text <- capture.output(A)
tnirp(text)
```

```
4.63215947 10.78627511 0.36108497 0.08292101 -0.52558196 -0.76430859
-0.75186542 -0.57632291 -0.58017174 -0.57048696 -0.56234135 -0.80645122
-0.77752524 -0.80909494 -0.56920540 -0.55317302 0.45757298 -0.64155368
-0.59119637 -0.66006794 -0.66582399 -0.66772684 -0.67351412 -0.68361820
-0.67038245 -0.68938726 -0.68889078 -0.68779016 -0.68604629 -0.68361820
```
universalmclapply

Run the function FUN on X using parallel computing

Description

Return the results of the function FUN applied to X. It uses forking in unix system and not in windows system.

Usage

universalmclapply(
  X,
  FUN,
  ..., 
  mc.cores = parallel::detectCores(),
  mc.preschedule = TRUE,
  clusterExport = list(),
  clusterEvalQ = list(),
  forking = ifelse(.Platform$OS.type == "windows", FALSE, TRUE),
  progressbar = FALSE
)

Arguments

X
A vector (atomic or list) or an expressions vector. Other objects (including classed objects) will be coerced by as.list.

FUN
The function to be applied to each element of X

... Optional arguments to FUN

mc.cores
The number of cores to use, i.e. at most how many child processes will be run simultaneously.

---

```
p1.36  p1.37  p2.09  p2.10  p2.11  p2.12
-0.67045238 -0.66115613 2.55403149 2.31060620 2.31348160 2.20958757
p2.13  p2.14  p2.15  p2.16  p2.17  p2.18
2.14304918 2.19699719 2.38095457 2.18740019 2.3205811 2.31668302
p2.19  p2.20  p2.21  p2.22  p2.23  p2.24
1.99424288 2.06613445 2.38092301 2.40551276 2.31987342 2.30344402
p2.25  p2.26  p2.27  p2.28  p2.29  p2.30
2.26860058 2.25008836 2.23385204 2.22768782 2.25341904 1.77043360
p2.31  p2.32  p2.33  p2.34  p2.35  p2.36
2.21606813 2.21581431 2.21153872 2.21118013 2.21375660 2.21182196
p2.37
1.86137833 
```

```r
tnirp(" 27.89  289.99
90.56", named=FALSE)
```
mc.preschedule  if set to TRUE then the computation is first divided to (at most) as many jobs are there are cores and then the jobs are started, each job possibly covering more than one value. If set to FALSE then one job is forked for each value of X. The former is better for short computations or large number of values in X, the latter is better for jobs that have high variance of completion time and not too many values of X compared to mc.cores.

clusterExport  List of clusterExport parameters as list
clusterEvalQ  List of clusterEvalQ parameters as list
forking  If TRUE will use forking
progressbar  If pbapply package is installed, show a progressbar

Details

universalmclapply runs the function FUN on X using parallel computing

Value

The results of the function FUN applied to X

Author(s)

Marc Girondot

Examples

```r
## Not run:
library(HelpersMG)
x <- 1:1000
funx <- function(y) {
mint <- rep(NA, length(y))
for (i in seq_along(y)) {
k <- rnorm(runif(n = 1, 50, 50), mean=10, sd=2)
mint[i] <- mean(k)
}
mint
tp <- system.time({
m <- universalmclapply(X=x, FUN=funx, forking=FALSE)
})
tp <- system.time({
m <- universalmclapply(X=x, FUN=funx, forking=TRUE)
})

### An example using clusterExport
# Here no error is generated because environment was exported
# However forking is not possible in windows and non parallel code is ran
pp <- runif(100)
x <- 1:100
funx1 <- function(y) {pp[y]*10}
u <- universalmclapply(x, FUN=funx1, forking=TRUE)
```
wget  Download a file from internet and save it locally

Description

Download a file from internet and save it locally. This function is a wrapper for download.files() that keep the name identical and can get several files at once.

Usage

wget(url = stop("At least one internet adress is required"), ...)

Arguments

url The url where to download file

... The parameters send to download.file()
**Details**

wget download a file from internet and save it locally

**Value**

Nothing

**Author(s)**

Marc Girondot

**Examples**

