

Package ‘HelpersMG’

February 17, 2022

Type Package

Title Tools for Environmental Analyses, Ecotoxicology and Various R Functions

Version 5.1

Date 2022-02-17

Author Marc Girondot

Maintainer Marc Girondot <marc.girondot@gmail.com>

Depends lme4, coda, R (>= 4.1), MASS, Matrix

Suggests RNetCDF, ncd4, maps, fields, shiny, ppcor, pbmccapply, pbapply, parallel, visNetwork, igraph, shinyWidgets, cranlogs

Description Contains miscellaneous functions useful for managing 'NetCDF' files (see <<https://en.wikipedia.org/wiki/NetCDF>>), get moon phase and time for sun rise and fall, tide level, 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

LazyLoad yes

Encoding UTF-8

RoxygenNote 7.1.2

NeedsCompilation no

Repository CRAN

Date/Publication 2022-02-17 17:52:03 UTC

R topics documented:

HelpersMG-package	4
as.mcmc.mcmcComposite	7
as.parameters	9
as.quantiles	11

asc	12
barplot_errbar	13
cArrows	15
ChangeCoordinate	16
chr	17
compare	18
compare_AIC	19
compare_AICc	20
compare_BIC	22
contingencyTable.compare	23
convert.tz	26
cutter	27
d	35
dbeta_new	36
dcutter	37
dgamma	39
DIx	41
dSnbinom	42
duplicated_packages	45
ellipse	46
ExtractAIC.glm	48
fitdistrquantiles	50
flexit	51
FormatCompareAIC	53
format_ncdf	54
iCutter	56
IC_clean_data	56
IC_correlation_simplify	58
IC_threshold_matrix	60
index.periodic	62
ind_long_lat	64
inside.search	65
invlogit	67
LD50	68
LD50_MHmcmc	70
LD50_MHmcmc_p	73
list.packages	74
local.search	75
logit	76
logLik.compareAIC	77
logLik.cutter	78
logLik.LD50	79
merge.mcmcComposite	80
MHalgoGen	82
minmax.periodic	86
modeled.hist	88
modifyVector	89
moon.info	90

MovingWindow	91
NagelkerkeScaledR2	92
newcompassRose	93
newmap.scale	94
openwd	95
plot.cutter	96
plot.IconoCorel	100
plot.LD50	102
plot.mcmcComposite	104
plot_add	109
plot_errbar	110
predict.LD50	112
print.cutter	113
pSnbinom	117
qSnbinom	119
qvlmer	120
r2norm	121
RandomFromHessianOrMCMC	122
rcutter	125
read_folder	127
RectangleRegression	128
RM_add	129
RM_delete	131
RM_duplicate	132
RM_get	133
RM_list	134
rnbinom_new	136
rSnbinom	137
ScalePreviousPlot	138
SEfromHessian	140
series.compare	141
show_name	144
similar	145
specify_decimal	146
summary.mcmcComposite	147
sun.info	148
symbol.Female	150
symbol.Male	151
symmetricize	152
tide.info	153
tnirp	155
universalmclapply	157
wget	159

HelpersMG-package	<i>Tools for Environmental Analyses, Ecotoxicology and Various R Functions</i>
-------------------	--

Description

Contains miscellaneous 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 their 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, calculate distribution of unobserved values in censored or truncated 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

Package:	HelpersMG
Type:	Package
Version:	5.1 build 1166
Date:	2022-02-17
License:	GPL (>= 2)
LazyLoad:	yes

Author(s)

Marc Girondot <marc.girondot@gmail.com>

Examples

```
## 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)
print('-----')
print('Examples for Daily patterns of temperature')
print('-----')
# 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")

# How many times this package has been download
library(cranlogs)
HelpersMG <- cran_downloads("HelpersMG", from = "2015-04-07",
                           to = Sys.Date() - 1)

sum(HelpersMG$count)
plot(HelpersMG$date, HelpersMG$count, type="l", bty="n")

## End(Not run)

```

as.mcmc.mcmcComposite *Extract mcmc object from a mcmcComposite object*

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)
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.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.

If index="median", the function will return the median value of the parameter.

if index="quantile", the function will return the probs defined by quantiles parameter.

If index="mode", the function will return the mode value of the parameter based on Asselin de Beauville (1978) method.

index can also be a numeric value.

This function uses the complete iterations available except the adaptation part, even if thin parameter is not equal to 1.

Usage

```
as.parameters(x, index = "best", chain = 1, probs = 0.025)
```

Arguments

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

Value

A vector with parameters at maximum likelihood or index position

Author(s)

Marc Girondot

References

Asselin de Beauville J.-P. (1978). Estimation non paramétrique de la densité et du mode, exemple de la distribution Gamma. *Revue de Statistique Appliquée*, 26(3):47-70.

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)
# With index being median, it returns the median value for each parameter
as.parameters(mcmc_run3, index="median")
as.parameters(mcmc_run3, index="mode")
as.parameters(mcmc_run3, index="best")
as.parameters(mcmc_run3, index="quantile", probs=0.025)
as.parameters(mcmc_run3, index="quantile", probs=0.975)
as.parameters(mcmc_run3, index="quantile", probs=c(0.025, 0.975))

## 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

```
## 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=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

```
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: [chr\(\)](#), [d\(\)](#), [tnirp\(\)](#)

Examples

```
asc("abcd")
asc("ABCD")
```

barplot_errbar	<i>Plot a barplot graph with error bar on y</i>
----------------	---

Description

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

Usage

```
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
)
```

Arguments

...	Parameters for <code>barplot()</code> such as <code>main=</code> or <code>ylim=</code>
<code>errbar.y</code>	The length of error bars for y. Recycled if necessary.
<code>errbar.y.plus</code>	The length of positive error bars for y. Recycled if necessary.

<code>errbar.y.minus</code>	The length of negative error bars for y. Recycled if necessary.
<code>y.plus</code>	The absolut position of the positive error bar for y. Recycled if necessary.
<code>y.minus</code>	The absolut position of the nagative error bar for y. Recycled if necessary.
<code>errbar.tick</code>	Size of small ticks at the end of error bars defined as a proportion of total width or height graph size.
<code>errbar.lwd</code>	Error bar line width, see <code>par("lwd")</code>
<code>errbar.lty</code>	Error bar line type, see <code>par("lwd")</code>
<code>errbar.col</code>	Error bar line color, see <code>par("col")</code>
<code>add</code>	If true, add the graph to the previous one.

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\(\)](#), [show_name\(\)](#)

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

Draw curved lines with arrowhead

Description

Draw a curved line with arrowhead.

Usage

```
cArrows(
  x1,
  y1,
  x2,
  y2,
  code = 2,
  size = 1,
  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
)
```

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 straight line, positive or 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 with 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

```
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

```
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

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

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

Value

Nothing

Author(s)

Marc Giron dot

References

Giron dot, M., Guillon, J.-M., 2018. The w-value: An alternative to t- and X2 tests. Journal of Biostatistics & Biometrics 1, 1-3.

See Also

Other w-value functions: [contingencyTable.compare\(\)](#), [series.compare\(\)](#)

Examples

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

## End(Not run)
```

`compare_AIC`*Compares the AIC of several outputs*

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 $2 * \text{factor.value} * \text{value} + 2 * \text{length}(\text{par})$.

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

```
compare_AIC(  
  ...,  
  factor.value = 1,  
  silent = FALSE,  
  FUN = function(x) specify_decimal(x, decimals = 2)  
)
```

Arguments

<code>...</code>	Successive results to be compared as lists.
<code>factor.value</code>	The <code>\$value</code> of the list object is multiplied by <code>factor.value</code> to calculate AIC.
<code>silent</code>	If TRUE, nothing is displayed.
<code>FUN</code>	Function used to show values

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

```
## 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 $2 * \text{factor.value} * \text{logLik}(x) + (2 * \text{attributes}(\text{logLik}(x))\$df * (\text{attributes}(\text{logLik}(x))\$nobs - 2)) / (\text{attributes}(\text{logLik}(x))\$nobs - 2)$

Usage

```
compare_AICc(
  ...,
  factor.value = -1,
  silent = FALSE,
  FUN = function(x) specify_decimal(x, decimals = 2)
)
```

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.
FUN	Function used to show values

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

```
## 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 $2 * \text{factor.value} * \sum(\log\text{Lik}(x)) + \sum(\text{attributes}(\log\text{Lik}(x))\$df) * \log(\text{attributes}(\log\text{Lik}(x))\$nobs)$

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

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

Usage

```
compare_BIC(
  ...,
  factor.value = -1,
  silent = FALSE,
  FUN = function(x) specify_decimal(x, decimals = 2)
)
```

Arguments

<code>...</code>	Successive results to be compared as lists.
<code>factor.value</code>	The <code>\$value</code> of the list object is multiplied by <code>factor.value</code> to calculate BIC.
<code>silent</code>	If TRUE, nothing is displayed.
<code>FUN</code>	Function used to show values

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)
```

contingencyTable.compare

Contingency table comparison using Akaike weight

Description

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

Usage

```
contingencyTable.compare(
  table,
  criterion = c("AIC", "AICc", "BIC"),
  probs = NULL
)
```

Arguments

<code>table</code>	A matrix or a data.frame with series in rows and number of each category in column
<code>criterion</code>	Which criterion is used for model selection
<code>probs</code>	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

Girondot, M., Guillon, J.-M., 2018. The w-value: An alternative to t- and X2 tests. *Journal of Biostatistics & Biometrics* 1, 1-4.

See Also

Other w-value functions: [compare\(\)](#), [series.compare\(\)](#)

Examples

```
## 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
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
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]

aw[l2]
pv[l2]

y[which(y2>0.05)[1]]
y[which(rev(y2)>0.05)[1]]

par(xpd=TRUE)
text(x=25, y=1.15, labels="Group A: 25 type P / 100", pos=1)

```

```
segments(x0=25, y0=0, x1=25, y1=1, lty=3)

# plot(1, 1)

v1 <- c(expression(italic("p")*"-value"), expression("after "*chi^2*" -test"))
v2 <- c(expression(italic("w")*"-value for A"), expression("and B identical models"))
legend("topright", legend=c(v1, v2),
       y.intersp = 1,
       col=c("black", "black", "red", "red"), bty="n", lty=c(1, 0, 1, 0))

segments(x0=0, x1=n, y0=0.05, y1=0.05, lty=2)
text(x=101, y=0.05, labels = "0.05", pos=4)

## End(Not run)
```

`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

<code>x</code>	The date-time in POSIXlt or POSIXct format
<code>tz</code>	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

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

cutter

Distribution of the fitted distribution without cut.

Description

If observations is a data.frame, it can have 4 columns:

A column for the measurements;

A column for the lower detection limit;

A column for the upper detection limit;

A column for the truncated or censored nature of the data.

The names of the different columns are in the observations.colname, lower_detection_limit.colname, upper_detection_limit.colname and cut_method.colname.

If lower_detection_limit.colname is NULL or if the column does not exist, the data are supposed to not be left-cut and if upper_detection_limit.colname is NULL or if the column does not exist, the data are supposed to not be right-cut.

If observations is a vector, then the parameters lower_detection_limit and/or upper_detection_limit must be given. Then cut_method must be also provided.

In observations, -Inf must be used to indicate a value below the lower detection limit and +Inf must be used for a value above the upper detection limit.

Be careful: NA is used to represent a missing data and not a value below or above the detection limit.

If lower_detection_limit, upper_detection_limit or cut_method are only one value, they are supposed to be used for all the observations.

Definitions for censored or truncated distribution vary, and the two terms are sometimes used interchangeably. Let the following data set:

```
1 1.25 2 4 5
```

Censoring: some observations will be censored, meaning that we only know that they are below (or above) some bound. This can for instance occur if we measure the concentration of a chemical in a water sample. If the concentration is too low, the laboratory equipment cannot detect the presence of the chemical. It may still be present though, so we only know that the concentration is below the laboratory's detection limit.

If the detection limit is 1.5, so that observations that fall below this limit is censored, our example data set would become:

```
<1.5 <1.5 2 4 5;
```

that is, we don't know the actual values of the first two observations, but only that they are smaller than 1.5.

Truncation: the process generating the data is such that it only is possible to observe outcomes above (or below) the truncation limit. This can for instance occur if measurements are taken using a detector which only is activated if the signals it detects are above a certain limit. There may be lots of weak incoming signals, but we can never tell using this detector.

If the truncation limit is 1.5, our example data set would become: 2 4 5; and we would not know that there in fact were two signals which were not recorded.

If `n.iter` is `NULL`, no Bayesian MCMC is performed but credible interval will not be available.

Usage

```
cutter(
  observations = stop("Observations must be provided"),
  observations.colname = "Observations",
  lower_detection_limit.colname = "LDL",
  upper_detection_limit.colname = "UDL",
  cut_method.colname = "Cut",
  par = NULL,
  lower_detection_limit = NULL,
  upper_detection_limit = NULL,
  cut_method = "censored",
  distribution = "gamma",
  n.mixture = 1,
  n.iter = 5000,
  debug = FALSE,
  progress.bar = TRUE,
  session = NULL
)
```

Arguments

<code>observations</code>	The observations; see description
<code>observations.colname</code>	If <code>observations</code> is a <code>data.frame</code> , the name of column with observations
<code>lower_detection_limit.colname</code>	If <code>observations</code> is a <code>data.frame</code> , the name of column with lower detection limit
<code>upper_detection_limit.colname</code>	If <code>observations</code> is a <code>data.frame</code> , the name of column with upper detection limit
<code>cut_method.colname</code>	If <code>observations</code> is a <code>data.frame</code> , the name of column with cut method, being "censored" or "truncated"
<code>par</code>	Initial values for parameters of distribution
<code>lower_detection_limit</code>	Value for lower detection limit
<code>upper_detection_limit</code>	Value for upper detection limit
<code>cut_method</code>	Value for cut method, being "censored" or "truncated"
<code>distribution</code>	Can be <code>gamma</code> , <code>normal</code> , <code>weibull</code> , <code>lognormal</code> , or <code>generalized.gamma</code>
<code>n.mixture</code>	Number of distributions
<code>n.iter</code>	Number of iteration for Bayesian MCMC and to estimate the goodness-of-fit

debug	If TRUE, show some information
progress.bar	If TRUE, show a progress bar for MCMC
session	The session of a shiny process

Details

cutter returns the fitted distribution without cut

Value

The parameters of distribution of values below or above the detection limit.

