

# Package ‘RMAWGEN’

December 12, 2019

**Maintainer** Emanuele Cordano <emanuele.cordano@gmail.com>

**License** GPL (>= 2)

**Title** Multi-Site Auto-Regressive Weather GENERator

**Type** Package

**Author** Emanuele Cordano, Emanuele Eccel

**Description** S3 and S4 functions are implemented for spatial multi-site stochastic generation of daily time series of temperature and precipitation. These tools make use of Vector AutoRegressive models (VARs). The weather generator model is then saved as an object and is calibrated by daily instrumental “Gaussianized” time series through the ‘vars’ package tools. Once obtained this model, it can be used for weather generations and be adapted to work with several climatic monthly time series.

**Version** 1.3.7

**Repository** CRAN

**Date/Publication** 2019-12-12 07:00:07 UTC

**Date** 2019-12-11

**Depends** R (>= 2.10),chron,date,vars,methods

**Suggests** RgoogleMaps,lubridate

**URL** <https://github.com/ecor/RMAWGEN>,  
<https://docs.google.com/file/d/0B66otCUk3Bv6V3RPbm1mUG4zVHc/edit>,  
[http://presentations.copernicus.org/EGU2012-14026\\_presentation.pdf](http://presentations.copernicus.org/EGU2012-14026_presentation.pdf),  
[http://presentations.copernicus.org/EGU2012-5404\\_presentation.pdf](http://presentations.copernicus.org/EGU2012-5404_presentation.pdf)

**RoxygenNote** 7.0.2

**NeedsCompilation** no

## R topics documented:

RMAWGEN-package . . . . .	3
acvWGEN . . . . .	5

adddate	6
addsuffices	7
arch_test	7
ComprehensivePrecipitationGenerator	8
ComprehensiveTemperatureGenerator	11
continuity_ratio	14
countNAs	16
covariance	16
ElevationOf	18
extractdays	18
extractmonths	19
extractTnFromAnomalies	20
extractTxFromAnomalies	21
extractyears	22
findDate	22
forecastEV	23
forecastResidual	24
generateTemperatureTimeseries	25
getDailyMean	26
getMonthlyMean	27
getVARmodel	28
GPCA	30
GPCA-class	31
GPCAiteration-class	32
GPCAvarest2-class	32
GPCA_iteration	33
inv_GPCA	34
inv_GPCA_iteration	35
is.monthly.climate	36
months_f	37
NewVAREventRealization	38
newVARmultieventRealization	39
normality_test	40
normalizeGaussian	40
normalizeGaussian_prec	41
normalizeGaussian_severalstations	43
normalizeGaussian_severalstations_prec	45
plotDailyClimate	47
plot_sample	48
PrecipitationEndDay	50
PrecipitationStartDay	51
print.GPCA	51
qqplot.lagged	52
qqplotprecWGEN	53
qqplotprecWGEN_seasonal	54
qqplotTnTxWGEN	55
qqplotTnTxWGEN_seasonal	56
qqplotWGEN	57

qqplot_RMAWGEN_Tx . . . . .	57
removeNAs . . . . .	59
rescaling_monthly . . . . .	60
residuals.varest2 . . . . .	61
serial_test . . . . .	61
setComprehensiveTemperatureGeneratorParameters . . . . .	62
splineInterpolateMonthlytoDaily . . . . .	64
splineInterpolateMonthlytoDailyforSeveralYears . . . . .	65
TemperatureEndDay . . . . .	66
TemperatureStartDay . . . . .	67
trentino . . . . .	68
varest-class . . . . .	69
varest2-class . . . . .	70
VAR_mod . . . . .	70
WhereIs . . . . .	71
<b>Index</b>	<b>72</b>

---

RMAWGEN-package

*R - Multi-site Autoregressive WEather Generator*


---

## Description

Multi-site autoregressive Models for Daily Weather Generation. The modeling in climate change applications for agricultural or hydrological purposes often requires daily time-series of precipitation and temperature. This is the case of downscaled series from monthly or seasonal predictions of Global Climate Models (GCMs). The R package RMAWGEN (R Multi-Sites Autoregressive Weather GENERator) is built to generate daily temperature and precipitation time series in several sites by using the theory of vectorial autoregressive models (VAR). The VAR model is used because it is able to maintain the temporal and spatial correlations among the several series. In particular, observed time series of daily maximum and minimum temperature and precipitation are used to calibrate the parameters of a VAR model (saved as "GPCAvarest2" or "varest2" classes, which inherit the "varest" S3 class defined in the package vars [Pfaff, 2008]). Therefore the VAR model, coupled with monthly mean weather variables downscaled by GCM predictions, allows to generate several stochastic daily scenarios. The structure of the package consists in functions that transform precipitation and temperature time series into Gaussian-distributed random variables through deseasonalization and Principal Component Analysis. Then a VAR model is calibrated on transformed time series. The time series generated by VAR are then inversely re transformed into precipitation and/or temperature series. An application dataset is included in the RMAWGEN package as an example; it is presented by using a dataset with daily weather time series recorded in 59 different sites of Trentino (Italy) and its neighborhoods for the period 1958-2007. The software is distributed as a Free Software with General Public License (GPL) and is available on CRAN website. (<https://cran.r-project.org/package=RMAWGEN>) . A presentation of the package is available on <https://docs.google.com/file/d/0B8xDtMCnW3dJU2JIemVqMnpKTHc/edit>. Example script files about package usage are available on <https://github.com/ecor/RMAWGENCodeCorner>.

**Details**

```

Package:           RMAWGEN
Type:              Package
Version:           1.3.6
Date:              2019-11-13
License:           GPL (>= 2)
LazyLoad:         yes
Depends: R(>=2.12),time,chron,vars

```

### Note

First release of RMAWGEN was created in the frame of ACE-SAP (<http://www.ace-sap.it/>) and ENVIROCHANGE (<http://www.envirochange.cc/>) projects funded by Provincia Autonoma di Trento (<http://www.provincia.tn.it/>).

RMAWGEN is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

RMAWGEN is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

### Author(s)

Emanuele Cordano <[emanuele.cordano@gmail.org](mailto:emanuele.cordano@gmail.org)>, Emanuele Eccel <[emanuele.eccel@fmach.it](mailto:emanuele.eccel@fmach.it)>

### References

Cordano E. and Eccel E. (2016), Tools for stochastic weather series generation in R environment, Italian Journal of Agrometeorology doi:10.19199/2016.3.2038-5625.031 <https://doi.org/10.19199/2016.3.2038-5625.031> Pfaff B. (2008). VAR, SVAR and SVEC Models: Implementation Within R Package vars. Journal of Statistical Software 27(4). <https://www.jstatsoft.org/v27/i04/>(doi:10.18637/jss.v027.i04)

---

acvWGEN

*Plots the auto- and cross- covariance functions between measured and simulated data for several stations*

---

### Description

Plots the auto- and cross- covariance functions between measured and simulated data for several stations

**Usage**

```
acvWGEN(measured, simulated, titles = c("Sim.", "Mes."),
        station = NULL)
```

**Arguments**

measured	matrix containing measured time series
simulated	matrix containing simulated time series
titles	title suffixes for the simulated and measured data respectively c("Sim.,"Mes.")
station	string vector containing the IDs of the meteorological stations where the auto-covariance is calculated. If it is NULL (default) all stations (corresponding to the columns of "simulated" and "measured") are applied

**Value**

0 in case of success

**Note**

It uses [acf](#) function

---

adddate	<i>Inserts three columns (year,month,day) passing dates to a matrix or to a dataframe</i>
---------	---

---

**Description**

Inserts three columns (year,month,day) passing dates to a matrix or to a dataframe

**Usage**

```
adddate(data, origin = "1961-1-1")
```

**Arguments**

data	matrix of daily data
origin	character string containing the date of the first row of data as YYYY-MM-DD

**Value**

a data frame with dates and data values

**See Also**

[findDate](#)

---

addsuffixes	<i>Adds suffixes for daily maximum and minimum temperature to the names of a column data frame</i>
-------------	--

---

**Description**

Adds suffixes for daily maximum and minimum temperature to the names of a column data frame

**Usage**

```
addsuffixes(names = c("T0001", "T0099", "T0001", "T0099"),
            suffix = c("_Tx", "_Tn"), sep = "")
```

**Arguments**

names	a character string vector with column names
suffix	suffixes to add to the first and second groups of column names respectively
sep	separation element

**Details**

This function is used for data frames with duplicated field names

**Value**

the vector of names with suffixes added

**See Also**

[getVARmodel](#)

**Examples**

```
names <- addsuffixes()
```

---

arch_test	<i>arch.test function for varest2 object</i>
-----------	--

---

**Description**

arch.test function for varest2 object

**Usage**

```
arch_test(object, interval = NULL, overlap = 20, list.output = FALSE,
          ...)
```

**Arguments**

<code>object</code>	a <code>varest2</code> object
<code>interval</code>	string or subset interval of time (e.g. days) or length of this subset interval to which the ARCH test is applied (see Note). Default is NULL.
<code>overlap</code>	number of time instants (e.g. days) which are overlapped on two different subsequent intervals. Default is 20. It is used only if <code>interval</code> has length 1.
<code>list.output</code>	logical value. If TRUE the function returns a list of the test results of each interval. It is used if <code>interval</code> is not NULL. Default is FALSE.
<code>...</code>	further arguments for <code>arch.test</code>

**Details**

This function is a wrapper of `arch.test`. It can compute the test also for some subsets (intervals) of the time-series or for all the time-series divided in overlapping intervals. The intervals considered for the ARCH test are defined with the argument `interval`. If `interval` is an integer number instead of a vector, it indicates the length of the intervals in which the time-series is split. If `interval` is set to NULL, the test is done on the comprehensive residual time-series without splitting.

**Value**

One object or a list of objects with class attribute `varcheck` as reported in `arch.test`

**See Also**

[arch.test](#)

---

ComprehensivePrecipitationGenerator

*The comprehensive Precipitation Generator*

---

**Description**

The comprehensive Precipitation Generator

**Usage**

```
ComprehensivePrecipitationGenerator(station = c("T0001", "T0010",
  "T0099"), prec_all, mean_climate_prec = NULL, year_max = 1990,
  year_min = 1961, leap = TRUE, nmonth = 12, cpf = NULL,
  verbose = TRUE, p = 1, type = "none", lag.max = NULL,
  ic = "AIC", activateVARselect = FALSE, exogen = NULL,
  exogen_sim = NULL, is_exogen_gaussian = FALSE,
  year_max_sim = year_max, year_min_sim = year_min,
  mean_climate_prec_sim = NULL, onlygeneration = FALSE,
  varmodel = NULL, type_quantile = 3, qnull = NULL, valmin = 0.5,
  step = 0, n_GPCA_iteration = 0,
```



```
n_GPCA_iteration_residuals = n_GPCA_iteration, sample = NULL,
extremes = TRUE, exogen_all = NULL, exogen_all_col = station,
no_spline = FALSE, nscenario = 1, seed = NULL, noise = NULL)
```

### Arguments

station	character vector of the IDs of the considered meteorological stations
prec_all	data frame containing daily precipitation of all meteorological stations. See <a href="#">PRECIPITATION</a> defined in the <a href="#">trentino</a> dataset for formatting.
mean_climate_prec	a matrix containing monthly mean daily precipitation for the considered station. If it is NULL, it is calculated. See input of <a href="#">is.monthly.climate</a>
year_max	start year of the recorded (calibration) period
year_min	end year of the recorded (calibration) period
leap	logical variables. If it is TRUE (default)(recommended), leap years are considered, otherwise all years have 365 days
nmonth	number of months in one year (default is 12)
cpf	see <a href="#">normalizeGaussian_severalstations</a>
verbose	logical variable
p, type, lag.max, ic, activateVARselect	see respective input parameter on <a href="#">getVARmodel</a>
exogen	data frame or matrix containing the (normalized or not) exogenous variables (predictors) for the recorded (calibration) period.
exogen_sim	data frame or matrix containing the (normalized or not) exogenous variables (predictors) for the simulation period. Default is NULL. If it is NULL, it is replaced with exogen within the function.
is_exogen_gaussian	logical value. If TRUE, exogen_sim and exogen are given as already normalized variables, otherwise they are not normalized. Default is FALSE
year_max_sim	last year of the simulation period. Default is equal to year_max
year_min_sim	first year of the simulation period. Default is equal to year_min
mean_climate_prec_sim	a matrix containing monthly mean daily precipitation for the simulation period. If is NULL (Default), it is set equal to mean_climate_prec.
onlygeneration	logical value. If TRUE the VAR model varmodel is given as input and only random generation is done, otherwise (default) is calculated from measured data
varmodel	the comprehensive VAR model as a <a href="#">varest2</a> S4 object or a NULL object. If NULL (default), the comprehensive VAR is estimated from measured data within the function, otherwise it is given as input and only random generation is done.
type_quantile	see type on <a href="#">quantile</a>
step	see <a href="#">normalizeGaussian_severalstations</a> . Default is 0.
n_GPCA_iteration	number of iterations of Gaussianization process for data. Default is 0 (no Gaussianization)

n_GPCA_iteration_residuals	number of iterations of Gaussianization process for VAR residuals. Default is 0 (no Gaussianization)
sample, extremes, qnull, valmin	see <a href="#">normalizeGaussian_severalstations</a>
exogen_all	data frame containing exogenous variable formatted like prec_all. Default is NULL. It is alternative to exogen and if it not NULL, is_exogen_gaussian is automatically set FALSE
exogen_all_col	vector of considered columns of exogen_all. Default is station.
no_spline	logical value. See <a href="#">splineInterpolateMonthlytoDailyforSeveralYears</a> . Default is TRUE.
nscenario	number of generated scenarios for daily maximum and minimum temperature
seed	seed for stochastic random generation see <a href="#">set.seed</a> .
noise	stochastic noise to add for variable generation. Default is NULL. See <a href="#">newVARmultieventRealization</a> . Not used in case that nscenario>1.

