Package ‘foreSIGHT’
October 13, 2022

Version 1.0.0
Depends R (>= 3.5.0), GA (>= 3.0.2), zoo
LinkingTo Rcpp
Imports doParallel, ggplot2 (>= 3.3.0), directlabels, cowplot, stats, graphics, grDevices, utils, moments, Rcpp, jsonlite, progress, rcorpora, scales, viridisLite, fields, RColorBrewer, rlang
Suggests knitr (>= 1.8), rmarkdown (>= 1.18), testthat, evd
Title Systems Insights from Generation of Hydroclimatic Timeseries
Author Bree Bennett [aut, cre] (<https://orcid.org/0000-0002-2131-088X>), Sam Culley [aut] (<https://orcid.org/0000-0003-4798-8522>), Anjana Devanand [aut] (<https://orcid.org/0000-0001-9422-3894>), Seth Westra [aut] (<https://orcid.org/0000-0003-4023-6061>), Danlu Guo [ctb] (<https://orcid.org/0000-0003-1083-1214>), Holger Maier [ths] (<https://orcid.org/0000-0002-0277-6887>)
Maintainer Bree Bennett <bree.bennett@adelaide.edu.au>
BugReports https://github.com/bsbennett/foreSIGHT/issues
tial evapotranspiration) and allows a variety of different hydroclimate variable properties, herein called attributes, to be perturbed. Options are included for the easy integration of existing system models both internally in R and externally for seamless ‘stress-testing’. A suite of visualization options for the results of a scenario-neutral analysis (e.g. plotting performance spaces and overlaying climate projection information) are also included. As further developments in scenario-neutral approaches occur the tool will be updated to incorporate these advances.

License GPL-3
NeedsCompilation yes
VignetteBuilder knitr
LazyData true
RoxygenNote 7.1.1
Repository CRAN
Date/Publication 2020-12-17 09:00:12 UTC

R topics documented:

calculateAttributes .................................................. 3
climdata ................................................................. 4
createExpSpace ......................................................... 4
egClimData ............................................................. 7
egScalPerformance .................................................... 7
egScalSummary ......................................................... 8
egSimOATPerformance ............................................... 8
egSimOATSsummary ..................................................... 9
egSimPerformance ..................................................... 9
egSimPerformance_systemB ....................................... 10
egSimSummary ......................................................... 10
generateScenario ..................................................... 11
generateScenarios .................................................... 12
getSimSummary ........................................................ 14
modCalibrator ......................................................... 15
modSimulator .......................................................... 16
plotExpSpace .......................................................... 17
plotOptions ............................................................ 19
plotPerformanceOAT .................................................. 21
plotPerformanceSpace .............................................. 22
plotPerformanceSpaceMulti ....................................... 25
plotScenarios ........................................................ 27
runSystemModel ....................................................... 29
tankPerformance ..................................................... 31
tankWrapper ........................................................... 32
tank_obs ............................................................... 33
viewAttributeDef ..................................................... 33
viewAttributes ........................................................ 34
viewDefaultOptimArgs .............................................. 35
calculateAttributes

Calculates the attributes of the hydroclimate time series

Description

calculateAttributes calculates the specified attributes of the input daily hydroclimate time series.

Usage

calculateAttributes(climateData, attSel, startYr = NULL, endYr = NULL)

Arguments

climateData data.frame; daily climate data, the attributes of which are to be calculated. climateData should be a data.frame with columns named `year`, `month`, `day`, `variable_name1`, `variable_name2`. Use `viewModels()` to view the valid variable names. Note that the first three columns of the data.frame contain the year, month, and day of the data. The columns have to be named as specified. Please refer data provided with the package that may be loaded using `data(tankDat)` for an example of the expected format of `climateData`.

attSel a vector; specifying the names of the attributes to be calculated.

startYr a number (default NULL); to specify the starting year to subset `climateData` if required. If NULL, startYr is starting year in the input `climateData`.

endYr a number (default NULL); to specify the ending year to subset `climateData` if required. If NULL, endYr is last year in the input `climateData`.

Value

The function returns a vector of length equal to the number of selected attributes (`attSel`), named using the names of the attributes.

See Also

`viewAttributes`, `viewAttributeDef`
Examples

# view the names of available attributes
viewAttributes()
# load example climate data available in the package
data("tankDat")
attSel <- c("P_ann_tot_m", "P_ann_nWet_m", "P_ann_R10_m", "Temp_ann_rng_m", "Temp_ann_avg_m")
tank_obs_atts <- calculateAttributes(tank_obs, attSel = attSel)

climdata

Example climate projection data

Description

A dataframe of climate projection data for superposition on performance spaces via plotLayers

Format

climdata is a dataframe with 12 rows and 3 columns

climdata A dataframe of climate attributes and performance in the form \( P_{\text{ann\_tot\_m}} \) \( \text{Temp\_ann\_avg\_m} \)

performance.

createExpSpace

Creates exposure space of hydroclimatic targets for generation of scenarios using 'generateScenarios'

Description

createExpSpace returns a list containing the targets (targetMat) and the metadata (input arguments) used to create the exposure space.