Author(s)

Marc Girondot

See Also

Other Distributions: [dbeta_new\(\)](#), [dcutter\(\)](#), [dggamma\(\)](#), [logLik.cutter\(\)](#), [plot.cutter\(\)](#), [print.cutter\(\)](#), [rcutter\(\)](#)

Examples

```
## Not run:
library(HelpersMG)
# -----
# right censored distribution with gamma distribution
# -----
# Detection limit
DL <- 100
# Generate 100 random data from a gamma distribution
obc <- rgamma(100, scale=20, shape=2)
# remove the data below the detection limit
obc[obc>DL] <- +Inf
# search for the parameters the best fit these censored data
result <- cutter(observations=obc, upper_detection_limit=DL,
                 cut_method="censored")

result
plot(result, xlim=c(0, 150), breaks=seq(from=0, to=150, by=10), col.mcmc=NULL)
plot(result, xlim=c(0, 150), breaks=seq(from=0, to=150, by=10))
# -----
# The same data seen as truncated data with gamma distribution
# -----
obc <- obc[is.finite(obc)]
# search for the parameters the best fit these truncated data
result <- cutter(observations=obc, upper_detection_limit=DL,
                 cut_method="truncated")

result
plot(result, xlim=c(0, 150), breaks=seq(from=0, to=150, by=10))
# -----
# left censored distribution with gamma distribution
```

```

# -----
# Detection limit
DL <- 10
# Generate 100 random data from a gamma distribution
obc <- rgamma(100, scale=20, shape=2)
# remove the data below the detection limit
obc[obc<DL] <- -Inf
# search for the parameters the best fit these truncated data
result <- cutter(observations=obc, lower_detection_limit=DL,
                cut_method="censored")

result
plot(result, xlim=c(0, 200), breaks=seq(from=0, to=300, by=10))
# -----
# left censored distribution with mixture of gamma distribution
# -----
#' # Detection limit
library(HelpersMG)
# Generate 200 random data from a gamma distribution
set.seed(1234)
obc <- c(rgamma(100, scale=10, shape=5), rgamma(100, scale=20, shape=10))
LDL <- 20
l <- seq(from=0, to=LDL, length.out=1001)
p <- pgamma(l, scale=10, shape=5)*0.5+pgamma(l, scale=20, shape=10)
deltal <- l[2]-l[1]
expected_LDL <- sum((l[-1]-deltal/2)*(p[-1]-p[-length(p)]))/sum((p[-1]-p[-length(p)]))
# remove the data below the detection limit
obc[obc<LDL] <- -Inf

UDL <- 300
l <- seq(from=UDL, to=1000, length.out=1001)
p <- pgamma(l, scale=10, shape=5)*0.5+pgamma(l, scale=20, shape=10)
deltal <- l[2]-l[1]
expected_UDL <- sum((l[-1]-deltal/2)*(p[-1]-p[-length(p)]))/sum((p[-1]-p[-length(p)]))
obc[obc>UDL] <- +Inf

# search for the parameters the best fit these truncated data
result1_gamma <- cutter(observations=obc, lower_detection_limit=LDL,
                      upper_detection_limit = UDL,
                      distribution="gamma",
                      cut_method="censored", n.iter=5000, debug=0)
result1_normal <- cutter(observations=obc, lower_detection_limit=LDL,
                       upper_detection_limit = UDL,
                       distribution="normal",
                       cut_method="censored", n.iter=5000)
result1_lognormal <- cutter(observations=obc, lower_detection_limit=LDL,
                           upper_detection_limit = UDL,
                           distribution="lognormal",
                           cut_method="censored", n.iter=5000)
result1_Weibull <- cutter(observations=obc, lower_detection_limit=LDL,
                        upper_detection_limit = UDL,
                        distribution="Weibull",
                        cut_method="censored", n.iter=5000)
result1_generalized.gamma <- cutter(observations=obc, lower_detection_limit=LDL,

```

```

        upper_detection_limit = UDL,
        distribution="generalized.gamma",
        cut_method="censored", n.iter=5000)
result2_gamma <- cutter(observations=obc, lower_detection_limit=LDL,
        upper_detection_limit = UDL,
        distribution="gamma",
        n.mixture=2,
        cut_method="censored", n.iter=5000)
result2_normal <- cutter(observations=obc, lower_detection_limit=LDL,
        upper_detection_limit = UDL,
        distribution="normal",
        n.mixture=2,
        cut_method="censored", n.iter=5000)
result2_lognormal <- cutter(observations=obc, lower_detection_limit=LDL,
        upper_detection_limit = UDL,
        distribution="lognormal",
        n.mixture=2,
        cut_method="censored", n.iter=5000)
result2_Weibull <- cutter(observations=obc, lower_detection_limit=LDL,
        upper_detection_limit = UDL,
        distribution="Weibull",
        n.mixture=2,
        cut_method="censored", n.iter=5000)
result2_generalized.gamma <- cutter(observations=obc, lower_detection_limit=LDL,
        upper_detection_limit = UDL,
        distribution="generalized.gamma",
        n.mixture=2,
        cut_method="censored", n.iter=5000)

compare_AIC(nomixture.gamma=result1_gamma,
            nomixture.normal=result1_normal,
            nomixture.lognormal=result1_lognormal,
            nomixture.Weibull=result1_Weibull,
            nomixture.generalized.gamma=result1_generalized.gamma,
            mixture.gamma=result2_gamma,
            mixture.normal=result2_normal,
            mixture.lognormal=result2_lognormal,
            mixture.Weibull=result2_Weibull,
            mixture.generalized.gamma=result2_generalized.gamma)

plot(result2_gamma, xlim=c(0, 600), breaks=seq(from=0, to=600, by=10))
plot(result2_generalized.gamma, xlim=c(0, 600), breaks=seq(from=0, to=600, by=10))

# -----
# left and right censored distribution
# -----
# Generate 100 random data from a gamma distribution
obc <- rgamma(100, scale=20, shape=2)
# Detection limit
LDL <- 10
# remove the data below the detection limit
obc[obc<LDL] <- -Inf
# Detection limit

```

```

UDL <- 100
# remove the data below the detection limit
obc[obc>UDL] <- +Inf
# search for the parameters the best fit these censored data
result <- cutter(observations=obc, lower_detection_limit=LDL,
                upper_detection_limit=UDL,
                cut_method="censored")

result
plot(result, xlim=c(0, 150), col.DL=c("black", "grey"),
     col.unobserved=c("green", "blue"),
     breaks=seq(from=0, to=150, by=10))

# -----
# Example with two values for lower detection limits
# corresponding at two different methods of detection for example
# with gamma distribution
# -----
obc <- rgamma(50, scale=20, shape=2)
# Detection limit for sample 1 to 50
LDL1 <- 10
# remove the data below the detection limit
obc[obc<LDL1] <- -Inf
obc2 <- rgamma(50, scale=20, shape=2)
# Detection limit for sample 1 to 50
LDL2 <- 20
# remove the data below the detection limit
obc2[obc2<LDL2] <- -Inf
obc <- c(obc, obc2)
# search for the parameters the best fit these censored data
result <- cutter(observations=obc,
                lower_detection_limit=c(rep(LDL1, 50), rep(LDL2, 50)),
                cut_method="censored")

result
# It is difficult to choose the best set of colors
plot(result, xlim=c(0, 150), col.dist="red",
     col.unobserved=c(rgb(red=1, green=0, blue=0, alpha=0.1),
                      rgb(red=1, green=0, blue=0, alpha=0.2)),
     col.DL=c(rgb(red=0, green=0, blue=1, alpha=0.5),
              rgb(red=0, green=0, blue=1, alpha=0.9)),
     breaks=seq(from=0, to=200, by=10))

# -----
# left censored distribution comparison of normal, lognormal,
# weibull, generalized gamma, and gamma without Bayesian MCMC
# Comparison with Akaike Information Criterion
# -----
# Detection limit
DL <- 10
# Generate 100 random data from a gamma distribution
obc <- rgamma(100, scale=20, shape=2)
# remove the data below the detection limit
obc[obc<DL] <- -Inf

result_gamma <- cutter(observations=obc, lower_detection_limit=DL,

```



```

                                cut_method="censored", distribution="gamma",
                                n.iter=NULL)
plot(result_gamma)

result_lognormal <- cutter(observations=obc, lower_detection_limit=DL,
                           cut_method="censored", distribution="lognormal",
                           n.iter=NULL)
plot(result_lognormal)

result_weibull <- cutter(observations=obc, lower_detection_limit=DL,
                        cut_method="censored", distribution="weibull",
                        n.iter=NULL)
plot(result_weibull)

result_normal <- cutter(observations=obc, lower_detection_limit=DL,
                       cut_method="censored", distribution="normal",
                       n.iter=NULL)
plot(result_normal)

result_generalized.gamma <- cutter(observations=obc, lower_detection_limit=DL,
                                   cut_method="censored", distribution="generalized.gamma",
                                   n.iter=NULL)
plot(result_generalized.gamma)

compare_AIC(gamma=result_gamma,
            lognormal=result_lognormal,
            normal=result_normal,
            Weibull=result_weibull,
            Generalized.gamma=result_generalized.gamma)

# -----
# left censored distribution comparison of normal, lognormal,
# weibull, generalized gamma, and gamma
# -----
# Detection limit
DL <- 10
# Generate 100 random data from a gamma distribution
obc <- rgamma(100, scale=20, shape=2)
# remove the data below the detection limit
obc[obc<DL] <- -Inf
# search for the parameters the best fit these truncated data
result_gamma <- cutter(observations=obc, lower_detection_limit=DL,
                      cut_method="censored", distribution="gamma")
result_gamma
plot(result_gamma, xlim=c(0, 250), breaks=seq(from=0, to=250, by=10))

result_lognormal <- cutter(observations=obc, lower_detection_limit=DL,
                           cut_method="censored", distribution="lognormal")
result_lognormal
plot(result_lognormal, xlim=c(0, 250), breaks=seq(from=0, to=250, by=10))

result_weibull <- cutter(observations=obc, lower_detection_limit=DL,
                        cut_method="censored", distribution="weibull")

```

```

result_weibull
plot(result_weibull, xlim=c(0, 250), breaks=seq(from=0, to=250, by=10))

result_normal <- cutter(observations=obc, lower_detection_limit=DL,
                        cut_method="censored", distribution="normal")
result_normal
plot(result_normal, xlim=c(0, 250), breaks=seq(from=0, to=250, by=10))

result_generalized.gamma <- cutter(observations=obc, lower_detection_limit=DL,
                                   cut_method="censored", distribution="generalized.gamma")
result_generalized.gamma
plot(result_generalized.gamma, xlim=c(0, 250), breaks=seq(from=0, to=250, by=10))

# -----
# Test for similarity in gamma left censored distribution between two
# datasets
# -----
obc1 <- rgamma(100, scale=20, shape=2)
# Detection limit for sample 1 to 50
LDL <- 10
# remove the data below the detection limit
obc1[obc1<LDL] <- -Inf
obc2 <- rgamma(100, scale=10, shape=2)
# remove the data below the detection limit
obc2[obc2<LDL] <- -Inf
# search for the parameters the best fit these censored data
result1 <- cutter(observations=obc1,
                  distribution="gamma",
                  lower_detection_limit=LDL,
                  cut_method="censored", n.iter=NULL)
logLik(result1)
plot(result1, xlim=c(0, 200),
      breaks=seq(from=0, to=200, by=10))
result2 <- cutter(observations=obc2,
                  distribution="gamma",
                  lower_detection_limit=LDL,
                  cut_method="censored", n.iter=NULL)
logLik(result2)
plot(result2, xlim=c(0, 200),
      breaks=seq(from=0, to=200, by=10))
result_tot1 <- cutter(observations=c(obc1, obc2),
                     distribution="gamma",
                     lower_detection_limit=LDL,
                     cut_method="censored", n.iter=NULL)
logLik(result_tot1)
plot(result_tot1, xlim=c(0, 200),
      breaks=seq(from=0, to=200, by=10))

compare_AIC(Separate=list(result1, result2),
            Common=result_tot1, factor.value=1)
compare_BIC(Separate=list(result1, result2),
            Common=result_tot1, factor.value=1)

```

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

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

```
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 <code>.deparseOpts</code> for their description.
collapse	Characters used to separate values.

Details

d Write an ASCII Representation of a vector object

Value

A string

Author(s)

Marc Girondot

See Also

Other Characters: [asc\(\)](#), [chr\(\)](#), [tnirp\(\)](#)

Examples

```
d(c(A=10, B=20))  
dput(c(A=10, B=20))
```

 dbeta_new

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).

The returned object has three attributes:

shape1, shape2, and ncp

Note that if x has attributes, they are preserved.

Usage

```
dbeta_new(
  x,
  mu = NULL,
  v = NULL,
  shape1,
  shape2,
  ncp = 0,
  log = FALSE,
  silent = 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).
silent	If FALSE, show the shape1 and shape 2 values.

Details

dbeta_new returns the density for the Beta distributions

The Beta distribution with parameters shape1 = a and shape2 = b has density

$$\frac{\text{gamma}(a+b)}{\text{gamma}(a)\text{gamma}(b)}x^{a-1}(1-x)^{b-1}$$

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.

Value

dbeta_new gives the density for the Beta distributions

Author(s)

Marc Girondot

See Also

Other Distributions: [cutter\(\)](#), [dcutter\(\)](#), [dggamma\(\)](#), [logLik.cutter\(\)](#), [plot.cutter\(\)](#), [print.cutter\(\)](#), [rcutter\(\)](#)

Examples

```
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()$ylim["end"]/
  max(dbeta_new(seq(from=0.01, to=0.99, by=0.01), mu = 0.8, v=0.1))
curve(dbeta_new(x, mu = 0.8, v=0.1)*mx, add=TRUE, col="red")
out <- dbeta_new(0.1, mu = 0.8, v=0.1)
out
attributes(out)$shape1; attributes(out)$shape2; attributes(out)$ncp
dbeta(0.1, shape1=attributes(out)$shape1, shape2=attributes(out)$shape2,
  ncp=attributes(out)$ncp)

# It can be used to generate random numbers using mu and v
out <- dbeta_new(0.1, mu = 0.8, v=0.1, silent=TRUE)
pi <- rbeta(100, shape1=attributes(out)$shape1, shape2=attributes(out)$shape2,
  ncp=attributes(out)$ncp)
hist(pi, freq=FALSE, breaks=seq(from=0, to=1, by=0.1), ylim=c(0, 8), las=1)
```

dcutter

Distribution of the fitted distribution without cut.

Description

If observations must be a data.frame with 4 columns:
 observations: A column for the measurements;
 LDL: A column for the lower detection limit;
 UDL: A column for the upper detection limit;
 Cut: A column for the truncated of censored nature of the data.

