Package ‘stUPscales’

July 14, 2020

Type Package

Title Spatio-Temporal Uncertainty Propagation Across Multiple Scales

Version 1.0.5.1

Date 2020-07-10

Author J.A. Torres-Matallana [aut, cre]
U. Leopold [ctb]
G.B.M. Heuvelink [ctb]

Maintainer J.A. Torres-Matallana <arturo.torres@list.lu>

Description Integrated environmental modelling requires coupling sub-models at different spatial and temporal scales, thus accounting for change of support procedures (aggregation and disaggregation). We contribute to state-of-the-art open source tools that support uncertainty propagation analysis in temporal and spatio-temporal domains. We implement the tool for uncertainty propagation in environmental modelling, with examples in the urban water domain.
The main functionalities of the class setup and the methods and functions MC.setup, MC.sim, MC.analysis, MC.analysis_generic and Agg.t are contained, which are used for setting up, running and analysing Monte Carlo uncertainty propagation simulations, and for spatio-temporal aggregation. We also implement functionalities to model and predict variables that vary in space and time. stUPscales takes uncertainty characterisation and propagation a step further by including temporal and spatio-temporal auto- and cross-correlation, resulting in more realistic (spatio-)temporal series of environmental variables. Due to its modularity, the package allows the implementation of additional methods and functions for spatio-temporal disaggregation of model inputs and outputs, when linking models across multiple space-time scales.

License GPL (>=3)

Depends R (>= 2.10),
graphics,
grDevices,
mAr,
methods,
lmom,
stats,
utils,

Imports data.table,
DescTools,
doParallel,
foreach,
gstat,
ggplot2,
hydroGOF,
lattice,
moments,
msm,
parallel,
raster,
rgdal,
sp,
spacetime,
xts,
zoo

NeedsCompilation no

R topics documented:

`stUPscales-package` *Spatio-Temporal Uncertainty Propagation Across Multiple Scales*

Description

Integrated environmental modelling requires coupling sub-models at different spatial and temporal scales, thus accounting for change of support procedures (aggregation and disaggregation). We contribute to state-of-the-art open source tools that support uncertainty propagation analysis in temporal and spatio-temporal domains. We implement the tool for uncertainty propagation in environmental modelling, with examples in the urban water domain. The main functionalities of the class setup and the methods and functions MC.setup, MC.sim, MC.analysis, MC.analysis_generic and Agg.t are contained, which are used for setting up, running and analysing Monte Carlo uncertainty propagation simulations, and for spatio-temporal aggregation. We also implement functionalities to model and predict variables that vary in space and time. stUPscales takes uncertainty characterisation and propagation a step further by including temporal and spatio- temporal auto- and cross-correlation, resulting in more realistic (spatio-)temporal series of environmental variables. Due to its modularity, the package allows the implementation of additional methods and functions for spatio-temporal disaggregation of model inputs and outputs, when linking models across multiple space-time scales.

Details

The DESCRIPTION file:

Package:	stUPscales
Type:	Package
Version:	1.1.0.0
Date:	2019-05-30
License:	GPL (>= 3)
Depends:	R (>= 2.10), methods, stats, graphics, grDevices, utils, mAr, lmom
Imports:	parallel, doParallel, foreach, lattice, msm, ggplot2, moments, hydroGOF, zoo, data.table, xts, EmiStatR, rgdal, spacetime
Suggests:	sp

Author(s)

J.A. Torres-Matallana [aut, cre]; U. Leopold [ctb]; G.B.M. Heuvelink [ctb].

Maintainer: J.A. Torres-Matallana.

References

J.A. Torres-Matallana, U. Leopold, and G.B.M. Heuvelink (2018). “stUPscales: An R-Package for Spatio-Temporal Uncertainty Propagation across Multiple Scales with Examples in Urban Water Modelling”. Water, vol. 10, no. 7, p. 837. <doi:10.3390/w10070837>

`Agg.t` *Temporal aggregation of environmental variables*

Description

Function for temporal aggregation of environmental variables. `Agg` is a wrapper function of `aggregate` from `stats` package.

Usage

```
Agg.t(data, nameData, delta, func, namePlot)
```

Arguments

`data`	A `data.frame` that contains the time series of the environmental variable to be aggregated, e.g. precipitation. This data.frame should have at two columns: the first one, Time [y-m-d h:m:s]; the second one, a numeric value equal to the magnitude of the environmental variable. If the environmental variable is different than precipitation, then the column name of the values can be named as the name of the variable itself.
`nameData`	A `character` string that defines the name of the environmental variable to be aggregated.
`delta`	A `numeric` value that specifies the level of aggregation required in minutes.
`func`	The name of the function of aggregation e.g. `mean`, `sum`.
`namePlot`	A `character` string that defines the title of the plot generated.

Value

A `data.frame` with two columns:

`time`	the date-time time series of the aggregated variable
`value`	time series with the magnitude of the aggregated variable.

Author(s)

J.A. Torres-Matallana

Examples

```
## temporal aggregation
library(EmiStatR)
data(P1)
colnames(P1) <- c("time", "P1")
head(P1)

library(stUPscales)
P1.agg <- Agg.t(data = P1, nameData = "P1", delta = 120 , func = sum,
             namePlot = "Temporal aggregation of precipitation P1")

head(P1.agg)
tail(P1.agg)
```

`Correct.radar` *Correction of precipitation radar imagery from the German Weather Service (DWD)*

Description

Given a set of calibrated radar imagery for precipitation (RW product, one hour temporal resolution) in format .tif, this function corrects precipitation radar imagery non-calibrated (RY product, five minutes temporal resolution) in format .tif for the same spatial extend. This function aggregates temporally the non-calibrated data to 10 minutes and makes a simple temporal linear disaggregation of the calibrated radar data to match the required temporal resolution of the non-calibrated data (10 minutes in this case). It is specifically designed for the radar imagery from the German Weather Service (DWD).

Usage

```
Correct.radar(path.calibrated, path.non.calibrated, time.ini.calib,
                                 format = "%y%m%d%H%M", n, NAcutoff = 250)
```

Arguments

`path.calibrated`
: A `character`, which defines the path to the folder which contains the calibrated imagery (RW product, one hour temporal resolution) in .tif format.

`path.non.calibrated`
: A `character`, path to the folder which contains the non-calibrated imagery (RY product, five minutes temporal resolution) in .tif format.

`time.ini.calib`
: A `character`, initial time for calibration e.g. "1112160055".

`format`
: A `character`, initial time format e.g. "%y%m%d%H%M".

`n`
: A `numeric`, number of images to process including time.ini.calib.

`NAcutoff`
: A `numeric`, cutoff value for NA assignation in individual image (file).

Value

A STFDF which contains the corrected data.

Author(s)

J.A. Torres-Matallana

`Germany_precipitation_201112`

Sample precipitation time series in Germany

Description

A 1-minute sample event for precipitation time series measured in 37 rain gauge stations distributed over the territory of Germany close to the frontier to the Grand-Duchy of Luxembourg.

Usage

```
data("Germany_precipitation_201112")
```

Format

The format is:
An 'xts' object on 2011-12-01/2011-12-31 23:59:00 containing:
Data: num [1:44640, 1:37] NA NA NA NA NA NA NA NA NA NA ...
- attr(*, "dimnames")=List of 2
..$: NULL
..$: chr [1:37] "Station.1327" "Station.1964" "Station.200" "Station.2170" ...
Indexed by objects of class: [POSIXct,POSIXt] TZ:
xts Attributes:
NULL

Source

https://www.dwd.de/

Examples

```
library(stUPscales)
library(spacetime)

data(Germany_precipitation_201112)

summary(Germany_precipitation_201112)

library(zoo)
par(mfrow = c(5, 1))
```

```
for(i in 1:ncol(Germany_precipitation_201112)){
  plot(index(Germany_precipitation_201112), Germany_precipitation_201112[,i],
       typ="l", col="blue", xlab="Time", ylab='Precip. [mm]',
       main=colnames(Germany_precipitation_201112[,i]))
}
```

Germany_stations *A SpatialPointsDataFrame with the location of 37 rain gauges in Germany*

Description

A SpatialPointsDataFrame with the location of 37 rain gauges distributed over the territory of Germany close to frontier with the Grand-Duchy of Luxembourg. These 37 stations are the same related to the "Germany_precipitation_201112" dataset.

Usage

```
data("Germany_stations")
```

Format

The format is:
Formal class 'SpatialPointsDataFrame' [package "sp"] with 5 slots
..@ data :'data.frame': 37 obs. of 9 variables:
.. ..$ Stations_id : int [1:37] 200 450 460 523 603 723 902 942 953 1327 ...
.. ..$ von_datum : int [1:37] 20020924 20050920 19930930 20020807 20071024 20020717 20060618 20020925 19970730 20040707 ...
.. ..$ bis_datum : int [1:37] 20180820 20121204 20180820 20180604 20180820 20180820 20180820 20180820 20180820 20180820 ...
.. ..$ Stationshoehe: int [1:37] 517 120 363 359 159 290 573 308 481 147 ...
.. ..$ geoBreite : num [1:37] 50.1 49.9 49.3 50 50.7 ...
.. ..$ geoLaenge : num [1:37] 6.32 7.07 6.69 6.53 7.19 ...
.. ..$ Stationsname : Factor w/ 1109 levels "Aachen","Aachen-Orsbach",..: 535 90 92 104 118 456 172 179 180 1046 ...
.. ..$ Bundesland : Factor w/ 16 levels "Baden-Wuerttemberg",..: 11 11 12 11 10 11 10 11 11 10 ...
.. ..$ d : logi [1:37] NA NA NA NA NA NA ...
..@ coords.nrs : num(0)
..@ coords : num [1:37, 1:2] 90590 144511 117752 106135 152358 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$: NULL
.. .. ..$: chr [1:2] "coords.x1" "coords.x2"
..@ bbox : num [1:2, 1:2] 82780 31291 156554 200669
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$: chr [1:2] "coords.x1" "coords.x2"
.. .. ..$: chr [1:2] "min" "max"
..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot

.. .. ..@ projargs: chr "+init=epsg:2169 +proj=tmerc +lat_0=49.8333333333334 +lon_0=6.166666666666667 +k=1 +x_0=80000 +y_0=100000 +ellp"| __truncated__

Source

https://www.dwd.de/

Examples

```
library(stUPscales)
library(sp)

data(Germany_stations)

str(Germany_stations)

data(Lux_boundary)
plot(Germany_stations)
plot(boundary.Lux, add=TRUE) # Luxembourg boundary
```

Goe_catchment *A SpatialPolygonsDataFrame for the boundaries of the catchment Goesdorf (Goe)*

Description

A SpatialPolygonsDataFrame for the boundaries of the catchment Goesdorf (Goe), located in the catchment Haute-Sûre in the North-West of the Grand-Duchy of Luxembourg.