**Value**

A list of the following variables:

prec\_mes matrix containing measured daily precipitation (the data is copied by the measured data given as input for the period and the station considered for varmodel estimation)

prec\_spline matrix containing climatic "spline-interpolated" daily precipitation from mean\_climate\_prec

data\_prec matrix containing normalized measured precipitation variable

prec\_gen matrix containing generated daily precipitation [mm]

prec\_spline\_sim matrix containing climatic "spline-interpolated" daily precipitation from mean\_climate\_prec\_sim

data\_prec\_gen matrix containing normalized generated precipitation variable

mean\_climate\_prec matrix containing monthly means of daily precipitation (historical scenario)

mean\_climate\_prec\_sim matrix containing monthly means of daily precipitation (predicted/simulated scenario)

var a varest object containing the used VAR model

**Note**

It pre-processes and generates a multi-site precipitation fields. It uses [getVARmodel](#). Detailed examples can be viewed of this function in [this presentation](#). Unfortunately, using this approach, the spatial correlations are underestimated. This is due to the persistence of zeros in the precipitation records. This problem is known in literature and can be solved in the future versions of RMAW-GEN. See the R code for further details

**Author(s)**

Emanuele Cordano, Emanuele Eccel

**See Also**

[splineInterpolateMonthlytoDailyforSeveralYears](#)

**Examples**

```
data(trentino)
set.seed(1222) # set the seed for random generations!
year_max <- 1990
year_min <- 1961
year_max_sim <- 1982
year_min_sim <- 1981

n_GPCA_iter <- 2
p <- 1
nscenario=1
station <- c("T0090","T0083")
## Not Run: the call to ComprehensivePrecipitationGenerator may elapse too
## long time (more than 5 eseconds) and is not executed by CRAN check.
## Please uncomment the following line to run the example on your own PC.
# generation00 <- ComprehensivePrecipitationGenerator(station=station,
# prec_all=PRECIPITATION,year_min=year_min,year_max=year_max,
# year_min_sim=year_min_sim,year_max_sim=year_max_sim,p=p,
# n_GPCA_iteration=n_GPCA_iter,n_GPCA_iteration_residuals=0,
# sample="monthly",nscenario=nscenario,no_spline=TRUE)

#
```

---

ComprehensiveTemperatureGenerator

*The Comprehensive Temperature Generator*

---

**Description**

The Comprehensive Temperature Generator

**Usage**

```
ComprehensiveTemperatureGenerator(station = c("T0001", "T0010", "T0099"),
  Tx_all, Tn_all, mean_climate_Tn = NULL, mean_climate_Tx = NULL,
  Tx_spline = NULL, Tn_spline = NULL, year_max = 1990,
  year_min = 1961, leap = TRUE, nmonth = 12, verbose = TRUE,
  p = 1, type = "none", lag.max = NULL, ic = "AIC",
  activateVARselect = FALSE, year_max_sim = year_max,
  year_min_sim = year_min, mean_climate_Tn_sim = NULL,
  mean_climate_Tx_sim = NULL, Tn_spline_sim = NULL,
  Tx_spline_sim = NULL, onlygeneration = FALSE, varmodel = NULL,
```

```

normalize = TRUE, type_quantile = 3, sample = NULL,
extremes = TRUE, option = 2, yearly = FALSE, yearly_sim = yearly,
n_GPCA_iteration = 0, n_GPCA_iteration_residuals = n_GPCA_iteration,
exogen = NULL, exogen_sim = exogen, is_exogen_gaussian = FALSE,
exogen_all = NULL, exogen_all_col = station, nscenario = 1,
seed = NULL, noise = NULL)

```

## Arguments

**station** see respective input parameter on [setComprehensiveTemperatureGeneratorParameters](#)  
**Tx\_all, Tn\_all, mean\_climate\_Tn, mean\_climate\_Tx, Tx\_spline, Tn\_spline**  
 see respective input parameter on [setComprehensiveTemperatureGeneratorParameters](#)  
**year\_max, year\_min, leap, nmonth, verbose**  
 see respective input parameter on [setComprehensiveTemperatureGeneratorParameters](#)  
**p, type, lag.max, ic, activateVARselect**  
 see respective input parameter on [getVARmodel](#)  
**year\_max\_sim** last year of the simulation period. Default is equal to year\_max  
**year\_min\_sim** first year of the simulation period. Default is equal to year\_min  
**mean\_climate\_Tn\_sim**  
 monthly averaged daily minimum temperatures for the simulated scenario and  
 used by the random generator . Default is mean\_climate\_Tn  
**mean\_climate\_Tx\_sim**  
 monthly averaged daily maximum temperatures for the simulated scenario and  
 used by the random generator . Default is mean\_climate\_Tx  
**Tn\_spline\_sim** daily timeseries (from the first day of year\_min\_sim to the last day of year\_max\_sim)  
 of averaged minimum temperature which can be obtained by a spline interpola-  
 tion of monthly mean values (for the generation period). Default is Tn\_spline.  
 See for spline interpolation utilized [splineInterpolateMonthlytoDailyforSeveralYears](#).  
**Tx\_spline\_sim** daily timeseries (from the first day of year\_min\_sim to the last day of year\_max\_sim)  
 of averaged maximum temperature which can be obtained by a spline interpola-  
 tion of monthly mean values (for the generation period). Default is Tx\_spline.  
 See for spline interpolation utilized [splineInterpolateMonthlytoDailyforSeveralYears](#).  
**onlygeneration** logical variable. If TRUE the VAR model varmodel is given as input and only  
 random generation is done, otherwise (default) is calculated from measured data  
**varmodel** the comprehensive VAR model as a [varest2](#) or [GPCAvarest2](#) S4 object or a  
 NULL object. If NULL (default), the comprehensive VAR is estimated from mea-  
 sured data within the function, otherwise it is given as input and only random  
 generation is done.  
**normalize, sample, extremes**  
 see [normalizeGaussian\\_severalstations](#) or [setComprehensiveTemperatureGeneratorParameters](#)  
**type\_quantile** see type on [quantile](#)  
**option** integer value. If 1, the generator works with minimum and maximum tempera-  
 ture, if 2 (default) it works with the average value between maximum and mini-  
 mum temperature and the respective daily thermal range.

yearly	logical value. If TRUE the monthly mean values are calculated for each year from year_min to year_max separately. Default is FALSE.
yearly_sim	logical value. If TRUE the monthly mean values are calculated for each year from year_min_sim to year_max_sim separately. Default is yearly.
n_GPCA_iteration	number of iterations of Gaussianization process for data. Default is 0 (no Gaussianization)
n_GPCA_iteration_residuals	number of iterations of Gaussianization process for VAR residuals. Default is 0 (no Gaussianization)
exogen	data frame or matrix containing the (normalized or not) exogenous variables (predictors) for the recorded (calibration) period. Default is NULL.
exogen_sim	data frame or matrix containing the (normalized or not) exogenous variables (predictors) for the simulation period. Default is NULL. If it is NULL, exogen_sim is set equal to exogen within the function.
is_exogen_gaussian	logical value, If TRUE, exogen_sim and exogen are given as already normalized variables, otherwise they are not normalized. Default is FALSE
exogen_all	data frame containing exogenous variable formatted like Tx_all and Tn_all. Default is NULL. It is alternative to exogen and if it not NULL, is_exogen_gaussian is automatically set to FALSE
exogen_all_col	vector of considered columns of exogen_all. Default is station.
nscenario	number of generated scenarios for daily maximum and minimum temperature
seed	seed for stochastic random generation see <a href="#">set.seed</a>
noise	stochastic noise to add for variable generation. Default is NULL. See <a href="#">newVARmultieventRealization</a> . Not used in case that nscenario>1.

**Value**

A list of the following variables:

- input list of variables returned by [setComprehensiveTemperatureGeneratorParameters](#)
- var varest object containing the used VAR model (if useVAR is true), NULL (otherwise)
- output list variables returned by [generateTemperatureTimeseries](#) (i.e. generated timeseries)

**Note**

It pre-processes series and generates multi-site temperature fields by using [setComprehensiveTemperatureGeneratorParameters](#) and [generateTemperatureTimeseries](#). Detailed examples can be viewed of this function in [this presentation](#).

**Author(s)**

Emanuele Cordano, Emanuele Eccel

**See Also**

[setComprehensiveTemperatureGeneratorParameters](#), [generateTemperatureTimeseries](#), [generateTemperatureTime](#)

**Examples**

```
data(trentino)

set.seed(1222) # set the seed for random generations!
year_min <- 1961
year_max <- 1990

year_min_sim <- 1982
year_max_sim <- 1983

n_GPCA_iter <- 5
n_GPCA_iteration_residuals <- 5
p <- 1
vstation <- c("B2440", "B6130", "B8570", "B9100", "LAVIO", "POLSA", "SMICH", "T0001",
  "T0010", "T0014", "T0018", "T0032", "T0064", "T0083", "T0090", "T0092",
  "T0094", "T0099", "T0102", "T0110", "T0129", "T0139", "T0147", "T0149",
  "T0152", "T0157", "T0168", "T0179", "T0189", "T0193", "T0204", "T0210",
  "T0211", "T0327", "T0367", "T0373")
## Not Run: the call to ComprehensiveTemperatureGenerator may elapse
## too long time (more than 5 esconds) and is not executed by CRAN check.
## Please uncomment the following line to run the example on your own PC.
# generation00 <- ComprehensiveTemperatureGenerator(station=vstation[16],
# Tx_all=TEMPERATURE_MAX, Tn_all=TEMPERATURE_MIN, year_min=year_min, year_max=year_max,
# p=p, n_GPCA_iteration=n_GPCA_iter, n_GPCA_iteration_residuals=n_GPCA_iteration_residuals,
# sample="monthly", year_min_sim=year_min_sim, year_max_sim=year_max_sim)
```

---

continuity_ratio	<i>Calculates the continuity ratio of a set of precipitation measured or generated data in several sites as defined by Wilks, 1998 (see reference link)</i>
------------------	---

---

**Description**

Calculates the continuity ratio of a set of precipitation measured or generated data in several sites as defined by Wilks, 1998 (see reference link)

**Usage**

```
continuity_ratio(data, lag = 0, valmin = 0.5)
```

**Arguments**

data	containing daily precipitation time series for several gauges (one gauge time series per column)
lag	numeric lag (expressed as number of days) used for computation for "cross" continuity ratio and joint probability of precipitation (no)occurrence.
valmin	threshold precipitation value [mm] for wet/dry day indicator. If precipitation is lower than valmin, day is considered dry. Default is 0.5 mm.

**Value**

A list containing the following matrices:

continuity\_ratio : lag-day lagged continuity ratio ,

occurrence : joint probability of lag-day lagged precipitation occurrence

nooccurrence : joint probability of lag-day lagged no precipitation occurrence.

nooccurrence\_occurrence : joint probability of lag-day lagged no precipitation and precipitation occurrence respectively.

occurrence\_nooccurrence : joint probability of lag-day lagged precipitation and no precipitation occurrence respectively.

probability\_continuity\_ratio: lag-day lagged ratio about precipitation probability conditioned to no precipitation/precipitation occurrence in the other site

**Note**

If lag==0 the function returns the continuity ratio and joint probability as described by Wilks, 1998. Otherwise the precipitation values for each couple of rain gauges are taken with lag-day lag.

**References**

see the following URL references: <http://onlinelibrary.wiley.com/doi/10.1002/joc.2305/abstract> and <http://www.sciencedirect.com/science/article/pii/S0022169498001863>

**Examples**

```
data(trentino)

year_min <- 1961
year_max <- 1990
origin <- paste(year_min,1,1,sep="-")

period <- PRECIPITATION$year>=year_min & PRECIPITATION$year<=year_max
station <- names(PRECIPITATION)[!(names(PRECIPITATION) %in% c("day","month","year"))]
prec_mes <- PRECIPITATION[period,station]

## removing nonworking stations (e.g. time series with NA)
accepted <- array(TRUE,length(names(prec_mes)))
names(accepted) <- names(prec_mes)
```

```

for (it in names(prec_mes)) {
  accepted[it] <- (length(which(!is.na(prec_mes[,it])))==length(prec_mes[,it]))
}

prec_mes <- prec_mes[,accepted]
## the dataset is reduced!!!
prec_mes <- prec_mes[,1:2]

continuity_ratio <-continuity_ratio(data=prec_mes,lag=0, valmin=0.5)
continuity_ratio1 <-continuity_ratio(data=prec_mes,lag=-1, valmin=0.5)

```

---

countNAs	<i>counts NAs in each row of data</i>
----------	---------------------------------------

---

### Description

counts NAs in each row of data

### Usage

```
countNAs(data)
```

### Arguments

data	a data input matrix @export
------	--------------------------------

### Value

the vector with numbers of NA values for each data column

---

covariance	<i>Calculates the covariance matrix of the normally standardized variables obtained from the columns of x</i>
------------	---

---

### Description

Calculates the covariance matrix of the normally standardized variables obtained from the columns of x

### Usage

```

covariance(x, data = x, cpf = NULL, mean = 0, sd = 1,
  step = NULL, prec = 10^-4, use = "pairwise.complete.obs",
  type = 3, extremes = TRUE, sample = NULL, origin_x = NULL,
  origin_data = origin_x)

```



**Arguments**

x	variable
data	a sample of data on which a non-parametric probability distribution is estimated
cpf	cumulative probability distribution. If NULL (default) is calculated as <code>ecdf(data)</code>
mean	mean (expected value) of the normalized random variable. Default is 0.
sd	standard deviation of the normalized random variable. Default is 1.
step	vector of values in which step discontinuities of the cumulative probability function occur. Default is NULL
prec	amplitude of the neighbourhood of the step discontinuities where cumulative probability function is treated as non continuous.
use	see <code>cov</code>
type	see <code>quantile</code>
extremes	logical variable. If TRUE (default) the probability or frequency is multiplied by

$$\frac{N}{N + 1}$$

where  $N$  is the length of data

sample	information about sample or probability distribution. Default is NULL
origin_x	date corresponding to the first row of x
origin_data	date corresponding to the first row of data

**Value**

a matrix with the normalized variable or its inverse

**Author(s)**

Emanuele Cordano, Emanuele Eccel

**See Also**

[normalizeGaussian\\_severalstations](#), [normalizeGaussian](#)

@note It applies [normalizeGaussian\\_severalstations](#) to x and data and then calculates the covariances among the column. See the R code for further details

---

ElevationOf	<i>Extracts the elevation of a meteorological station expressed in meters above a reference (sea level)</i>
-------------	---

---

**Description**

Extracts the elevation of a meteorological station expressed in meters above a reference (sea level)

**Usage**

```
ElevationOf(name, station_names, elevation)
```

**Arguments**

name	character ID of the station
station_names	vector of the IDs (characters) of the considered meteorological stations. An example is STATION_NAMES, which is defined in the <a href="#">trentino</a> dataset.
elevation	vector of the elevation of the considered meteorological stations. An example is ELEVATION, which is defined in the <a href="#">trentino</a> dataset.