Usage

createExpSpace(
  attPerturb,
  attPerturbSamp,
  attPerturbMin,
  attPerturbMax,
  attPerturbType = "regGrid",
  attPerturbBy = NULL,
  attHold = NULL,
  attTargetsFile = NULL
)
createExpSpace

Arguments

attPerturb A char vector; the names of the attributes to be perturbed. This vector can contain attributes of different hydroclimatic variables.

attPerturbSamp An integer vector; the number of samples for each attribute attPerturb. The length of this vector should be equal to the length of attPerturb.

attPerturbMin A numeric vector; the minimum bounds for sampling of attPerturb. The length of this vector should be equal to the length of attPerturb. For variables like precipitation, evapotranspiration, radiation, etc. attPerturbMin should specified as a fraction of the original (e.g: 0.9 = 90% of the original attribute). For temperature, attPerturbMin should be specified in K (e.g: 0.9 = 0.9 K).

attPerturbMax A numeric vector; the maximum bounds for sampling of attPerturb. The length of this vector should be equal to the length of attPerturb. For variables like precipitation, evapotranspiration, radiation, etc. attPerturbMax should specified as a fraction of the original (e.g: 0.9 = 90% of the original attribute). For temperature, attPerturbMax should be specified in K (e.g: 0.9 = 0.9 K). Note that to create a single sample of the attribute, attPerturbSamp could be specified as 1 with attPerturbMin and attPerturbMax specified as equal.

attPerturbType A string to specify the type of sampling, defaults to regular spacing. Valid sampling types are:

- "regGrid" a regular grid sampling all the attributes specified in attPerturb simultaneously
- "OAT" one-at-a-time sampling of the attributes specified in attPerturb

attPerturbBy A numeric vector; increment of values to create samples between attPerturbMin and attPerturbMax. If attPerturbBy is specified, attPerturbSamp should be set as NULL.

attHold A char vector; the names of the attributes to be held at historical levels. This vector can contain attributes of different hydroclimatic variables.

attTargetsFile String specifying the full path to a CSV file containing the target exposure space. The column names in the file should correspond to the attributes specified in attPerturb and attHold. attTargetsFile is alternate way to specify exposure space targets that do not form a regular grid. If attTargetsFile is specified, the inputs arguments attPerturbSamp, attPerturbMin, attPerturbMax, and attPerturbType should be set to NULL and will not be used by the function.

Details

The list of valid attributes that may be specified using attPerturb or attHold can be viewed using the function viewAttributes(). The definition of the attribute can be viewed using the function viewAttributeDef.

Value

The exposure space as a list containing the following fields:

- targetMat a dataframe or matrix; each column is a perturb/hold attribute, each row is a point in the exposure space.
- attRot a char vector containing the one-at-a-time ("OAT") attributes associated with targetMat, attRot is NULL for other types of sampling.
- attPerturb, attHold, attPerturbSamp, attPerturbMin, attPerturbMax, attPerturbType in the function input arguments, if not NULL.

See Also

generateScenarios, viewAttributes, viewAttributeDef

Examples

# To view the valid attributes. The function does not take any input arguments.
viewAttributes()

# To view the definition of any valid attribute
viewAttributeDef("P_ann_tot_m")

# To create an exposure space of points on a regular grid
attPerturb <- c("P_ann_tot_m", "P_ann_nWet_m", "P_ann_R10_m")
attPerturbType <- "regGrid"
attPerturbSamp <- c(3, 1, 1)
attPerturbMin <- c(0.9, 1, 1)
attPerturbMax <- c(1.1, 1, 1)
attHold <- c("P_Feb_tot_m", "P_SON_dyWet_m", "P_JJA_avgWSD_m", "P_MAM_tot_m", "P_DJF_avgDSD_m", "Temp_ann_rng_m", "Temp_ann_avg_m")
expSpace <- createExpSpace(attPerturb = attPerturb, attPerturbSamp = attPerturbSamp, attPerturbMin = attPerturbMin, attPerturbMax = attPerturbMax, attPerturbType = attPerturbType, attHold = attHold, attTargetsFile = NULL)

# Using attPerturbBy to specify the increment of perturbation (attPerturbSamp set to NULL)
attPerturb <- c("P_ann_tot_m", "P_ann_nWet_m", "P_ann_R10_m")
attPerturbType <- "regGrid"
attPerturbMin <- c(0.9, 1, 1)
attPerturbMax <- c(1.1, 1, 1)
attPerturbBy <- c(0.1, 0, 0)
attHold <- c("P_Feb_tot_m", "P_SON_dyWet_m", "P_JJA_avgWSD_m", "P_MAM_tot_m", "P_DJF_avgDSD_m", "Temp_ann_rng_m", "Temp_ann_avg_m")
expSpace <- createExpSpace(attPerturb = attPerturb, attPerturbSamp = NULL, attPerturbMin = attPerturbMin, attPerturbMax = attPerturbMax, attPerturbType = attPerturbType, attPerturbBy = attPerturbBy, attHold = attHold, attTargetsFile = NULL)

# To create an exposure space of observed attributes without perturbation
# Note that attPerturbMin and attPerturbMax values are set to 1 for variables like precipitation,
# and 0 for temperature
attPerturb <- c("P_ann_tot_m", "P_ann_nWet_m", "P_ann_R10_m", "Temp_DJF_avg_m")
attPerturbType <- "regGrid"
attPerturbSamp <- c(1, 1, 1, 1)
attPerturbMin <- c(1, 1, 1, 0)
attPerturbMax <- c(1, 1, 1, 0)
expSpace <- createExpSpace(attPerturb = attPerturb, attPerturbSamp = attPerturbSamp, attPerturbMin = attPerturbMin, attPerturbMax = attPerturbMax, attPerturbType = attPerturbType,
egClimData

Description
A example dataset containing the climate attribute values in fraction/additive change

Usage
egClimData

Format
A data frame with 6 rows and 6 variables:
P_ann_tot_m change in mean annual total P, fraction
P_ann_seasRatio_m change in seasonal ratio of P, fraction
P_ann_nWet_m change in the number of wet days, fraction
Temp_ann_avg_m change in average annual Temp, additive
Name name of the climate model
Avg. Deficit performance metric values

egScalPerformance

Description
Performance metrics of the tank model using simple scaled scenarios.