Usage

```
dcutter(  
  par,  
  observations = NULL,  
  distribution = "gamma",  
  n.mixture = NULL,  
  debug = FALSE,  
  limits.lower = NULL,  
  limits.upper = NULL,  
  log = TRUE  
)
```

Arguments

par	Values for parameters of distribution
observations	The observations; see description.
distribution	Can be gamma, normal, weibull, lognormal, or generalized.gamma.
n.mixture	Number of distributions
debug	If TRUE, show some information. If 2, show more information.
limits.lower	Value for lower detection limit
limits.upper	Value for upper detection limit
log	If TRUE, return the log likelihood

Details

dcutter returns the density of the cutter function

Value

The density of the cutter function according to observations.

Author(s)

Marc Girondot

See Also

Other Distributions: [cutter\(\)](#), [dbeta_new\(\)](#), [dggamma\(\)](#), [logLik.cutter\(\)](#), [plot.cutter\(\)](#), [print.cutter\(\)](#), [rcutter\(\)](#)

Examples

```
## Not run:  
library(HelpersMG)  
par <- c('shape1' = 0.42265849507444225,  
        'scale1' = 14.139457094879594,  
        'shape2' = 1.667131542489706,  
        'scale2' = 0.10763344388223803,
```

```

'p1' = 0.12283307526788023)
obs <- data.frame(Observations=c(0.755, 1.013, 2.098, 6.265, 4.708, 0.078, 2.169, 0.403, 1.251,
                                0.008, 1.419, 1.078, 2.744, 81.534, 1.426, 13.486, 7.813, 0.165,
                                0.118, 0.864, 0.369, 7.159, 2.605, 1.579, 1.646, 0.484, 4.492,
                                0.139, 0.28, 0.154, 0.106, 0.104, 4.185, 0.735, 0.149, 0.183,
                                0.062, 8.246, 0.165, 0.121, 0.109, 0.092, 0.162, 0.108, 0.139,
                                0.141, 0.124, 0.124, 0.151, 0.141, 0.364, 0.295, 0.09, 0.135,
                                0.154, 0.218, 0.167, -Inf, 0.203, 0.228, 0.107, 0.162, 0.194,
                                0.322, 0.351, 0.17, 0.236, 0.176, 0.107, 0.12, 0.095, 0.27, 0.194,
                                0.125, 0.123, 0.085, 0.164, 0.106, 0.079, 0.162),
                  LDL=0.001, UDL=NA, Cut="censored")
dcutter(par=par, observations=obs, distribution="gamma",
        n.mixture=NULL, debug=FALSE, limits.lower=NULL,
        limits.upper=NULL, log=FALSE)
dcutter(par=par, observations=obs, distribution="gamma",
        n.mixture=NULL, debug=FALSE, limits.lower=NULL,
        limits.upper=NULL, log=TRUE)

## End(Not run)

```

dggamma

Generalized gamma distribution.

Description

Generalized gamma distribution

Usage

```
dggamma(x, theta, kappa, delta, log = FALSE)
```

```
pgamma(q, theta, kappa, delta, lower.tail = TRUE, log.p = FALSE)
```

```
qgamma(p, theta, kappa, delta, lower.tail = TRUE, log.p = FALSE)
```

```
rggamma(n, theta, kappa, delta)
```

Arguments

x, q	vector of quantiles.
theta	scale parameter.
kappa	shape parameter.
delta	shape parameter.
log, log.p	logical; if TRUE, probabilities p are given as $\log(p)$.
lower.tail	logical; if TRUE (default), probabilities are $P[X \leq x]$ otherwise, $P[X > x]$.
p	vector of probabilities.
n	number of observations.

Details

pgamma, qgamma, dgamma, and rgamma are used to model the generalized gamma distribution. The code is modified from <https://rpubs.com/FJRubio/GG>.

Value

dgamma gives the density, pgamma gives the distribution function, qgamma gives the quantile function, and rgamma generates random deviates.

Functions

- dgamma: Density of the generalized gamma.
- pgamma: Distribution function of the generalized gamma.
- qgamma: Quantile of the generalized gamma.
- rgamma: Random of the generalized gamma.

More details here

The generalized gamma is described here https://en.wikipedia.org/wiki/Generalized_gamma_distribution.

With a being θ , b being κ , and p being δ . θ , κ and δ must be all > 0 .

Author(s)

Marc Girondot

See Also

Other Distributions: [cutter\(\)](#), [dbeta_new\(\)](#), [dcutter\(\)](#), [logLik.cutter\(\)](#), [plot.cutter\(\)](#), [print.cutter\(\)](#), [rcutter\(\)](#)

Examples

```
# To reproduce the wikipedia page graphic
x <- seq(from=0, to=8, by=0.1)
plot(x, dgamma(x, theta=2, kappa=0.5, delta=0.5), lty=1, col="blue",
      type="l", lwd=2, xlab="x", ylab="PDF")
lines(x, dgamma(x, theta=1, kappa=1, delta=0.5), lty=1, col="green", lwd=2)
lines(x, dgamma(x, theta=2, kappa=1, delta=2), lty=1, col="red", lwd=2)
lines(x, dgamma(x, theta=5, kappa=1, delta=5), lty=1, col="yellow", lwd=2)
lines(x, dgamma(x, theta=7, kappa=1, delta=7), lty=1, col="grey", lwd=2)
legend("topright", legend=c("a=2, d=0.5, p=0.5", "a=1, d=1, p=0.5",
                             "a=2, d=1, p=2", "a=5, d=1, p=5", "a=7, d=1, p=7"),
      col=c("blue", "green", "red", "yellow", "grey"),
      lty=1, lwd=2, bty="n")
```

DIx *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 p1 and p2 two vectors of relative measures of objects with $\text{sum}(p1) = 1$ and $\text{sum}(p2)=1$ and n1 being the number of objects in p1 and n2 being the number of objects in p2.

Edwards distance for all permutations of p1 and p2 objects are computed and the average value E is calculated.

The maximum possible Shannon index for identical n1 is $\text{max1} = \text{sum}((1/n1) * \log(1/n1))$.

Shannon index is $v1 = \text{sum}(p1 * \log(p1))$.

If version == 2, the complementary of Shannon index for these n1 objects is used: $c1 = 2 * \text{max1} - v1$

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

The geometry mean between both sides defined the measure of diversity within each side: $S = \text{sqrt}(c1 * c2)$

The Developmental Instability Index is then $S * E$

Usage

```
DIx(l1, l2, details = FALSE, version = 1)
```

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

Details

DIx returns an index of quantitative asymmetry and complexity

Value

A numeric value

Author(s)

Marc Girondot

References

- Edwards, A.W.F., 1971. Distances between populations on the basis of gene frequencies. *Biometrics* 27, 873–881.
- Shannon C.E. 1948 A mathematical theory of communication. *Bell System Technical Journal* 27(3), 379-423.

Examples

```
## 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(0.2, 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, 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)
```

dSnbinom

Density for the sum of random variable with negative binomial distributions.

Description

Density for the sum of random variable with negative binomial distributions.
If all prob values are the same, infinite is automatically set to 0.

Usage

```
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

References

Furman, E., 2007. On the convolution of the negative binomial random variables. *Statistics & Probability Letters* 77, 169-172.

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)
```

duplicated_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 <- duplicated_packages()
if (nrow(li) != 0)
for (i in 1:nrow(li))
remove.packages(rownames(li)[i], lib=li[i, "Lib1"])
```

Usage

```
duplicated_packages()
```

Details

duplicated_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)
duplicated_packages()

## End(Not run)
```

 ellipse

Plot an ellipse

Description

Plot an ellipse defined 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

```
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")
extractAIC(g)
options(AIC="AICc")
extractAIC(g)
```

fitdistrquantiles *Parameters of beta, normal or gamma distribution based on quantiles.*

Description

Return the parameters of beta or gamm that fits the best the quantiles. The vector of probabilities can be obtained from names of quantiles.

Usage

```
fitdistrquantiles(
  quantiles = stop("At least two quantiles must be provided"),
  probs = NULL,
  scaled = FALSE,
  distribution = "beta"
)
```

Arguments

quantiles	Vector of quantiles.
probs	Numeric vector of probabilities with values in [0,1].
scaled	Used scaled least-square.
distribution	Distribution to be fitted: beta, normal, or gamma.

Details

fitdistrquantiles returns the parameters of beta, normal or gamma distribution

Value

Parameters of beta, normal or gamma distribution based on quantiles.

Author(s)

Marc Girondot

Examples

```
rd <- rbeta(100000, shape1 = 0.7, shape2 = 6.2, ncp=0)
(q <- quantile(rd, probs=c(0.025, 0.5, 0.975)))

(best <- fitdistrquantiles(quantiles = q, probs = c(0.025, 0.5, 0.975),
  scaled=FALSE, distribution = "beta"))
rd10000 <- rbeta(10000, shape1 = best["shape1"], shape2 = best["shape2"], ncp=best["ncp"])
quantile(rd10000, probs=c(0.025, 0.5, 0.975))

# Here the probabilities are obtained from names of quantiles
(best <- fitdistrquantiles(quantiles = q, scaled=FALSE, distribution = "beta"))
```

```

rd10000 <- rbeta(10000, shape1 = best["shape1"], shape2 = best["shape2"], ncp=best["ncp"])
quantile(rd10000, probs=c(0.025, 0.5, 0.975))

# If only two quantiles are provided, ncp cannot be fitted
(q2 <- quantile(rd, probs=c(0.025, 0.975)))
(best <- fitdistrquantiles(quantiles = q2, scaled=FALSE, distribution = "beta"))
rd10000 <- rbeta(10000, shape1 = best["shape1"], shape2 = best["shape2"])
quantile(rd10000, probs=c(0.025, 0.975))
x <- seq(from=0.00, to=1, by=0.001)
plot(x=x, y=pbeta(x, shape1 = best["shape1"], shape2 = best["shape2"]),
     las=1, bty="n", type="l", ylim=c(0, 1))
segments(x0=q2[1], x1=q2[1], y0=0, y1=1, lty=2)
segments(x0=q2[2], x1=q2[2], y0=0, y1=1, lty=2)

(best <- fitdistrquantiles(quantiles = q, probs = c(0.025, 0.5, 0.975),
                          scaled=FALSE, distribution = "gamma"))
rd10000 <- rgamma(10000, shape = best["shape"], scale = best["scale"])
quantile(rd10000, probs=c(0.025, 0.5, 0.975))

(best <- fitdistrquantiles(quantiles = c(10, 20, 30), probs = c(0.025, 0.5, 0.975),
                          scaled=FALSE, distribution = "normal"))
rd10000 <- rnorm(10000, mean = best["mean"], sd = best["sd"])
quantile(rd10000, probs=c(0.025, 0.5, 0.975))

```

flexit

Return the flexit

Description

Return a vector with the probabilities. The flexit equation is published in:

Abreu-Grobois, F.A., Morales-Mérida, B.A., Hart, C.E., Guillon, J.-M., Godfrey, M.H., Navarro, E. & Girondot, M. (2020) Recent advances on the estimation of the thermal reaction norm for sex ratios. *PeerJ*, 8, e8451.

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

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

with:

$$S1 = (2^{(K1 - 1)} * S * K1) / (2^{K1} - 1)$$

$$S2 = (2^{(K2 - 1)} * S * K2) / (2^{K2} - 1)$$

New in version 4.7-3 and larger:

If 2^{K1} is too large to be estimated, the approximation $S1 = S * K1/2$ is used.

Demonstration:

$$S1 = (2^{(K1 - 1)} * S * K1) / (2^{K1} - 1)$$

$$S1 = \exp(\log((2^{(K1 - 1)} * S * K1) / (2^{K1} - 1)))$$

$$S1 = \exp(\log(2^{K1} - 1)) + \log(S * K1) - \log(2^{K1} - 1))$$

When $K1$ is very large, $2^{K1} - 1 = 2^{K1}$ then

$$S1 = \exp((K1 - 1) * \log(2) + \log(S * K1) - K1 * \log(2))$$

$$S1 = \exp((K1 * \log(2) - \log(2) + \log(S * K1) - K1 * \log(2))$$

$$S1 = \exp(\log(S * K1) - \log(2))$$

$$S1 = S * K1/2$$

If 2^{K2} is too large to be estimated, the approximation $S2 = S * K2/2$ is used.

If $(1 + (2^{K1} - 1) * \exp(4 * S1 * (P - x)))^{(-1/K1)}$ is not finite, the following approximation is used:

$$\exp((-1/K1) * (K1 * \log(2) + (4 * S1 * (P - x))))$$

If $1 - ((1 + (2^{K2} - 1) * \exp(4 * S2 * (x - P)))^{(-1/K2)})$ is not finite, the following approximation is used:

$$1 - \exp((-1/K2) * (K2 * \log(2) + (4 * S2 * (x - P))))$$

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)

1/(1+exp(0.01*4*(50-1:100)))
flexit(1:100, P=50, S=0.01, K1=1, K2=1)
```

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

<code>logLik</code>	The log likelihood
<code>nobs</code>	Number of observations
<code>df</code>	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: [ExtractAIC.glm\(\)](#), [compare_AICc\(\)](#), [compare_AIC\(\)](#), [compare_BIC\(\)](#)

Examples

```
## 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)
```

format_ncdf

Return an array with ncdf data

Description

Return a list with two elements: data is an array and time is the POSIX.It time.
Or if label.time is NULL or if bathy is TRUE, a bathy object.
If varid is NULL, it shows the available variable and dimensions of the file.
Bathymetry data can be download here:
https://www.gebco.net/data_and_products/gridded_bathymetry_data/#global

Usage

```
format_ncdf(  
  ncdf,  
  label.latitude = "latitude",  
  label.longitude = "longitude",  
  label.time = "time",  
  varid = NULL,  
  longitude1 = NA,  
  latitude1 = NA,  
  longitude2 = NA,  
  latitude2 = NA,  
  package = "ncdf4",  
  bathy = TRUE  
)
```

Arguments

ncdf	An object read from package ncdf4 or a file name of ncdf file
label.latitude	Label of latitude
label.longitude	Label of longitude
label.time	Label of time
varid	Name of variable to extract
longitude1	Longitude for first corner
latitude1	latitude for first corner
longitude2	Longitude for second corner
latitude2	latitude for second corner
package	If ncdf is a file, give the package to use to open the file
bathy	If TRUE, return a bathy object

Details

format_ncdf is used extract information from ncdf file

Value

A list with two element: data is an array and time is the POSIX.It time

Author(s)

Marc Girondot

See Also

Other ncdf: [ind_long_lat\(\)](#)

Examples

```
## Not run:
url <- "https://downloads.psl.noaa.gov/Datasets/noaa.oisst.v2.highres/"
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)
format_ncdf(dest)

## End(Not run)
```

iCutter	<i>Run a shiny application to fit bone section</i>
---------	--

Description

Run a shiny application to fit bone section

Usage

```
iCutter()
```

Details

BP runs a shiny application to fit bone section

Value

Nothing

Author(s)

Marc Girondot <marc.girondot@gmail.com>

Examples

```
## Not run:  
# Not run:  
library(HelpersMG)  
iCutter()  
  
## End(Not run)
```

IC_clean_data	<i>Clean the dataframe before to be used with IC_threshold_matrix</i>
---------------	---

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_corr%C3%A9lations

Usage

```
IC_clean_data(
  data = stop("A dataframe 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

<code>data</code>	The data.frame to be cleaned
<code>use</code>	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".
<code>method</code>	a character string indicating which correlation coefficient (or covariance) is to be computed. One of "pearson" (default), "kendall", or "spearman": can be abbreviated.
<code>variable.retain</code>	a vector with the name of columns to keep
<code>test.partial.correlation</code>	should the partial correlations be tested ?
<code>progress</code>	Show a progress bar
<code>debug</code>	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

Lesty, M., 1999. Une nouvelle approche dans le choix des régresseurs de la régression multiple en présence d'interactions et de colinéarités. *Revue de Modulad* 22, 41-77.