**Value**

the elevation given the vectors of station IDs and the respective elevations

**Examples**

```
data(trentino)
ElevationOf("T0099",station_names=STATION_NAMES,elevation=ELEVATION)
```

---

extractdays	<i>Extracts the rows of a matrix corresponding to the requested days (expressed as dates YYYY-MM-DD) given the date (origin) of the first row</i>
-------------	---

---

**Description**

Extracts the rows of a matrix corresponding to the requested days (expressed as dates YYYY-MM-DD) given the date (origin) of the first row

**Usage**

```
extractdays(data = array(1:ndim_max, dim = c(ndim_max, 1)),
  ndim_max = 1e+05, when = "1990-1-1", origin = "1961-1-1",
  nday = 1)
```

**Arguments**

data	an input data matrix where each row corresponds to a daily record
ndim_max	maximum (integer) number of rows in data where to find when. Default is 100000 and works if data is missing.
when	desired dates for which the data are requested
origin	date corresponding to the first row of data
nday	(optional) number of days since when to extract the data

**Value**

a matrix containing the requested rows

**Note**

It uses [julian](#)

**Examples**

```
extractdays()
```

---

extractmonths	<i>Extracts the rows of a matrix corresponding to requested months of a year given the date (origin) of the first row</i>
---------------	---

---

**Description**

Extracts the rows of a matrix corresponding to requested months of a year given the date (origin) of the first row

**Usage**

```
extractmonths(data = array(1:ndim_max, dim = c(ndim_max, 1)),
  ndim_max = 1e+05, when = c("Dec", "Jan", "Feb"), year = NULL,
  origin = "1961-1-1")
```

**Arguments**

data	an input data matrix where each row corresponds to a daily record
ndim_max	maximum (integer) number of rows in data where to find when. Default is 100000 and works if data is missing.
when	character vector of months for which the data are required. It must be a subset of <code>c("Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec")</code>
year	year(s) when data must be extracted
origin	date corresponding to the first row of data

**Value**

a matrix containing the requested rows

**Note**

It uses [months](#) and [julian](#)

**Author(s)**

Emanuele Cordano, Emanuele Eccel

**See Also**

[extractdays](#)

**Examples**

```
extractmonths()

data(trentino)
dates <- sprintf("%02d-%02d-%02d", TEMPERATURE_MAX$year, TEMPERATURE_MAX$month, TEMPERATURE_MAX$day)
origin <- dates[1]
out <- extractmonths(data=TEMPERATURE_MAX,origin=origin)
```

---

extractTnFromAnomalies

*Extracts generated time series of Daily Minimum Temperature from a random multi-realization obtained by [generateTemperatureTimeseries](#) function*

---

**Description**

Extracts generated time series of Daily Minimum Temperature from a random multi-realization obtained by [generateTemperatureTimeseries](#) function

**Usage**

```
extractTnFromAnomalies(res_multigen, std, SplineAdv)
```

**Arguments**

res_multigen	matrix containing standardized values of daily temperature as returned by <a href="#">generateTemperatureTimeseries</a> (first item)
std	vector containing standard deviation for each minimum temperature anomalies
SplineAdv	matrix containing the averaged daily values of minimum temperature obtained by a spline interpolation of the monthly climate

**Value**

a matrix with generated minimum temperature

**Author(s)**

Emanuele Cordano, Emanuele Eccel

---

extractTxFromAnomalies

*Extracts generated time series of Daily Maximum Temperature from a random multi-realization obtained by [generateTemperatureTimeseries](#) function*

---

**Description**

Extracts generated time series of Daily Maximum Temperature from a random multi-realization obtained by [generateTemperatureTimeseries](#) function

**Usage**

```
extractTxFromAnomalies(res_multigen, std, SplineAdv)
```

**Arguments**

res_multigen	matrix containing standardized values of daily temperature as returned by <a href="#">generateTemperatureTimeseries</a> (first item)
std	vector containing standard deviation for each maximum temperature anomalies
SplineAdv	matrix containing the averaged values of maximum temperature obtained by a spline interpolation of monthly climate

**Value**

a matrix with generated maximum temperature

**Author(s)**

Emanuele Cordano, Emanuele Eccel

---

extractyears	<i>Extracts the elements of a data frame corresponding to a period between year_min and year_max for the stations listed in station</i>
--------------	---

---

### Description

Extracts the elements of a data frame corresponding to a period between year\_min and year\_max for the stations listed in station

### Usage

```
extractyears(data, year_min = 1961, year_max = 1990,
             station = c("T0001", "T0014", "T0129"))
```

### Arguments

data	a dataframe containing daily data.
year_min	start year
year_max	end year
station	character vector of the IDs of the station where the data are required

### Value

a matrix containing the requested daily data where each day corresponds to a row and each station corresponds to a column

### Note

The input data frame data must have the following fields: year, month, day, variables\_ID1, variables\_ID2, ... where the fields ,variables\_ID1, variables\_ID2, ... contain the daily variables referred to the respective stations and the field names are replaced with the respective station ID.

---

findDate	<i>Finds the date corresponding a row index of a matrix given the date (origin) of the first row</i>
----------	--

---

### Description

Finds the date corresponding a row index of a matrix given the date (origin) of the first row

### Usage

```
findDate(k, origin = "1961-1-1", data.frame = TRUE, decimal = FALSE,
         character = FALSE)
```

**Arguments**

k	integer or decimal value corresponding to number of days since origin
origin	origin date. See also <a href="#">extractdays</a>
data.frame	logical variable. If TRUE (default) the date is returned as data frame (like data in <a href="#">extractyears</a> ), otherwise it is returned as character or POSIXct.
decimal	logical variable. If FALSE (default) k is integer and starts from 1, otherwise is consider as the decimal julian day since origin (deprecated)
character	logical variable. It is used if data.frame is FALSE, if it is FALSE, the date is returned as POSIXct, otherwise it is a character in the following form: YYYY-MM-DD

**Value**

the date(s) corresponding to k under different formats

**Note**

It uses functions of [time](#) package. It works like an inverse functions of [extractdays](#). If k is a vector, the function returns several dates for each element of k

**See Also**

[date.mdy](#), [extractdays](#)

**Examples**

```
findDate <- findDate(100,origin="1961-1-1",data.frame=FALSE,character=TRUE)
```

---

forecastEV	<i>Forecasts the expected value of a VAR realization given the previous one</i>
------------	---

---

**Description**

Forecasts the expected value of a VAR realization given the previous one

**Usage**

```
forecastEV(var, xprev = NULL, exogen = NULL)
```

**Arguments**

var	A VAR model represented by a varest object as returned by <a href="#">getVARmodel</a> or <a href="#">VAR</a>
xprev	previous status of the random variable
exogen	vector containing the values of the "exogen" variables (predictor) for the generation

**Value**

a vector of values

**See Also**

[forecastResidual](#)

@export

---

forecastResidual	<i>Forecasts the residual value of a VAR realization given the white noise covariance matrix</i>
------------------	--

---

**Description**

Forecasts the residual value of a VAR realization given the white noise covariance matrix

**Usage**

```
forecastResidual(var, xprev = NULL, B = NULL)
```

**Arguments**

var	A VAR model represented by a varest object as returned by <a href="#">getVARmodel</a> or <a href="#">VAR</a>
xprev	previous status of the random variable, in this case the "current instant" white-noise". Default is NULL and then randomly generated.
B	matrix of coefficients for the vectorial white-noise component

**Value**

a vector of values

**Author(s)**

Emanuele Cordano, Emanuele Eccel

**See Also**

[forecastEV,NewVAReventRealization](#)



---

```
generateTemperatureTimeseries
```

*Returns time series of Daily Maximum and Minimum with a random multi-realization obtained by using [newVARmultieventRealization](#). This function is called by [ComprehensiveTemperatureGenerator](#).*

---

## Description

Returns time series of Daily Maximum and Minimum with a random multi-realization obtained by using [newVARmultieventRealization](#). This function is called by [ComprehensiveTemperatureGenerator](#).

## Usage

```
generateTemperatureTimeseries(std_tn, std_tx, SplineTx, SplineTn, SplineTm,
    SplineDeltaT, std_tm, var = NULL, exogen = NULL, normalize = TRUE,
    type = 3, extremes = TRUE, sample = NULL, option = 1,
    original_data, origin_x = NULL, origin_data = NULL, noise = NULL)
```

## Arguments

std_tn	vector containing standard deviation of daily minimum temperature anomalies. stdTn is default, see <a href="#">setComprehensiveTemperatureGeneratorParameters</a> .
std_tx	vector containing standard deviation of daily maximum temperature anomalies. stdTx is default, see <a href="#">setComprehensiveTemperatureGeneratorParameters</a> .
SplineTx	matrix containing the averaged daily maximum temperature obtained by a spline interpolation of monthly means . SplineAdvTx is default, see <a href="#">setComprehensiveTemperatureGeneratorParameters</a> .
SplineTn	matrix containing the averaged daily minimum temperature obtained by a spline interpolation of monthly means . SplineAdvTn is default, see <a href="#">setComprehensiveTemperatureGeneratorParameters</a> .
SplineTm	matrix containing the averaged daily "mean" temperature obtained by a spline interpolation of monthly means . SplineAdvTm is default, see <a href="#">setComprehensiveTemperatureGeneratorParameters</a> .
SplineDeltaT	matrix containing the rescaled averaged daily temperature range obtained by a spline interpolation of monthly means. SplineAdvDelta_T_sim/SplineAdvDelta_T is default, see <a href="#">setComprehensiveTemperatureGeneratorParameters</a> .
std_tm	vector containing standard deviation of daily "mean" temperature anomalies. stdTn is default, see <a href="#">setComprehensiveTemperatureGeneratorParameters</a> .
var	A VAR model represented by a varest object as returned by <a href="#">getVARmodel</a> or <a href="#">VAR</a>
exogen	see <a href="#">VAR</a>
normalize	logical variable If TRUE <a href="#">normalizeGaussian_severalstations</a> is used, otherwise not. If option is 2, it is always TRUE.
type	see <a href="#">quantile</a>
sample, origin_x, origin_data, extremes	see <a href="#">normalizeGaussian_severalstations</a>

option	integer value. If 1, the generator works with minimum and maximum temperature, if 2 (Default) it works with the average value between maximum and minimum temperature and the respective daily Thermal Range.
original_data	matrix containing the measured standardized temperature anomalies
noise	stochastic noise to add for variable generation. Default is NULL. See <a href="#">newVARmultieventRealization</a> .

### Value

This function returns a list of the following variables:

res\_multigen matrix containing standardized values of daily maximum and minimum temperature anomalies

Tx\_spline matrix containing climatic "spline-interpolated" daily maximum temperature

Tn\_spline matrix containing climatic "spline-interpolated" daily minimum temperature

Tx\_gen matrix containing generated daily maximum daily temperature ( $Tx_{gen}$ )

Tn\_gen matrix containing generated daily minimum daily temperature ( $Tn_{gen}$ )

Tm\_gen matrix containing generated "mean" daily temperature defined as  $\frac{Tx_{gen} + Tn_{gen}}{2}$

DeltaT\_gen matrix containing generated daily thermal range defined as  $Tx_{gen} - Tn_{gen}$

See the R code for further details

### Author(s)

Emanuele Cordano, Emanuele Eccel

### See Also

[newVARmultieventRealization](#), [normalizeGaussian\\_severalstations](#)

---

getDailyMean	<i>Calculates the daily means of a range of days around each date of a data frame corresponding to a period between year_min and year_max for stations listed in station</i>
--------------	--

---

### Description

Calculates the daily means of a range of days around each date of a data frame corresponding to a period between year\_min and year\_max for stations listed in station

### Usage

```
getDailyMean(data, year_min = 1961, year_max = 1990,
              station = c("T0001", "T0010"), origin = "1961-1-1", lag = 5)
```

**Arguments**

data	a data frame containing daily data.
year_min	start year
year_max	end year
station	character vector of the IDs of the station where the data are requested
origin	origin date of time-series
lag	lag (number of days) on which daily mean is calculated. The mean is calculated considering lag days before and after each day.

**Value**

a matrix containing the requested daily mean data where each day corresponds to a row and each station corresponds to a column

**Note**

The input data frame data must have the following fields: year, month, day, variables\_ID1, variables\_ID2, ... where the fields , variables\_ID1, variables\_ID2, ... contain the daily variables referred to the respective stations and the field names are replaced with the respective station ID.