Usage
egScalPerformance

Format
A list with 2 elements
Avg. Deficit average daily deficit of water, litres
Reliability reliability of the tank, fraction
egScalSummary

Summary of a simple scaled scenario.

Description

Summary generated using the function getSimSummary.

Usage

egScalSummary

Format

A list containing 3 elements

- **simDates** the dates of the simulation
- **expSpace** the exposure space of the simulation
- **controlFile** "scaling"

egSimOATPerformance

Performance metrics of the tank model using OAT scenarios.

Description

Performance metrics of the tank model using OAT scenarios.

Usage

egSimOATPerformance

Format

A list with 2 elements

- **Avg. Deficit** average daily deficit of water, litres
- **Reliability** reliability of the tank, fraction
egSimOATSummary

Summary of a OAT scenario.

Description

Summary generated using the function getSimSummary for a scenarios generated using stochastic models for an OAT exposure space.

Usage

egSimOATSummary

Format

A list containing 13 elements

egSimPerformance

Performance metrics of the tank model using regGrid scenarios.

Description

Performance metrics of the tank model using regGrid scenarios.

Usage

egSimPerformance

Format

A list with 2 elements

- Avg. Deficit: average daily deficit of water, litres
- Reliability: reliability of the tank, fraction
### egSimPerformance_systemB

Performance metrics of an alternate tank model using regGrid scenarios.

#### Description
Performance metrics of an alternate tank model using regGrid scenarios.

#### Usage

```
egSimPerformance_systemB
```

#### Format

A list with 2 elements

- **Avg. Deficit** average daily deficit of water, litres
- **Reliability** reliability of the tank, fraction

### egSimSummary

Summary of a regGrid scenario.

#### Description

Summary generated using the function getSimSummary for a scenarios generated using stochastic models for a regGrid exposure space

#### Usage

```
egSimSummary
```

#### Format

A list containing 13 elements
generateScenario

**Description**

generateScenario is the base function used by generateScenarios. The function produces time series of hydroclimatic variables using requested climate attributes that correspond to a single target in the exposure space. The function argument definitions are detailed in the documentation of generateScenarios; please refer to that documentation using ?generateScenarios.

**Usage**

generateScenario(
  reference,  
  expTarg,    
  simLengthNyrs = NULL, 
  seedID = NULL, 
  controlFile = NULL
)

**Arguments**

*reference*  
data.frame; contains reference daily climate data in a data frame with columns named `year` `month` `day` *variable_name1* *variable_name2*.* Use `viewModels()` to view the valid variable names. Note that the first three columns of the data.frame contain the year, month, and day of the reference. The columns have to be named as specified. Please refer data provided with the package that may be loaded using `data(tankDat)` for an example of the expected format of `reference`.

*expTarg*  
a named vector; the attributes at the target location in the exposure space `generateScenario` is intended to be used to adapt the functionality of `generateScenarios` for use in a parallel computing environment.

*simLengthNyrs*  
a number; a scalar that specifies the length in years of each generated scenario. This argument is used only with stochastic generation. If NULL (the default), the generated simulation will be as long as `reference`.

*seedID*  
a number; a scalar that specifies the seed to be used for the first replicate. Subsequent replicates will use seeds incremented by one. If `seedID` is NULL (which is the default), the function will use a random seed for stochastic time series generation. The seed used will be specified in the output. This argument is intended for use in cases that aim to reproduce an existing simulation.

*controlFile*  
a string; to specify the model/optimisation options used for simulating time series data. The valid values are:
  - NULL: the simulation uses the foreSIGHT default stochastic model settings.
  - "scaling" : the simulation uses simple scaling instead of a stochastic model.
• path to a JSON file: the JSON file contains advanced options specify the stochastic model and optimisation inputs. These options can be used to change stochastic model types, overwrite default model parameter bounds, change default optimisation arguments, and set penalty attributes to be used in optimisation. Please refer to the function writeControlFile in order to create an controlFile JSON file.

See Also

generateScenarios

---

### Description

generateScenarios produces time series of hydroclimatic variables for an exposure space.

### Usage

```r
generateScenarios(
  reference,
  expSpace,
  simLengthNyrs = NULL,
  numReplicates = 1,
  seedID = NULL,
  controlFile = NULL
)
```

### Arguments

- **reference**: data.frame; contains reference daily climate data in a data frame with columns named `year month day *variable_name1* *variable_name2*`. Use `viewModels()` to view the valid variable names. Note that the first three columns of the data.frame contain the year, month, and day of the reference. The columns have to be named as specified. Please refer data provided with the package that may be loaded using `data(tankDat)` for an example of the expected format of `reference`.

