Package 'stUPscales'

April 3, 2018

Type Package

Title Spatio-Temporal Uncertainty Propagation Across Scales

Version 1.0.3

Date 2018-03-02

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 The Spatio-
Temporal Uncertainty Propagation Across Scales, stUPscales, provides several R functions for temporal aggregation of environmental variables used in Urban Drainage Models (UDMs), as precipitation and pollutants.
Also, it provides methods and functions for uncertainty propagation via Monte Carlo simulation.
This package, moreover, provides specific analysis functions for urban drainage system simulation to evaluate water quantity and quality in combined sewer overflows (CSOs).

License GPL (>=3)

Depends methods,
stats,
graphics,
grDevices,
utils,
data.table,
xts,
mAr,
lmom,
EmiStatR

Imports parallel,
doParallel,
foreach,
lattice,
msm,
ggplot2,
moments,

hydroGOF,
zoo

NeedsCompilation no

R topics documented:

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

Description

The Spatio-Temporal Uncertainty Propagation Across Scales, stUPscales, provides several R functions for temporal aggregation of environmental variables used in Urban Drainage Models (UDMs), as precipitation and pollutants. Also, it provides methods and functions for uncertainty propagation via Monte Carlo simulation. This package, moreover, provides specific analysis functions for urban drainage system simulation to evaluate water quantity and quality in combined sewer overflows (CSOs).

Details

The DESCRIPTION file:

Package:	stUPscales
Type:	Package
Version:	1.0.3
Date:	2018-03-02
License:	GPL (>= 3)
Depends:	R (>= 2.10), methods, mAr, data.table, lmom, xts

Author(s)

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

Maintainer: J.A. Torres-Matallana.

`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(stUPscales)
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)
```

Class-setup *Class* "setup"

Description

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

Usage

```
setup(id = "MC_sim_1", nsim = 1, seed = 0.7010607, mcCores = 1, ts.input = NULL,
      rng = NULL, ar.model  = list(NULL), var.model = list(NULL))
```

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).

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)
```

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 for the plot created. If empty string ("") then no plot is 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
```

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

Description

"IsReg" is a wrapping Function for Function "is.regular" from "zoo" package. Given 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(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(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]")
```

`Level2Volume` *Function for linear interpolation given the relationship of two variables*

Description

Given a relationship between two variables (e.g. level and volume), this function interpolates the corresponding second variable (e.g. volume) for a known value of the first variable (e.g. level). This function is suitable to represented e.g. the dynamics of water storage in a combined sewer overflow chamber.

Usage

```
Level2Volume(lev, lev2vol)
```

Arguments

`lev`	A `numeric` object that represents the know variable e.g. the level in a storage chamber.
`lev2vol`	A `list` of two elements. The first element is a `vector` named "lev" that contains the interpolation steps of the first variable (e.g. level). The second element is a `vector` that contains the interpolation steps of the second variable (e.g. volume).

Details

The function uses the `approx` function from the `stats` package with "yleft" argument equal to the minimum value of the second variable and "yright" argument equal to the maximum value of the second variable.

Value

A `numeric` object with one element representing the interpolated value of the second variable (e.g. volume).

Author(s)

J.A. Torres-Matallana

Examples

```
library(stUPscales)

# definition of relationship level to volume
lev2vol <- list(lev = c(.06, 1.10, 1.30, 3.30), vol = c(0, 31, 45, 200))

interpolated_volume <- Level2Volume(lev=c(0, .25, 1.25, 2.25, 4.25), lev2vol = lev2vol)
interpolated_volume
```

`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`	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 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)
data(Esch_Sure2010)

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

library(stUPscales)

setting_EmiStatR <-  setup(id       = "MC_sim1",
                           nsim     = 5,
                           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)

## 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("2010-08-15 15:30:00")
event.end <- as.POSIXct("2010-08-16 09:30:00")


new_analysis <- MC.analysis(sims, delta, qUpper,  P1, sim.det, event.ini, event.end, 5)
str(new_analysis)
```

`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(Esch_Sure2010)

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

library(stUPscales)

setting_EmiStatR <-  setup(id       = "MC_sim1",
                           nsim     = 5,
                           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(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")
new_data <- matrix(data = NA, nrow = nrow(P1), ncol = 500)
for(i in 1:500){
  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)
head(new_summary)
dim(new_summary)

# 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)
head(new_summary_agg)

# definition of the second summary.agg object
new_data1 <- matrix(data = NA, nrow = nrow(P1), ncol = 500)
for(i in 1:500){
  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)
```

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

# 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.

Author(s)

J.A. Torres-Matallana

References

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

Examples

```
library(stUPscales)
library(EmiStatR)

data("P1")
new_data <- matrix(data = NA, nrow = nrow(P1), ncol = 500)
for(i in 1:500){
  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")
```

Index