**Author(s)**

Emanuele Cordano, Emanuele Eccel

**See Also**

[extractyears](#)

---

getMonthlyMean	<i>Calculates the monthly means of a data frame corresponding to a period between year_min and year_max for stations listed in station</i>
----------------	--

---

**Description**

@author Emanuele Cordano, Emanuele Eccel

**Usage**

```
getMonthlyMean(data, year_min = 1961, year_max = 1990,  
  station = names(data), no_date = FALSE, origin = "1961-1-1",  
  yearly = FALSE)
```

**Arguments**

data	a dataframe containing daily data.
year_min	start year
year_max	end year
station	character vector of the IDs of the station where the data are requested
no_date	logical value if TRUE the function extractmonths is used. Default is FALSE. It is recommended if data does not contain columns for the dates.
origin	date corresponding to the first row
yearly	logical value. If TRUE the monthly mean values are calculated for each year from year_min to year_max separately. Default is FALSE.

**Value**

a matrix containing the requested monthly means where each month corresponds to a row and each station corresponds to a column or a list of such matrices in case the monthly mean values are calculated separately for each year (if yearly is TRUE)

**Note**

The input data frame data must have the following fields: year, month, day, variables\_ID1, variables\_ID2, ... where the fields , variables\_ID1, variables\_ID2, ... contain the daily variables referred to the respective stations and the field names are replaced with the respective station ID. In case yearly is TRUE the returned output is a list of matrices whose names are the corresponding year.

**See Also**

[extractyears](#)

---

getVARmodel	<i>Either creates a VAR model or chooses a VAR model by using VAR or VARselect commands of vars package</i>
-------------	---

---

**Description**

Either creates a VAR model or chooses a VAR model by using VAR or VARselect commands of vars package

**Usage**

```
getVARmodel(data, suffix = c("_Tx", "_Tn"), sep = "", p = 1,
  type = "none", season = NULL, exogen = NULL, lag.max = NULL,
  ic = "AIC", activateVARselect = FALSE, na.rm = TRUE,
  n_GPCA_iteration = 0, n_GPCA_iteration_residuals = n_GPCA_iteration,
  extremes = TRUE)
```

**Arguments**

data	see <a href="#">VAR</a> and <a href="#">addsuffixes</a>
suffix	see <a href="#">addsuffixes</a>
sep	separator element. See <a href="#">addsuffixes</a> ).
p	lag considered for the auto-regression see <a href="#">VAR</a>
type	see <a href="#">VAR</a>
season	see <a href="#">VAR</a>
exogen	see <a href="#">VAR</a>
lag.max	see <a href="#">VARselect</a>
ic	see <a href="#">VAR</a>
activateVARselect	logical variables. If TRUE, the function <a href="#">VARselect</a> is run. Default and recommended use is FALSE.
na.rm	logical variables. If TRUE (default), it takes into account NA values
n_GPCA_iteration	number of iterations of Gaussianization process for data. Default is 0 (no Gaussianization)
n_GPCA_iteration_residuals	number of iterations of Gaussianization process for data. Default is 0 (no Gaussianization)
extremes	see <a href="#">normalizeGaussian_severalstations</a> and <a href="#">GPCA</a>

**Value**

a varest2 or GPCAvarest2 object representing a VAR model or a GPCA-varest object which also contains the GPCA transformation parameters

**Note**

It inherits input parameters of [VAR](#), [VARselect](#) and [addsuffixes](#). The variable data contains the measured data on which the vector auto-regressive models is estimated. It is a matrix where each row is a realization of the vector random variable. In some application of this package, the random variables may be the daily maximum and minimum temperature anomalies for different stations. Often the the columns of data are called with the IDs of the stations whithout specifying the type of variable (e.g. minimun or maximum temperature anomalies). This means that two or more columns may have the same name. Therefore the function [addsuffixes](#), which is called from this function, adds suitable suffixes to the column names.

**Author(s)**

Emanuele Cordano, Emanuele Eccel

---

GPCA                      *This function makes a Gaussianization procedure based on PCA iteration ( see [GPCA\\_iteration](#))*

---

### Description

This function makes a Gaussianization procedure based on PCA iteration ( see [GPCA\\_iteration](#))

### Usage

```
GPCA(x_prev, n = 30, extremes = TRUE)
```

### Arguments

x_prev	previous set of the random variable x. If it is a varest object, the residuals are taken into account.
n	number of reiterations
extremes	see <a href="#">normalizeGaussian_severalstations</a>

### Value

A [GPCA-class](#) S3 object returned by [GPCA\\_iteration](#) at each iteration and the final results of the G-PCA procedure (matrix final\_results)

### Note

This function re-iterates the equation (1) of "PCA Gaussianization for One-Class Remote Sensing Image" by V. Laparra et al., [http://www.uv.es/lapeva/papers/SPIE09\\_one\\_class.pdf](http://www.uv.es/lapeva/papers/SPIE09_one_class.pdf), [http://www.uv.es/vista/vistavalencia/papers/SPIE\\_09\\_Gaussianization\\_presentation.pdf](http://www.uv.es/vista/vistavalencia/papers/SPIE_09_Gaussianization_presentation.pdf)

### Author(s)

Emanuele Cordano

### See Also

[GPCA](#), [GPCA\\_iteration](#), [inv\\_GPCA\\_iteration](#), [inv\\_GPCA](#), [GPCA-class](#) for 'GPCA' S3 class

### Examples

```
library(RMAWGEN)
set.seed(1222)
nIterations <- 30
N <- 20
x <- rexp(N)
y <- x+rnorm(N)
df <- data.frame(x=x,y=y)
```

```
GPCA <- GPCA(df,n=nIterations,extremes=TRUE)

x <- rnorm(N)
y <- x+rnorm(N)
dfn <- data.frame(x=x,y=y)

GPCAn <- GPCA(dfn,n=nIterations,extremes=TRUE)
```

---

GPCA-class

*GPCA-class*

---

## Description

GPCA S3 class returned by [GPCA](#)

## Details

**list of** GPCA\_iteration subsequent GPCA iterations

**final\_results** data.frame or matrix of the "gaussianized" data

## Note

Formal definition with [setOldClass](#) for the S3 class GPCA

## Author(s)

Emanuele Cordano

## Examples

```
showClass("GPCA")
```

---

GPCAiteration-class    *GPCAiteration-class*

---

### Description

GPCAiteration S3 class returned by [GPCA\\_iteration](#)

### Details

`x_prev` Previous set of random variable, `x_prev` input variable of [GPCA\\_iteration](#)

`x_gauss_prev` Marginal Gaussianization of `x_prev` obtained through [normalizeGaussian\\_severalstations](#)

`B_prev` rotation matrix (i. e. eigenvector matrix of the covariance matrix of `x_gauss_prev`)

`x_next` results obtained by multiplying `B_prev` by `x_gauss_prev` (see equation 1 of the reference in [GPCA\\_iteration](#))

### Note

Formal definition with [setOldClass](#) for the S3 class GPCAiteration

### Author(s)

Emanuele Cordano

### Examples

```
showClass("GPCAiteration")
```

---

GPCAvarest2-class    *GPCAvarest2-class*

---

### Description

This class inherits `varest2` and contains all information about GPCA ([GPCA](#) transformation).

### Details

`GPCA_data`: A "GPCA" S3 object containing the parameters of the Multi-variate Gaussianization of the time series, it is the result of [GPCA](#) function applied to the input data of [getVARmodel](#)

`GPCA_residuals`: A "GPCA" S3 object containing the parameters of the Multi-variate Gaussianization of the residuals of the VAR model contained in the VAR slot; it is NULL if no Gaussianization of residuals is applied. Object of class "list"

`VAR`: S3 Object of class "varest"

#' @note A GPCAvarest2 object can be created by `new("GPCAvarest2", ...)` or returned by the function [getVARmodel](#)



**Author(s)**

Emanuele Cordano

**Examples**

```
showClass("GPCAvarest2")
```

---

GPCA_iteration	<i>This function makes an iteration of PCA-Gaussianization process</i>
----------------	--

---

**Description**

This function makes an iteration of PCA-Gaussianization process

**Usage**

```
GPCA_iteration(x_prev, extremes = TRUE)
```

**Arguments**

x\_prev            previous set of random variable x  
extremes          see [normalizeGaussian\\_severalstations](#)

**Value**

A GPCA\_iteration S3 object which contains the following objects:

x\_prev Previous set of random variable, x\_prev input variable

x\_gauss\_prev Marginal Gaussianization of x\_prev obtained through [normalizeGaussian\\_severalstations](#)

B\_prev rotation matrix (i. e. eigenvector matrix of the covariance matrix of x\_gauss\_prev)

x\_next results obtained by multiplying B\_prev by x\_gauss\_prev (see equation 1 of the reference)

**Note**

This function is based on equation (1) of "PCA Gaussianization for One-Class Remote Sensing Image" by V. Laparra et al., [https://www.uv.es/lapeva/papers/SPIE09\\_one\\_class.pdf](https://www.uv.es/lapeva/papers/SPIE09_one_class.pdf) and <http://ieeexplore.ieee.org/document/5413808/>

**Author(s)**

Emanuele Cordano

**See Also**

[GPCA](#), [GPCA\\_iteration](#), [inv\\_GPCA\\_iteration](#), [inv\\_GPCA](#)

**Examples**

```

library(RMAWGEN)
set.seed(1222)
N <- 20
x <- rexp(N)
y <- x+rnorm(N)
df <- data.frame(x=x,y=y)

GPCA <- GPCA_iteration(df,extremes=TRUE)

x <- rnorm(N)
y <- x+rnorm(N)
dfn <- data.frame(x=x,y=y)

GPCAn <- GPCA_iteration(dfn,extremes=TRUE)

```

---

inv_GPCA	<i>This function makes an inverse Gaussianization procedure based on PCA iteration ( see <a href="#">inv_GPCA_iteration</a></i>
----------	---

---

**Description**

This function makes an inverse Gaussianization procedure based on PCA iteration ( see [inv\\_GPCA\\_iteration](#)

**Usage**

```
inv_GPCA(x = NULL, GPCA_param, type = 3, extremes = TRUE)
```

**Arguments**

x	gaussian random variable to transform
GPCA_param	<a href="#">GPCA-class</a> S3 object returned by the function <a href="#">GPCA</a>
type	see <a href="#">normalizeGaussian_severalstations</a>
extremes	see <a href="#">normalizeGaussian_severalstations</a>

**Value**

the non-Gaussian random variable

**Note**

This function re-iterates the inverse of equation (1) of "PCA Gaussianization for One-Class Remote Sensing Image" by V. Laparra et al., <http://ieeexplore.ieee.org/document/5413808/>

**Author(s)**

Emanuele Cordano

**See Also**[GPCA](#), [GPCA\\_iteration](#), [inv\\_GPCA\\_iteration](#), [inv\\_GPCA](#)**Examples**

```

library(RMAWGEN)
set.seed(1222)
nIterations <- 30
N <- 20
x <- rexp(N)
y <- x+rnorm(N)
df <- data.frame(x=x,y=y)

GPCA <- GPCA(df,n=nIterations,extremes=TRUE)

x <- rnorm(N)
y <- x+rnorm(N)
dfn <- data.frame(x=x,y=y)

GPCAn <- GPCA(dfn,n=nIterations,extremes=TRUE)

df_out <- inv_GPCA(GPCA_param=GPCA,extremes=TRUE)
dfn_out <- inv_GPCA(GPCA_param=GPCAn,extremes=TRUE)

```

---

inv_GPCA_iteration	<i>This function makes an inverse iteration of PCA-Gaussianization process</i>
--------------------	--

---

**Description**

This function makes an inverse iteration of PCA-Gaussianization process

**Usage**

```
inv_GPCA_iteration(x = GPCA_iter_param$x_next, GPCA_iter_param,
  type = 3, extremes = TRUE)
```

**Arguments**

**x** matrix of gaussian random variable to transform

**GPCA\_iter\_param** [GPCAiteration](#) S3 object returned by the function [GPCA\\_iteration](#) corresponding the related direct iteration

type            see [normalizeGaussian\\_severalstations](#)  
 extremes        see [normalizeGaussian\\_severalstations](#)

**Value**

the non-Gaussian random variable

**Note**

This function is based on the inverse of the equation (1) of "PCA Gaussianization for One-Class Remote Sensing Image" by V. Laparra et al., <http://ieeexplore.ieee.org/document/5413808/>

**See Also**

[GPCA](#), [GPCA\\_iteration](#), [inv\\_GPCA\\_iteration](#), [inv\\_GPCA](#), [GPCA-class](#) for 'GPCA' S3 class

**Examples**

```
library(RMAWGEN)
set.seed(1222)
N <- 20
x <- rexp(N)
y <- x+rnorm(N)
df <- data.frame(x=x,y=y)

GPCA <- GPCA_iteration(df,extremes=TRUE)

x <- rnorm(N)
y <- x+rnorm(N)
dfn <- data.frame(x=x,y=y)

GPCAn <- GPCA_iteration(dfn,extremes=TRUE)

df_out <- inv_GPCA_iteration(GPCA_iter_param=GPCA,extremes=TRUE)
dfn_out <- inv_GPCA_iteration(GPCA_iter_param=GPCAn,extremes=TRUE)
```

---

is.monthly.climate	<i>Verifies if 'climate' represents the monthly climatology in one year, i.e 'climate' is monthly.climate type matrix whose rows represent months and each column represents a station. It is also used in <a href="#">setComprehensiveTemperatureGeneratorParameters</a>.</i>
--------------------	--

---

**Description**

Verifies if 'climate' represents the monthly climatology in one year, i.e 'climate' is monthly.climate type matrix whose rows represent months and each column represents a station. It is also used in [setComprehensiveTemperatureGeneratorParameters](#).

**Usage**

```
is.monthly.climate(climate, nstation = 3, nmonth = 12,
  verbose = TRUE)
```

**Arguments**

climate	matrix containing the 'monthly climatology' data
nstation	number of variable measurement stations (columns of the matrix 'climate')
nmonth	number of months in one year (it can be different if climate is represented by seasonal averages or others), Default is 12 (recommended). (it can be different if climate is represented by seasonal averages, in this case 4)
verbose	Prints output and warning messages only if is TRUE.