```r
## Not run:
library(HelpersMG)
# Save locally the files send in the parameter url
wget(c("https://cran.r-project.org/web/packages/HelpersMG/HelpersMG.pdf",
       "https://cran.r-project.org/web/packages/embryogrowth/embryogrowth.pdf"))

## End(Not run)
```
Index

*Topic Lunar
moon.info, 69
*Topic Lune
moon.info, 69
*Topic Moon
moon.info, 69
*Topic Tide
tide.info, 110

as.mcmc.mcmcComposite, 5, 8, 9, 60, 63, 79, 104
as.parameters, 6, 7, 9, 60, 63, 79, 104
as.quantiles, 6, 8, 9, 60, 63, 79, 104
asc, 10, 15, 26, 112

barplot_errbar, 11, 84, 85, 98
cArrows, 12
ChangeCoordinate, 14
chr, 11, 15, 26, 112
clean.knitr, 16
compare, 16, 22, 101
compare_AIC, 17, 19, 21, 34, 37
compare_AICc, 18, 19, 21, 34, 37
compare_BIC, 18, 19, 20, 34, 37
contingencyTable.compare, 17, 21, 101
convert.tz, 24
d, 11, 15, 25, 112
Dlx, 26
dSnbinc, 28, 88, 90, 97
duplicate.packages, 30

ellipse, 31
ExtractAIC.glm, 18, 19, 21, 33, 37

flexit, 35, 48, 57
FormatCompareAIC, 18, 19, 21, 34, 36
growlnotify, 37

HelpersMG-package, 3
IC_clean_data, 38, 40, 42, 75
IC_correlation_simplify, 39, 40, 42, 75
IC_threshold_matrix, 39, 40, 41, 75
ind_long_lat, 45
index.periodic, 44, 66, 69, 106, 111
inside.search, 47
invlogit, 36, 48, 57
LD50, 49, 53, 54, 59, 77, 87
LD50_MHmcmc, 50, 51, 54, 59, 77, 87
LD50_MHmcmc_p, 50, 53, 54, 59, 77, 87
list.packages, 55
local.search, 56
logit, 36, 48, 57
logLik.compareAIC, 58
logLik.LD50, 50, 53, 54, 59, 77, 87

merge.mcmcComposite, 6, 8, 9, 60, 63, 79, 104
MHalgoGen, 6, 8, 9, 60, 61, 79, 104
minmax.periodic, 45, 65, 69, 106, 111
modeled.hist, 67
modifyVector, 68
moon.info, 45, 66, 69, 106, 111
MovingWindow, 70

newcompassRose, 71
newdbeta, 72
newmap.scale, 73

plot.IconoCorel, 39, 40, 42, 74
plot.LD50, 50, 53, 54, 59, 76, 87
plot.mcmcComposite, 6, 8, 9, 60, 63, 78, 104
plot_add, 12, 83, 85, 98
plot_errbar, 12, 84, 84, 98
predict.LD50, 50, 53, 54, 59, 77, 86
pSnbinc, 29, 87, 90, 97
qSnbinc, 29, 88, 89, 97
qvlmer, 90
r2norm, 91
RandomFromHessianOrMCMC, 92
read_folder, 94
RectangleRegression, 95
rSnbinom, 29, 88, 90, 97
ScalePreviousPlot, 12, 84, 85, 98
SEfromHessian, 99
series.compare, 17, 22, 100
similar, 103
summary.mcmcComposite, 6, 8, 9, 60, 63, 79, 104
sun.info, 45, 66, 69, 105, 111
symbol.Female, 107, 108
symbol.Male, 107, 108
symmetricize, 109
tide.info, 45, 66, 69, 106, 110
tnirp, 11, 15, 26, 112
universalmclapply, 113
wget, 115