Usage

```
data("Goe_catchment")
```

Format

The format is:
Formal class 'SpatialPolygonsDataFrame' [package "sp"] with 5 slots
..@ data :'data.frame': 10 obs. of 3 variables:
.. ..$ cat : int [1:10] 2 1 3 4 6 7 5 8 9 10
.. ..$ value: int [1:10] 14 10 16 12 18 8 6 4 8 2
.. ..$ label: Factor w/ 0 levels: NA NA NA NA NA NA NA NA NA NA
..@ polygons :List of 10
.. ..$:Formal class 'Polygons' [package "sp"] with 5 slots
.. ..@ Polygons :List of 1
.. ..$:Formal class 'Polygon' [package "sp"] with 5 slots
.. ..@ labpt : num [1:2] 64972 110572
.. ..@ area : num 334500

```
.. .. .. .. .. .. ..@ hole : logi FALSE
.. .. .. .. .. .. ..@ ringDir: int 1
.. .. .. .. .. .. ..@ coords : num [1:332, 1:2] 64570 64645 64645 64650 64650 ...
.. .. .. ..@ plotOrder: int 1
.. .. .. ..@ labpt : num [1:2] 64972 110572
.. .. .. ..@ ID : chr "0"
.. .. .. ..@ area : num 334500
.. ..$ :Formal class 'Polygons' [package "sp"] with 5 slots
.. .. .. ..@ Polygons :List of 1
.. .. .. .. ..$ :Formal class 'Polygon' [package "sp"] with 5 slots
.. .. .. .. .. .. ..@ labpt : num [1:2] 63815 110371
.. .. .. .. .. .. ..@ area : num 769525
.. .. .. .. .. .. ..@ hole : logi FALSE
.. .. .. .. .. .. ..@ ringDir: int 1
.. .. .. .. .. .. ..@ coords : num [1:453, 1:2] 64570 64570 64575 64575 64580 ...
.. .. .. ..@ plotOrder: int 1
.. .. .. ..@ labpt : num [1:2] 63815 110371
.. .. .. ..@ ID : chr "1"
.. .. .. ..@ area : num 769525
.. ..$ :Formal class 'Polygons' [package "sp"] with 5 slots
.. .. .. ..@ Polygons :List of 1
.. .. .. .. ..$ :Formal class 'Polygon' [package "sp"] with 5 slots
.. .. .. .. .. .. ..@ labpt : num [1:2] 65293 110235
.. .. .. .. .. .. ..@ area : num 378100
.. .. .. .. .. .. ..@ hole : logi FALSE
.. .. .. .. .. .. ..@ ringDir: int 1
.. .. .. .. .. .. ..@ coords : num [1:410, 1:2] 64840 64840 64835 64835 64840 ...
.. .. .. ..@ plotOrder: int 1
.. .. .. ..@ labpt : num [1:2] 65293 110235
.. .. .. ..@ ID : chr "2"
.. .. .. ..@ area : num 378100
.. ..$ :Formal class 'Polygons' [package "sp"] with 5 slots
.. .. .. ..@ Polygons :List of 1
.. .. .. .. ..$ :Formal class 'Polygon' [package "sp"] with 5 slots
.. .. .. .. .. .. ..@ labpt : num [1:2] 64435 109996
.. .. .. .. .. .. ..@ area : num 516350
.. .. .. .. .. .. ..@ hole : logi FALSE
.. .. .. .. .. .. ..@ ringDir: int 1
.. .. .. .. .. .. ..@ coords : num [1:557, 1:2] 64585 64585 64590 64590 64595 ...
.. .. .. ..@ plotOrder: int 1
.. .. .. ..@ labpt : num [1:2] 64435 109996
.. .. .. ..@ ID : chr "3"
.. .. .. ..@ area : num 516350
.. ..$ :Formal class 'Polygons' [package "sp"] with 5 slots
.. .. .. ..@ Polygons :List of 1
.. .. .. .. ..$ :Formal class 'Polygon' [package "sp"] with 5 slots
.. .. .. .. .. .. ..@ labpt : num [1:2] 64962 109448
.. .. .. .. .. .. ..@ area : num 922000
```

```
.. .. .. .. .. .. ..@ hole : logi FALSE
.. .. .. .. .. .. ..@ ringDir: int 1
.. .. .. .. .. .. ..@ coords : num [1:613, 1:2] 65100 65110 65110 65115 65115 ...
.. .. .. ..@ plotOrder: int 1
.. .. .. ..@ labpt : num [1:2] 64962 109448
.. .. .. ..@ ID : chr "4"
.. .. .. ..@ area : num 922000
.. ..$ :Formal class 'Polygons' [package "sp"] with 5 slots
.. .. .. ..@ Polygons :List of 1
.. .. .. .. ..$ :Formal class 'Polygon' [package "sp"] with 5 slots
.. .. .. .. .. .. ..@ labpt : num [1:2] 63795 109872
.. .. .. .. .. .. ..@ area : num 16075
.. .. .. .. .. .. ..@ hole : logi FALSE
.. .. .. .. .. .. ..@ ringDir: int 1
.. .. .. .. .. .. ..@ coords : num [1:132, 1:2] 63660 63660 63665 63665 63670 ...
.. .. .. ..@ plotOrder: int 1
.. .. .. ..@ labpt : num [1:2] 63795 109872
.. .. .. ..@ ID : chr "5"
.. .. .. ..@ area : num 16075
.. ..$ :Formal class 'Polygons' [package "sp"] with 5 slots
.. .. .. ..@ Polygons :List of 1
.. .. .. .. ..$ :Formal class 'Polygon' [package "sp"] with 5 slots
.. .. .. .. .. .. ..@ labpt : num [1:2] 63309 109732
.. .. .. .. .. .. ..@ area : num 334700
.. .. .. .. .. .. ..@ hole : logi FALSE
.. .. .. .. .. .. ..@ ringDir: int 1
.. .. .. .. .. .. ..@ coords : num [1:224, 1:2] 63115 63125 63125 63130 63130 ...
.. .. .. ..@ plotOrder: int 1
.. .. .. ..@ labpt : num [1:2] 63309 109732
.. .. .. ..@ ID : chr "6"
.. .. .. ..@ area : num 334700
.. ..$ :Formal class 'Polygons' [package "sp"] with 5 slots
.. .. .. ..@ Polygons :List of 1
.. .. .. .. ..$ :Formal class 'Polygon' [package "sp"] with 5 slots
.. .. .. .. .. .. ..@ labpt : num [1:2] 64030 109425
.. .. .. .. .. .. ..@ area : num 589825
.. .. .. .. .. .. ..@ hole : logi FALSE
.. .. .. .. .. .. ..@ ringDir: int 1
.. .. .. .. .. .. ..@ coords : num [1:422, 1:2] 63785 63790 63790 63795 63795 ...
.. .. .. ..@ plotOrder: int 1
.. .. .. ..@ labpt : num [1:2] 64030 109425
.. .. .. ..@ ID : chr "7"
.. .. .. ..@ area : num 589825
.. ..$ :Formal class 'Polygons' [package "sp"] with 5 slots
.. .. .. ..@ Polygons :List of 1
.. .. .. .. ..$ :Formal class 'Polygon' [package "sp"] with 5 slots
.. .. .. .. .. .. ..@ labpt : num [1:2] 63952 109802
.. .. .. .. .. .. ..@ area : num 25
```

.. ..@ hole : logi FALSE
.. ..@ ringDir: int 1
.. ..@ coords : num [1:5, 1:2] 63950 63955 63955 63950 63950 ...
.. ..@ plotOrder: int 1
.. ..@ labpt : num [1:2] 63952 109802
.. ..@ ID : chr "8"
.. ..@ area : num 25
.. ..$:Formal class 'Polygons' [package "sp"] with 5 slots
.. ..@ Polygons :List of 1
.. ..$:Formal class 'Polygon' [package "sp"] with 5 slots
.. ..@ labpt : num [1:2] 64458 108753
.. ..@ area : num 356625
.. ..@ hole : logi FALSE
.. ..@ ringDir: int 1
.. ..@ coords : num [1:211, 1:2] 63925 63935 63935 63950 63950 ...
.. ..@ plotOrder: int 1
.. ..@ labpt : num [1:2] 64458 108753
.. ..@ ID : chr "9"
.. ..@ area : num 356625
..@ plotOrder : int [1:10] 5 2 8 4 3 10 7 1 6 9
..@ bbox : num [1:2, 1:2] 62935 108355 65735 111040
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$: chr [1:2] "x" "y"
.. .. ..$: chr [1:2] "min" "max"
..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot
.. .. ..@ projargs: chr "+proj=tmerc +lat_0=49.83333333333334 +lon_0=6.166666666666667 +k=1 +x_0=80000 +y_0=100000 +ellps=intl +units=m +no_defs"

Examples

```
data(Goe_catchment)
str(Goe_catchment)
plot(Goe_catchment)
```

`GoF` *Wrapper function for the* `gof` *function from* `hydroGOF` *package*

Description

A wrapper function for the `gof` function from `hydroGOF` package

Usage

```
GoF(eval, col_sim, col_obs, name)
```

Arguments

- `eval` — A `matrix` or `data.frame` with n observations of at least two variables: simulations and observations.
- `col_sim` — A numeric value defining the column in `eval` data.frame that contains the simulated vector time series.
- `col_obs` — A numeric value defining the column in `eval` data.frame that contains the observed vector time series.
- `name` — A character string that defines the name of the files (.csv and .RData) created with the results. If missing then no files are created.