- **expSpace**: a list; created using the function `createExpSpace`

- **simLengthNyrs**: a number; a scalar that specifies the length in years of each generated scenario. This argument is used only with stochastic generation. If `NULL` (the default), the generated simulation will be as long as `reference`.

- **numReplicates**: a number; a scalar that specific the number of stochastic replicates to be generated. The default is 1.
seedID a number; a scalar that specifies the seed to be used for the first replicate. Subsequent replicates will use seeds incremented by one. If seedID is NULL (which is the default), the function will use a random seed for stochastic time series generation. The seed used will be specified in the output. This argument is intended for use in cases that aim to reproduce an existing simulation.

controlFile a string; to specify the model/optimisation options used for simulating time series data. The valid values are:

- NULL: the simulation uses the foreSIGHT default stochastic model settings.
- "scaling": the simulation uses simple scaling instead of a stochastic model.
- path to a JSON file: the JSON file contains advanced options specify the stochastic model and optimisation inputs. These options can be used to change stochastic model types, overwrite default model parameter bounds, change default optimisation arguments, and set penalty attributes to be used in optimisation. Please refer to the function writeControlFile in order to create an controlFile JSON file.

Value

The function returns a list containing the time series data generated. The list can contain multiple replicates (named as Rep1, Rep2 etc.) equal to the numReplicates function argument. Each replicate can contain multiple targets (named as Target1, Target2 etc.) based on the specified exposure space (expSpace). The expSpace and controlFile are also returned as part of this output list.

See Also

createExpSpace, writeControlFile, viewModels

Examples

# Example 1: Stochastic simulation using foreSIGHT default settings
#---------------------------------------------------------------
## Not run:
# create an exposure space
attPerturb <- c("P_ann_tot_m", "P_ann_nWet_m", "P_ann_R10_m")
attHold <- c("P_Feb_tot_m", "P_SON_dyWet_m", "P_JJA_avgWSD_m", "P_MAM_tot_m",
             "P_DJF_avgDSD_m", "Temp_ann_rng_m", "Temp_ann_avg_m")
attPerturbType = "regGrid"
attPerturbSamp = c(2, 1, 1)
attPerturbMin = c(0.8, 1, 1)
attPerturbMax = c(1.1, 1, 1)
expSpace <- createExpSpace(attPerturb = attPerturb,
                           attPerturbSamp = attPerturbSamp,
                           attPerturbMin = attPerturbMin,
                           attPerturbMax = attPerturbMax,
                           attPerturbType = attPerturbType,
                           attHold = attHold,
                           attTargetsFile = NULL)
# load example data available in foreSIGHT
data(tankDat)
# perform stochastic simulation
simStochastic <- generateScenarios(reference = tank_obs, 
    expSpace = expSpace, 
    simLengthNyrs = 30)

## End(Not run)

# Example 2: Simple scaling
#-----------------------------------------------------------------------
attPerturb<-c("P_ann_tot_m", "Temp_ann_avg_m")
attPerturbType = "regGrid"
attPerturbSamp = c(2, 2)
attPerturbMin = c(0.8, -1)
attPerturbMax = c(1.1, 1)
expSpace <- createExpSpace(attPerturb = attPerturb, 
    attPerturbSamp = attPerturbSamp, 
    attPerturbMin = attPerturbMin, 
    attPerturbMax = attPerturbMax, 
    attPerturbType = attPerturbType)
data(tankDat)
simScaling <- generateScenarios(reference = tank_obs, 
    expSpace = expSpace, 
    controlFile = "scaling")

getSimSummary

Produces a summary object containing the metadata of a full simulation

Description

getSimSummary uses a full simulation generated using the function generateScenarios as input and outputs a summary object containing the metadata of the full simulation. The output summary object may be used as an input to the plotting functions in this package. The output summary object will be much smaller in size than the full simulation for ease of storage and use with the plotting functions.

Usage

getSimSummary(sim)

Arguments

sim: list; a simulation containing the scenarios generated using the function generateScenarios.

See Also
generateScenarios, plotPerformanceSpace, plotPerformanceOAT
modCalibrator

Description
Calibrates weather generator models specified using modelTag.

Usage
modCalibrator(obs = NULL,
             modelTag = NULL,
             window = NULL)

Arguments
obs A dataframe of observed climate data in the form Year Month Day P Temp.
modelTag A character vector of which stochastic models to use to create each climate variable. Supported tags are shown in under details below.
window moving average window to calibrate daily gamma parameters for the modelTag "P-har-WGEN".

Details
modelTag provides the main function with requested models. modelTag is vector of any of the following supported models:

- "Simple-ann"a simple annual scaling
- "P-ann-wgen"a four parameter annual rainfall model
- "P-seas-wgen"a 16 parameter seasonal rainfall model
- "P-har6-wgen"a harmonic rainfall model with 6 periods
- "P-har12-wgen"a harmonic rainfall model
- "P-har12-wgen-FS"a harmonic rainfall model where seasonality is fixed (phase angles must be specified via modelInfoMod=list("P-har12-wgen-FS"=fixedPars=c(x,x,x,x))
- "P-har26-wgen"a harmonic rainfall model
- "P-2har26-wgen"a double harmonic rainfall model
- "Temp-har26-wgen"a harmonic temperature model not conditional on rainfall
- "Temp-har26-wgen-wd"a harmonic temperature model dependent on wet or dry day
- "Temp-har26-wgen-wdsd"a harmonic temperature model where standard deviation parameters are dependent on wet or dry day
- "PET-har12-wgen"a harmonic potential evapotranspiration model
- "PET-har26-wgen"a harmonic potential evapotranspiration model
- "PET-har26-wgen-wd"a harmonic potential evapotranspiration model dependent on wet or dry day
- "Radn-har26-wgen"a harmonic solar radiation model (MJ/m2)
modSimulator

Examples

```r
data(tankDat) # Load tank data (tank_obs)
modelTag=c("P-ann-wgen","Temp-har26-wgen") # Select a rainfall and a temperature generator
out<- modCalibrator(obs = tank_obs, modelTag = modelTag) # Calibrate models
```

modSimulator

Description

Simulates using weather generator models specified using modelTag.