**Value**

A logical variable if the matrix 'climate' is monthly.climate type

**Author(s)**

Emanuele Cordano, Emanuele Eccel

**See Also**

[setComprehensiveTemperatureGeneratorParameters](#)

---

months_f	<i>months REPLACEMENT</i>
----------	---------------------------

---

**Description**

months REPLACEMENT

**Usage**

```
months_f(x, ...)
```

**Arguments**

x	an object. See <a href="#">months</a>
...	arguments

---

NewVAReventRealization

*Generates a new realization of a VAR model*

---

### Description

Generates a new realization of a VAR model

### Usage

```
NewVAReventRealization(var, xprev, noise, exogen = NULL, B = NULL)
```

### Arguments

var	A VAR model represented by a varest object as returned by <a href="#">getVARmodel</a> or <a href="#">VAR</a>
xprev	previous status of the random variable
noise	uncorrelated or white noise (residual). Default is <code>rnorm(length(xprev))</code> (or <code>rnorm(ncol(B))</code> )
exogen	vector containing the values of the "exogen" variables (predictor) for the generation
B	matrix of coefficients for the vectorial white-noise component

### Value

a vector of values

### Author(s)

Emanuele Cordano, Emanuele Eccel

### See Also

[forecastEV](#), [forecastResidual](#)

---

`newVARmultieventRealization`*Generates several realizations of a VAR model*

---

**Description**

Generates several realizations of a VAR model

**Usage**

```
newVARmultieventRealization(var, xprev = rnorm(var@VAR$K * var@VAR$p),  
  exogen = NULL, nrealization = 10, B = t(chol(cov(residuals(var)))),  
  extremes = TRUE, type = 3, noise = NULL)
```

**Arguments**

<code>var</code>	A VAR model represented by a <code>varest2</code> object as returned by <a href="#">getVARmodel</a>
<code>xprev</code>	previous status of the random variable
<code>exogen</code>	matrix containing the values of the "exogen" variables (predictor) for the generation
<code>nrealization</code>	number of realization (e.g. days to simulate). If <code>exogen</code> is not <code>NULL</code> and it is a matrix, it must be lower or equal to the number of rows of <code>exogen</code>
<code>B</code>	matrix of coefficients for the vector white-noise component
<code>extremes, type</code>	see <a href="#">inv_GPCA</a>
<code>noise</code>	stochastic noise to add for variable generation. Default is <code>NULL</code> and it is automatically randomly generated according to matrix <code>B</code> . If the VAR model ( <code>var</code> argument) does not fit well the residuals (e.g. non-normality, non-seriality or heteroskedasticity) and the white noise is manually inserted, in this case argument <code>B</code> is not taken into account.

**Value**

a matrix of values

**Author(s)**

Emanuele Cordano, Emanuele Eccel

---

normality_test	normality.test <i>method for varest2 object</i>
----------------	---

---

**Description**

normality.test method for varest2 object

**Usage**

```
normality_test(object, ...)
```

**Arguments**

object	a varest2 object
...	passed arguments

**See Also**

[normality.test](#)

---

normalizeGaussian	<i>Converts a random variable x extracted by a population represented by the sample data or sample to a normally-distributed variable with assigned mean and standard deviation or vice versa in case inverse is TRUE</i>
-------------------	---

---

**Description**

Converts a random variable x extracted by a population represented by the sample data or sample to a normally-distributed variable with assigned mean and standard deviation or vice versa in case inverse is TRUE

**Usage**

```
normalizeGaussian(x = 0, data = x, cpf = NULL, mean = 0, sd = 1,
  inverse = FALSE, step = NULL, prec = 10^-4, type = 3,
  extremes = TRUE, sample = NULL)
```

**Arguments**

x	value or vector of values to be converted
data	a sample of data on which a non-parametric probability distribution is estimated
cpf	cumulative probability distribution. If NULL (default) is calculated as <a href="#">ecdf</a> (data)
mean	mean (expected value) of the normalized random variable. Default is 0.



sd	standard deviation of the normalized random variable. Default is 1.
inverse	logical value. If TRUE the function works inversely (the opposite way). Default is FALSE.
step	vector of values in which step discontinuities of the cumulative probability function occur. Default is NULL
prec	amplitude of the neighbourhood of the step discontinuities where cumulative probability function is treated as non-continuous.
type	see <a href="#">quantile</a>
extremes	logical variable. If TRUE (default) the probability or frequency is multiplied by

$$\frac{N}{N + 1}$$

where  $N$  is the length of data

sample	a character string or NULL containing sample or probability distribution information. Default is NULL
--------	---

### Value

the normalized variable or its inverse

@note This function makes a Marginal Gaussianization. See the R code for further details

### Author(s)

Emanuele Cordano, Emanuele Eccel

---

normalizeGaussian\_prec

*Converts precipitation values to "Gaussinized" normally-distributed values taking into account the probability of no precipitation occurrences. values or vice versa in case inverse is TRUE*

---

### Description

Converts precipitation values to "Gaussinized" normally-distributed values taking into account the probability of no precipitation occurrences. values or vice versa in case inverse is TRUE

### Usage

```
normalizeGaussian_prec(x = 0, data = x, cpf = NULL, mean = 0,
  sd = 1, inverse = FALSE, type = 3, extremes = TRUE,
  sample = NULL, qnull = 0, valmin = 1)
```

**Arguments**

x	value or vector of values to be converted
data	a sample of data on which a non-parametric probability distribution is estimated
cpf	cumulative probability distribution. If NULL (default) is calculated as <code>ecdf(data)</code>
mean	mean (expected value) of the normalized random variable. Default is 0.
sd	standard deviation of the normalized random variable. Default is 1.
inverse	logical value. If TRUE the function works inversely (the opposite way). Default is FALSE.
type	see <a href="#">quantile</a>
extremes	logical variable. If TRUE (default) the probability or frequency is multiplied by

$$\frac{N}{N + 1}$$

where  $N$  is the length of data

sample	a character string or NULL containing sample or probability distribution information. Default is NULL
qnull	probability of no precipitation occurrence
valmin	minimum value of precipitation to consider a wet day

**Value**

the normalized variable or its inverse

**Note**

In the version 1.2.5 of **RMAWGEN** This function is deprecated and not used.

**Author(s)**

Emanuele Cordano, Emanuele Eccel

**See Also**

[normalizeGaussian](#)

**Examples**

```
library(RMAWGEN)
NDATA <- 1000
occurrence <- as.logical(runif(NDATA)>0.5)
prec <- rexp(NDATA,rate=1/3)
prec[!occurrence] <- 0
valmin <- 0.5 #0.01
x <- normalizeGaussian_prec(x=prec, valmin=valmin)
prec2 <- normalizeGaussian_prec(x=x, data=prec, valmin=valmin, inverse=TRUE)
qqplot(prec,prec2)
```

```

occurrence3 <- as.logical(runif(NDATA)>0.5)
prec3 <- rexp(NDATA,rate=1/3)
prec3[!occurrence3] <- 0
x3 <- normalizeGaussian_prec(x=prec3, valmin=valmin)

qqplot(x,x3)
abline(0,1)

```

---

normalizeGaussian\_severalstations

*Converts several samples x random variable extracted by populations represented by the columns of data respectively or sample to a normally-distributed samples with assigned mean and standard deviation or vice versa in case inverse is TRUE*

---

## Description

Converts several samples x random variable extracted by populations represented by the columns of data respectively or sample to a normally-distributed samples with assigned mean and standard deviation or vice versa in case inverse is TRUE

## Usage

```

normalizeGaussian_severalstations(x, data = x, cpf = NULL, mean = 0,
  sd = 1, inverse = FALSE, step = NULL, prec = 10^-4, type = 3,
  extremes = TRUE, sample = NULL, origin_x = NULL,
  origin_data = NULL)

```

## Arguments

x	value to be converted
data	a sample of data on which a non-parametric probability distribution is estimated
cpf	cumulative probability distribution. If NULL (default) is calculated as <a href="#">ecdf</a> (data)
mean	mean (expected value) of the normalized random variable. Default is 0.
sd	standard deviation of the normalized random variable. Default is 1.
inverse	logical value. If TRUE the function works inversely (the opposite way). Default is FALSE.
step	vector of values in which step discontinuities of the cumulative probability function occur. Default is NULL
prec	amplitude of the neighbourhood of the step discontinuities where cumulative probability function is treated as non-continuous.
type	see <a href="#">quantile</a>

extremes logical variable. If TRUE (default) the probability or frequency is multiplied by

$$\frac{N}{N+1}$$

where  $N$  is the length of data

sample information on how to sample  $x$  and  $data$ . Default is NULL, this means that the values of each column of  $x$  and  $data$  belong to the same sample. If  $x$  and  $data$  are sampled for each month separately, it is set to monthly.

origin\_x date corresponding to the first row of  $x$

origin\_data date corresponding to the first row of  $data$

### Value

a matrix with the normalized variable or its inverse

### Note

It applies [normalizeGaussian](#) for each column of  $x$  and  $data$ . See the R code for further details

### Author(s)

Emanuele Cordano, Emanuele Eccel

### See Also

[normalizeGaussian](#)

### Examples

```
## Not run:
library(RMAWGEN)

set.seed(1234)
N <- 30
x <- rexp(N)
y <- x+rnorm(N)
df <- data.frame(x=x,y=y)

dfg <- normalizeGaussian_severalStations(df,data=df,extremes=TRUE,inverse=FALSE)

dfi <- normalizeGaussian_severalStations(dfg,data=df,extremes=TRUE,inverse=TRUE)

N <- 365*2
origin <- "1981-01-01"
x <- rexp(N)
y <- x+rnorm(N)
df <- data.frame(x=x,y=y)

dfgm <- normalizeGaussian_severalStations(df,data=df,extremes=TRUE,
```

```

inverse=FALSE,origin_x=origin,origin_data=origin,sample="monthly")

dfim <- normalizeGaussian_severalstations(dfg,data=df,extremes=TRUE,
inverse=TRUE,origin_x=origin,origin_data=origin,sample="monthly")

## Compatibility with 'lubridate' package

library(lubridate)

N <- 30
x <- rexp(N)
y <- x+rnorm(N)
df <- data.frame(x=x,y=y)

dfg <- normalizeGaussian_severalstations(df,data=df,extremes=TRUE,inverse=FALSE)

dfi <- normalizeGaussian_severalstations(dfg,data=df,extremes=TRUE,inverse=TRUE)

N <- 365*2
origin <- "1981-01-01"
x <- rexp(N)
y <- x+rnorm(N)
df <- data.frame(x=x,y=y)

dfgm <- normalizeGaussian_severalstations(df,data=df,extremes=TRUE,
inverse=FALSE,origin_x=origin,origin_data=origin,sample="monthly")

dfim <- normalizeGaussian_severalstations(dfg,data=df,extremes=TRUE,
inverse=TRUE,origin_x=origin,origin_data=origin,sample="monthly")

## End(Not run)

```

---

```
normalizeGaussian_severalstations_prec
```

*DEPRECATED* Converts several samples x random variable (daily precipitation values) extracted by populations represented by the columns of data respectively or sample to a normally-distributed samples with assigned mean and standard deviation or vice versa in case inverse is TRUE using the function [normalizeGaussian\\_prec](#)

---

### Description

DEPRECATED Converts several samples x random variable (daily precipitation values) extracted by populations represented by the columns of data respectively or sample to a normally-distributed

samples with assigned mean and standard deviation or vice versa in case inverse is TRUE using the function [normalizeGaussian\\_prec](#)

### Usage

```
normalizeGaussian_severalstations_prec(x, data = x, cpf = NULL,
  mean = 0, sd = 1, inverse = FALSE, qnull = NULL, valmin = 0.5,
  type = 3, extremes = TRUE, sample = NULL, origin_x = NULL,
  origin_data = NULL)
```

### Arguments

x	value to be converted
data	a sample of data on which a non-parametric probability distribution is estimated
cpf	cumulative probability distribution. If NULL (default) is calculated as <a href="#">ecdf</a> (data)
mean	mean (expected value) of the normalized random variable. Default is 0.
sd	standard deviation of the normalized random variable. Default is 1.
inverse	logical value. If TRUE the function works inversely (the opposite way). Default is FALSE.
qnull	probability of no precipitation occurrence. (It can be a matrix in case sample="monthly")
valmin	minimum value of precipitation to consider a wet day
type	see <a href="#">quantile</a>
extremes	logical variable. If TRUE (default) the probability or frequency is multiplied by $\frac{N}{N + 1}$ where $N$ is the length of data
sample	information about sample or probability distribution. Default is NULL
origin_x	date corresponding to the first row of x
origin_data	date corresponding to the first row of data

### Value

a matrix or a data.frame with the normalized variable or its inverse

### Note

In the version 1.2.5 of **RMAWGEN** This function is deprecated and not used.

### Author(s)

Emanuele Cordano, Emanuele Eccel

### See Also

[normalizeGaussian\\_prec](#)

---

plotDailyClimate	<i>Plots daily climatology through one year</i>
------------------	---

---

## Description

Plots daily climatology through one year

## Usage

```
plotDailyClimate(data, title = "Daily_Averaged_Temperature_in_one_year",  
  origin = "1961-1-1", when = "1979-1-1",  
  ylab = "Temperature [degC]", xlab = "Time [days]", nday = 365,  
  bicolor = FALSE, col = "black", lwd = 1)
```

## Arguments

data	matrix whose columns contain daily-averaged climatic series of variables (e.g. maximum or minimum daily averaged temperature obtained by spline interpolation of monthly climatology)
title, xlab, ylab, col, lwd	see <a href="#">plot.default</a>
origin	origin date corresponding to the first row of data
when	start day for daily climatology plot
nday	number of days in one year. Default is 365.
bicolor	logical variable. If TRUE and data represents climatologies of minimum and maximum daily temperature, the lines are plotted with blue and red colors respectively.