Value

A vector with 20 elements for each one of the following measures of godness-of-fit: 1) ME, mean error; 2) MAE, mean absolute error; 3) MSE, mean squared error; 4) RMSE, root mean square error; 5) NRMSE %, normalized root square error; 6) PBIAS %, percent bias; 7) RSR, Ratio of RMSE to the standard deviation of the observations; 8) rSD, Ratio of Standard Deviations; 9) NSE, Nash-Sutcliffe Efficiency; 10) mNSE, modified Nash-Sutcliffe efficiency; 11) rNSE, relative Nash-Sutcliffe efficiency; 12) d, Index of Agreement; 13) md, Modified index of agreement; 14) rd, Relative Index of Agreement; 15) cp,Coefficient of persistence; 16) r, Pearson product-moment correlation coefficient; 17) R2, Coefficient of Determination; 18) bR2, Coefficient of determination (r2) multiplied by the slope of the regression line between sim and obs; 19) KGE,Kling-Gupta Efficiency; 20) VE, Volumetric Efficiency.

Author(s)

J.A. Torres-Matallana

References

Mauricio Zambrano-Bigiarini, 2014. hydroGOF: Goodness-of-fit functions for comparison of simulated and observed hydrological time series. R package version 0.3-8.
https://CRAN.R-project.org/package=hydroGOF.

Examples

```
library(stUPscales)

data_new <- rnorm(230, .25, .1)
data_new <- cbind(data_new, data_new*1.2)
colnames(data_new) <- c("sim", "obs")
head(data_new)

gof.new <- GoF(data_new, 1, 2)
gof.new

# writing files
gof.new <- GoF(data_new, 1, 2, "GoF_results")
```

HS_RW20111216_stfdf *1-hour DWD precipitation radar imagery calibrated in STFDF format*

Description

Calibrated hourly precipitation radar imagery from the German Weather Service (DWD from the initials in German) at one-kilometer spatial resolution over the Haute-Sure catchment of the Grand-Duchy of Luxembourg. The data was recorded at Neuheilenbach radar station located in territory of Germany, which covers the entire territory of the Grand-Duchy of Luxembourg and surroundings. This sample STFDF (spatio-temporal full data.frame) corresponds to 1-day sample event for precipitation recorded on 16th December 2011.

Usage

```
data("HS_RW20111216_stfdf")
```

Format

The format is: Formal class 'STFDF' [package "spacetime"] with 4 slots
..@ data :'data.frame': 52900 obs. of 1 variable:
.. ..$ raa01.rw_10000.1112160050.dwd...bin: num [1:52900] 1 1 1 1 1 1 1 1 1 1 ...
..@ sp :Formal class 'SpatialPixels' [package "sp"] with 5 slots
.. .. ..@ grid :Formal class 'GridTopology' [package "sp"] with 3 slots
.. ..@ cellcentre.offset: Named num [1:2] -327 -4347
.. ..- attr(*, "names")= chr [1:2] "s1" "s2"
.. ..@ cellsize : num [1:2] 1 1
.. ..@ cells.dim : int [1:2] 46 46
.. .. ..@ grid.index : int [1:2116] 1 2 3 4 5 6 7 8 9 10 ...
.. .. ..@ coords : num [1:2116, 1:2] -327 -326 -325 -324 -323 ...
.. ..- attr(*, "dimnames")=List of 2
.. ..$: NULL
.. ..$: chr [1:2] "s1" "s2"
.. .. ..@ bbox : num [1:2, 1:2] -327 -4348 -281 -4302
.. ..- attr(*, "dimnames")=List of 2
.. ..$: chr [1:2] "s1" "s2"
.. ..$: chr [1:2] "min" "max"
.. .. ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot
.. ..@ projargs: chr "+proj=stere +lat_0=90 +lat_ts=90 +lon_0=10 +k=0.93301270189 +x_0=0 +y_0=0 +a=6370040 +b=6370040 +units=km +no_defs"
..@ time :An 'xts' object on 2011-12-16 00:50:00/2011-12-17 00:50:00 containing:
Data: int [1:25, 1] 1 2 3 4 5 6 7 8 9 10 ...
- attr(*, "dimnames")=List of 2
..$: NULL
..$: chr "timeIndex"
Indexed by objects of class: [POSIXct,POSIXt] TZ:
xts Attributes:
NULL

..@ endTime: POSIXct[1:25], format: "2011-12-16 00:50:00" "2011-12-16 01:50:00" "2011-12-16 02:50:00" "2011-12-16 03:50:00" ...

Source

https://www.dwd.de/

Examples

```
library(stUPscales)
data(HS_RW20111216_stfdf)

library(spacetime)
stplot(HS_RW20111216_stfdf,
       scales=list(draw=TRUE),
       key.space="right", colorkey=TRUE,
       main="1-hour DWD sample precipitation radar imagery calibrated in STFDF format",
       cex=.74, par.strip.text=list(cex=.74))
```

HS_RY20111216_stfdf *5-minute DWD precipitation radar imagery non-calibrated in STFDF format*

Description

Non-calibrated 5-minute precipitation radar imagery from the German Weather Service (DWD from the initials in German) at one-kilometer spatial resolution over the Haute-Sure catchment of the Grand-Duchy of Luxembourg. The data was recorded at Neuheilenbach radar station located in territory of Germany, which covers the entire territory of the Grand-Duchy of Luxembourg and surroundings. This sample STFDF (spatio-temporal full data.frame) corresponds to 1-day sample event for precipitation recorded on 16th December 2011.

Please note that these are un-calibrated radar data.

Usage

```
data("HS_RY20111216_stfdf")
```

Format

The format is:
Formal class 'STFDF' [package "spacetime"] with 4 slots
..@ data :'data.frame': 609408 obs. of 1 variable:
.. ..$ raa01.ry_10000.1112160000.dwd...bin: num [1:609408] 1 2 1 1 1 1 1 1 1 1 ...
..@ sp :Formal class 'SpatialPixels' [package "sp"] with 5 slots
.. .. ..@ grid :Formal class 'GridTopology' [package "sp"] with 3 slots
.. ..@ cellcentre.offset: Named num [1:2] -327 -4347
.. ..- attr(*, "names")= chr [1:2] "s1" "s2"

.. ..@ cellsize : num [1:2] 1 1
.. ..@ cells.dim : int [1:2] 46 46
.. .. ..@ grid.index : int [1:2116] 1 2 3 4 5 6 7 8 9 10 ...
.. .. ..@ coords : num [1:2116, 1:2] -327 -326 -325 -324 -323 ...
.. ..- attr(*, "dimnames")=List of 2
.. ..$: NULL
.. ..$: chr [1:2] "s1" "s2"
.. .. ..@ bbox : num [1:2, 1:2] -327 -4348 -281 -4302
.. ..- attr(*, "dimnames")=List of 2
.. ..$: chr [1:2] "s1" "s2"
.. ..$: chr [1:2] "min" "max"
.. .. ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot
.. ..@ projargs: chr "+proj=stere +lat_0=90 +lat_ts=90 +lon_0=10 +k=0.93301270189 +x_0=0 +y_0=0 +a=6370040 +b=6370040 +units=km +no_defs"
..@ time :An 'xts' object on 2011-12-16/2011-12-16 23:55:00 containing:
Data: int [1:288, 1] 1 2 3 4 5 6 7 8 9 10 ...
- attr(*, "dimnames")=List of 2
..$: NULL
..$: chr "timeIndex"
Indexed by objects of class: [POSIXct,POSIXt] TZ:
xts Attributes:
NULL
..@ endTime: POSIXct[1:288], format: "2011-12-16 00:00:00" "2011-12-16 00:05:00" "2011-12-16 00:10:00" "2011-12-16 00:15:00" ...

Source

https://www.dwd.de/

Examples

```
library(stUPscales)
library(spacetime)

data(HS_RY20111216_stfdf)

sample.idx <- seq.default(from = 1, to = 25, by = 1)
sample     <- HS_RY20111216_stfdf[, sample.idx]

stplot(sample,
       scales=list(draw=TRUE),
       key.space="right", colorkey=TRUE,
     main="5-minute DWD sample precipitation radar imagery non-calibrated in STFDF format",
       cex=.74, par.strip.text=list(cex=.74))
```

inputObs-class *Class* "inputObs"

Description

The class provides a container for inputs required to invoke Validation_Quantity and Validation_Quantity_Agg methods.

Objects from the Class

Objects can be created by calls of the form inputObs(...).