Usage

```r
modSimulator(datStart=NULL, datFinish=NULL, modelTag=NULL, parS=NULL, seed=NULL, file=NULL, IOmode="suppress")
```

Arguments

- `datStart`: A date string in an accepted date format e.g. "01-10-1990".
- `datFinish`: A date string in an accepted date format e.g. "01-10-1990". Must occur after datStart.
- `modelTag`: A character vector of which stochastic models to use to create each climate variable. Supported tags are shown in details below.
- `parS`: A list (names must match supplied modelTags) containing numeric vectors of model parameters.
- `seed`: Numeric. Seed value supplied to weather generator.
- `file`: Character. Specifies filename for simulation output.
- `IOmode`: A string that specifies the input-output mode for the time series = "verbose", "dev" or "suppress".

Details

modelTag provides the main function with requested models. modelTag is vector of any of the following supported models:

- "Simple-ann" - a simple annual scaling
- "P-ann-wgen" - a four parameter annual rainfall model
plotExpSpace

- "P-seas-wgen" a 16 parameter seasonal rainfall model
- "P-har6-wgen" a harmonic rainfall model with 6 periods
- "P-har12-wgen" a harmonic rainfall model
- "P-har12-wgen-FS" a harmonic rainfall model where seasonality is fixed (phase angles must be specified via modelInfoMod=list("P-har12-wgen-FS"=fixedPars=c(x,x,x,x))
- "P-har26-wgen" a harmonic rainfall model
- "P-2har26-wgen" a double harmonic rainfall model
- "Temp-har26-wgen" a harmonic temperature model not conditional on rainfall
- "Temp-har26-wgen-wd" a harmonic temperature model dependent on wet or dry day
- "Temp-har26-wgen-wdsd" a harmonic temperature model where standard deviation parameters are dependent on wet or dry day
- "PET-har12-wgen" a harmonic potential evapotranspiration model
- "PET-har26-wgen" a harmonic potential evapotranspiration model
- "PET-har26-wgen-wd" a harmonic potential evapotranspiration model dependent on wet or dry day
- "Radn-har26-wgen" a harmonic solar radiation model (MJ/m2)

Examples

```r
## Not run:
data(tankDat); obs=tank_obs #Get observed data
modelTag=c("P-har12-wgen","Temp-har26-wgen") #Select models
pars=modCalibrator(obs=obs,modelTag=modelTag) #Calibrate models
sim=modSimulator(datStart="1970-01-01", #Simulate!
datFinish="1999-12-31", modelTag=modelTag,
parS=pars, seed=123,
file=paste0(" tester.csv"), IOmode="verbose")
plot(sim$P[1:365]) #Plot first year of rainfall
## End(Not run)
```

plotExpSpace

Plots the location of points in a two-dimensional exposure space

Description

The function uses an exposure space created using the function createExpSpace as input and creates a plot of the two dimensional (2D) exposure space. plotExpSpace plots only 2D spaces consisting of samples of 2 attributes.
Usage

plotExpSpace(
  expSpace,
  y = expSpace[["attPerturb"]][1],
  x = expSpace[["attPerturb"]][2]
)

Arguments

expSpace list; an exposure space created using the function createExpSpace
y a string; tag of a perturbed attribute to plot on the y-axis. Defaults to expSpace[["attPerturb"]][1].
x a string; tag of a perturbed attribute to plot on the x-axis. Defaults to expSpace[["attPerturb"]][2].

Details

The number of dimensions of an exposure space is equal to the number of perturbed attributes in that space. If the exposure space has more than 2 dimensions (perturbed attributes), this function can be used to plot 2D slices of the space. Note that the default arguments of this function is defined to plot a slice showing the first two dimensions of the space, arguments x and y may be specified to plot alternate dimensions.

See Also

createExpSpace

Examples

# create an exposure space that has more than 2 dimensions
attPerturb <- c("P_ann_tot_m", "P_ann_nWet_m", "P_Feb_tot_m")
attHold <- c("P_SON_dyWet_m", "P_JJA_avgWSD_m", "P_MAM_tot_m", "P_DJF_avgDSD_m",
             "Temp_ann_rng_m", "Temp_ann_avg_m")
attPerturbType = "regGrid"
attPerturbSamp = c(5, 5, 5)
attPerturbMin = c(0.8, 0.9, 0.85)
attPerturbMax = c(1, 1.1, 1.05)
expSpace <- createExpSpace(attPerturb = attPerturb,
                           attPerturbSamp = attPerturbSamp,
                           attPerturbMin = attPerturbMin,
                           attPerturbMax = attPerturbMax,
                           attPerturbType = attPerturbType,
                           attHold = attHold,
                           attTargetsFile = NULL)

# plot the first two dimensions
plotExpSpace(expSpace)
# plot another slice
plotExpSpace(expSpace, y = "P_ann_tot_m", x = "P_Feb_tot_m")
plotOptions  

Plots the differences in performance metrics from two system options

Description

plotOptions uses the system model performances calculated using the function runSystemModel for two alternate system model options, and the summary of the simulation generated using the functions generateScenarios & getSimSummary as input. The function plots the differences in the performance metrics between the two options, and the changes in performance thresholds in the space. The user may specify the attributes to be used as the axes of the plot. The function contains arguments to control the finer details of the plot.