See Also

Other Iconography of correlations: [IC_correlation_simplify\(\)](#), [IC_threshold_matrix\(\)](#), [plot.IconoCorel\(\)](#)

Examples

```
## Not run:
library("HelpersMG")
# based on https://fr.wikipedia.org/wiki/Iconographie_des_corrélations
es <- structure(list(Student = c("e1", "e2", "e3", "e4", "e5", "e6", "e7", "e8"),
                    Mass = c(52, 59, 55, 58, 66, 62, 63, 69),
                    Age = c(12, 12.5, 13, 14.5, 15.5, 16, 17, 18),
                    Assiduity = c(12, 9, 15, 5, 11, 15, 12, 9),
                    Note = c(5, 5, 9, 5, 13.5, 18, 18, 18)),
               row.names = c(NA, -8L), class = "data.frame")

es

df_clean <- IC_clean_data(es, debug = TRUE)
cor_matrix <- IC_threshold_matrix(data=df_clean, threshold = NULL, progress=FALSE)
cor_threshold <- IC_threshold_matrix(data=df_clean, threshold = 0.3)
plot(cor_threshold, show.legend.strength=FALSE, show.legend.direction = FALSE)
cor_threshold_Note <- IC_correlation_simplify(matrix=cor_threshold, variable="Note")
plot(cor_threshold_Note, show.legend.strength=FALSE, show.legend.direction = FALSE)

cor_threshold <- IC_threshold_matrix(data=df_clean, threshold = 0.6)
plot(cor_threshold,
     layout=matrix(data=c(53, 53, 55, 55,
                        55, 53, 55, 53), ncol=2, byrow=FALSE),
     show.legend.direction = FALSE,
     show.legend.strength = FALSE, xlim=c(-2, 2), ylim=c(-2, 2))

## End(Not run)
```

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_c

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

Lesty, M., 1999. Une nouvelle approche dans le choix des régresseurs de la régression multiple en présence d'interactions et de colinéarités. *Revue de Modulad* 22, 41-77.

See Also

Other Iconography of correlations: [IC_clean_data\(\)](#), [IC_threshold_matrix\(\)](#), [plot.IconoCorel\(\)](#)

Examples

```
## Not run:
library("HelpersMG")
es <- structure(list(Student = c("e1", "e2", "e3", "e4", "e5", "e6", "e7", "e8"),
                      Mass = c(52, 59, 55, 58, 66, 62, 63, 69),
                      Age = c(12, 12.5, 13, 14.5, 15.5, 16, 17, 18),
                      Assiduity = c(12, 9, 15, 5, 11, 15, 12, 9),
                      Note = c(5, 5, 9, 5, 13.5, 18, 18, 18)),
                row.names = c(NA, -8L), class = "data.frame")

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")
plot(cor_threshold_Note)

## End(Not run)
```

 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: The link between A and B is “remarkable” if and only if the total correlation between them is higher than a given threshold and if the partial correlation between A and B in respect to any other variable C is also higher in absolute values than this threshold and with the same sign as the total correlation. 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.

https://fr.wikipedia.org/wiki/Iconographie_des_corr%C3%A9lations

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"),
  significance.level = FALSE,
  correction.multiple.comparisons = "fdr",
  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).

significance.level
 if FALSE, does not use significance level; or use this significance level.
 correction.multiple.comparisons
 "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", or "none".
 progress show a progress bar
 debug display information about progression of computing

Details

IC_threshold_matrix calculates correlation matrix thresholded by partial correlation

Value

A list

Author(s)

Marc Girondot

References

Lesty, M., 1999. Une nouvelle approche dans le choix des régresseurs de la régression multiple en présence d'interactions et de colinéarités. *Revue de Modulad* 22, 41-77.

See Also

Other Iconography of correlations: [IC_clean_data\(\)](#), [IC_correlation_simplify\(\)](#), [plot.IconoCorel\(\)](#)

Examples

```
## Not run:
library("HelpersMG")
es <- structure(list(Student = c("e1", "e2", "e3", "e4", "e5", "e6", "e7", "e8"),
  Mass = c(52, 59, 55, 58, 66, 62, 63, 69),
  Age = c(12, 12.5, 13, 14.5, 15.5, 16, 17, 18),
  Assiduity = c(12, 9, 15, 5, 11, 15, 12, 9),
  Note = c(5, 5, 9, 5, 13.5, 18, 18, 18)),
  row.names = c(NA, -8L), class = "data.frame")

es

df_clean <- IC_clean_data(es, debug = TRUE)
cor_matrix <- IC_threshold_matrix(data=df_clean, threshold = NULL, progress=FALSE)
cor_threshold <- IC_threshold_matrix(data=df_clean, threshold = 0.3)
plot(cor_threshold, show.legend.strength=FALSE, show.legend.direction = FALSE)
cor_threshold_Note <- IC_correlation_simplify(matrix=cor_threshold, variable="Note")
plot(cor_threshold_Note)

cor_threshold <- IC_threshold_matrix(data=df_clean, threshold = 0.8, progress=FALSE)
gr <- plot(cor_threshold, plot=FALSE)
ly <- getFromNamespace("layout_nicely", ns="igraph")(gr)
```

```

plot(cor_threshold,
layout=matrix(data=c(53, 53, 55, 55,
                    55, 53, 55, 53), ncol=2, byrow=FALSE),
show.legend.direction = FALSE,
show.legend.strength = FALSE, xlim=c(-2, 2), ylim=c(-2, 2))

# Using significance level

cor_threshold <- IC_threshold_matrix(data=df_clean, threshold = 0.3,
significance.level=0.05)
plot(cor_threshold, show.legend.strength=FALSE, show.legend.direction = FALSE)
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_clean, threshold = 0.3, model="AAT")
par(mar=c(1,1,1,1))
set.seed(4)
plot(cor_threshold_AAT, show.legend.strength="bottomleft")

#####
dta <- structure(list(Student = c("e1", "e2", "e3", "e4", "e5", "e6", "e7", "e8"),
Mass = c(52, 59, 55, 58, 66, 62, 63, 69),
Age = c(12, 12.5, 13, 14.5, 15.5, 16, 17, 18),
Assiduity = c(12, 9, 15, 5, 11, 15, 12, 9),
Note = c(5, 5, 9, 5, 13.5, 18, 18, 18)),
row.names = c(NA, -8L), class = "data.frame")

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)

## End(Not run)

```

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

```
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>

See Also

Other Periodic patterns of indices: [minmax.periodic\(\)](#), [moon.info\(\)](#), [sun.info\(\)](#), [tide.info\(\)](#)

Examples

```
## 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

Return or the index in ncdf object from lat/longitude or inverse

Description

Return or the index in ncdf object from lat/longitude or reverse.

Usage

```

ind_long_lat(
  ncdf = stop("The ncdf data must be supplied"),
  long = NULL,
  lat = NULL,
  indice.long = NULL,
  indice.lat = NULL,
  label.longitude = "lon",
  label.latitude = "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
label.longitude	Name of argument for longitude, default is lon
label.latitude	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

See Also

Other ncdf: [format_ncdf\(\)](#)

Examples

```
## Not run:
url <- "https://downloads.psl.noaa.gov/Datasets/noaa.oisst.v2.highres/"
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)
# 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

Author(s)

Marc Girondot

Examples

```
## Not run:
library(HelpersMG)
# Search for files in path with names based on pattern that have the string search inside.
inside.search(path=".", pattern="*\\\.R$", search="embryogrowth")

## End(Not run)
```

invlogit	<i>Return the inverse logit</i>
----------	---------------------------------

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)
```

LD50

*Estimate the parameters that best describe LD50***Description**

Estimate the parameters that best describe LD50

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

logistic = $1/(1+\exp((1/S)*(P-d)))$

logit = $1/(1+\exp(P+d*S))$

See these publications for the description of equations:

Girondot, M. 1999. Statistical description of temperature-dependent sex determination using maximum likelihood. *Evolutionary Ecology Research*, 1, 479-486.

Godfrey, M.H., Delmas, V., Girondot, M., 2003. Assessment of patterns of temperature-dependent sex determination using maximum likelihood model selection. *Ecoscience* 10, 265-272.

Hulin, V., Delmas, V., Girondot, M., Godfrey, M.H., Guillon, J.-M., 2009. Temperature-dependent sex determination and global change: are some species at greater risk? *Oecologia* 160, 493-506.

The flexit equation is not still published :

$$\text{ifdose} < P \text{ then } (1 + (2^{K1} - 1) * \exp(4 * S1 * (P - x)))^{(- 1/K1)}$$

$$\text{ifdose} > P \text{ then } 1 - ((1 + (2^{K2} - 1) * \exp(4 * S2 * (x - P)))^{(- 1/K2)})$$

with:

$$S1 = S / ((4/K1) * (2^{(- K1)} - 1))^{(1/K1 + 1)} * (2^{K1} - 1)$$

$$S2 = S / ((4/K2) * (2^{(- K2)} - 1))^{(1/K2 + 1)} * (2^{K2} - 1)$$

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

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

```
## 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 $\sqrt{n.iter}$.

Jones, G.L., Haran, M., Caffo, B.S. and Neath, R. (2006) Fixed Width Output Analysis for Markov

chain Monte Carlo , Journal of the American Statistical Association, 101:1537-1547.
 coda package is necessary for this function.

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

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

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

```
## 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
```



```

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(-40, 40))
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))

## End(Not run)

```

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	<i>Return path of file searched for in local disk based on its file name</i>
--------------	--

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

Examples

```
## Not run:
RnwFiles <- local.search("*.Rnw")
nc.files <- local.search("*.nc", folder=paste0("'",getwd(),"'"))

## End(Not run)
```

logit*Return the 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)
```

logLik.compareAIC	<i>Return Log Likelihood generated by FormatCompareAIC</i>
-------------------	--

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.cutter	<i>Return log likelihood of a cutter fitted model</i>
---------------	---

Description

Return log likelihood of a cutter fitted model.

Usage

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

Arguments

object	A result file generated by cutter
...	Not used

Details

logLik.cutter return log likelihood of a cutter fitted model

Value

Nothing

Author(s)

Marc Girondot

See Also

Other Distributions: [cutter\(\)](#), [dbeta_new\(\)](#), [dcutter\(\)](#), [dggamma\(\)](#), [plot.cutter\(\)](#), [print.cutter\(\)](#), [rcutter\(\)](#)

Examples

```
## Not run:  
# -----  
# Test for similarity in gamma left censored distribution between two  
# datasets  
# -----  
obc1 <- rgamma(100, scale=20, shape=2)  
# Detection limit for sample 1 to 50  
LDL <- 10  
# remove the data below the detection limit  
obc1[obc1<LDL] <- -Inf  
obc2 <- rgamma(100, scale=10, shape=2)  
# remove the data below the detection limit  
obc2[obc2<LDL] <- -Inf
```

```

# search for the parameters the best fit these censored data
result1 <- cutter(observations=obc1,
                 lower_detection_limit=LDL,
                 cut_method="censored")

logLik(result1)
result2 <- cutter(observations=obc2,
                 lower_detection_limit=LDL,
                 cut_method="censored")

logLik(result2)
result_totl <- cutter(observations=c(obc1, obc2),
                    lower_detection_limit=LDL,
                    cut_method="censored")

logLik(result_totl)
compare_AICc(Separate=list(result1, result2),
             Common=result_totl, factor.value=1)
compare_BIC(Separate=list(result1, result2),
            Common=result_totl, factor.value=1)

## 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)
```

merge.mcmcComposite *Merge two mcmcComposite results*

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

```
## 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

```
## 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)
```

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, dbeta
- 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:

Rosenthal, J. S. 2011. Optimal Proposal Distributions and Adaptive MCMC. Pages 93-112 in 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,
  progress.bar.ini = NULL,
  progress.bar = NULL,
  adaptive = FALSE,
  adaptive.lag = 500,
  adaptive.fun = function(x) { ifelse(x > 0.234, 1.3, 0.7) },
  intermediate = NULL,
  filename = "intermediate.Rdata",
```

```

    previous = NULL,
    session = 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
progress.bar.ini	The command to initialize progress bar
progress.bar	The command to run the progress bar
adaptive	Should an adaptive process for SDProp be used
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
session	The shiny session

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$resultMCMC[[1]][, 1]
y <- mcmc_run$resultMCMC[[1]][, 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
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)$statistics[, "Time-series SE"]
summary(mcmc_run)
as.parameters(mcmc_run)
lastp <- as.parameters(mcmc_run, index="last")
parameters_mcmc[, "Init"] <- lastp
# The n.adapt set to 1 is used to not record the first set of parameters
# then it is not duplicated (as it is also the last one for
# the object mcmc_run)
mcmc_run2 <- MHalgoGen(n.iter=1000, parameters=parameters_mcmc, x=x, data=val,
likelihood=dnormx, n.chains=1, n.adapt=1, thin=1, trace=1)
mcmc_run3 <- merge(mcmc_run, mcmc_run2)
##### no adaptation, n.adapt must be 0
parameters_mcmc[, "Init"] <- c(mean(x), sd(x))
mcmc_run3 <- MHalgoGen(n.iter=1000, parameters=parameters_mcmc, x=x, data=val,
likelihood=dnormx, n.chains=1, n.adapt=0, thin=1, trace=1)
# Here is how to use adaptive mcmc
mcmc_run <- MHalgoGen(n.iter=50000, parameters=parameters_mcmc, data=val, adaptive = FALSE,
likelihood=dnormx, n.chains=1, n.adapt=100, thin=1, trace=1)
1-rejectionRate(as.mcmc(mcmc_run))
mcmc_run <- MHalgoGen(n.iter=50000, parameters=parameters_mcmc, data=val, adaptive = TRUE,
likelihood=dnormx, n.chains=1, n.adapt=100, thin=1, trace=1)
1-rejectionRate(as.mcmc(mcmc_run))
# To see the dynamics :
var <- "mean"
par(mar=c(4, 4, 1, 1)+0.4)
plot(1:nrow(mcmc_run$resultMCMC[[1]]), mcmc_run$resultMCMC[[1]][, var], type="l",
xlab="Iterations", ylab=var, bty="n", las=1)
# Exemple with a progress bar