Slots

id: Object of class "numeric" to define an unique index for the object.

plot: Object of class "numeric". One of 0 (no plots are cretated) or 1 (to create plots).

delta: Object of class "list" to define the time step in minutes for temporal aggregation required e.g. list(P1 = 10, wlt_obs = 10, vol_sim = 10) for defining the time steps of 10 minutes for the three variables P1, wlt_obs, vol_sim.

observations: Object of class "list" to define the observed time series.

lev2vol: Object of class "list" to define the curve for the relationship level to volume.

namePlot: Object of class "character" to define the name of the plot to create.

legendPosition: Object of class "list" with three character objects, which define the posistion of the legend for the top, second and bottom insets of the plot.

var: Object of class "character" to define the name of the variable from which the time series simulated are defined.

Methods

Validation_Quantity_Agg signature(x = "input",y = "inputObs"): ...

Validation_Quantity signature(x = "input",y = "inputObs"): ...

Author(s)

J.A. Torres-Matallana

Examples

```
showClass("inputObs")

inputObs()
```

IsReg.files — *Check regulatiry of time series in a folder*

Description

This function checks the regulatiry of the time series which are contained in files in a specific folder.

Usage

```
IsReg.files(path, sep = "-", pattern = ".tif", format = "%y%m%d%H%M")
```

Arguments

`path`	`character`, path to the folder which contains the files with the imagery.
`sep`	`character`, separation character for spliting the file names (e.g. "-").
`pattern`	`character`, extension of the files to check (e.g. .tif).
`format`	`character`, time format in the file name (e.g. "%y%m%d%H%M").

Value

Object of class `"list"`. This object contains 2 elements, the first one contains a character string "_TSregular" if the xts object created is strict regular, or "_TSirregular" if it is strict irregular. More details can be found in the "is.regular" function of the "zoo" package.

Warning

This function was specifically created for file names following the naming nomenclature of the German Weather Service (DWD) for radar imagery e.g. "raa01-rw_10000-1112160050-dwd—bin.tif" for RW products or "raa01-ry_10000-1112160000-dwd—bin.tif" for RY products.

Author(s)

J.A. Torres-Matallana

See Also

See also `IsReg.ts`.

IsReg.ts *Wrapper function for function* `is.regular` *from* `zoo` *package for* `data.frame` *objects*

Description

"IsReg.ts" is a wrapping Function for Function "is.regular" from "zoo" package. Given a time series (ts) as a "data.frame" object, it is converted into a "xts" object, while the regularity of the object is checked. The first column of the "data.frame" should contain a character string vector to be converted via as.POSIXct accordingly with the date format (format) and time zone (tz).

Usage

```
IsReg.ts(data, format, tz)
```

Arguments

`data`	an object of class `data.frame` containing in its first column a character string vector to be converted via as.POSIXct into a date vector accordingly with the date format (format) and time zone (tz) defined
`format`	character string giving a date-time format as used by `strptime`.
`tz`	a time zone specification to be used for the conversion, if one is required. System-specific, but "" is the current time zone, and "GMT" is UTC (Universal Time, Coordinated). Invalid values are most commonly treated as UTC, on some platforms with a warning.

Details

"IsReg" calls the as.POSIXct function from `base` package to convert an object to one of the two classes used to represent date/times (calendar dates plus time to the nearest second). More details can be found in the "is.regular" function of the "zoo" package.

Value

Object of class `"list"`. This object contains 2 elements, the first one contains a character string "_TSregular" if the xts object created is strict regular, or "_TSirregular" if it is strict irregular. More details can be found in the "is.regular" function of the "zoo" package.

Author(s)

J.A. Torres-Matallana

Examples

```
library(EmiStatR)
data("P1")

class(P1)
head(P1)

ts <- IsReg.ts(data = P1, format = "%Y-%m-%d %H:%M:%S", tz = "UTC")
str(ts)

ts[[1]]

head(ts[[2]]); tail(ts[[2]])

plot(ts[[2]], ylab = "Precipitation [mm]")
```

`Lux_boundary` *A SpatialPolygonsDataFrame for the boundary of the Grand-Duchy of Luxembourg*

Description

A SpatialPolygonsDataFrame for the country boundary of the Grand-Duchy of Luxembourg

Usage

```
data("Lux_boundary")
```

Format

The format is:
Formal class 'SpatialPolygonsDataFrame' [package "sp"] with 5 slots
..@ data :'data.frame': 1 obs. of 3 variables:
.. ..$ cat: int 1
.. ..$ X_ : Factor w/ 1 level "?": 1
.. ..$ X_1: Factor w/ 1 level "?": 1
..@ polygons :List of 1
.. ..$:Formal class 'Polygons' [package "sp"] with 5 slots
.. ..@ Polygons :List of 1
.. ..$:Formal class 'Polygon' [package "sp"] with 5 slots
.. ..@ labpt : num [1:2] 74692 93669
.. ..@ area : num 2.6e+09
.. ..@ hole : logi FALSE
.. ..@ ringDir: int 1
.. ..@ coords : num [1:1141, 1:2] 82274 82437 82449 82765 82782 ...
.. ..@ plotOrder: int 1
.. ..@ labpt : num [1:2] 74692 93669
.. ..@ ID : chr "0"

.. ..@ area : num 2.6e+09
..@ plotOrder : int 1
..@ bbox : num [1:2, 1:2] 49034 57132 106245 138879
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$: chr [1:2] "x" "y"
.. .. ..$: chr [1:2] "min" "max"
..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot
.. .. ..@ projargs: chr "+proj=tmerc +lat_0=49.83333333333334 +lon_0=6.166666666666667 +k=1 +x_0=80000 +y_0%%=100000 +ellps=intl +units=m +no_defs"

Examples

```
library(stUPscales)

data(Lux_boundary)

str(boundary.Lux)
```

Lux_precipitation *Sample precipitation time series in the Grand-Duchy of Luxembourg*

Description

A 10-hour sample event for precipitation time series measured in 25 rain gauge stations distributed over the territory of the Grand-Duchy of Luxembourg.

Usage

```
data("Lux_precipitation")
```

Format

The format is:
An 'xts' object on 2011-12-16/2011-12-16 10:00:00 containing:
Data: num [1:61, 1:25] 0 0 0 0 0.1 0.1 0.1 0.1 0.1 0.2 ...
- attr(*, "dimnames")=List of 2
..$: NULL
..$: chr [1:25] "Dahl" "Echternach" "Esch-Sure" "Eschdorf" ...
Indexed by objects of class: [POSIXct,POSIXt] TZ:
xts Attributes:
NULL

Source

http://agrimeteo.lu/

Examples

```
data(Lux_precipitation)

library(xts)
head(event.subset.xts)
tail(event.subset.xts)

plot(event.subset.xts)
```

`Lux_precipitation_2010_2011`

Sample precipitation time series in the Grand-Duchy of Luxembourg (2-year period)

Description

A 2-year period sample for precipitation time series measured at 10-minute time step in 25 rain gauge stations distributed over the territory of the Grand-Duchy of Luxembourg.

Usage

```
data("Lux_precipitation_2010_2011")
```

Format

The format is:
An 'xts' object on 2010-01-01/2011-12-31 23:50:00 containing:
Data: num [1:105120, 1:25] 0 0 0 0 0 0 0 0 0 0 ...
- attr(*, "dimnames")=List of 2
..$: NULL
..$: chr [1:25] "Dahl" "Echternach" "Esch-Sure" "Eschdorf" ...
Indexed by objects of class: [POSIXct,POSIXt] TZ:
xts Attributes:
NULL

Source

http://agrimeteo.lu/

Examples

```
library(stUPscales)
data(Lux_precipitation_2010_2011)

library(xts)
head(Lux_precipitation_2010_2011)
tail(Lux_precipitation_2010_2011)
```

```
plot(Lux_precipitation_2010_2011)
```

Lux_stations *A SpatialPointsDataFrame with the location of 25 rain gauges in Luxembourg*

Description

A SpatialPointsDataFrame with the location of 25 rain gauges distributed over the territory of the Grand-Duchy of Luxembourg. These 25 stations are the same related to the "event.subset.xts" dataset.

Usage

```
data("Lux_stations")
```

Format

The format is:
Formal class 'SpatialPointsDataFrame' [package "sp"] with 5 slots
..@ data :'data.frame': 25 obs. of 8 variables:
.. ..$ id : Factor w/ 25 levels "1","11","12",..: 4 5 6 7 8 9 10 11 25 1 ...
.. ..$ name : Factor w/ 25 levels "Arsdorf","Christnach",..: 4 5 7 6 8 9 10 11 22 1 ...
.. ..$ north_lure: Factor w/ 24 levels "101950","102913",..: 6 NA 4 3 1 5 22 20 21 2 ...
.. ..$ east_luref: Factor w/ 25 levels "56584","56990",..: 10 25 5 7 16 19 20 24 17 1 ...
.. ..$ elev_luref: Factor w/ 25 levels "190","202","207",..: 22 5 16 25 2 11 14 1 6 20 ...
.. ..$ station_ty: Factor w/ 1 level "1": 1 1 1 1 1 1 1 1 1 1 ...
.. ..$ management: Factor w/ 1 level "ASTA": 1 1 1 1 1 1 1 1 1 1 ...
.. ..$ telemetry : Factor w/ 0 levels: NA NA NA NA NA NA NA NA NA NA ...
..@ coords.nrs : num(0)
..@ coords : num [1:25, 1:2] 66562 99810 62258 63363 74929 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$: NULL
.. .. ..$: chr [1:2] "coords.x1" "coords.x2"
..@ bbox : num [1:2, 1:2] 56584 64215 99810 132012
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$: chr [1:2] "coords.x1" "coords.x2"
.. .. ..$: chr [1:2] "min" "max"
..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot
.. .. ..@ projargs: chr "+proj=tmerc +lat_0=49.8333333333334 +lon_0=6.16666666666667 +k=1 +x_0=80000 +y_0=100000 +ellps=intl +units=m +no_defs"

Source

http://agrimeteo.lu/

Examples

```
library(stUPscales)
data(Lux_stations)

str(stations)

library(sp)
plot(stations)
```

MC.analysis *Analysis of the Monte Carlo simulation*

Description

Function for running the analysis of the Monte Carlo simulation.