Usage

plotOptions(
  performanceOpt1,   
  performanceOpt2,   
  sim,              
  metric = NULL,    
  attX = NULL,      
  attY = NULL,      
  topReps = NULL,   
  opt1Label = "Option 1", 
  opt2Label = "Option 2", 
  titleText = paste0(opt2Label, " - ", opt1Label), 
  perfThresh = NULL, 
  perfThreshLabel = "Threshold", 
  attSlices = NULL,  
  climData = NULL,   
  colMap = NULL,     
  colLim = NULL
)

Arguments

performanceOpt1

a named list; contains the system model performance calculated using runSystemModel for system model option 1. If the list contains more than one performance metric, the argument metric can be used to specify the metric to be used.

performanceOpt2

a named list; contains the system model performance calculated using runSystemModel for system model option 2. If the list contains more than one performance metric, the argument metric can be used to specify the metric to be used.

sim

a list; summary of the simulation containing the scenarios generated using the function generateScenarios that is used to run the system model using runSystemModel. The summary of the simulation may be obtained by using the function getSimSummary.
The summary object is much smaller in size for ease of storage and use with the performance plotting functions like `plotPerformanceSpace`.

- **metric**: a string; the name of the performance metric to be plotted. The argument can be used to select the metric from `performanceOpt1` and `performanceOpt2` lists for plotting. If `NULL` (the default), the first metric in the lists will be used.

- **attX**: a string; the tag of the perturbed attribute to plot on the x-axis. The attribute must be one of the perturbed attributes of `sim`. Type `sim$expSpace$attPerturb` to view all perturbed attributes of `sim`. If `NULL` (default), the first perturbed attribute of `sim` will be used.

- **attY**: a string; the tag of the perturbed attribute to plot on the y-axis. The attribute must be another perturbed attribute of `sim`. If `NULL`, the second perturbed attribute of `sim` will be used.

- **topReps**: an integer (default is `NULL`); the number of "top" replicates in terms of simulation fitness to be used. If `topReps` is specified, `topReps` number of replicates will be identified for each target and the average performance across these replicates will be plotted. If `NULL`, the average performance across all the replicates will be plotted.

- **opt1Label**: a string; the text to label `performanceOpt1`.

- **opt2Label**: a string; the text to label `performanceOpt2`.

- **titleText**: a string; text for the title of the plot. The default is `paste0(opt2Label, " - ", opt1Label)`.

- **perfThresh**: a number; the minimum or maximum threshold value of the performance metric. A line will be drawn to mark this threshold value in the performance space.

- **perfThreshLabel**: a string; the text to label `perfThresh`.

- **attSlices**: a list; used to subset perturbed attributes in `sim` for the plot. This argument would typically be used in cases where there are more than two perturbed attributes. The elements of the list correspond to the perturbed attributes to be subsetted and must be named using the attribute tag. Each element may contain a single value or a two-element vector specifying the minimum-maximum values. If the element is a single value, the exposure space is sliced on this single value of the attribute. If minimum-maximum values are specified, the exposure space will be sliced to subset this range. If `attSlices` includes `attX` or `attY`, these attributes will be sliced and the resulting plot will be a "zoomed-in" space.

- **climData**: data.frame; the values of `attX` and `attY` from other sources like climate models. This data will be plotted as points in the performance space. The data frame may contain columns with values of the performance metric to be plotted and the "Name" of the dataset. If the performance metric is available in the data frame, the points will be coloured based on the performance `colMap` scale. If the `Name` of the data is available in the data frame, the points will be identified using the `Name`. Please refer data provided with the package that may be loaded using `data("egClimData")` for an example of the expected format of `climData`.

- **colMap**: a vector of colours; to specify the colourmap to be used. If `NULL`, the default `foreSIGHT` colurmap is used.

- **colLim**: a vector of 2 values; the minimum and maximum limits of the colour scale.
Value

The plot of the differences in the performance metrics (option 2 - option 1) in a ggplot object.

See Also

runSystemModel, plotPerformanceSpace, generateScenarios, getSimSummary

Examples

# load example datasets
data("egSimSummary")
data("egSimPerformance") # performance of option1
data("egSimPerformance_systemB") # performance of option2
data("egClimData")
plotOptions(egSimPerformance[1], egSimPerformance_systemB[1], egSimSummary,
attX = "P_ann_seasRatio_m", attY = "P_ann_tot_m", topReps = 7, perfThreshLabel = "Threshold (28L)",
perfThresh = 28, opt1Label = "System A", opt2Label = "System B", climData = egClimData)

plotPerformanceOAT

Displays performance for one-at-a-time (OAT) perturbations in attributes

Description

plotPerformanceOAT uses the system model performance calculated using the function runSystemModel and the summary of the simulation generated using the function generateScenarios & getSimSummary as input. The function creates line plots, each panel shows the variations in performance with perturbations in a single attribute. The function is intended for use with simulations with attributes perturbed on a one-at-a-time (OAT) grid.