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'))
# Set up the progress bar
mcmc_run <- MHALgoGen(n.iter=50000, parameters=parameters_mcmc, data=val,
likelihood=dnormx, n.chains=1, n.adapt=100, thin=1, trace=FALSE,
progress.bar.ini=function(n.iter) {
assign("pb", txtProgressBar(min=0, max=n.iter, style=3),
env = parent.frame())},
progress.bar=function(iter) {setTxtProgressBar(get("pb", envir = parent.frame()), iter)})

## End(Not run)

```

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
)

```

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

```
## 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

Examples

```
## 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)
```

moon.info

Moon phase based on a date

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 quarter

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

```
## 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	<i>Return a moving average of a vector.</i>
--------------	---

Description

Return a moving average of a vector./cr 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

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)

```

NagelkerkeScaledR2 *Return the scaled R2 defined by Nagelkerke (1991)*

Description

Return the scaled R2 of a binomial model based on:

Nagelkerke NJD (1991) A note on a general definition of the coefficient of determination. *Biometrika* 78:691-192.

This definition of scaled R2 by Nagelkerke (1991) has the following properties:

- (i) It is consistent with classical R2, that is the general definition applied to e.g. linear regression yields the classical R2.
- (ii) It is consistent with maximum likelihood as an estimation method, i.e. the maximum likelihood estimates of the model parameters maximize R2.
- (iii) It is asymptotically independent of the sample size n.
- (iv) 1-R2 has the interpretation of the proportion of unexplained 'variation'.
- (v) It is dimensionless, i.e. it does not depend on the units used.

The reported value is similar to the value estimated with `nagelkerke()` function from `rcompanion` package but not from the `NagelkerkeR2()` function from `fmsb` package. I don't know why.

Usage

```
NagelkerkeScaledR2(x, size, prediction, scaled = TRUE)
```

Arguments

<code>x</code>	The number of observations
<code>size</code>	Number of trials
<code>prediction</code>	Prediction of <code>x/size</code>
<code>scaled</code>	If TRUE, return the scaled R2

Details

`NagelkerkeScaledR2` returns the scaled R2 defined by Nagelkerke (1991)

Value

The scaled R2 value

Author(s)

Marc Girondot

Examples

```
x <- c(10, 9, 6, 4, 3, 1, 0)
size <- c(10, 10, 10, 10, 10, 10, 10)
prediction <- c(0.9, 0.8, 0.7, 0.5, 0.4, 0.3, 0.2)
NagelkerkeScaledR2(x, size, prediction)

# Using the example in fmsb::NagelkerkeR2
res <- glm(cbind(ncases,ncontrols) ~ agegp+alcgp+tobgp, data=esoph, family=binomial())
NagelkerkeScaledR2(x=esoph$ncases, size = esoph$ncases+esoph$ncontrols,
  prediction = res$fitted.values)
```

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

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

`newmap.scale`

Add Scale to Existing Unprojected Map

Description

Adds a scale to an existing map, both as a ratio and a distance gauge. If `x` or `y` are not specified, this will be taken to be near the lower left corner of the map.

Usage

```
newmap.scale(  
  x,  
  y,  
  relwidth = 0.15,  
  metric = TRUE,  
  ratio = TRUE,  
  col.line = "black",  
  ...  
)
```

Arguments

x	Location of left end of distance gauge.
y	Location of left end of distance gauge.
relwidth	Proportion of width of display to be used for the scale. The default is 0.15.
metric	If TRUE, the distance gauge will be in km, otherwise miles.
ratio	If FALSE, the scale ratio of the map is not displayed.
col.line	The color of lines for the gauge.
...	Further plotting parameters may be specified as for the command text().

Details

newmap.scale Add Scale to Existing Unprojected Map

Value

The exact calculated scale is returned.

Author(s)

See map.scale maps

Examples

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

## End(Not run)
```

openwd

Open a finder window with current working directory in MacOS X

Description

This function opens a finder window with directory files in MacOS X. It has not been tested in Windows. In linux, it just returns the list of files in directory. By default, it uses the current working directory.

Usage

```
openwd(directory = getwd())
```

Arguments

directory The directory you want to open

Details

openwd will open a finder window with current working directory

Value

A vector with the list of files.

Author(s)

Marc Girondot

Examples

```
## Not run:
openwd()

## End(Not run)
```

plot.cutter

Plot results of cutter that best describe distribution

Description

Plot the estimates of cut distribution.

Usage

```
## S3 method for class 'cutter'
plot(
  x,
  col.hist = "grey",
  col.DL = "blue",
  col.dist = "black",
  col.unobserved = "green",
  col.mcmc = rgb(red = 0.6, green = 0, blue = 0, alpha = 0.01),
  legend = TRUE,
  ...
)
```

Arguments

x	A result file generated by cutter
col.hist	The color of histogram
col.DL	The color of below of above samples
col.dist	The color of distribution
col.unobserved	The color of unobserved states

col.mcmc	The color of mcmc outputs
legend	If TRUE, a legend is shown
...	Parameters for plot

Details

plot.cutter plot result of cutter

Value

Nothing

Author(s)

Marc Girondot

See Also

Other Distributions: [cutter\(\)](#), [dbeta_new\(\)](#), [dcutter\(\)](#), [dgamma\(\)](#), [logLik.cutter\(\)](#), [print.cutter\(\)](#), [rcutter\(\)](#)

Examples

```
## Not run:
library(HelpersMG)
# -----
# right censored distribution with gamma distribution
# -----
# Detection limit
DL <- 100
# Generate 100 random data from a gamma distribution
obc <- rgamma(100, scale=20, shape=2)
# remove the data below the detection limit
obc[obc>DL] <- +Inf
# search for the parameters the best fit these censored data
result <- cutter(observations=obc, upper_detection_limit=DL,
                 cut_method="censored")

result
plot(result, xlim=c(0, 150), breaks=seq(from=0, to=150, by=10))
# -----
# The same data seen as truncated data with gamma distribution
# -----
obc <- obc[is.finite(obc)]
# search for the parameters the best fit these truncated data
result <- cutter(observations=obc, upper_detection_limit=DL,
                 cut_method="truncated")

result
plot(result, xlim=c(0, 150), breaks=seq(from=0, to=150, by=10))
# -----
# left censored distribution with gamma distribution
# -----
```

```

# Detection limit
DL <- 10
# Generate 100 random data from a gamma distribution
obc <- rgamma(100, scale=20, shape=2)
# remove the data below the detection limit
obc[obc<DL] <- -Inf
# search for the parameters the best fit these truncated data
result <- cutter(observations=obc, lower_detection_limit=DL,
                 cut_method="censored")

result
plot(result)
plot(result, xlim=c(0, 200), breaks=seq(from=0, to=200, by=10))
# -----
# left and right censored distribution
# -----
# Generate 100 random data from a gamma distribution
obc <- rgamma(100, scale=20, shape=2)
# Detection limit
LDL <- 10
# remove the data below the detection limit
obc[obc<LDL] <- -Inf
# Detection limit
UDL <- 100
# remove the data below the detection limit
obc[obc>UDL] <- +Inf
# search for the parameters the best fit these censored data
result <- cutter(observations=obc, lower_detection_limit=LDL,
                 upper_detection_limit=UDL,
                 cut_method="censored")

result
plot(result, xlim=c(0, 150), col.DL=c("black", "grey"),
      col.unobserved=c("green", "blue"),
      breaks=seq(from=0, to=150, by=10))
# -----
# Example with two values for lower detection limits
# corresponding at two different methods of detection for example
# with gamma distribution
# -----
obc <- rgamma(50, scale=20, shape=2)
# Detection limit for sample 1 to 50
LDL1 <- 10
# remove the data below the detection limit
obc[obc<LDL1] <- -Inf
obc2 <- rgamma(50, scale=20, shape=2)
# Detection limit for sample 1 to 50
LDL2 <- 20
# remove the data below the detection limit
obc2[obc2<LDL2] <- -Inf
obc <- c(obc, obc2)
# search for the parameters the best fit these censored data
result <- cutter(observations=obc,
                 lower_detection_limit=c(rep(LDL1, 50), rep(LDL2, 50)),
                 cut_method="censored")

```

```

result
# It is difficult to choose the best set of colors
plot(result, xlim=c(0, 150), col.dist="red",
      col.unobserved=c(rgb(red=1, green=0, blue=0, alpha=0.1),
                        rgb(red=1, green=0, blue=0, alpha=0.2)),
      col.DL=c(rgb(red=0, green=0, blue=1, alpha=0.5),
               rgb(red=0, green=0, blue=1, alpha=0.9)),
      breaks=seq(from=0, to=200, by=10))

# -----
# left censored distribution comparison of normal, lognormal and gamma
# -----
# Detection limit
DL <- 10
# Generate 100 random data from a gamma distribution
obc <- rgamma(100, scale=20, shape=2)
# remove the data below the detection limit
obc[obc<DL] <- -Inf
# search for the parameters the best fit these truncated data
result_gamma <- cutter(observations=obc, lower_detection_limit=DL,
                      cut_method="censored", distribution="gamma")
result_gamma
plot(result_gamma, xlim=c(0, 250), breaks=seq(from=0, to=250, by=10))

result_lognormal <- cutter(observations=obc, lower_detection_limit=DL,
                          cut_method="censored", distribution="lognormal")
result_lognormal
plot(result_lognormal, xlim=c(0, 250), breaks=seq(from=0, to=250, by=10))

result_normal <- cutter(observations=obc, lower_detection_limit=DL,
                       cut_method="censored", distribution="normal")
result_normal
plot(result_normal, xlim=c(0, 250), breaks=seq(from=0, to=250, by=10))

compare_AICc(gamma=result_gamma,
             lognormal=result_lognormal,
             normal=result_normal)

# -----
# Test for similarity in gamma left censored distribution between two
# datasets
# -----
obc1 <- rgamma(100, scale=20, shape=2)
# Detection limit for sample 1 to 50
LDL <- 10
# remove the data below the detection limit
obc1[obc1<LDL] <- -Inf
obc2 <- rgamma(100, scale=10, shape=2)
# remove the data below the detection limit
obc2[obc2<LDL] <- -Inf
# search for the parameters the best fit these censored data
result1 <- cutter(observations=obc1,
                 distribution="gamma",
                 lower_detection_limit=LDL,
                 cut_method="censored")

```

```

logLik(result1)
plot(result1, xlim=c(0, 200),
      breaks=seq(from=0, to=200, by=10))
result2 <- cutter(observations=obc2,
                 distribution="gamma",
                 lower_detection_limit=LDL,
                 cut_method="censored")
logLik(result2)
plot(result2, xlim=c(0, 200),
      breaks=seq(from=0, to=200, by=10))
result_totl <- cutter(observations=c(obc1, obc2),
                    distribution="gamma",
                    lower_detection_limit=LDL,
                    cut_method="censored")
logLik(result_totl)
plot(result_totl, xlim=c(0, 200),
      breaks=seq(from=0, to=200, by=10))

compare_AICc(Separate=list(result1, result2),
             Common=result_totl, factor.value=1)
compare_BIC(Separate=list(result1, result2),
            Common=result_totl, factor.value=1)

## End(Not run)

```

plot.IconoCorel

Clean the dataframe before to be used with IC_threshold_matrix

Description

This function plots the data as a network. It returns an invisible object that can be used with visI-graph from package visNetwork. https://fr.wikipedia.org/wiki/Iconographie_des_corr%C3%A9lations

Usage

```

## 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",
  vertex.label.cex = 1,
  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
vertex.label.cex a vector of cex
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

Lesty, M., 1999. Une nouvelle approche dans le choix des régresseurs de la régression multiple en présence d'interactions et de colinéarités. *Revue de Modulad* 22, 41-77.

See Also

Other Iconography of correlations: [IC_clean_data\(\)](#), [IC_correlation_simplify\(\)](#), [IC_threshold_matrix\(\)](#)

Examples

```
## Not run:
library("HelpersMG")
es <- structure(list(Student = c("e1", "e2", "e3", "e4", "e5", "e6", "e7", "e8"),
  Mass = c(52, 59, 55, 58, 66, 62, 63, 69),
  Age = c(12, 12.5, 13, 14.5, 15.5, 16, 17, 18),
  Assiduity = c(12, 9, 15, 5, 11, 15, 12, 9),
  Note = c(5, 5, 9, 5, 13.5, 18, 18, 18)),
  row.names = c(NA, -8L), class = "data.frame")
```

```

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)
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)

# You can record the position of elements and use them later
ly <- layout_nicely(kk)
plot(cor_threshold, vertex.color="red", layout=ly)

## 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
)

```

Arguments

x	A result file generated by IC50()
...	Parameters for plot()

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)

show.prior	would the prior 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 a 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

```
## 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=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,
  adaptive = TRUE,
  likelihood=dnormx, n.chains=1, n.adapt=0, thin=1, trace=1)

#####
## Example with transform
#####

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")

```

```

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(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,

```

```
        'm2' = 1.3,
        's2' = 0.1,
        'p' = 0.5), stringsAsFactors = FALSE,
        row.names=c('m1', 's1', 'm2', 's2', 'p'))

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\(\)](#), [show_name\(\)](#)

Examples

```
## 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

```
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),
  names = NULL,
  add = FALSE
)
```

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 absolut position of the positive error bar for x. Recycled if necessary.
x.minus	The absolut position of the negative error bar for x. 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 nagative 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 filed polygon.
errbar.y.polygon.list	List of parameters to be used for polygon.
names	The names of the points to be used with show_name().
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\(\)](#), [show_name\(\)](#)

Examples

```

## 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=1:100, 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=1:100, 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

```

## 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

```
## 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)
```

print.cutter

Print results of cutter that best describe distribution

Description

Print the estimates of cut distribution.

Usage

```
## S3 method for class 'cutter'
print(x, silent = FALSE, ...)
```

Arguments

x	A result file generated by cutter
silent	If TRUE does not show the output
...	Not used

Details

print.cutter plot result of cutter

Value

Nothing

Author(s)

Marc Girondot

See Also

Other Distributions: [cutter\(\)](#), [dbeta_new\(\)](#), [dcutter\(\)](#), [dgamma\(\)](#), [logLik.cutter\(\)](#), [plot.cutter\(\)](#), [rcutter\(\)](#)

Examples