## Value

a matrix containing the plotted variables

## Author(s)

Emanuele Cordano, Emanuele Eccel

---

plot\_sample

*It makes a plot by sampling (e.g. monthly) the variables x and y*


---

### Description

It makes a plot by sampling (e.g. monthly) the variables x and y

### Usage

```
plot_sample(x, y = normalizeGaussian_severalstations(x =
  as.data.frame(x), data = as.data.frame(data), origin_x = origin_x,
  origin_data = origin_data, sample = sample, step = step, prec = prec)[,
  1], xlim = range(x, na.rm = TRUE), legend_position = "topleft",
  ylim = range(y, na.rm = TRUE), pch = 1, col = 1, col_max = 0.9,
  col_min = 0.1, origin, sample = NULL, xhist = hist(x, breaks =
  breaks, plot = FALSE), yhist = hist(y, breaks = breaks, plot = FALSE),
  axes = FALSE, step = NULL, prec = 1e-04, breaks = 50,
  origin_x = origin, origin_data = origin, data = x, xlab = "",
  ylab = "", color = FALSE, gray = TRUE, sort = FALSE,
  valmin_x = valmin, valmin_y = valmin, valmin = -9999,
  abline = c(0, 1), ...)
```

### Arguments

x	vector of input data
y	vector of second input data. Default is <code>normalizeGaussian_severalstations(x=as.data.frame(x),</code>
xlim, ylim, xlab, ylab	see <a href="#">plot.default</a> (Graphic)
legend_position	legend position. Default is "topleft". See <a href="#">legend</a> .
pch	integer single or multi values for pch (see <a href="#">plot.default</a> ). Default is 1.
col	integer single or multi values for col (see <a href="#">plot.default</a> ). Default is 1.
col_max	maximum value for color scale to apply to <a href="#">rainbow</a> or <a href="#">rainbow</a> . Utilized if col is not a vector and both gray or color are TRUE. Default is 0.9 .
col_min	minimum value for color scale to apply to <a href="#">rainbow</a> or <a href="#">rainbow</a> . Utilized if col is not a vector and both gray or color are TRUE. Default is 0.1 .
origin	date of the first row of x. See <a href="#">normalizeGaussian_severalstations</a> .
sample	string character containing information how to sample x and y. Default is NULL. If NULL no sampling is done. see <a href="#">normalizeGaussian_severalstations</a> . Only NULL or "monthly" options are implemented.
xhist	frequency histogram for x. Default is <code>hist(x,breaks=breaks,plot=FALSE)</code> . If it is NULL, no marginal histograms appear.
yhist	frequency histogram for y. Default is <code>hist(y,breaks=breaks,plot=FALSE)</code> . If it is NULL, no marginal histograms appear. = <code>hist(y,breaks=breaks,plot=FALSE)</code> ,



axes	see <a href="#">barplot</a>
step, prec	see <a href="#">normalizeGaussian_severalstations</a>
breaks	see <a href="#">hist</a>
origin_x	see <a href="#">normalizeGaussian_severalstations</a> . Default value is set equal to origin.
origin_data	<a href="#">normalizeGaussian_severalstations</a> . Default value is set equal to origin.
data	<a href="#">normalizeGaussian_severalstations</a> . Default value is set equal to x.
color	logical value. If TRUE and if col is unspecified, a color scale is applied according to col_min and col_max (see <a href="#">rainbow</a> ). Default is FALSE.
gray	logical value. If TRUE and if col is unspecified, a color scale is applied according to col_min and col_max (see <a href="#">gray</a> ). Default is TRUE.
sort	logical value. If TRUE, x and y are sorted and a Q-Q plot is presented. Deafault is FALSE.
valmin_x	numerical threshold value over which the variable x is plotted. It is enabled only if sort is set TRUE.
valmin_y	numerical threshold value over which the variable y is plotted. It is enabled only if sort is set TRUE.
valmin	numerical threshold value for valmin_y and valmin_x if there are not specified.
abline	arguments for <a href="#">abline</a> function. Default is c(0,1). If it is NULL, <a href="#">abline</a> is disabled and not called.
...	see graphical parametes on <a href="#">plot.default</a> <pre>@usage plot_sample(x, y = normalizeGaussian_severalstations(x = as.data.frame(x), data = as.data.frame(data), origin_x = origin_x, origin_data = origin_data, sam- ple = sample, step = step, prec = prec)[, 1], xlim = range(x, na.rm = TRUE), legend_position = "topleft", ylim = range(y, na.rm = TRUE), pch = 1, col = 1, col_max = 0.9, col_min = 0.1, origin, sample = NULL, xhist = hist(x, breaks = breaks, plot = FALSE), yhist = hist(y, breaks = breaks, plot = FALSE), axes = FALSE, step = NULL, prec = 1e-04, breaks = 50, origin_x = origin, origin_data = origin, data = x, xlab = "", ylab = "", color = FALSE, gray = TRUE, sort = FALSE, valmin_x = valmin, valmin_y = valmin, valmin = -9999, abline = c(0, 1), ...)</pre>

**Value**

0 in case of success

**Note**

It makes a plot between x and y and shows their respective probibilty histograms. If y is missing, it is automatically calculated as one-dimensional Gaussianization of x through the function [normalizeGaussian\\_severalstations](#).

**See Also**

[plot.default](#), [extractmonths](#), see [normalizeGaussian\\_severalstations](#)

**Examples**

```
## Not run:

library(lubridate)
data(trentino)
plot_sample(x=TEMPERATURE_MIN$T0090, sample="monthly",
  origin="1958-1-1", axes=FALSE, xlab="Tn [ degC]",
  ylab="x")

set.seed(123456)
z <- rexp(10000, rate=0.5)
x <- normalizeGaussian(x=z, data=z)
plot_sample(x=z, xlab="z", ylab="x")

## End(Not run)
```

---

PrecipitationEndDay *Gets the last day in a precipitation time series, expressed in decimal julian days since 1970-1-1 00:00 UTC*

---

**Description**

@author Emanuele Cordano, Emanuele Eccel

**Usage**

```
PrecipitationEndDay(name, station_names, end_day)
```

**Arguments**

name	character ID of the station
station_names	vector containing the IDs (characters) of the considered meteorological stations. An example is STATION_NAMES defined in <a href="#">trentino</a> .
end_day	vector containing the measurement end day. An example is TEMPERATURE_MEASUREMENT_END_DAY defined in <a href="#">trentino</a> .

**Value**

the precipitation measurement end day given the vectors of station IDs and the precipitation measurement end days

**Examples**

```
data(trentino)
PrecipitationEndDay("T0099", station_names=STATION_NAMES, end_day=PRECIPITATION_MEASUREMENT_END_DAY)
```

---

PrecipitationStartDay *Gets the first day in a precipitation time series, expressed in decimal julian days since 1970-1-1 00:00 UTC*

---

### Description

@author Emanuele Cordano

### Usage

```
PrecipitationStartDay(name, station_names, start_day)
```

### Arguments

name	character ID of the station
station_names	vector containing the IDs (characters) of the considered meteorological stations. An example is STATION_NAMES defined in the <a href="#">trentino</a> dataset.
start_day	vector containing the precipitation measurement start day. An example is TEMPERATURE_MEASUREMENT_START_DAY defined in the <a href="#">trentino</a> dataset.

### Value

the precipitation measurement start day given the vectors of station IDs and the respective precipitation measurement start days

### Examples

```
data(trentino)
PrecipitationStartDay("T0099",
  station_names=STATION_NAMES,
  start_day=PRECIPITATION_MEASUREMENT_START_DAY)
```

---

print.GPCA                      print S3 method for GPCA or GPCA\_iteration object

---

### Description

print S3 method for GPCA or GPCA\_iteration object

**Usage**

```
## S3 method for class 'GPCA'
print(x, rmin = 1, rmax = 4, cmin = rmin,
      cmax = rmax, ...)

## S3 method for class 'GPCAiteration'
print(x, rmin = 1, rmax = 4, cmin = rmin,
      cmax = rmax, ...)
```

**Arguments**

x a GPCA or GPCAiteration object  
 rmin, rmax, cmin, cmax maximum and minimum rows and columns to be printed  
 ... passed arguments

**See Also**

[GPCA, GPCA\\_iteration](#)  
[GPCA\\_iteration](#)

---

qqplot.lagged	<i>This function creates a Q-Q plot of the lag-lag moving cumulative addition of the values in the samples x, y, z</i>
---------------	--

---

**Description**

This function creates a Q-Q plot of the lag-lag moving cumulative addition of the values in the samples x, y, z

**Usage**

```
qqplot.lagged(x = rnorm(1000), y = rnorm(1000), z = NULL,
              when = 1:length(x), lag = 1, pch = 1, ...)
```

**Arguments**

x, y samples. If x is a data frame, y and z can be omitted.  
 z further samples organized as a list  
 when (integer) indices of x and y on which the Q-Q plot is made.  
 lag lag (current index included) on whose value the addition is made.  
 pch a vector of plotting characters or symbols: see [points](#)  
 ... further arguments for [qqplot](#)

**Value**

the Q-Q plot

**See Also**

[qqplot](#)

---

qqplotprecWGEN	<i>Makes a qqplot of measured and simulated data for several stations.</i>
----------------	--

---

**Description**

Makes a qqplot of measured and simulated data for several stations.

**Usage**

```
qqplotprecWGEN(measured, simulated, xlab = "simulated[mm]",
  ylab = "measured[mm]", title = "daily precipitation",
  station = NULL, diff = FALSE, quantile = 0)
```

**Arguments**

measured	matrix containing measured data (each station corresponds to a column)
simulated	matrix containing respective generated data (each station corresponds to a column)
xlab, ylab	see <a href="#">plot.default,qqplotWGEN</a>
title	title
station	character vector containing IDs of analyzed stations. If NULL (default) all stations (columns of simulated and measured) are considered
diff, quantile	see <a href="#">qqplotWGEN</a>

**Value**

0 in case of success

**Note**

It uses [qqplotWGEN](#) and makes a figure for each pair of columns from measured and simulated. See the R code for further details.

**Author(s)**

Emanuele Cordano, Emanuele Eccel

---

qqplotprecWGEN\_seasonal

*Makes four seasonal qqplots (winter, spring, summer and autumn) of measured and simulated data for several stations.*

---

### Description

Makes four seasonal qqplots (winter, spring, summer and autumn) of measured and simulated data for several stations.

### Usage

```
qqplotprecWGEN_seasonal(measured, simulated, origin = "1961-1-1",
  xlab = "simulated[mm]", ylab = "measured[mm]",
  title = "daily_precipitation", directorypdf,
  station = names(simulated))
```

### Arguments

measured	matrix containing measured data (each station corresponds to a column)
simulated	matrix containing respective generated data (each station corresponds to a column)
origin	first day of data, see <a href="#">extractmonths</a> for format and other information
xlab, ylab	see <a href="#">plot.default,qqplotWGEN</a>
title	title
directorypdf	name of the directory (path included) where to save the outputs
station	character vector containing IDs of analyzed stations. If NULL (default) all stations (columns of simulated and measured) are considered

### Value

0 in case of success

### Note

Uses [qqplotprecWGEN](#) for each season of collected data and saves the output on pdf files. See the R code for further details.

### Author(s)

Emanuele Cordano, Emanuele Eccel

### See Also

[qqplotprecWGEN](#), [extractmonths](#)

---

qqplotTnTxWGEN	<i>Makes a qqplot of measured and simulated data for several stations.</i>
----------------	--

---

**Description**

Makes a qqplot of measured and simulated data for several stations.

**Usage**

```
qqplotTnTxWGEN(measured, simulated, xlab = "simulated[degC]",  
               ylab = "measured[degC]", titles = c("Q-Qplot_An._Tx",  
               "Q-Qplot_An._Tn"), station = NULL, diff = FALSE, quantile = 0)
```

**Arguments**

measured	matrix containing measured data (each station corresponds to a column)
simulated	matrix containing respective generated data (each station corresponds to a column)
xlab, ylab	see <a href="#">plot.default,qqplotWGEN</a>
titles	titles that will be added to main argument of <a href="#">plot.default</a>
station	character vector containing IDs of analyzed station. If NULL (default) all station (columns of simulated and measured) are considered
diff, quantile	see <a href="#">qqplotWGEN</a>

**Value**

0 in case of success

**Note**

It uses [qqplotWGEN](#) and makes a figure for each pair of columns from measured and simulated. See the R code for further details.

**Author(s)**

Emanuele Cordano, Emanuele Eccel

---

qqplotTnTxWGEN\_seasonal

*Makes four seasonal qqplots (winter, spring, summer and autumn) of measured and simulated data for several stations.*

---

### Description

Makes four seasonal qqplots (winter, spring, summer and autumn) of measured and simulated data for several stations.

### Usage

```
qqplotTnTxWGEN_seasonal(measured, simulated, origin = "1961-1-1",
  xlab = "simulated[degC]", ylab = "measured[degC]",
  titles = c("Q-Qplot_An._Tx", "Q-Qplot_An._Tn"), directorypdf,
  station = NULL)
```

### Arguments

measured	matrix containing measured data (each station corresponds to a column)
simulated	matrix containing respective generated data (each station corresponds to a column)
origin	first day of data, see <a href="#">extractmonths</a> for format and other information
xlab, ylab	see <a href="#">plot.default,qqplotWGEN</a>
titles	titles that will be added
directorypdf	name of the directory (path included) where to save the outputs
station	character vector containing IDs of analyzed station. If NULL (default) all station (columns of simulated and measured) are considered

### Value

0 in case of success

### Note

Uses [qqplotTnTxWGEN](#) for each seasons of collected data and saves the output on pdf files. See the R code for further details.