Usage

```
MC.analysis(x, delta, qUpper, p1.det, sim.det, event.ini, event.end,
ntick, summ.data = NULL)
```

Arguments

`x`	A `list`.
`delta`	A `numeric` value that specifies the level of aggregation required in minutes.
`qUpper`	A `character` string that defines the upper percentile to plot the confidence band of results, several options are possible `"q999"` the 99.9th percentile, `"q995"` the 99.5th percentile, `"q99"` the 99th percentile, `"q95"` the 95th percentile, `"q50"` the 50th percentile. The lower boundary of the confidence band (showed in gray in the output plots) is the 5th percentile in all cases.
`p1.det`	A `data.frame` that contains the time series of the main driving force of the system to be simulated deterministically, e.g. precipitation. This data.frame should have only two columns: the first one, Time [y-m-d h:m:s]; the second one, a numeric value equal to the magnitude of the environmental variable.
`sim.det`	A `list` that contains the results of the deterministic simulation, here the output of EmiStatR given `p1.det`. See the method `EmiStatR` from the homonym package for details.
`event.ini`	A time-date string in `POSIXct` format that defines the initial time for event analysis.
`event.end`	A time-date string in `POSIXct` format that defines the final time for event analysis.
`ntick`	A `numeric` value to specify the number of ticks in the x-axis for the event time-window plots.
`summ.data`	A `list` by default NULL. If provided, the list should contain an output of the MC.analysis function, and the analysis is done again without the calculation of some of the internal variables, therefore the analysis is faster.

Value

A `list` of length 2:

summm
: A `list` that contains the summary statistics of the Monte Carlo simulation per output variable. Each output variable is summarised by calculating the mean "Mean", standard deviation "sd", variance "Variance", 5th, 25th, 50th, 75th, 95th, 99.5th, 99.9th percentiles "q05","q25","q50","q75","q95","q995", "q999", the max "Max", the sum "Sum", time "time", and the deterministic precipitation "p1", all variables as time series.

variance
: A `data.frame` that contains the summary statistics of the variance of the Monte Carlo simulation per output variable.

Author(s)

J.A. Torres-Matallana

See Also

See also `setup-class`, `MC.setup-methods`, `MC.sim-methods`.

Examples

```
## the Monte Carlo simulation: MC.sim
library(EmiStatR)

# library(xts)
# data(Esch_Sure2010)
# P <- IsReg(Esch_Sure2010, format = "%Y-%m-%d %H:%M:%S", tz = "CET")
# P1 <- P[[2]]
# P1 <- P1["2010-08",][1:55]
# P1 <- cbind.data.frame(time=index(P1), P1 = coredata(P1))

data(P1)
P1 <- P1[165:(110*2),]
plot(P1[,2], typ="l")

library(stUPscales)

setting_EmiStatR <-  setup(id       = "MC_sim1",
                           nsim     = 3, # # use a larger number to have
                                             # a proper confidence band of simulatios
                           seed     = 123,
                           mcCores  = 1,
                           ts.input = P1,
                           rng      = rng <- list(
                             qs   = 150,    # [l/PE/d]
                         CODs = c(pdf = "nor", mu = 4.378, sigma = 0.751),    # log[g/PE/d]
                         NH4s = c(pdf = "nor", mu = 1.473, sigma = 0.410),    # log[g/PE/d]
                             qf   = 0.04,     # [l/s/ha]
                             CODf = 0,              # [g/PE/d]
                             NH4f = 0,              # [g/PE/d]
```

```
                              CODr = c(pdf = "nor", mu = 3.60, sigma = 1.45),    # 71 log[mg/l]
                                NH4r = 1,            # [mg/l]
                                nameCSO = "E1",      # [-]
                                id      = 1,         # [-]
                                ns      = "FBH Goesdorf",  # [-]
                                nm      = "Goesdorf",  # [-]
                                nc      = "Obersauer", # [-]
                                numc    = 1,         # [-]
                                use     = "R/I",  # [-]
                                Atotal  = 36,            # [ha]
                                Aimp    = c(pdf = "uni", min = 4.5, max = 25),       # [ha]
                                Cimp    = c(pdf = "uni", min = 0.25, max = 0.95),  # [-]
                                Cper    = c(pdf = "uni", min = 0.05, max = 0.60),  # [-]
                                tfS     = 1,             # [time steps]
                                pe      = 650,           # [PE]
                                Qd      = 5,             # [l/s]
                                Dd      = 0.150,         # [m]
                                Cd      = 0.18,          # [-]
                                V       = 190,           #  [m3]
                                lev.ini = 0.10,          # [m]
                                lev2vol = list(lev = c(.06, 1.10, 1.30, 3.30),   # [m]
                                               vol = c(0, 31, 45, 190))          # [m3]
                              ),
                              ar.model  = ar.model <- list(
                                CODs    = 0.5,
                                NH4s    = 0.5,
                                CODr    = 0.7),
                              var.model = var.model <- list(
                                inp     = c("", ""), # c("CODs", "NH4s"), # c("", ""),
                                w       = c(0.04778205, 0.02079010),
                                A       = matrix(c(9.916452e-01, -8.755558e-05,
                                               -0.003189094, 0.994553910), nrow=2, ncol=2),
                                C       = matrix(c(0.009126591, 0.002237936,
                                               0.002237936, 0.001850941), nrow=2, ncol=2)))

MC_setup <- MC.setup(setting_EmiStatR)

sims <- MC.sim(x = MC_setup, EmiStatR.cores = 0)


## Monte Carlo simulation analysis: MC.analysis

# Deterministic simulation
# Definition of structure 1, E1:

E1 <- list(id = 1, ns = "FBH Goesdorf", nm = "Goesdorf", nc = "Obersauer", numc = 1,
           use = "R/I", Atotal = 36, Aimp = 25.2, Cimp = 0.80, Cper = 0.30,
           tfS = 0, pe = 650, Qd = 5,
           Dd = 0.150, Cd = 0.18, V = 190, lev.ini = 0.10,
           lev2vol = list(lev = c(.06, 1.10, 1.30, 3.30),
                          vol = c(0, 31, 45, 190))
           )
```

```
# Defining deterministic input:
library(EmiStatR)
# data(P1)

input.det <- input(spatial = 0, zero = 1e-5,
                    folder = system.file("shiny", package = "EmiStatR"),
                    cores = 0,
                    ww = list(qs = 150, CODs = 104, NH4s = 4.7),
                    inf = list(qf= 0.04, CODf = 0, NH4f = 0),
                    rw = list(CODr = 71, NH4r = 1, stat = "Dahl"),
                    P1 = P1, st = list(E1=E1), export = 0)

# Invoking `EmiStatR` with the deterministic input:
sim.det   <- EmiStatR(input.det)

# further arguments
delta <- 10 # minutes
qUpper <- "q999"

event.ini <- as.POSIXct("2016-01-02 03:20:00")
event.end <- as.POSIXct("2016-01-02 12:30:00")

# uncomment to run:
# new_analysis <- MC.analysis(x = sims, delta = delta, qUpper = qUpper,  p1.det = P1,
#                          sim.det = sim.det, event.ini = event.ini, event.end = event.end,
#                               ntick = 5, summ.data = NULL)
```

`MC.analysis_generic` *Analysis of the Monte Carlo simulation (general function)*

Description

General function for running the analysis of the Monte Carlo simulation.

Usage

```
MC.analysis_generic(x, delta, qUpper, data.det, sim.det, event.ini, event.end,
              ntick, summ.data = NULL)
```

Arguments

`x`	A `list` of 1, which contains the output of the Monte Carlo simulation as a data.frame with n rows as time steps and the first column is time in format POSIXct and m columns named e.g. V1, V2, V3... Vm, where m is the number of Monte Carlo runs results.
`delta`	A `numeric` value that specifies the level of aggregation required in minutes.

qUpper A `character` string that defines the upper percentile to plot the confidence band of results, several options are possible "q999" the 99.9th percentile, "q995" the 99.5th percentile, "q99" the 99th percentile, "q95" the 95th percentile, "q50" the 50th percentile. The lower boundary of the confidence band (showed in gray in the output plots) is the 5th percentile in all cases.

data.det A `data.frame` that contains the time series of the main driving force of the system to be simulated deterministically, e.g. precipitation. This data.frame should have only two columns: the first one, Time [y-m-d h:m:s] in POSIXct format; the second one, a numeric value equal to the magnitude of the variable.

sim.det A `list` of 1 that contains the results of the deterministic simulation, here the output given `data.det`. The format is the same as `data.det`.

event.ini A time-date string in `POSIXct` format that defines the initial time for event analysis.

event.end A time-date string in `POSIXct` format that defines the final time for event analysis.

ntick A `numeric` value to specify the number of ticks in the x-axis for the event time-window plots.

summ.data A `list` by default NULL. If provided, the list should contain an output of the MC.analysis function, and the analysis is done again without the calculation of some of the internal variables, therefore the analysis is faster.

Value

A `list` of length 2:

summ A `list` that contains the summary statistics of the Monte Carlo simulation per output variable. Each output variable is summarised by calculating the mean "Mean", standard deviation "sd", variance "Variance", 5th, 25th, 50th, 75th, 95th, 99.5th, 99.9th percentiles "q05","q25","q50","q75","q95","q995", "q999", the max "Max", the sum "Sum", time "time", and the deterministic data "p1", all variables as time series.

variance A `data.frame` that contains the summary statistics of the variance of the Monte Carlo simulation per output variable.