```
## Not run:
library(HelpersMG)
# -----
# right censored distribution with gamma distribution
# -----
# Detection limit
DL <- 100
# Generate 100 random data from a gamma distribution
obc <- rgamma(100, scale=20, shape=2)
# remove the data below the detection limit
obc[obc<DL] <- +Inf
# search for the parameters the best fit these censored data
result <- cutter(observations=obc, upper_detection_limit=DL,
                 cut_method="censored")

result
plot(result, xlim=c(0, 150), breaks=seq(from=0, to=150, by=10))
# -----
# The same data seen as truncated data with gamma distribution
# -----
obc <- obc[is.finite(obc)]
# search for the parameters the best fit these truncated data
```

```

result <- cutter(observations=obc, upper_detection_limit=DL,
                cut_method="truncated")
result
plot(result, xlim=c(0, 150), breaks=seq(from=0, to=150, by=10))
# -----
# left censored distribution with gamma distribution
# -----
# Detection limit
DL <- 10
# Generate 100 random data from a gamma distribution
obc <- rgamma(100, scale=20, shape=2)
# remove the data below the detection limit
obc[obc<DL] <- -Inf
# search for the parameters the best fit these truncated data
result <- cutter(observations=obc, lower_detection_limit=DL,
                cut_method="censored")
result
plot(result, xlim=c(0, 200), breaks=seq(from=0, to=200, by=10))
# -----
# left and right censored distribution
# -----
# Generate 100 random data from a gamma distribution
obc <- rgamma(100, scale=20, shape=2)
# Detection limit
LDL <- 10
# remove the data below the detection limit
obc[obc<LDL] <- -Inf
# Detection limit
UDL <- 100
# remove the data below the detection limit
obc[obc>UDL] <- +Inf
# search for the parameters the best fit these censored data
result <- cutter(observations=obc, lower_detection_limit=LDL,
                upper_detection_limit=UDL,
                cut_method="censored")
result
plot(result, xlim=c(0, 150), col.DL=c("black", "grey"),
      col.unobserved=c("green", "blue"),
      breaks=seq(from=0, to=150, by=10))
# -----
# Example with two values for lower detection limits
# corresponding at two different methods of detection for example
# with gamma distribution
# -----
obc <- rgamma(50, scale=20, shape=2)
# Detection limit for sample 1 to 50
LDL1 <- 10
# remove the data below the detection limit
obc[obc<LDL1] <- -Inf
obc2 <- rgamma(50, scale=20, shape=2)
# Detection limit for sample 1 to 50
LDL2 <- 20
# remove the data below the detection limit

```

```

obc2[obc2<LDL2] <- -Inf
obc <- c(obc, obc2)
# search for the parameters the best fit these censored data
result <- cutter(observations=obc,
                 lower_detection_limit=c(rep(LDL1, 50), rep(LDL2, 50)),
                 cut_method="censored")

result
# It is difficult to choose the best set of colors
plot(result, xlim=c(0, 150), col.dist="red",
      col.unobserved=c(rgb(red=1, green=0, blue=0, alpha=0.1),
                        rgb(red=1, green=0, blue=0, alpha=0.2)),
      col.DL=c(rgb(red=0, green=0, blue=1, alpha=0.5),
               rgb(red=0, green=0, blue=1, alpha=0.9)),
      breaks=seq(from=0, to=200, by=10))

# -----
# left censored distribution comparison of normal, lognormal and gamma
# -----
# Detection limit
DL <- 10
# Generate 100 random data from a gamma distribution
obc <- rgamma(100, scale=20, shape=2)
# remove the data below the detection limit
obc[obc<DL] <- -Inf
# search for the parameters the best fit these truncated data
result_gamma <- cutter(observations=obc, lower_detection_limit=DL,
                      cut_method="censored", distribution="gamma")
result_gamma
plot(result_gamma, xlim=c(0, 250), breaks=seq(from=0, to=250, by=10))

result_lognormal <- cutter(observations=obc, lower_detection_limit=DL,
                           cut_method="censored", distribution="lognormal")
result_lognormal
plot(result_lognormal, xlim=c(0, 250), breaks=seq(from=0, to=250, by=10))

result_normal <- cutter(observations=obc, lower_detection_limit=DL,
                       cut_method="censored", distribution="normal")
result_normal
plot(result_normal, xlim=c(0, 250), breaks=seq(from=0, to=250, by=10))

compare_AICc(gamma=result_gamma,
             lognormal=result_lognormal,
             normal=result_normal)

# -----
# Test for similarity in gamma left censored distribution between two
# datasets
# -----
obc1 <- rgamma(100, scale=20, shape=2)
# Detection limit for sample 1 to 50
LDL <- 10
# remove the data below the detection limit
obc1[obc1<LDL] <- -Inf
obc2 <- rgamma(100, scale=10, shape=2)
# remove the data below the detection limit

```

```

obc2[obc2<LDL] <- -Inf
# search for the parameters the best fit these censored data
result1 <- cutter(observations=obc1,
                 distribution="gamma",
                 lower_detection_limit=LDL,
                 cut_method="censored")

logLik(result1)
plot(result1, xlim=c(0, 200),
     breaks=seq(from=0, to=200, by=10))
result2 <- cutter(observations=obc2,
                 distribution="gamma",
                 lower_detection_limit=LDL,
                 cut_method="censored")

logLik(result2)
plot(result2, xlim=c(0, 200),
     breaks=seq(from=0, to=200, by=10))
result_tot1 <- cutter(observations=c(obc1, obc2),
                    distribution="gamma",
                    lower_detection_limit=LDL,
                    cut_method="censored")

logLik(result_tot1)
plot(result_tot1, xlim=c(0, 200),
     breaks=seq(from=0, to=200, by=10))

compare_AICc(Separate=list(result1, result2),
             Common=result_tot1, factor.value=1)
compare_BIC(Separate=list(result1, result2),
            Common=result_tot1, factor.value=1)

## 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. $0 < \text{prob} \leq 1$.
mu	alternative parametrization via mean.
lower.tail	logical; if TRUE (default), probabilities are $P[X \leq x]$, 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

```
## 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)

## End(Not run)
```

qSnbinom	<i>Quantile function for the sum of random variable with negative binomial distributions.</i>
----------	---

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. $0 < \text{prob} \leq 1$.
mu	alternative parametrization via mean.
lower.tail	logical; if TRUE (default), probabilities are $P[X \leq x]$, 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

See Also

Other Distribution of sum of random variable with negative binomial distributions: [dSnbinom\(\)](#), [pSnbinom\(\)](#), [rSnbinom\(\)](#)

Examples

```
## Not run:
alpha <- c(2.1, 2.05, 2)
mu <- c(10, 30, 20)
q <- qSnbinom(p=0.1, size=alpha, mu=mu, lower.tail = TRUE)

## End(Not run)
```

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

Usage

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

Arguments

<code>object</code>	A object obtained using <code>lmer</code> from package <code>lme4</code>
<code>factorname</code>	Either <code>NULL</code> , or a character vector of length 1
<code>coef.indices</code>	Either <code>NULL</code> , or a numeric vector of length at least 3
<code>dispersion</code>	An optional scalar multiplier for the covariance matrix, to cope with overdispersion for example
<code>...</code>	Other arguments to pass to <code>qvcalc.default</code>

Details

`qvlmer` is Quasi Variances for lmer Model Coefficients

Value

A list of class `qv`.

Author(s)

marc.girondot@u-psud.fr

References

- Easton, D. F, Peto, J. and Babiker, A. G. A. G. (1991) Floating absolute risk: an alternative to relative risk in survival and case-control analysis avoiding an arbitrary reference group. *Statistics in Medicine* 10, 1025–1035.
- Firth, D. (2000) Quasi-variances in Xlisp-Stat and on the web. *Journal of Statistical Software* 5.4, 1–13. At <http://www.jstatsoft.org>
- Firth, D. (2003) Overcoming the reference category problem in the presentation of statistical models. *Sociological Methodology* 33, 1–18.
- Firth, D. and de Mezezes, R. X. (2004) Quasi-variances. *Biometrika* 91, 65–80.
- McCullagh, P. and Nelder, J. A. (1989) *Generalized Linear Models*. London: Chapman and Hall.
- Menezes, R. X. de (1999) More useful standard errors for group and factor effects in generalized linear models. D.Phil. Thesis, Department of Statistics, University of Oxford.
- Ridout, M.S. (1989). Summarizing the results of fitting generalized linear models to data from designed experiments. In: *Statistical Modelling: Proceedings of GLIM89 and the 4th International Workshop on Statistical Modelling held in Trento, Italy, July 17–21, 1989* (A. Decarli et al., eds.), pp 262–269. New York: Springer.

Examples

```
## 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)
```

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

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

hist(n)

## End(Not run)
```

RandomFromHessianOrMCMC

Random numbers based on Hessian matrix or MCMC

Description

If it is very long, use silent parameter to check if something goes wrong.

If replicates is null or is 0, or if method is NULL, parameters are just copied into data.frame.

Usage

```
RandomFromHessianOrMCMC(
  se = NULL,
  Hessian = NULL,
  mcmc = NULL,
  chain = 1,
  regularThin = TRUE,
  MinMax = NULL,
  fitted.parameters = NULL,
  fixed.parameters = NULL,
```

```

    method = NULL,
    probs = c(0.025, 0.5, 0.975),
    replicates = 10000,
    fn = NULL,
    silent = FALSE,
    ParTofn = "par",
    ...
)

```

Arguments

<code>se</code>	A named vector with SE of parameters
<code>Hessian</code>	An Hessian matrix
<code>mcmc</code>	A result from <code>MHalgogen()</code>
<code>chain</code>	MCMC chain to be used
<code>regularThin</code>	If TRUE, use regular thin for MCMC
<code>MinMax</code>	A data.frame with at least two columns: Min and Max and rownames being the variable names
<code>fitted.parameters</code>	The fitted parameters
<code>fixed.parameters</code>	The fixed parameters
<code>method</code>	Can be NULL, "SE", "Hessian", "MCMC", or "PseudoHessianFromMCMC"
<code>probs</code>	Probability for quantiles
<code>replicates</code>	Number of replicates to generate the randoms
<code>fn</code>	The function to apply to each replicate
<code>silent</code>	Should the function display some information
<code>ParTofn</code>	Name of the parameter to send random values to fn
<code>...</code>	Parameters send to fn function

Details

`RandomFromHessianOrMCMC` returns random numbers based on Hessian matrix or MCMC

Value

Returns a list with three data.frames named `random`, `fn`, and `quantiles`

Author(s)

Marc Girondot

Examples

```

## 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,
                             method="Hessian")$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,
                              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,
                    parameters_name = "par",
                    likelihood=fitnorm, n.chains=1, n.adapt=100, thin=1, trace=1)
df <- RandomFromHessianOrMCMC(mcmc=mcmc_run, fitted.parameters=NULL,
                             method="MCMC")$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 fn
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,
                             method="Hessian",
                             fn=function(par) (par["a"]*(x)+par["b"]))

plot(1:100, val)
lines(1:100, df$quantiles["50%", ])
lines(1:100, df$quantiles["2.5%", ], lty=2)

```

```
lines(1:100, df$quantiles["97.5%", ], lty=2)

## End(Not run)
```

rcutter	<i>Random values of unobserved values of cut distribution.</i>
---------	--

Description

Return n random numbers.

It can be used to get the posterior predictive distribution; see example.

If random_method is "ML", the parameter values obtained using maximum likelihood are used.

If random_method is "medianMCMC", the parameter values obtained using median of posterior distribution are used.

If random_method is "MCMC", the parameter values are one sample of the MCMC posterior distribution.

if observed_detection_limit is set to TRUE, the number of random number is equal to the number of observations; n is not used.

rcutter is the abbreviation for random-cutter.

Usage

```
rcutter(
  cutter = stop("A result of cutter() must be provided"),
  n = 1,
  lower_detection_limit = NULL,
  upper_detection_limit = NULL,
  method_cut = c("censored", "truncated"),
  observed_detection_limit = FALSE,
  random_method = c("medianMCMC", "MCMC", "ML"),
  index_mcmc = NULL
)
```

Arguments

cutter	The fitted model obtained with cutter()
n	number of random numbers
lower_detection_limit	The lower detection limit
upper_detection_limit	The upper detection limit
method_cut	What method is used to cut the distribution: "censored", "truncated"?
observed_detection_limit	If TRUE, will use the pattern of detection limit as in observations
random_method	How to get parameters; it can be "ML", "medianMCMC", or "MCMC"
index_mcmc	For MCMC random_method, the index of data to be used.

Details

rcutter returns random values based on fitted distribution with cut.

Value

A vector with the random numbers.

Author(s)

Marc Girondot

See Also

Other Distributions: [cutter\(\)](#), [dbeta_new\(\)](#), [dcutter\(\)](#), [dggamma\(\)](#), [logLik.cutter\(\)](#), [plot.cutter\(\)](#), [print.cutter\(\)](#)

Examples

```
## Not run:
library(HelpersMG)
# -----
# right censored distribution with gamma distribution
# -----
# Detection limit
DL <- 100
# Generate 100 random data from a gamma distribution
obc <- rgamma(100, scale=20, shape=2)
# remove the data below the detection limit
obc[obc>DL] <- +Inf
# search for the parameters the best fit these censored data
result <- cutter(observations=obc, upper_detection_limit=DL,
                 cut_method="censored")

result
# Posterior predictive distribution
r <- rcutter(cutter=result, upper_detection_limit=DL, n=100)
hist(r)
# -----
# left censored distribution with gamma distribution
# -----
# Detection limit
DL <- 10
# Generate 100 random data from a gamma distribution
obc <- rgamma(100, scale=20, shape=2)
# remove the data below the detection limit
obc[obc<DL] <- -Inf
# search for the parameters the best fit these truncated data
result <- cutter(observations=obc, lower_detection_limit=DL,
                 cut_method="censored")

result
plot(result, breaks=seq(from=0, to=200, by=10))
r <- rcutter(cutter=result, n=100)
hist(r, breaks=seq(from=0, to=200, by=10))
```

```

r <- rcutter(cutter=result, lower_detection_limit=DL, n=100)
hist(r, breaks=seq(from=0, to=250, by=10))
# With censored method, some values are replaced with +Inf or -Inf
any(is.infinite(r))
r <- rcutter(cutter=result, upper_detection_limit=DL, n=100,
             method_cut="truncated")
# With truncated method, the values below LDL or upper UDL are not present
any(is.infinite(r))
hist(r, breaks=seq(from=0, to=10, by=0.25))
r <- rcutter(cutter=result, observed_detection_limit=TRUE)
hist(r, breaks=seq(from=0, to=300, by=10))

## End(Not run)

```

read_folder	<i>Read files present in a folder and creates a list with the content of these files</i>
-------------	--

Description

To create a list, the syntax is:

```
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

```

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
contentaslist <- read_folder(folder=".", wildcard="*.csv", read=read.csv2)
# Read all the files from the current folder/directory
contentaslist <- read_folder(folder=".", wildcard="*.*", read=read.csv2)
# Read two files from the current folder/directory
files <- c("filename1.csv", "filename2.csv")
contentaslist <- 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%", ])
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")
```

RM_add

Create a results management or add a value in a results management to an object

Description

Return original object with a new value or a new results management.