### Author(s)

Emanuele Cordano, Emanuele Eccel

### See Also

[qqplotTnTxWGEN](#), [extractmonths](#)



---

qqplotWGEN	<i>Makes a qqplot and Wilcoxon test between the two columns of val</i>
------------	--

---

**Description**

Makes a qqplot and Wilcoxon test between the two columns of val

**Usage**

```
qqplotWGEN(val, xlab = "simulated", ylab = "measured",
  main = "title", ylim = c(min(val), max(val)), xlim = c(min(val),
  max(val)), diff = FALSE, quantile = 0)
```

**Arguments**

val	a matrix with two columns containing the two samples to be compared
xlab, ylab, main	see <a href="#">plot.default</a>
xlim, ylim	see <a href="#">plot.default</a>
diff	logical variable, if TRUE the function is applied to diff(val) instead of val. See <a href="#">diff</a>
quantile	quantile value on which data samples in val are considered. Default is 0.

**Value**

Wilcoxon test between the two columns of 'val'

**Author(s)**

Emanuele Cordano, Emanuele Eccel

---

qqplot_RMAWGEN_Tx	<i>It makes the Q-Q plots observed vs generated time series of daily maximum, minimum temperature and daily thermal range for a list of collected stochastic generations</i>
-------------------	--

---

**Description**

It makes the Q-Q plots observed vs generated time series of daily maximum, minimum temperature and daily thermal range for a list of collected stochastic generations

**Usage**

```
qqplot_RMAWGEN_Tx(Tx_mes, Tx_gen, Tn_gen, Tn_mes, Tx_spline = NULL,
  Tn_spline = NULL, xlab = "observed", ylab = "simulated",
  when = 1:nrow(Tx_mes), main = names(Tx_gen), station, pdf = NULL,
  xlim = range(Tx_mes), ylim = xlim, cex = 0.4, cex.main = 1,
  cex.lab = 1, cex.axis = 1)
```

```
qqplot_RMAWGEN_Tn(Tx_mes, Tx_gen, Tn_gen, Tn_mes, Tx_spline = NULL,
  Tn_spline = NULL, xlab = "observed", ylab = "simulated",
  when = 1:nrow(Tn_mes), main = names(Tn_gen), station, pdf = NULL,
  xlim = range(Tn_mes), ylim = xlim, cex = 0.4, cex.main = 1,
  cex.lab = 1, cex.axis = 1)
```

```
qqplot_RMAWGEN_deltaT(Tx_mes, Tx_gen, Tn_gen, Tn_mes, xlab = "observed",
  ylab = "simulated", when = 1:nrow(Tx_mes), main = names(Tx_gen),
  station, pdf = NULL, xlim = range(Tx_mes - Tn_mes), ylim = xlim,
  cex = 0.4, cex.main = 1, cex.lab = 1, cex.axis = 1)
```

```
qqplot_RMAWGEN_prec(prec_mes, prec_gen, xlab = "observed",
  ylab = "simulated", when = 1:nrow(prec_mes),
  main = names(prec_gen), station, pdf = NULL,
  xlim = range(prec_mes), ylim = xlim, cex = 0.4, cex.main = 1,
  cex.lab = 1, cex.axis = 1, lag = 1)
```

**Arguments**

Tx_mes	data frame containing measured daily maximum temperature
Tx_gen	data frame containing generated daily maximum temperature
Tn_gen	data frame containing generated daily minimum temperature
Tn_mes	data frame containing measured daily minimum temperature
Tx_spline	data frame containing spline-interpolated daily maximum temperature. Default is NULL and not considered for Q-Q plot.
Tn_spline	data frame containing spline-interpolated daily minimum temperature Default is NULL and not considered for Q-Q plot.
xlab, ylab	lables of x and y axes. See <a href="#">qqplot</a> .
when	day indices on which the data frame are extracted for Q-Q plot. Default is 1:nrow(Tn_mes) (in <a href="#">qqplot_RMAWGEN_Tn</a> ) or 1:nrow(Tx_mes) (otherwise)
main	main titles for each plot. Default is names(Tn_gen) (in <a href="#">qqplot_RMAWGEN_Tn</a> ) or names(Tx_gen) (otherwise)
station	identification name (ID) of the station used for the Q-Q plot
pdf	name of pdf file if output is written in a pdf file
xlim	see <a href="#">qqplot</a> . Default is range(Tn_mes) (in <a href="#">qqplot_RMAWGEN_Tn</a> ) or range(Tx_mes) (in <a href="#">qqplot_RMAWGEN_Tx</a> ).or range(Tx_mes-Tn_mes) (in <a href="#">qqplot_RMAWGEN_deltaT</a> )
ylim, cex, cex.main, cex.lab, cex.axis	see <a href="#">qqplot</a> and <a href="#">plot</a>

prec_mes	data frame containing measured daily precipitation (in millimeters)
prec_gen	data frame containing generated daily precipitation (in millimeters)
lag	lag (current index included) on whose value the precipitation addition is made. See <a href="#">qqplot.lagged</a> .

**Note**

Tx\_gen, Tn\_gen and main must have an even number of elements.

**Author(s)**

Emanuele Cordano

---

removeNAs	<i>Replaces each entry of the rows containing NA values with NA</i>
-----------	---

---

**Description**

Replaces each entry of the rows containing NA values with NA

**Usage**

```
removeNAs(data)
```

**Arguments**

data	a matrix @author Emanuele Cordano, Emanuele Eccel
------	--

**Value**

the matrix data with the modified rows of NA values

**Note**

In [getVARmodel](#), when using [VAR](#) or [VARselect](#), all NAs will be removed

**See Also**

[getVARmodel](#)

---

rescaling_monthly	<i>This function adjusts the monthly mean to a daily weather dataset (e. g. spline-interpolated temperature)</i>
-------------------	--

---

### Description

This function adjusts the monthly mean to a daily weather dataset (e. g. spline-interpolated temperature)

### Usage

```
rescaling_monthly(data, val, origin = "1961-1-1")
```

### Arguments

data	data frame of wheather variables)
val	monthly means returned by <a href="#">getMonthlyMean</a>
origin	character string containing the gregorian date of the first day of data

### Value

A data frame with data of data rescaled with val for each month

### Note

It uses [months](#) and [julian](#)

### Author(s)

Emanuele Cordano  
@export

### See Also

[extractdays](#)

---

residuals.varest2      residuals S3 method for varest2 object

---

**Description**

residuals S3 method for varest2 object

**Usage**

```
## S3 method for class 'varest2'  
residuals(object, squared = FALSE, ...)
```

**Arguments**

object	a blockmatrix object
squared	logical value. Default is FALSE. If TRUE the method returns the squared residuals.
...	passed arguments

**Value**

residuals of object as a data frame. In case squared=TRUE , the squared residuals are returned, otherwise simple residuals are returned. The squared residuals can be useful in case of ARCH analysis.

**Author(s)**

Emanuele Cordano

---

serial\_test      serial.test function for varest2 object

---

**Description**

serial.test function for varest2 object

**Usage**

```
serial_test(object, ...)
```

**Arguments**

object	a varest2 object
...	passed arguments

**See Also**

[serial.test](#)

---

```
setComprehensiveTemperatureGeneratorParameters
```

*Computes climatic and correlation information useful for creating an auto-regressive random generation of maximum and minimum daily temperature. This function is called by [ComprehensiveTemperatureGenerator](#).*

---

## Description

Computes climatic and correlation information useful for creating an auto-regressive random generation of maximum and minimum daily temperature. This function is called by [ComprehensiveTemperatureGenerator](#).

## Usage

```
setComprehensiveTemperatureGeneratorParameters(station, Tx_all, Tn_all,
  mean_climate_Tn = NULL, mean_climate_Tx = NULL, Tx_spline = NULL,
  Tn_spline = NULL, year_max = 1990, year_min = 1961, leap = TRUE,
  nmonth = 12, verbose = FALSE, cpf = NULL, normalize = TRUE,
  sample = NULL, option = 2, yearly = FALSE)
```

## Arguments

station	character vector of the IDs of the considered meteorological stations
Tx_all	data frame containing daily maximum temperature of all meteorological station. See <a href="#">TEMPERATURE_MAX</a> for formatting.
Tn_all	data frame containing daily minimum temperature of all meteorological station. See <a href="#">TEMPERATURE_MIN</a> for formatting.
mean_climate_Tn	a matrix containing monthly mean minimum daily temperature for the considered station or an object as returned by <a href="#">getMonthlyMean</a> . If NULL, it is calculated. See input of <a href="#">is.monthly.climate</a>
mean_climate_Tx	a matrix containing monthly mean maximum daily temperature for the considered station or an object as returned by <a href="#">getMonthlyMean</a> . If NULL, it is calculated. See input of <a href="#">is.monthly.climate</a>
Tx_spline	daily timeseries (from the first day of year_min to the last day of year_max) of averaged maximum temperature which can be obtained by a spline interpolation of monthly mean values. Default is NULL and returned as output. See for spline interpolation utilized: <a href="#">splineInterpolateMonthlytoDailyforSeveralYears</a> .
Tn_spline	daily timeseries (from the first day of year_min to the last day of year_max) of averaged minimum temperature which can be obtained by a spline interpolation of monthly mean values. Default is NULL and returned as output. See for spline interpolation utilized: <a href="#">splineInterpolateMonthlytoDailyforSeveralYears</a> .
year_max	start year of the recorded (calibration) period
year_min	end year of the recorded (calibration) period

leap	logical variables. It is TRUE (Default) if leap years are considered
nmonth	number of months in one year. Default is 12.
verbose	logical variable
cpf	see <a href="#">normalizeGaussian_severalstations</a>
normalize	logical variable If TRUE <a href="#">normalizeGaussian_severalstations</a> is used, otherwise it is not. If option is 2, it is always TRUE.
sample	see <a href="#">normalizeGaussian_severalstations</a>
option	integer value. If 1, the generator works with minimum and maximum temperature, if 2 (default) it works with the average value between maximum and minimum temperature and the respective daily thermal range.
yearly	logical value. If TRUE the monthly mean values are calculated for each year from year_min to year_max separately. Default is FALSE.

### Value

This function creates and returns the following global variables:

data\_original matrix containing normalized and standardized data (i.e. data\_original)

data\_for\_var matrix returned from [normalizeGaussian\\_severalstations](#) by processing data\_original if normalize is TRUE), otherwise it is equal to data\_original.

Tn\_mes matrix containing measured minimum daily temperature in the analyzed time period ( $Tn_{mes}$ )

Tx\_mes matrix containing measured maximum daily temperature in the analyzed time period ( $Tx_{mes}$ )

Tm\_mes matrix calculated as to

$$\frac{Tx_{mes} + Tn_{mes}}{2}$$

DeltaT\_mes matrix corresponding to  $Tx_{mes} - Tn_{mes}$

monthly\_mean\_Tn matrix containing monthly means of minimum daily temperature for the considered station. It is calculated according to the input format [is.monthly.climate](#) if saveMonthlyClimate is TRUE.

monthly\_mean\_Tx matrix containing monthly means of maximum daily temperature for the considered station. It is calculated according to the input format [is.monthly.climate](#) if saveMonthlyClimate is TRUE.

Tx\_spline matrix containing the averaged daily values of maximum temperature obtained by a spline interpolation of the monthly climate monthly\_mean\_Tx or mean\_climate\_Tx using [splineInterpolateMonthlytoData](#) ( $Tx_s$ )

Tn\_spline matrix containing the averaged daily values of minimum temperature obtained by a spline interpolation of the monthly climate monthly\_mean\_Tn or mean\_climate\_Tn using [splineInterpolateMonthlytoData](#) ( $Tn_s$ )

SplineAdvTm matrix calculated as  $\frac{Tx_s + Tn_s}{2}$

SplineAdvDeltaT, matrix corresponding to  $Tx_s - Tn_s$

stdTn vector containing the standard deviation of minimum temperature anomalies  $Tn_{mes} - Tn_s$  ( $\sigma_{Tn}$ )

stdTx vector containing the standard deviation of maximum temperature anomalies  $Tx_{mes} - Tx_s$  ( $\sigma_{Tx}$ )

stdTm vector containing the standard deviation of "mean" temperature anomalies  $Tm_{mes} - Tm_s$  ( $\sigma_{Tm}$ )

Tn\_mes\_res standard core (standardization) of  $Tn_{mes}$  obtained by solving column by column the expression

$$\frac{Tn_{mes} - Tn_s}{\sigma_{Tn}}$$

Tx\_mes\_res standard core (standardization) of  $Tx_{mes}$  obtained by solving column-by-column the expression

$$\frac{Tx_{mes} - Tn_s}{sd_{Tm}}$$

Tm\_mes\_res standard core (standardization) of  $Tm_{mes}$  obtained by solving column-by-column the expression

$$\frac{Tm_{mes} - Tn_s}{sd_{Tm}}$$

DeltaT\_mes\_res equal to DeltaT\_mes

data\_original matrix obtained as cbind(Tx\_mes\_res, Tn\_mes\_res) if option==1, or cbind(Tm\_mes\_res, DeltaT\_mes\_res) if option==2

See the R code for further details.