Author(s)

J.A. Torres-Matallana

See Also

See also `setup-class`, `MC.setup-methods`, `MC.sim-methods`.

Examples

```
## Creating meta-model
Model <- function(A, B, variable.1, variable.2){
  lum <- A*variable.1 + B*variable.2
}
```

```
## Model input and parameter set-up

time <- data.frame(time = seq.POSIXt(from = as.POSIXct("2019-01-01"),
                                     to = as.POSIXct("2019-01-02"), by = 60*60*6))
data <- cbind(time, data = 25)
data

new.setup <- setup(id = "MC_1",
                   nsim = 10,
                   seed = 123,
                   mcCores = 1,
                   ts.input = data,
                   rng = rng <- list(
                     A = 1.25,
                     B = 0.75,
                     variable.1 = c(pdf = "uni", min = 0, max = 4),
                     variable.2 = c(pdf = "uni", min = 2.2, max = 3.2)
                   )
)

str(new.setup)

## Monte Carlo simulation set-up
set.seed(slot(new.setup, "seed"))

new.mc.setup <- MC.setup(new.setup)
str(new.mc.setup)

## Monte Carlo simulation
output <- data.frame(time = new.mc.setup$ts.input[,1])
output[,2:(new.mc.setup$nsim + 1)] <- NA

for(i in 1:new.mc.setup$nsim){
  for(j in 1:nrow(new.mc.setup$ts.input)){

    ## model parameter definition
    A <- new.mc.setup$par$A
    B <- new.mc.setup$par$B

    ## model input definition
    variable.1 <- new.mc.setup$par$variable.1[i,j]
    variable.2 <- new.mc.setup$par$variable.2[i,j]

    ## model evaluation
    output[j,i+1] <- Model(A, B, variable.1, variable.2)
  }
}

output <- list(output1 = output)
output

## Deterministic simulation
```

```
# model parameter definition
A <- new.mc.setup$par$A
B <- new.mc.setup$par$B

# model input definition
variable.1.det <- apply(X = new.mc.setup$par$variable.1, MARGIN = 2, FUN = mean)
variable.2.det <- apply(X = new.mc.setup$par$variable.2, MARGIN = 2, FUN = mean)

output.det      <- Model(A, B, variable.1.det, variable.2.det)
output.det      <- cbind(time, output.det)
output.det      <- list(out1 = output.det)
str(output.det)

## Monte Carlo analysis
delta     <- 60*6 # minutes
qUpper    <- "q95"
event.ini <- data$time[1]
event.end <- data$time[nrow(data)]
ntick     <- 1

analysis <- MC.analysis_generic(x = output, delta = delta, qUpper = qUpper, data.det = data,
                       sim.det = output.det, event.ini = event.ini, event.end = event.end,
                                ntick = ntick)
```

MC.calibra-methods *Methods for Function* MC.calibra

Description

Given the arguments of the method a calibration routine takes place. Method used only for internal purpose.

Methods

signature(x = "list", obs = "inputObs", EmiStatR.cores = "numeric")

MC.setup-methods *Methods for Function* MC.setup

Description

Given an object of class setup, the method can be invoked for setting-up the Monte Carlo simulation. The variables are sampled accordingly to their parameters specified in the slot rng of the setup object. If ar.model is defined in slot ar.model, then the specified variables are sampled from the pdf nor as an autorregresive (AR) model via the function arima.sim from base package stats. If var.model is defined in slot var.model, then the specified variables are sampled from the pdf nor as an vector autorregresive (VAR) model via the function mAr.sim from package mAr (see Barbosa, 2015, and Luetkepohl, 2005, for details). See setup-class for further details to define the AR and VAR models.

Usage

```
MC.setup(x)
```

Arguments

x	an object of class setup.

Methods

```
signature(x = "setup")
```

Author(s)

J.A Torres-Matallana

References

S. M. Barbosa, Package "mAr": Multivariate AutoRegressive analysis, 1.1-2, The Comprehensive R Archive Network, CRAN, 2015.

H. Luetkepohl, New Introduction to Multiple Time Series Analysis, Springer, 2005.

Examples

```
# loading a precipitation time series as input for the setup class

library(EmiStatR)
data(P1)

# A setup with three variables to be considered in the Monte Carlo simulation:
# var1, a constant value variable; var2, a variable sampled from a uniform (uni)
# probability distribution function (pdf) with parameters min and max;
# var3, a variable sampled from a normal (nor) pdf with parameteres mu and sigma

ini <- setup(id = "MC_sim1", nsim = 500, seed = 123, mcCores = 1, ts.input = P1,
             rng = list(var1 = 150, var2 = c(pdf = "uni", min = 50, max = 110),
                        var3 = c(pdf = "nor", mu = 90, sigma = 2.25))
)

MC_setup <- MC.setup(ini)
str(MC_setup)

## definition of AR models for variables var2 and var3 with AR coefficients 0.995 and 0.460

library(EmiStatR)
data(P1)

ini_ar <- setup(id = "MC_sim1_ar", nsim = 500, seed = 123, mcCores = 1, ts.input = P1,
                rng = list(var1 = 150, var2 = c(pdf = "nor", mu = 150, sigma = 5),
                           var3 = c(pdf = "nor", mu = 90, sigma = 2.25)),
                ar.model = ar.model <- list(var2 = 0.995, var3 = 0.460)
```

```
)

MC_setup_ar <- MC.setup(ini_ar)
str(MC_setup_ar)

## definition of a bi-variate VAR model for variables var2 and var3

ini_var <- setup(id = "MC_sim1_ar", nsim = 500, seed = 123, mcCores = 1, ts.input = P1,
                 rng = rng <- list(var1 = 150,
                                   var2 = c(pdf = "nor", mu = 150, sigma = 5),
                                   var3 = c(pdf = "nor", mu = 90, sigma = 2.25)),
                 var.model = var.model <- list( inp = c("var2", "var3"),
                                                w = c(0.048, 0.021),
                                           A = matrix(c(0.992, -8.8e-05, -31e-4, 0.995),
                                                nrow=2, ncol=2),
                                           C = matrix(c(0.0091, 0.0022, 0.0022, 0.0019),
                                                nrow=2, ncol=2))
)


MC_setup_var <- MC.setup(ini_var)
str(MC_setup_var)
```

`MC.sim-methods` *~~ Methods for Function* `MC.sim` *~~*

Description

Method to be invoked for running the Monte Carlo simulation. The simulator used is the method `EmiStatR` from the homonym package. This method should be rewritted for working with another simulator.

Usage

```
MC.sim(x, EmiStatR.cores)
```

Arguments

`x`	an object of class `list` as is defined by method `MC.setup`.
`EmiStatR.cores`	a `numeric` value for specifying the number of cores (CPUs) to be used in the EmiStatR method. Use zero for not use parallel computation. See class `input` of package `EmiStatR` for details.

Value

A `list` of length 2:

`mc`	A `list` that contains the `MC_setup`, `timing` and `lap` objects.
`sim1`	A `list` that contains the Monte Carlo matrices of the simulator output.

Methods

```
signature(x = "list", EmiStatR.cores = "numeric")
```

Examples

```
## the Monte Carlo simulation: MC.sim

library(EmiStatR)
data(P1)
P1 <- P1[165:(110*2),]
plot(P1[,2], typ="l")

library(stUPscales)

setting_EmiStatR <-  setup(id       = "MC_sim1",
                           nsim     = 3, # use a larger number to have
                                         # a proper confidence band of simulations
                           seed     = 123,
                           mcCores  = 1,
                           ts.input = P1,
                           rng      = rng <- list(
                             qs   = 150,    # [l/PE/d]
                        CODs = c(pdf = "nor", mu = 4.378, sigma = 0.751),    # log[g/PE/d]
                        NH4s = c(pdf = "nor", mu = 1.473, sigma = 0.410),    # log[g/PE/d]
                             qf   = 0.04,     # [l/s/ha]
                             CODf = 0,              # [g/PE/d]
                             NH4f = 0,              # [g/PE/d]
                        CODr = c(pdf = "nor", mu = 3.60, sigma = 1.45),    # 71 log[mg/l]
                             NH4r = 1,              # [mg/l]
                             nameCSO = "E1",        # [-]
                             id      = 1,           # [-]
                             ns      = "FBH Goesdorf",  # [-]
                             nm      = "Goesdorf",  # [-]
                             nc      = "Obersauer", # [-]
                             numc    = 1,          # [-]
                             use     = "R/I",  # [-]
                             Atotal  = 36,              # [ha]
                             Aimp    = c(pdf = "uni", min = 4.5, max = 25),       # [ha]
                             Cimp    = c(pdf = "uni", min = 0.25, max = 0.95),  # [-]
                             Cper    = c(pdf = "uni", min = 0.05, max = 0.60),  # [-]
                             tfS     = 1,               # [time steps]
                             pe      = 650,             # [PE]
                             Qd      = 5,               # [l/s]
                             Dd      = 0.150,           # [m]
                             Cd      = 0.18,            # [-]
                             V       = 190,             #  [m3]
                             lev.ini = 0.10,            # [m]
                             lev2vol = list(lev = c(.06, 1.10, 1.30, 3.30),   # [m]
                                            vol = c(0, 31, 45, 190))          # [m3]
                           ),
                           ar.model  = ar.model <- list(
                             CODs    = 0.5,
```

```
                                  NH4s    = 0.5,
                                  CODr    = 0.7),
                                var.model = var.model <- list(
                                  inp     = c("", ""), # c("CODs", "NH4s"), # c("", ""),
                                  w       = c(0.04778205, 0.02079010),
                                  A       = matrix(c(9.916452e-01, -8.755558e-05,
                                                 -0.003189094, 0.994553910), nrow=2, ncol=2),
                                  C       = matrix(c(0.009126591, 0.002237936,
                                                 0.002237936, 0.001850941), nrow=2, ncol=2)))

MC_setup <- MC.setup(setting_EmiStatR)

sims <- MC.sim(x = MC_setup, EmiStatR.cores = 0)
str(sims)
```

`MC.summary` *Summary statistics computation of Monte Carlo simulation*

Description

A function that computes the summary statistics of a Monte Carlo simulation result.