Usage

plotPerformanceOAT(
  performance,
  sim,
  metric = NULL,
  topReps = NULL,
  col = NULL,
  ylim = NULL
)

Arguments

performance a named list; contains the system model performance calculated using runSystemModel. If the list contains more than one performance metric, the first metric will be plotted.

sim a list; a summary of a simulation containing the scenarios generated using the function generateScenarios that is used to run the system model using runSystemModel. The summary may be obtained using the function getSimSummary
metric  a string; the name of the performance metric to be plotted. The argument can be used to select a metric from performance for plotting.

topReps  an integer (default = NULL); the number of "top" replicates to be used. The "top" replicates will be identified for each target based on the simulation fitness. The average performance across topReps replicates will be plotted.

col  a colour; the colour of the lines. If NULL, the a default colour is used.

ylim  a vector of 2 values; the minimum and maximum limits of the y-axis (performance) scale.

Details

The plots show the mean value of performance across replicates. The ranges between the minimum and maximum values of performance across replicates are shaded. The function is intended for use with simulations containing attributes perturbed on an "OAT" grid. If the perturbations are on a "regGrid", this function will subset OAT perturbations, if available, to create the plots. The function creates separate plots for perturbations in attributes of temperature and other variables. The function may be called with performance argument specifying the metric to be plotted to plot other metrics.

Value

The plot of the performance space and the ggplot object.

See Also

runSystemModel, generateScenarios, plotPerformanceSpace, getSimSummary

Examples

# load example datasets
data("egSimSummary")
data("egSimPerformance")
plotPerformanceOAT(egSimPerformance[2], egSimSummary)
plotPerformanceOAT(egSimPerformance[1], egSimSummary)

# using the metric argument
plotPerformanceOAT(egSimPerformance, egSimSummary, metric = "Reliability (-)")

Description

plotPerformanceSpace uses the system model performance calculated using the function runSystemModel and the summary of the simulation generated using the functions generateScenarios & getSimSummary as input to plot the performance space of the system. The user may specify the attributes to be used as the axes of the performance space.
plotPerformanceSpace

Usage

plotPerformanceSpace(
  performance,
  sim,
  metric = NULL,
  attX = NULL,
  attY = NULL,
  topReps = NULL,
  perfThresh = NULL,
  perfThreshLabel = "Threshold",
  attSlices = NULL,
  climData = NULL,
  colMap = NULL,
  colLim = NULL
)

Arguments

performance a named list; contains the system model performance calculated using runSystemModel. If the list contains more than one performance metric, the argument metric can be used to specify the metric to be used.

sim a list; summary of the simulation containing the scenarios generated using the function generateScenarios that is used to run the system model using runSystemModel. The summary of the simulation may be obtained by using the function getSimSummary on the full simulation. The summary object is much smaller in size for ease of storage and use with the performance plotting functions like plotPerformanceSpace.

metric a string; the name of the performance metric to be plotted. The argument can be used to select a metric from performance for plotting. If NULL (the default), the first metric in the list will be used.

attX a string; the tag of the perturbed attribute to plot on the xaxis. The attribute must be one of the perturbed attributes of sim. Type sim$expSpace$attPerturb to view all perturbed attributes of sim. If NULL (default), the first perturbed attribute of sim will be used.

attY a string; the tag of the perturbed attribute to plot on the yaxis. The attribute must be another perturbed attribute of sim. If NULL, the second perturbed attribute of sim will be used.

topReps an integer (default is NULL); the number of "top" replicates in terms of simulation fitness to be used. If topReps is specified, topReps number of replicates will be identified for each target and the average performance across these replicates will be plotted. If NULL, the average performance across all the replicates will be plotted.

perfThresh a number; the minimum or maximum threshold value of the performance metric. A line will be drawn to mark this threshold value in the performance space.

perfThreshLabel a string; the text to label perfThresh.
**attSlices**  
a list; used to subset perturbed attributes in `sim` for the plot. This argument would typically be used in cases where there are more than two perturbed attributes. The elements of the list correspond to the perturbed attributes to be subsetted and must be named using the attribute tag. Each element may contain a single value or a two-element vector specifying the minimum-maximum values. If the element is a single value, the exposure space is sliced on this single value of the attribute. If minimum-maximum values are specified, the exposure space will be sliced to subset this range. If `attSlices` includes `attX` or `attY`, these attributes will be sliced and the resulting plot will be a "zoomed-in" space.

**climData**  
data.frame; the values of `attX` and `attY` from other sources like climate models. This data will be plotted as points in the performance space. The data frame may contain columns with values of the performance metric to be plotted and the "Name" of the dataset. If the performance metric is available in the data.frame, the points will be coloured based on the performance `colMap` scale. If the Name of the data is available in the data.frame, the points will be identified using the Name. Please refer data provided with the package that may be loaded using `data("egClimData")` for an example of the expected format of `climData`.

**colMap**  
a vector of colours; to specify the colourmap to be used. If NULL, the default `foreSIGHT` colourmap is used.