Usage

```
RM_add(
  x = stop("An object with results management must be provided"),
  RM = "RM",
  RMname = stop("A results management name must be provided"),
  valuname = NULL,
  value = NULL
)
```

Arguments

x	The object to add a results management or a result in a results management
RM	The name of results management stored
RMname	The name of the results management to be modified or created
valuename	The name of the new value to be added
value	The value to be added

Details

RM_add adds a results management or a value in results management to an object

Value

The original object with a new value in a results management object or a new results management

Author(s)

Marc Girondot

See Also

Other Results Management: [RM_delete\(\)](#), [RM_duplicate\(\)](#), [RM_get\(\)](#), [RM_list\(\)](#)

Examples

```
## Not run:
library("HelpersMG")
# Let an object of class objclass being created
obj <- list(A=100, name="My object")
class(obj) <- "objclass"
# And now I create a RM to this object
obj <- RM_add(x=obj, RMname="NewAnalysis1")
RM_list(obj)
obj <- RM_add(x=obj, RMname="NewAnalysis2")
RM_list(obj)
obj <- RM_add(x=obj, RMname="NewAnalysis2", valuename="V1", value=100)
RM_get(x=obj, RMname="NewAnalysis2", valuename="V1")
obj <- RM_add(x=obj, RMname="NewAnalysis2", valuename="V1", value=200)
RM_get(x=obj, RMname="NewAnalysis2", valuename="V1")
obj <- RM_add(x=obj, RMname="NewAnalysis2", valuename="V2", value=300)
RM_get(x=obj, RMname="NewAnalysis2", valuename="V2")
RM_list(obj)

## End(Not run)
```

RM_delete	<i>Delete a results management or a result within a results management from an object</i>
-----------	---

Description

Return the original object with the deleted results management or result.

Usage

```
RM_delete(  
  x = stop("An object with results management must be provided"),  
  RM = "RM",  
  RMname = stop("A name must be provided"),  
  valuename = NULL  
)
```

Arguments

x	The object to delete a results management
RM	The name of results management stored
RMname	The name of the result that will be deleted or its rank
valuename	The name of the result that will be deleted

Details

RM_delete deletes a results management or a result within a results management from an object

Value

The original object with the deleted results management

Author(s)

Marc Girondot

See Also

Other Results Management: [RM_add\(\)](#), [RM_duplicate\(\)](#), [RM_get\(\)](#), [RM_list\(\)](#)

Examples

```
## Not run:  
library("HelpersMG")  
# Let an object of class objclass being created  
obj <- list(A=100, name="My object")  
class(obj) <- "objclass"  
# And now I create a RM to this object
```

```

obj <- RM_add(x=obj, RMname="NewAnalysis1")
obj <- RM_add(x=obj, RMname="NewAnalysis2")
RM_list(obj)
obj <- RM_delete(x=obj, RMname="NewAnalysis1")
RM_list(obj)
obj <- RM_delete(x=obj, RMname=1)
RM_list(obj)
obj <- RM_add(x=obj, RMname="NewAnalysis1", valuename="V1", value=100)
RM_list(obj)
RM_get(x=obj, RMname="NewAnalysis1", valuename="V1")
obj <- RM_add(x=obj, RMname="NewAnalysis1", valuename="V2", value=200)
RM_get(x=obj, RMname="NewAnalysis1", valuename="V2")
obj <- RM_delete(x=obj, RMname="NewAnalysis1", valuename="V1")
RM_get(x=obj, RMname="NewAnalysis1", valuename="V1")
RM_get(x=obj, RMname="NewAnalysis1", valuename="V2")

## End(Not run)

```

RM_duplicate

Duplicate a results management within an object.

Description

RM_duplicate duplicates a results management within an object.

Usage

```

RM_duplicate(
  x = stop("An object with results management must be provided"),
  RM = "RM",
  RMnamefrom = 1,
  RMnameto = 2
)

```

Arguments

x	The object to duplicate a results management
RM	The name of results management stored
RMnamefrom	The name of the results management to be duplicated
RMnameto	The new name of the results management

Details

RM_duplicate duplicates a results management within an object

Value

The original object with a duplicated results management.

Author(s)

Marc Girondot

See Also

Other Results Management: [RM_add\(\)](#), [RM_delete\(\)](#), [RM_get\(\)](#), [RM_list\(\)](#)

Examples

```
## Not run:
library("HelpersMG")
# Let an object of class objclass being created
obj <- list(A=100, name="My object")
class(obj) <- "objclass"
# And now I create a RM to this object
obj <- RM_add(x=obj, RMname="NewAnalysis1")
RM_list(obj)
obj <- RM_duplicate(x=obj, RMnamefrom="NewAnalysis1", RMnameto="NewAnalysis2")
RM_list(obj)

## End(Not run)
```

RM_get

Get a value in a results management to an object

Description

Return the value valurname of the results management RMname.

Usage

```
RM_get(
  x = stop("An object with results managment must be provided"),
  RM = "RM",
  RMname = stop("A results managment name must be provided"),
  valurname = NULL
)
```

Arguments

x	The object in which to get a result in a results managment
RM	The name of results management stored
RMname	The name of the results management to be read
valurname	The name of the value to be read

Details

RM_get gets a value in results management to an object

Value

Return a value in a results management object

Author(s)

Marc Girondot

See Also

Other Results Management: [RM_add\(\)](#), [RM_delete\(\)](#), [RM_duplicate\(\)](#), [RM_list\(\)](#)

Examples

```
## Not run:
library("HelpersMG")
# Let an object of class objclass being created
obj <- list(A=100, name="My object")
class(obj) <- "objclass"
# And now I create a RM to this object
obj <- RM_add(x=obj, RMname="NewAnalysis1")
RM_list(obj)
obj <- RM_add(x=obj, RMname="NewAnalysis2")
RM_list(obj)
obj <- RM_add(x=obj, RMname="NewAnalysis2", valuename="V1", value=100)
RM_get(x=obj, RMname="NewAnalysis2", valuename="V1")

## End(Not run)
```

RM_list

Return the list of results management of an object.

Description

RM_list returns the list of results management of an object.

Usage

```
RM_list(
  x = stop("An object with results management must be provided"),
  RM = "RM",
  silent = FALSE,
  max.level = FALSE
)
```

Arguments

x	The object to add a results management
RM	The name of results management stored
silent	Should the results be shown ?
max.level	If TRUE, will return all list element of the objects

Details

RM_list returns the list of results management of an object

Value

A list with the names of results stored in an object

Author(s)

Marc Girondot

See Also

Other Results Management: [RM_add\(\)](#), [RM_delete\(\)](#), [RM_duplicate\(\)](#), [RM_get\(\)](#)

Examples

```
## Not run:
library("HelpersMG")
# Let an object of class objclass being created
obj <- list(A=100, name="My object")
class(obj) <- "objclass"
# And now I create a RM to this object
obj <- RM_add(x=obj, RMname="NewAnalysis1")
RM_list(obj)
obj <- RM_add(x=obj, RMname="NewAnalysis2")
RM_list(obj)
obj <- RM_add(x=obj, RMname="NewAnalysis2", valuename="V1", value=100)
RM_get(x=obj, RMname="NewAnalysis2", valuename="V1")
obj <- RM_add(x=obj, RMname="NewAnalysis2", valuename="V1", value=200)
RM_get(x=obj, RMname="NewAnalysis2", valuename="V1")
obj <- RM_add(x=obj, RMname="NewAnalysis2", valuename="V2", value=300)
RM_get(x=obj, RMname="NewAnalysis2", valuename="V2")
RM_list(obj)
rmlist <- RM_list(obj, max.level=TRUE)
rmlist

## End(Not run)
```

`rnbinom_new`*Random numbers for the negative binomial distribution.*

Description

See `rbinom`.

Usage

```
rnbinom_new(n, size = NULL, prob = NULL, mu = NULL, sd = NULL, var = NULL)
```

Arguments

<code>n</code>	number of observations.
<code>size</code>	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.
<code>prob</code>	probability of success in each trial. $0 < \text{prob} \leq 1$.
<code>mu</code>	alternative parametrization via mean.
<code>sd</code>	alternative parametrization via standard deviation.
<code>var</code>	alternative parametrization via variance.

Details

`rnbinom_new` returns random numbers for the negative binomial distribution

Value

Random numbers for the negative binomial distribution

Author(s)

Marc Girondot

Examples

```
## Not run:  
library("HelpersMG")  
set.seed(1)  
x <- rnbinom_new(n=1000, prob=6.25/(5+6.25), size=6.25)  
mean(x)  
sd(x)  
set.seed(1)  
x <- rnbinom_new(n=1000, mu=5, sd=3)  
mean(x)  
sd(x)  
set.seed(1)
```



```

x <- rnbinom_new(n=1000, mu=5, var=3^2)
mean(x)
sd(x)
set.seed(1)
x <- rnbinom_new(n=1000, mu=5, size=6.25)
mean(x)
sd(x)
set.seed(1)
x <- rnbinom_new(n=1000, size=6.25, var=3^2)
mean(x)
sd(x)
set.seed(1)
x <- rnbinom_new(n=1000, prob=6.25/(5+6.25), var=3^2)
mean(x)
sd(x)
# Example of wrong parametrization
set.seed(1)
x <- rnbinom_new(n=1000, sd=3, var=3^2)
set.seed(1)
x <- rnbinom_new(n=1000, mu=10, var=3^2)

## End(Not run)

```

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. $0 < \text{prob} \leq 1$.
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

```
## 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[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")

## 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(x = NULL, y = NULL)
```

Arguments

x	The position in x as relative position
y	The position in y as relative position

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\(\)](#), [show_name\(\)](#)

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
plot(x=1:100, y=sin(1:100), type="l", bty="n", xlim=c(1,200), xlab="", ylab="")
text(x=ScalePreviousPlot()$xlim["label"], y=ScalePreviousPlot()$ylim["center"],
     xpd=TRUE, "Legend for Y axes", pos=3, srt=90)
text(x=ScalePreviousPlot()$xlim["center"], y=ScalePreviousPlot()$ylim["label"],
     xpd=TRUE, "Legend for X axes", pos=1)
Example to plot legend always in the same place
layout(1:2)
plot(x=1:100, y=sin(1:100), type="l", bty="n", xlim=c(1,200), xlab="", ylab="")
text(x=ScalePreviousPlot(x=0.95, y=0.05)$x,
     y=ScalePreviousPlot(x=0.95, y=0.05)$y,
     labels="A", cex=2)
plot(x=0:1, y=0:1, type="p", bty="n")
text(x=ScalePreviousPlot(x=0.95, y=0.05)$x,
     y=ScalePreviousPlot(x=0.95, y=0.05)$y,
     labels="B", cex=2)

## End(Not run)
```

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 (2000) will be used to generate the Cholesky matrix.

Usage

```
SEfromHessian(a, hessian = FALSE, silent = FALSE)
```

Arguments

a	An Hessian matrix
hessian	If TRUE, return a list with the hessian and SE
silent	If TRUE, report some problems

Details

SEfromHessian returns standard error of parameters based on Hessian matrix

Value

SEfromHessian returns a vector with standard errors

Author(s)

Marc Girondot

References

Gill J. and G. King 2004. What to do when your Hessian is not invertible: Alternatives to model respecification in nonlinear estimation. *Sociological Methods & Research* 33: 54-87.

Rebonato and Jackel, "The most general methodology for creating a valid correlation matrix for risk management and option pricing purposes", *Journal of Risk*, Vol 2, No 2, 2000.

Examples

```
## 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

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

Arguments

<code>...</code>	Series of data (at least two or data are in a table with series in different rows)
<code>criterion</code>	Which criterion is used for model selection. can be AIC, AICc or BIC
<code>var.equal</code>	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

Girondot, M., Guillon, J.-M., 2018. The w-value: An alternative to t- and X2 tests. *Journal of Biostatistics & Biometrics* 1, 1-4.

See Also

Other w-value functions: [compare\(\)](#), [contingencyTable.compare\(\)](#)

Examples

```
## 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="", xlim=c(0, 1), las=1, breaks = (0:20)/20,
     freq=FALSE, xlab = expression(italic("p")*"~value"))
hist(w, main="", xlim=c(0, 1), las=1, breaks = (0:20)/20,
     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,
      col=c("black", "black", "red", "red"), bty="n", lty=c(1, 0, 1, 0))

segments(x0=min(x), x1=max(x), y0=0.05, y1=0.05, lty=2)
par(xpd = TRUE)

```

```
text(x=13.05, y=0.05, labels = "0.05", pos=4)
## End(Not run)
```

show_name	<i>Show the name of a point</i>
-----------	---------------------------------

Description

Click on a point in plot region and it will tell you what is the point.