Usage

```
MC.summary(p1, data)
```

Arguments

p1	The independient variable. A `dataframe` object with two columns and number of rows equal to the number of rows of the Monte Carlo simulated data. The first column, named "time", contains the vector of time of the time series in format `POSIXct`. The second column contains the observations of the time series.
data	A `matrix` or a `dataframe` that contains the results of a Monte Carlo simulation, with number of rows equal to the number of Monte Carlo realizations and number of columns equal to the number of oservations i.e. equal to the number of rows of "p1".

Details

This function is internally invoked by the `MC.analysis` function to compute the summary statistics of the Monte Carlo simulation under analysis.

Value

A `dataframe` with n observations of 15 variables, where n is the number of columns of the "data" argument. The 15 variables are time series with the summary statistics of the Monte Carlo data: 1) `idx`: an index for the dataset equal to 1; 2) `Mean`: the mean; 3) `Sd`: the standard deviation; 4) `Variance`, the variance; 5) `q05`: the five percent quantile; 6) `q25`: the 25 percent quantile; 7) `q50`:

the 50 percent quantile; 8) q75: the 75 percent quantile; 9) q95: the 95 percent quantile; 10) q995: the 99.5 percent quantile; 11) q999: the 99.9 percent quantile; 12) Max: the maximum; 13) Sum: the sum; 14) time: the time; 15) p1: the independient variable.

Author(s)

J.A. Torres-Matallana

Examples

```
library(stUPscales)
library(EmiStatR)

data(P1)
colnames(P1)

new_data <- t(matrix(data = rep(runif(nrow(P1), 10, 100), 5), nrow = nrow(P1), ncol = 5))
new_summary <- MC.summary(p1 = P1, data = new_data)
str(new_summary)
head(new_summary)
```

MC.summary.agg *Summary statistics computation of aggregated Monte Carlo simulation*

Description

A function that computes the summary statistics of aggregated Monte Carlo simulation result.

Usage

```
MC.summary.agg(summ, det, delta, func.agg, func.agg.p)
```

Arguments

summ	A dataframe with n observations of 15 variables, where n is the number observations or time steps of the data. The 15 variables are time series with the summary statistics of the Monte Carlo data. This dataframe is in the format as is described in the MC.summary function value.
det	A dataframe that contains the deterministic simulation.
delta	A numeric value that represents the level of aggregation (required time stemp) in minutes.
func.agg	The aggregation function to be applied to the summ dataframe.
func.agg.p	The aggregation function to be applied to the independient variable p1 from summ dataframe.

Value

A `dataframe` containing the `summ` data aggregated to the level defined by `delta`

Author(s)

J.A. Torres-Matallana

See Also

See Also as `MC.summary`

Examples

```
library(stUPscales)
library(EmiStatR)

data(P1)
colnames(P1)

new_data <- t(matrix(data = rep(runif(nrow(P1), 10, 100), 5), nrow = nrow(P1), ncol = 5))
new_summary <- MC.summary(p1 = P1, data = new_data)
str(new_summary)
head(new_summary)

# deterministic simulation
det <- rnorm(nrow(P1), 45, .15)

# level of aggregation
delta <- 60*2 # 2 hours

new_summary_agg <- MC.summary.agg(summ = new_summary, det, delta, func.agg = mean, func.agg.p = sum)
str(new_summary_agg)
head(new_summary_agg)
```

`PlotEval` *Function to execute evaluation plot*

Description

This function creates an evaluation plot for the Monte Carlo simulation result.

Usage

```
PlotEval(eval, ts, gof1, namePlot, pos1, pos2, pos3)
```

Arguments

- **eval** — A data.frame with n observations of seven variables: 1) time: A POSIXct object with format "%Y-%m-%d %H:%M:%S" defining the time vector; 2) column 2: a numeric vector containing the values of the observed variable, which is the first variable of the Level2Volume relationship; 3) column 3: a numeric vector containing the values for the second variable of the Level2Volume relationship; 4) column 4: a numeric vector containing the corresponding simulated values for the second variable of the Level2Volume relationship; 5) column 5: a numeric vector containing the difference between the vectors volT_sim and volT_obs. 6) Rainfall: a numeric vector named "Rainfall" containing the values of the driving force variable used in the simulations, e.g. rainfall. 7) column 7: (Optional) a numeric vector containing the values of the driving force variable used in the simulations in other measurement units, e.g. rainfall in intensity units if rainfall is the driving force of the simulations.
- **ts** — An xts object representing the eval data.frame indexed by the time vector of the eval argument: containing six data variables as it is defined by the eval argument: 1) column 2; 2) column 3; 3) column 4; 4) column 5; 5) Rainfall; 6) column 7.
- **gof1** — A matrix with the output of GoF function.
- **namePlot** — A character string defining the name of the plot to be created.
- **pos1** — Location to place the legend on the inside of the first sub-plot frame. Can be one of "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right" and "center".
- **pos2** — Location to place the legend on the inside of the second sub-plot frame. Can be one of "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right" and "center".
- **pos3** — Location to place the legend on the inside of the third sub-plot frame. Can be one of "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right" and "center".

Value

The function creates a plot in the current working directory with the goodness-of-fit between simulations and observations. The plot is provided in pdf format.

Author(s)

J.A. Torres-Matallana

Examples

```
time <- seq(from = as.POSIXct("2017-11-09"), by = 60*60*24, length.out = 230) # the time vector
data <- cbind.data.frame(time, NA) # a NA vector
data[,3] <- rnorm(230, .25, .1) # random normal distributed data, obs
data[,4] <- data[,3]*1.2  # positive correlated data, sim
data[,5] <- data[,4] - data[,3] # difference sim and obs
data[,6] <- 0 # driving force
```

```
data[,7] <- NA # a NA vector

colnames(data) <- c("time", "var1", "obs", "sim", "difference", "Rainfall", "Rainfall2")
head(data)

ts <- IsReg.ts(data, "%Y-%m-%d", "ECT")
ts <- ts[[2]]

gof.new <- GoF(data, 4, 3, "")
gof.new

PlotEval(data, ts, gof.new, "ExamplePlot", "topright", "topright", "topright")
```

`PlotMC.event` *A plot function for time series events*

Description

This is an internal function invoked by `MC.analysis` function to generate an event plot of the time series under analysis. A event means a time series with length lower to one month i.e. sub-montly time series.

Usage

```
PlotMC.event(summ, summ1, obs, det.var, det.var1, namePlot, ylab, ylab1, ntick, qUpper)
```

Arguments

`summ`	A data.frame with n observations of m variables as is provided by the output of function `MC.summary.agg` for the first variable to be plotted.
`summ1`	A data.frame with n observations of m variables as is provided by the output of function `MC.summary.agg` for the second variable to be plotted.
`obs`	A numeric value equal to 0. used for internal use.
`det.var`	A character string defining the name of the first variable from `summ` object to be plotted.
`det.var1`	A character string defining the name of the second variable from `summ` object to be plotted.
`namePlot`	A character string defining the name of the plot. The file created with the plot has this name.
`ylab`	A character string to define the label of the axes y for the first variable sub-plot.
`ylab1`	A character string to define the label of the axes y for the second variable sub-plot.
`ntick`	A numeric value integer which defines the number of tick marks in the axis x of the sub-plots.

qUpper A character string that defines the upper percentile to plot the confidence band of results, several options are possible "q999" the 99.9th percentile, "q995" the 99.5th percentile, "q99" the 99th percentile, "q95" the 95th percentile, "q50" the 50th percentile. The lower boundary of the confidence band (showed in gray in the output plots) is the 5th percentile in all cases.

Value

The function creates the plot in the current working directory. The format of the plot is pdf.

Author(s)

J.A. Torres-Matallana

Examples

```
library(stUPscales)
library(EmiStatR)

# definition of the first summary.agg object
data("P1")
P1 <- P1[1:1100,]

new_data <- matrix(data = NA, nrow = nrow(P1), ncol = 55)
for(i in 1:55){
  new_data[,i] <- matrix(data = rnorm(nrow(P1), 45, 15),
                         nrow = nrow(P1), ncol = 1)
}
new_data <- t(new_data)

new_summary <- MC.summary(p1 = P1, data = new_data)

# deterministic simulation
det <- rnorm(nrow(P1), 45, 15)
det <- cbind(det, rnorm(nrow(P1), 55, 23))
colnames(det) <- c("det1", "det2")

# level of aggregation
delta <- 60*2 # 2 hours

new_summary_agg <- MC.summary.agg(summ = new_summary, det, delta,
                                  func.agg = mean, func.agg.p = sum)

# definition of the second summary.agg object
new_data1 <- matrix(data = NA, nrow = nrow(P1), ncol = 55)
for(i in 1:55){
  new_data1[,i] <- matrix(data = rnorm(nrow(P1), 55, 23),
                          nrow = nrow(P1), ncol = 1)
}
new_data1 <- t(new_data1)

new_summary1 <- MC.summary(p1 = P1, data = new_data1)
```

```
new_summary_agg1 <- MC.summary.agg(summ = new_summary1, det, delta,
                                    func.agg = mean, func.agg.p = sum)

# creating the plot for the event
PlotMC.event(summ = new_summary_agg, summ1 = new_summary_agg1, obs = 0,
             det.var = "det1", det.var1 = "det2", namePlot = "ExamplePlot",
             ylab = "Variable 1 [units]", ylab1 = "Variable 2 [units]",
             ntick=10, qUpper= "q95")
```

PlotMC.season *A plot function for time series seasons*

Description

This is an internal function invoked by MC.analysis function to generate a season plot of the time series under analysis. A season means a time series with length greater to one month e.g. montly, yearly, decadal time series.