#### Author(s)

Emanuele Cordano, Emanuele Eccel

#### See Also

[splineInterpolateMonthlytoDailyforSeveralYears,ComprehensiveTemperatureGenerator](#)

---

splineInterpolateMonthlytoDaily

*Interpolates monthly data to daily data using [spline](#) and preserving monthly mean values*

---

#### Description

Interpolates monthly data to daily data using [spline](#) and preserving monthly mean values

#### Usage

```
splineInterpolateMonthlytoDaily(nday = 365, val = as.matrix(cbind(1 *
  (0.5:11.5) * nday/12, 2 * (0.5:11.5) * nday/12)), origin = "1961-1-1",
  first_row = 1, last_row = nday, no_spline = FALSE,
  no_mean = FALSE)
```



**Arguments**

nday	number of days on which the daily data is requested, e.g. number of days in one year
val	matrix containing monthly mean data
origin	date corresponding to the first row of the returned matrix
first_row	row corresponding the first day of time interval where monthly mean conservation is applied
last_row	corresponding the last day of time interval where monthly mean conservation is applied
no_spline	logical value. If TRUE no spline interpolation is calculated and the daily value corresponds to the monthly average value. Default is FALSE.
no_mean	logical value. Default is FALSE. If TRUE the function output is not rescaled in order to maintain observed mean monthly values. @export

**Value**

a matrix or data frame with interpolated daily data

**Author(s)**

Emanuele Cordano, Emanuele Eccel

**See Also**

[spline](#), [splineInterpolateMonthlytoDailyforSeveralYears](#)

---

splineInterpolateMonthlytoDailyforSeveralYears

*Interpolates monthly data to daily data using  
[splineInterpolateMonthlytoDaily](#) for several years*

---

**Description**

Interpolates monthly data to daily data using [splineInterpolateMonthlytoDaily](#) for several years

**Usage**

```
splineInterpolateMonthlytoDailyforSeveralYears(val, start_year = 2010,  
  nyear = 1, leap = TRUE, offset = 2, no_spline = FALSE,  
  yearly = FALSE)
```

**Arguments**

val	matrix containing monthly mean data for one year
start_year	first year
nyear	number of years since start_year
leap	logical variable If TRUE (default) leap years are considered, otherwise they are not
offset	integer values. Default is 2. Number of years considered beyond the extremes in order to avoid edge errors
no_spline	logical value. If TRUE no spline interpolation is calculated and the daily value corresponds to the monthly average value. Default is FALSE.
yearly	logical value. If TRUE the result with men value per each month per each year. Default is FALSE.
	@return a matrix or data frame with interpolated daily data

**Author(s)**

Emanuele Cordano, Emanuele Eccel

**See Also**

[spline](#), [splineInterpolateMonthlytoDaily](#)

---

TemperatureEndDay	<i>Gets the last day in a temperature time series, expressed as decimal julian days since 1970-1-1 00:00 UTC</i>
-------------------	--

---

**Description**

Gets the last day in a temperature time series, expressed as decimal julian days since 1970-1-1 00:00 UTC

**Usage**

```
TemperatureEndDay(name, station_names, end_day)
```

**Arguments**

name	character ID of the station
station_names	vector containing the IDs (characters) of the considered meteorological stations. An example is STATION_NAMES defined in the <a href="#">trentino</a> dataset.
end_day	vector containing the measurement end day. An example is TEMPERATURE_MEASUREMENT_END_DAY defined in the <a href="#">trentino</a> dataset.

**Value**

the temperature measurement end day given the vectors of station IDs and the temperature measurement end days

**Author(s)**

Emanuele Cordano, Emanuele Eccel

**Examples**

```
data(trentino)
TemperatureEndDay("T0099",station_names=STATION_NAMES,end_day=TEMPERATURE_MEASUREMENT_END_DAY)
```

---

TemperatureStartDay *Gets the first day in a temperature time series, expressed as decimal julian days since 1970-1-1 00:00 UTC*

---

**Description**

@author Emanuele Cordano, Emanuele Eccel

**Usage**

```
TemperatureStartDay(name, station_names, start_day)
```

**Arguments**

name	character ID of the station
station_names	vector containing the IDs (characters) of the considered meteorological stations. An example is STATION_NAMES defined in the <a href="#">trentino</a> dataset.
start_day	vector containing the temperature measurement start day. Default is TEMPERATURE_MEASUREMENT_START_ defined in the <a href="#">trentino</a> dataset.
	@export

**Value**

the temperature measurement start day given the vectors of station IDs and the respective temperature measurement start days

@examples data(trentino) TemperatureStartDay("T0099",station\_names=STATION\_NAMES,start\_day=TEMPERATURE\_

---

trentino

*Trentino Dataset*

---

### **Description**

It contains the following variables:

TEMPERATURE\_MIN Data frame containing year, month, day and daily minimum temperature in 59 stations in Trentino region

TEMPERATURE\_MAX Data frame containing year, month, day and daily maximum temperature in 59 stations in Trentino region

PRECIPITATION Data frame containing year, month, day and daily precipitation in 59 stations in Trentino region

STATION\_NAMES Vector containing the names of the meteorological stations

ELEVATION Vector containing the elevations of the meteorological stations respectively

STATION\_LATLON Matrix containing the latitude and longitude coordinates, respectively, of the meteorological stations

LOCATION Vector containing the names of the location of each meteorological station

TEMPERATURE\_MEASUREMENT\_START\_DAY Vector containing the first days referred to midday (expressed as decimal julian day since 1970-1-1 00:00 UTC) of temperature measurement of each meteorological station

TEMPERATURE\_MEASUREMENT\_END\_DAY Vector containing the last days referred to midday (expressed as decimal julian day since 1-1-1970 00:00 UTC) of temperature measurement of each meteorological station

PRECIPITATION\_MEASUREMENT\_START\_DAY Vector containing the first days referred to midday (expressed as decimal julian day since 1-1-1970 00:00 UTC) of precipitation measurement of each meteorological station

PRECIPITATION\_MEASUREMENT\_END\_DAY Vector containing the last days referred to midday (expressed as decimal julian day since 1-1-1970) of precipitation measurement of each meteorological station

### **Usage**

```
data(trentino)
```

**Format**

Data frames and vectors

**Details**

This dataset stores all information about meteorological stations and instrumental timeseries. The user can easily use the package with his/her own data after replacing the values of such variables.

**Source**

Original data are provided by Provincia Autonoma di Trento (<https://www.meteotrentino.it/>), Fondazione Edmund Mach (<https://www.fmach.it>), Provincia Autonoma di Bolzano/Autome Provinz Bozen (<http://www.provincia.bz.it/meteo>), ARPA Lombardia (<https://www.arpalombardia.it/>), ARPA Veneto (<https://www.arpa.veneto.it/previsioni/it/html/index.php>).

This dataset is intended for research purposes only, being distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY.

---

varest-class

*varest-class*

---

**Description**

varest S3 class (formal definition) see [VAR](#)

**Details**

The details of the class are reported on [VAR](#) documentation in "vars" package

**Note**

Formal definition with [setOldClass](#) for the S3 class varest

**Author(s)**

Bernhard Pfaff

**Examples**

```
showClass("varest")
```

---

varest2-class	<i>varest2-class</i>
---------------	----------------------

---

**Description**

This class derives from a varest S3 class which is a list of objects describing a Vectorial Autoregressive Model (see [VAR](#))

**Details**

**VAR:** a varest S3 object created by [VAR](#)

**Note**

A varest2 object can be created by `new("varest2", ...)` or returned by the function [getVARmodel](#)

**Author(s)**

Emanuele Cordano

**Examples**

```
showClass("varest2")
```

---

VAR_mod	<i>Modified version of <a href="#">VAR</a> function allowing to describe white-noise as VAR-(0) model (i. e. varest objects)</i>
---------	--

---

**Description**

Modified version of [VAR](#) function allowing to describe white-noise as VAR-(0) model (i. e. varest objects)

**Usage**

```
VAR_mod(y, p = 1, type = c("const", "trend", "both", "none"),
        season = NULL, exogen = NULL, lag.max = NULL, ic = c("AIC", "HQ",
        "SC", "FPE"))
```

**Arguments**

y, p, type, season, exogen, lag.max, ic  
see [VAR](#) function

**Value**

a Vector Auto-Regressive model (VAR) as `varest` object

---

WhereIs	<i>Gets the toponym where a meteorological station is located</i>
---------	---

---

**Description**

Gets the toponym where a meteorological station is located

**Usage**

```
WhereIs(name, station_names, location)
```

**Arguments**

<code>name</code>	character ID of the station
<code>station_names</code>	vector containing the IDs (characters) of the considered meteorological stations. An example is <code>STATION_NAMES</code> defined in the <a href="#">trentino</a> dataset.
<code>location</code>	vector containing the toponyms. An example is <code>LOCATION</code> defined in the <a href="#">trentino</a> dataset.

**Value**

the location toponym given the vectors of station IDs and the respective location toponyms

**Author(s)**

Emanuele Cordano, Emanuele Eccel

**Examples**

```
data(trentino)
WhereIs("T0099", station_names=STATION_NAMES, location=LOCATION)
```

# Index

- \*Topic **auto-regression**
    - RMAWGEN-package, 3
  - \*Topic **classes**
    - GPCA-class, 31
    - GPCAiteration-class, 32
    - GPCAvarest2-class, 32
    - varest-class, 69
    - varest2-class, 70
  - \*Topic **dataset**
    - trentino, 68
  - \*Topic **models**
    - RMAWGEN-package, 3
  - \*Topic **package**
    - RMAWGEN-package, 3
  - \*Topic **precipitation**
    - RMAWGEN-package, 3
  - \*Topic **temperature**
    - RMAWGEN-package, 3
  - \*Topic **time-series**
    - RMAWGEN-package, 3
  - \*Topic **vector**
    - RMAWGEN-package, 3
- abline, 49
- acf, 6
- acvWGEN, 5
- adddate, 6
- addsuffices, 7, 29
- arch.test, 8
- arch\_test, 7
- barplot, 49
- ComprehensivePrecipitationGenerator, 8
- ComprehensiveTemperatureGenerator, 11, 25, 62, 64
- continuity\_ratio, 14
- countNAs, 16
- cov, 17
- covariance, 16
- date.mdy, 23
- diff, 57
- ecdf, 17, 40, 42, 43, 46
- ELEVATION (trentino), 68
- ElevationOf, 18
- extractdays, 18, 20, 23, 60
- extractmonths, 19, 49, 54, 56
- extractTnFromAnomalies, 20
- extractTxFromAnomalies, 21
- extractyears, 22, 23, 27, 28
- findDate, 6, 22
- forecastEV, 23, 24, 38
- forecastResidual, 24, 24, 38
- generateTemperatureTimeseries, 13, 14, 20, 21, 25
- getDailyMean, 26
- getMonthlyMean, 27, 60, 62
- getVARmodel, 7, 9, 10, 12, 13, 23–25, 28, 32, 38, 39, 59, 70
- GPCA, 29, 30, 30, 31–36, 52
- GPCA-class, 31
- GPCA\_iteration, 30, 32, 33, 33, 35, 36, 52
- GPCAiteration, 35
- GPCAiteration (GPCAiteration-class), 32
- GPCAiteration-class, 32
- GPCAvarest2, 12
- GPCAvarest2 (GPCAvarest2-class), 32
- GPCAvarest2-class, 32
- gray, 49
- hist, 49
- inv\_GPCA, 30, 33, 34, 35, 36, 39
- inv\_GPCA\_iteration, 30, 33–35, 35, 36
- is.monthly.climate, 9, 36, 62, 63
- julian, 19, 20, 60



- legend, 48
- LOCATION (trentino), 68
- months, 20, 37, 60
- months\_f, 37
- NewVAReventRealization, 24, 38
- newVARMultieventRealization, 10, 13, 25, 26, 39
- normality.test, 40
- normality\_test, 40
- normalizeGaussian, 17, 40, 42, 44
- normalizeGaussian\_prec, 41, 45, 46
- normalizeGaussian\_severalstations, 9, 10, 12, 17, 25, 26, 29, 30, 32–34, 36, 43, 48, 49, 63
- normalizeGaussian\_severalstations\_prec, 45
- plot, 58
- plot.default, 47–49, 53–57
- plot\_sample, 48
- plotDailyClimate, 47
- points, 52
- PRECIPITATION, 9
- PRECIPITATION (trentino), 68
- PRECIPITATION\_MEASUREMENT\_END\_DAY (trentino), 68
- PRECIPITATION\_MEASUREMENT\_START\_DAY (trentino), 68
- PrecipitationEndDay, 50
- PrecipitationStartDay, 51
- print (print.GPCA), 51
- print.GPCA, 51
- qqplot, 52, 53, 58
- qqplot.lagged, 52, 59
- qqplot\_RMAWGEN\_deltaT (qqplot\_RMAWGEN\_Tx), 57
- qqplot\_RMAWGEN\_prec (qqplot\_RMAWGEN\_Tx), 57
- qqplot\_RMAWGEN\_Tn (qqplot\_RMAWGEN\_Tx), 57
- qqplot\_RMAWGEN\_Tx, 57
- qqplotprecWGEN, 53, 54
- qqplotprecWGEN\_seasonal, 54
- qqplotTnTxWGEN, 55, 56
- qqplotTnTxWGEN\_seasonal, 56
- qqplotWGEN, 53–56, 57
- quantile, 9, 12, 17, 25, 41–43, 46
- rainbow, 48, 49
- removeNAs, 59
- rescaling\_monthly, 60
- residuals (residuals.varest2), 61
- residuals.varest2, 61
- RMAWGEN (RMAWGEN-package), 3
- RMAWGEN-package, 3
- serial.test, 61
- serial\_test, 61
- set.seed, 10, 13
- setComprehensiveTemperatureGeneratorParameters, 12–14, 25, 36, 37, 62
- setOldClass, 31, 32, 69
- spline, 64–66
- splineInterpolateMonthlytoDaily, 64, 65, 66
- splineInterpolateMonthlytoDailyforSeveralYears, 10–12, 14, 62–65, 65
- STATION\_LATLON (trentino), 68
- STATION\_NAMES (trentino), 68
- TEMPERATURE\_MAX, 62
- TEMPERATURE\_MAX (trentino), 68
- TEMPERATURE\_MEASUREMENT\_END\_DAY (trentino), 68
- TEMPERATURE\_MEASUREMENT\_START\_DAY (trentino), 68
- TEMPERATURE\_MIN, 62
- TEMPERATURE\_MIN (trentino), 68
- TemperatureEndDay, 66
- TemperatureStartDay, 67
- time, 23
- trentino, 9, 18, 50, 51, 66, 67, 68, 71
- VAR, 23–25, 29, 38, 59, 69, 70
- VAR\_mod, 70
- varest (varest-class), 69
- varest-class, 69
- varest2, 9, 12
- varest2 (varest2-class), 70
- varest2-class, 70
- VARselect, 29, 59
- WhereIs, 71