Usage

```
show_name(
  points = NULL,
  x = NULL,
  y = NULL,
  names = NULL,
  col = "red",
  silent = FALSE
)
```

Arguments

points	A list with x, y and names elements
x	The x coordinates
y	The y coordinates.
names	The names of the points
col	Color of the legend.
silent	TRUE or FALSE

Details

Show the name of a point

Value

Name of the point

Author(s)

Marc Girondot

See Also

plot_errorbar

Other plot and barplot functions: [ScalePreviousPlot\(\)](#), [barplot_errbar\(\)](#), [plot_add\(\)](#), [plot_errbar\(\)](#)

Examples

```
## Not run:
k <- 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))
show_name(k)
k <- plot_errbar(1:10, rnorm(10, 1, 2),
  xlab="axe x", ylab="axe y", bty="n", xlim=c(1,10),
  errbar.x=2, errbar.y=rnorm(10, 1, 0.1),
  names=LETTERS[1:10])
show_name(k)
k <- plot_errbar(1:10, rnorm(10, 1, 2),
  xlab="axe x", ylab="axe y", bty="n", xlim=c(1,10),
  errbar.x=2, errbar.y=rnorm(10, 1, 0.1))
show_name(k, names=LETTERS[1:10])

## End(Not run)
```

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)
```

specify_decimal

Return a number as character with specified number of decimals

Description

Return a number as character with specified number of decimals. If a is a matrix, it will return a matrix of the same size and the same attributes.

Usage

```
specify_decimal(x, decimals = NULL, decimal.point = ".")
```

Arguments

x	The numbers to be formatted
decimals	Number of decimals to print
decimal.point	Character to be used as decimal point

Details

specify_decimals format a number with specified number of decimals

Value

A character

Author(s)

Marc Girondot

Examples

```
specify_decimal(x=pi, decimal.point=".")
specify_decimal(x=pi, decimals=4, decimal.point=".")
specify_decimal(x=c(pi, exp(1)), decimals=3, decimal.point=",")
specify_decimal(x=c(pi, exp(1)), decimal.point=",")
specify_decimal(x=c(pi*10, pi, pi/10, pi/100, pi/1000))
specify_decimal(x=c(pi=pi), decimal.point=".")
specify_decimal(x=matrix(pi*1:4, ncol=2), decimal.point=".")
m <- matrix(pi*1:4, ncol=2)
rownames(m) <- c("A", "B")
colnames(m) <- c("C", "D")
specify_decimal(x=m, decimal.point=".")
```

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

Description

Summary for the result of a mcmcComposite object.

Usage

```
## 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

```
## 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)
```

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

<code>date</code>	A vector with the time at which sun fates are needed
<code>latitude</code>	The latitude at which estimate the sun fates
<code>longitude</code>	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

Teets, D.A. 2003. Predicting sunrise and sunset times. *The College Mathematics Journal* 34(4):317-321.

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	<i>Plot a female symbol in the plotting region</i>
---------------	--

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
```

```
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)
```

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\(\)](#)

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
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

```
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

Annual tide information.

The columns are: Location, Longitude, Latitude, Phase, `DateTime.local`, `DateTime.UTC`, `Tide.meter`
 This function uses an API linking xtide software (<https://flaterco.com/xtide/>) with `tide.info()` function.

You must have a working internet connection for this function.

Usage

```
tide.info(  
  location = NULL,  
  year = 2021,  
  longitude = NULL,  
  latitude = NULL,  
  tz = NULL  
)
```

Arguments

location	Textual information about location name
year	Year to get the calendar
longitude	Longitude to search for
latitude	Latitude to search for
tz	The time zone to format the local time

Details

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

Value

Return a data.frame with annual tide calendar.

Author(s)

Marc Girondot <marc.girondot@gmail.com>

See Also

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

Examples

```
## Not run:  
library("HelpersMG")  
Location <- "Les Hattes"  
Year <- 2010  
tide <- tide.info(Location, Year, tz="America/Cayenne")  
plot(tide[, "DateTime.local"], tide[, "Tide.meter"],  
     type="l", bty="n", las=1,  
     main=tide[1, "Location"],  
     xlab=as.character(Year), ylab="Tide level in meter")  
  
Location <- "Hawaii"  
Year <- 2010  
tide <- tide.info(Location, Year)
```

```

Location <- "Hanamaulu Bay, Kauai Island, Hawaii"
Year <- 2010
tide <- tide.info(Location, Year)
plot(tide[, "DateTime.local"], tide[, "Tide.meter"],
     type="l", bty="n", las=1,
     main=tide[1, "Location"],
     xlab=as.character(Year), ylab="Tide level in meter")

tide <- tide.info(year=2010, longitude=-32, latitude=-4)
library(maps)
map(database = "world", regions = "Brazil", asp=1,
     xlim=c(-80, -30), ylim=c(-33, 5))
points(tide[1, "Longitude"], tide[1, "Latitude"], col="red", pch=19)
points(-32, -4, col="blue", pch=19)
axis(1)
axis(2, las=1)

# Show the locations with data
library(maps)
map(xlim=c(-180, 180), ylim=c(-90, 90))
title("Locations with harmonics data")
axis(1, at=seq(from=-180, to=180, by=45))
axis(2, las=1, at=seq(from=-90, to=90, by=15))
points(getFromNamespace(x="tide_location", ns="HelpersMG")[, c("longitude")],
       getFromNamespace(x="tide_location", ns="HelpersMG")[, c("latitude")],
       pch=".", col="red", cex=2)

# Another example
tikei_lon <- (-144.5465183)
tikei_lat <- -14.9505897
Year <- 2021
tikei_tide <- tide.info(year=Year, longitude=tikei_lon, latitude=tikei_lat)
plot(tikei_tide[, "DateTime.local"], tikei_tide[, "Tide.meter"],
     type="l", bty="n", las=1,
     main=tikei_tide[1, "Location"],
     xlab=as.character(Year), ylab="Tide level in meter")

## 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

```
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

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

```
tnirp("      mu  mu_season      OTN      p1.09      p1.10      p1.11
4.63215947 10.78627511 0.36108497 0.08292101 -0.52558196 -0.76430859
  p1.12      p1.13      p1.14      p1.15      p1.16      p1.17
-0.75186542 -0.57632291 -0.58017174 -0.57048696 -0.56234135 -0.80645122
  p1.18      p1.19      p1.20      p1.21      p1.22      p1.23
-0.77752524 -0.80909494 -0.56920540 -0.55317302 0.45757298 -0.64155368
  p1.24      p1.25      p1.26      p1.27      p1.28      p1.29
-0.59119637 -0.66006794 -0.66582399 -0.66772684 -0.67351412 -0.66941992
  p1.30      p1.31      p1.32      p1.33      p1.34      p1.35
-0.67038245 -0.68938726 -0.68889078 -0.68779016 -0.68604629 -0.68361820
  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 2.30705457 2.18740019 2.32305811 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
2.26869058 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 ")
tnirp(" 27.89 289.99
90.56", named=FALSE)
```

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 = getOption("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.
mc.preschedule	if set to TRUE then the computation is first divided to (at most) as many jobs as 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

```
## 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
}
# Note that parallel computing is not always the best solution !
(tp <- system.time({
  m <- lapply(X=x, FUN=funx)
}))
(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)

# Here an error is generated because environment was not exported when parLapplyLB is used
pp <- runif(100)
x <- 1:100
u <- universalmclapply(x, FUN=funx1, forking=FALSE)

# here no error is generated because the variable pp is exported
pp <- runif(100)
x <- 1:100
u <- universalmclapply(x, FUN=funx1, forking=FALSE,
  clusterExport=list(varlist=c("pp"), envir=environment()))

### An example using clusterEvalQ
asc("a") # asc() is a function from packages HelpersMG
```

```

funx2 <- function(y) {asc("a")*10}
# In unix, the loaded packages are visible from all cores
x <- 1:100
u <- universalmcapply(x, FUN=funx2, forking=TRUE)
# In windows, the loaded packages are not visible from all cores
x <- 1:100
u <- universalmcapply(x, FUN=funx2, forking=FALSE)
# In windows, the loaded packages are not visible from all cores
x <- 1:100
u <- universalmcapply(x, FUN=funx2, forking=FALSE,
clusterEvalQ=list(expr=expression(library(HelpersMG)))
)

### If package pbapply is available, progress bar can be shown
m <- universalmcapply(X=x, FUN=funx, forking=FALSE, progressbar=TRUE)
m <- universalmcapply(X=x, FUN=funx, forking=TRUE, progressbar=TRUE)

## End(Not run)

```

 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. It was written to simplify downloading of file. It does not use the true `wget` function (<https://www.gnu.org/software/wget/>) which is much more complex but also powerful.

Usage

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

Arguments

<code>url</code>	The url where to download file
<code>...</code>	The parameters send to <code>download.file()</code>

Details

wget download a file from internet and save it locally

Value

Nothing

Author(s)

Marc Girondot

Examples

```
## 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

- * **AIC functions**
 - logLik.compareAIC, 77
- * **AIC**
 - compare_AIC, 19
 - compare_AICc, 20
 - compare_BIC, 22
 - ExtractAIC.glm, 48
 - FormatCompareAIC, 53
- * **Characters**
 - asc, 12
 - chr, 17
 - d, 35
 - tnirp, 155
- * **Distribution of sum of random variable with negative binomial distributions**
 - dSnbinom, 42
 - pSnbinom, 117
 - qSnbinom, 119
 - rSnbinom, 137
- * **Distributions**
 - cutter, 27
 - dbeta_new, 36
 - dcutter, 37
 - dgamma, 39
 - logLik.cutter, 78
 - plot.cutter, 96
 - print.cutter, 113
 - rcutter, 125
- * **Iconography of correlations**
 - IC_clean_data, 56
 - IC_correlation_simplify, 58
 - IC_threshold_matrix, 60
 - plot.IconoCorel, 100
- * **LD50 functions**
 - LD50, 68
 - LD50_MHmcmc, 70
 - LD50_MHmcmc_p, 73
 - logLik.LD50, 79
 - plot.LD50, 102
 - predict.LD50, 112
- * **Lunar**
 - moon.info, 90
- * **Lune**
 - moon.info, 90
- * **Moon**
 - moon.info, 90
- * **Periodic patterns of indices**
 - index.periodic, 62
 - minmax.periodic, 86
 - moon.info, 90
 - sun.info, 148
 - tide.info, 153
- * **Rectangle Regression**
 - RectangleRegression, 128
- * **Results Management**
 - RM_add, 129
 - RM_delete, 131
 - RM_duplicate, 132
 - RM_get, 133
 - RM_list, 134
- * **Symbol**
 - symbol.Female, 150
 - symbol.Male, 151
- * **Tide**
 - tide.info, 153
- * **logit**
 - flexit, 51
 - invlogit, 67
 - logit, 76
- * **mcmcComposite functions**
 - as.mcmc.mcmcComposite, 7
 - as.parameters, 9
 - as.quantiles, 11
 - merge.mcmcComposite, 80
 - MHalgoGen, 82
 - plot.mcmcComposite, 104
 - summary.mcmcComposite, 147

- * **ncdf**
 - format_ncdf, 54
 - ind_long_lat, 64
- * **plot and barplot functions**
 - barplot_errbar, 13
 - plot_add, 109
 - plot_errbar, 110
 - ScalePreviousPlot, 138
 - show_name, 144
- * **w-value functions**
 - compare, 18
 - contingencyTable.compare, 23
 - series.compare, 141
- as.mcmc.mcmcComposite, 7, 10, 12, 81, 83, 105, 147
- as.parameters, 8, 9, 12, 81, 83, 105, 147
- as.quantiles, 8, 10, 11, 81, 83, 105, 147
- asc, 12, 18, 35, 156
- barplot_errbar, 13, 110, 111, 139, 144
- cArrows, 15
- ChangeCoordinate, 16
- chr, 13, 17, 35, 156
- compare, 18, 24, 142
- compare_AIC, 19, 21, 22, 49, 54
- compare_AICc, 20, 20, 22, 49, 54
- compare_BIC, 20, 21, 22, 49, 54
- contingencyTable.compare, 18, 23, 142
- convert.tz, 26
- cutter, 27, 37, 38, 40, 78, 97, 114, 126
- d, 13, 18, 35, 156
- dbeta_new, 29, 36, 38, 40, 78, 97, 114, 126
- dcutter, 29, 37, 37, 40, 78, 97, 114, 126
- dgamma, 29, 37, 38, 39, 78, 97, 114, 126
- DIx, 41
- dSnbinom, 42, 118, 120, 138
- duplicated_packages, 45
- ellipse, 46
- ExtractAIC.glm, 20–22, 48, 54
- fitdistrquantiles, 50
- flexit, 51, 67, 76
- format_ncdf, 54, 65
- FormatCompareAIC, 20–22, 49, 53
- HelpersMG-package, 4
- IC_clean_data, 56, 59, 61, 101
- IC_correlation_simplify, 57, 58, 61, 101
- IC_threshold_matrix, 57, 59, 60, 101
- iCutter, 56
- ind_long_lat, 55, 64
- index.periodic, 62, 87, 90, 149, 154
- inside.search, 65
- invlogit, 53, 67, 76
- LD50, 68, 72, 73, 80, 103, 113
- LD50_MHmcmc, 69, 70, 73, 80, 103, 113
- LD50_MHmcmc_p, 69, 72, 73, 80, 103, 113
- list.packages, 74
- local.search, 75
- logit, 53, 67, 76
- logLik.compareAIC, 77
- logLik.cutter, 29, 37, 38, 40, 78, 97, 114, 126
- logLik.LD50, 69, 72, 73, 79, 103, 113
- merge.mcmcComposite, 8, 10, 12, 80, 83, 105, 147
- MHalgoGen, 8, 10, 12, 81, 82, 105, 147
- minmax.periodic, 63, 86, 90, 149, 154
- modeled.hist, 88
- modifyVector, 89
- moon.info, 63, 87, 90, 149, 154
- MovingWindow, 91
- NagelkerkeScaledR2, 92
- newcompassRose, 93
- newmap.scale, 94
- openwd, 95
- pgamma (dgamma), 39
- plot.cutter, 29, 37, 38, 40, 78, 96, 114, 126
- plot.IconoCorel, 57, 59, 61, 100
- plot.LD50, 69, 72, 73, 80, 102, 113
- plot.mcmcComposite, 8, 10, 12, 81, 83, 104, 147
- plot_add, 14, 109, 111, 139, 144
- plot_errbar, 14, 110, 110, 139, 144
- predict.LD50, 69, 72, 73, 80, 103, 112
- print.cutter, 29, 37, 38, 40, 78, 97, 113, 126
- pSnbinom, 43, 117, 120, 138
- qgamma (dgamma), 39
- qSnbinom, 43, 118, 119, 138
- qVlmer, 120

r2norm, 121
RandomFromHessianOrMCMC, 122
rcutter, 29, 37, 38, 40, 78, 97, 114, 125
read_folder, 127
RectangleRegression, 128
rggamma (dggamma), 39
RM_add, 129, 131, 133–135
RM_delete, 130, 131, 133–135
RM_duplicate, 130, 131, 132, 134, 135
RM_get, 130, 131, 133, 133, 135
RM_list, 130, 131, 133, 134, 134
rnbinom_new, 136
rSnbinom, 43, 118, 120, 137

ScalePreviousPlot, 14, 110, 111, 138, 144
SEfromHessian, 140
series.compare, 18, 24, 141
show_name, 14, 110, 111, 139, 144
similar, 145
specify_decimal, 146
summary.mcmcComposite, 8, 10, 12, 81, 83,
105, 147
sun.info, 63, 87, 90, 148, 154
symbol.Female, 150, 151
symbol.Male, 150, 151
symmetricize, 152

tide.info, 63, 87, 90, 149, 153
tnirp, 13, 18, 35, 155

universalmclapply, 157

wget, 159