Usage

```
PlotMC.season(summ1, namePlot, ylab, qUpper)
```

Arguments

summ1	A data.frame with n observations of m variables as is provided by the output of function MC.summary.agg for the variable to be plotted, which the summary was computed.
namePlot	A character string defining the name of the plot. The file created with the plot has this name.
ylab	A character string to define the label of the axes y for the variable to plot.
qUpper	A character string that defines the upper percentile to plot the confidence band of results, several options are possible "q999" the 99.9th percentile, "q995" the 99.5th percentile, "q99" the 99th percentile, "q95" the 95th percentile, "q50" the 50th percentile. The lower boundary of the confidence band (showed in gray in the output plots) is the 5th percentile in all cases.

Value

The function creates the plot in the current working directory. The format of the plot is pdf.

Author(s)

J.A. Torres-Matallana

Examples

```
library(stUPscales)
library(EmiStatR)

data("P1")
P1 <- P1[1:550,]
new_data <- matrix(data = NA, nrow = nrow(P1), ncol = 55)
for(i in 1:55){
  new_data[,i] <- matrix(data = rnorm(nrow(P1), 22, 11),
                         nrow = nrow(P1), ncol = 1)
}
new_data <- t(new_data)

new_summary <- MC.summary(p1 = P1, data = new_data)
head(new_summary)
dim(new_summary)

new_summary$month <- strftime(new_summary[,"time"], format = "%Y-%m")

PlotMC.season(summ1 = new_summary, namePlot = "ExamplePlot",
              ylab = "Variable 1 [units]", qUpper = "q95")
```

`setup-class` *Class* `"setup"`

Description

Class to create objects of signature setup. setup object should be passed to the method MC.setup.

Objects from the Class

Objects can be created by calls of the form `setup()`.

Slots

`id`**:** Object of class `"character"` to identify the Monte Carlo simulation.

`nsim`**:** Object of class `"numeric"` to specify the number of Monte Carlo runs.

`seed`**:** Object of class `"numeric"` to specify the seed of the random numbers generator.

`mcCores`**:** Object of class `"numeric"` to specify the number of cores (CPUs) to be used in the Monte Carlo simulation.

`ts.input`**:** Object of class `"data.frame"` that contains the time series of the main driving force of the system to be simulated, e.g. precipitation. This data.frame should have at least two columns: the first one, Time [y-m-d h:m:s]; the second one, a numeric value equal to the magnitude of the environmental variable. This data.frame can also contain more that one column to allow several time series in several columns. If the data.frame has more than two columns, then the number of columns should be at least equal to nsim. If the number of columns is greater than nsim, the columns in excess are not recycled because the simulation will last nsim iterations.

`rng`**:** Object of class `"list"` that contains the names and values of the variables to be used in the Monte Carlo simulation. Five modes are available: 1) constant value, i.e. this variable will have a constant value along the Monte Carlo simulation; 2) a variable sampled from a uniform (uni) probability distribution function (pdf) with parameters for the lower boundary min and upper boundary max; 3) a variable sampled from a normal (nor) pdf with parameteres mean mu and standard deviation sigma; 4) a variable sampled from an autorregresive (AR) model and normal (nor) pdf with parameteres mean mu and standard deviation sigma, the coefficients of the AR model should be defined in the slot ar.model; 5) a variable sampled from an vector autorregresive (VAR) model and normal (nor) pdf with parameteres mean mu and standard deviation sigma, this mode is enabled by defining the vector of intercept terms w, the matrix of AR coefficients A, and the noise covariance matrix C in the slot var.model. See examples for the definition of this slot.

`ar.model`**:** Object of class `"list"` containing the coefficients of the AR model as vectors which name is the variable to be modeled and length the order of the model as is required for function arima.sim from the base package stats. The named variables here should correspond to a pdf nor in the slot rng. See examples for the definition of this slot.

`var.model`**:** Object of class `"list"` containing the the vector of intercept terms w, the matrix of AR coefficients A, and the noise covariance matrix C of the VAR model which name is the variable to be modeled and length the order of the model as is required for function mAr.sim from the package mAr. The named variables in this slot should correspond to a pdf nor in the slot rng. The current implementation considers the bi-variate case. See examples for the definition of this slot. For mathemamtical details see Luetkepohl (2005).

Methods

MC.setup `signature(x = "setup")`: execute MC.setup function

Author(s)

J.A Torres-Matallana

References

S. M. Barbosa, Package "mAr": Multivariate AutoRegressive analysis, 1.1-2, The Comprehensive R Archive Network, CRAN, 2015.

H.Luetkepohl, New Introduction to Multiple Time Series Analysis, Springer, 2005.

Examples

```
# loading a precipitation time series as input for the setup class

library(EmiStatR)
data(P1)

# A setup with three variables to be considered in the Monte Carlo simulation:
# var1, a constant value variable; var2, a variable sampled from a uniform (uni)
# probability distribution function (pdf) with parameters min and max;
# var3, a variable sampled from a normal (nor) pdf with parameteres mu and sigma
```

```
ini <- setup(id = "MC_sim1", nsim = 500, seed = 123, mcCores = 1, ts.input = P1,
             rng = list(var1 = 150, var2 = c(pdf = "uni", min = 50, max = 110),
                        var3 = c(pdf = "nor", mu = 90, sigma = 2.25))
)

str(ini)

## definition of AR models for variables var2 and var3 with AR coefficients 0.995 and 0.460

library(EmiStatR)
data(P1)

ini_ar <- setup(id = "MC_sim1_ar", nsim = 500, seed = 123, mcCores = 1, ts.input = P1,
                rng = list(var1 = 150, var2 = c(pdf = "nor", mu = 150, sigma = 5),
                           var3 = c(pdf = "nor", mu = 90, sigma = 2.25)),
                ar.model = ar.model <- list(var2 = 0.995, var3 = 0.460)
)

str(ini_ar)

## definition of a bi-variate VAR model for variables var2 and var3

ini_var <- setup(id = "MC_sim1_ar", nsim = 500, seed = 123, mcCores = 1, ts.input = P1,
                 rng = rng <- list(var1 = 150,
                                   var2 = c(pdf = "nor", mu = 150, sigma = 5),
                                   var3 = c(pdf = "nor", mu = 90, sigma = 2.25)),
                 var.model = var.model <- list( inp = c("var2", "var3"),
                                                w = c(0.048, 0.021),
                                           A = matrix(c(0.992, -8.8e-05, -31e-4, 0.995),
                                                nrow=2, ncol=2),
                                           C = matrix(c(0.0091, 0.0022, 0.0022, 0.0019),
                                                nrow=2, ncol=2))
)

str(ini_var)
```

Smooth.ts *Kernel smoothing for time series*

Description

Given one or more time series in xts format, this function applies a Daniel kernel function for smoothing the time series.

Usage

```
Smooth.ts(data.xts, kernel = kernel("daniell", c(1, 1)), sm.threshold = 1)
```

Arguments

data.xts — xts object with the time series to smooth.

kernel — tskernel object which defines the smoothing kernel to apply.

sm.threshold — numeric, defines the threshold value to apply the Kernel smoothing. Only values lower than sm.threshol are smoothed.

Value

A xts object with the time series smoothed.

Author(s)

J.A. Torres-Matallana

Examples

```
data("Lux_precipitation_2010_2011")

data.xts <- Lux_precipitation_2010_2011

my.smoothed.ts <- Smooth.ts(data.xts, kernel = kernel("daniell", c(1, 1)),
                            sm.threshold = 1)

# plot
par(mfrow=c(2,1))
plot(data.xts['2011-06-20 14:00/2011-06-22 16:00'])
plot(my.smoothed.ts['2011-06-20 14:00/2011-06-22 16:00'])
```

Validation_Quantity-methods

Methods for Function Validation_Quantity

Description

Given the arguments of the method a validation routine takes place. Method used only for internal purpose.

Methods

signature(x = "input", y = "inputObs")

`Validation_Quantity_Agg-methods` *Methods for Function* `Validation_Quantity_Agg`

Description

Given the arguments of the method a validation routine takes place. Method used only for internal purpose.

Methods

`signature(x = "input", y = "inputObs")`

`xts2STFDF` *Creation of STFDF object from xts and SpatialPointsDataFrame objects*

Description

Given one or more time series as an xts object from xts packakge and its associated geospatial domain represented as a SpatialPointsDataFrame from package sp, this function creates a spacetime full data.frame (STFDF) object as defined by package spacetime.

Usage

```
xts2STFDF(data.xts, point)
```

Arguments

`data.xts`	`xts` object which defines the time series.
`point`	`SpatialPointsDataFrame`, point geospatial domain of data.xts.

Value

A `STFDF` object.

Author(s)

J.A. Torres-Matallana

Examples

```
data("Lux_stations")

data("Lux_precipitation_2010_2011")
data.xts <- Lux_precipitation_2010_2011

my.stfdf <- xts2STFDF(data.xts, point = stations)

library(spacetime)
stplot(my.stfdf[,1:16,"data"])
stplot(my.stfdf[,"2011-12-16", drop = FALSE], mode = "ts")

data("Lux_boundary")
library(sp)
spplot(my.stfdf@sp, "elev_luref",
       sp.layout=list(sp.polygons = boundary.Lux),
       scales=list(arrows=FALSE),
       xlim=bbox(boundary.Lux)[1,],
       ylim=bbox(boundary.Lux)[2,],
       key.space = "right")
```

Index