**colLim**  
a vector of 2 values; the minimum and maximum limits of the colour scale.

**Details**

If the space contains more than two perturbed attributes, the performance values are averaged across the perturbations in the attributes other than `attX` and `attY`. The user may specify argument `attSlices` to slice the performance space at specific values of the other perturbed attributes. If `attSlices` are used to specify minimum-maximum values to subset other perturbed attributes, the performance values are averaged across the subsetted perturbations in these attributes. If the input performance list contains multiple performance metrics, the function plots the first metric. The function may be called with performance argument specifying the metric to be plotted `plotPerformanceSpace(performance[2], sim)` to plot other metrics.

**Value**

The plot of the performance space and the ggplot object.

**See Also**

`runSystemModel`, `generateScenarios`, `getSimSummary`, `plotPerformanceOAT`

**Examples**

```r
# load example datasets
data("egSimSummary")  # summary of stochastic simulation
data("egSimPerformance")  # system performance calculated using the stochastic simulation
data("egClimData")  # alternate climate data and system performance

plotPerformanceSpace(egSimPerformance[2], egSimSummary)
```
plotPerformanceSpaceMulti

Plots contours of the number of performance thresholds exceeded in the perturbation space

Description

plotPerformanceSpaceMulti uses multiple system model performances calculated using the function runSystemModel and the summary of the simulation generated using the functions generateScenarios & getSimSummary as input to plot filled contours showing the number of performance thresholds exceeded in the perturbation space. The user may specify the attributes to be used as the axes of the perturbation space.

Usage

plotPerformanceSpaceMulti(
  performance,
  sim,
  perfThreshMin,
  perfThreshMax,
  attX = NULL,
  attY = NULL,
  attSlices = NULL,
  topReps = NULL,
  climData = NULL,
  col = NULL
)
plotPerformanceSpaceMulti

Arguments

- **performance**
a list; each element of the list should be a performance metric. May be calculated using the function `runSystemModel`.

- **sim**
a list; summary of the simulation containing the scenarios generated using the function `generateScenarios` that is used to run the system model using `runSystemModel`. The summary of the simulation may be obtained by using the function `getSimSummary` on the full simulation. The summary object is much smaller in size for ease of storage and use with the performance plotting functions like `plotPerformanceSpace`.

- **perfThreshMin**
a vector; the minimum threshold value of each performance metric. The length of the vector should be equal to `length(performance)`. If the metric does not have a minimum threshold, specify the corresponding element in `perfThreshMin` as NA.

- **perfThreshMax**
a vector; the maximum threshold value of each performance metric. The length of the vector should be equal to `length(performance)`. If the metric does not have a maximum threshold, specify the corresponding element in `perfThreshMax` as NA.

- **attX**
a string; the tag of the perturbed attribute to plot on the xaxis. The attribute must be one of the perturbed attributes of `sim`. Type `sim$expSpace$attPerturb` to view all perturbed attributes of `sim`. If NULL (default), the first perturbed attribute of `sim` will be used.

- **attY**
a string; the tag of the perturbed attribute to plot on the yaxis. The attribute must be another perturbed attribute of `sim`. If NULL, the second perturbed attribute of `sim` will be used.

- **attSlices**
a list; used to subset perturbed attributes in `sim` for the plot. This argument would typically be used in cases where there are more than two perturbed attributes. The elements of the list correspond to the perturbed attributes to be subsetted and must be named using the attribute tag. Each element may contain a single value or a two-element vector specifying the minimum-maximum values. If the element is a single value, the exposure space is sliced on this single value of the attribute. If minimum-maximum values are specified, the exposure space will be sliced to subset this range. If `attSlices` includes `attX` or `attY`, these attributes will be sliced and the resulting plot will be a "zoomed-in" space.

- **topReps**
an integer (default is NULL); the number of "top" replicates in terms of simulation fitness to be used. If `topReps` is specified, `topReps` number of replicates will be identified for each target and the average performance across these replicates will be plotted. If NULL, the average performance across all the replicates will be plotted.

- **climData**
data.frame; the values of `attX` and `attY` from other sources like climate models. This data will be plotted as points in the perturbation space. If the Name of the data is available in the data.frame, the points will be identified using the Name. Please refer data provided with the package that may be loaded using `data("egClimData")` for an example of the expected format of `climData`.

- **col**
a vector of colours; The length of the vector should at least be sufficient to assign unique colours to all the different values in the generated plot. If NULL, the default foreSIGHT colours is used.
plotScenarios

Details

If the space contains more than two perturbed attributes, the performance values are averaged across the perturbations in the attributes other than attX and attY. The user may specify argument attSlices to slice the performance space at specific values of the other perturbed attributes. If attSlices are used to specify minimum-maximum values to subset other perturbed attributes, the performance values are averaged across the subsetted perturbations in these attributes. This function cannot be used with sim perturbed on an "OAT" grid since contours of the number of performance thresholds exceeded cannot be calculated for an irregular perturbation space.

Value

The plot showing the number of thresholds exceeded and the ggplot object.

See Also

runSystemModel, generateScenarios, getSimSummary, plotPerformanceSpace

Examples

# load example datasets
data("egSimPerformance")
data("egSimSummary")
data("egClimData")

plotPerformanceSpaceMulti(egSimPerformance, egSimSummary,
perfThreshMin = c(NA, 0.80), perfThreshMax = c(30, NA))

# add alternate climate data and specify different colours for the plot
plotPerformanceSpaceMulti(egSimPerformance, egSimSummary, perfThreshMin = c(NA, 0.80),
perfThreshMax = c(30, NA), climData = egClimData, col = viridisLite::magma(3))

# example using simple scaled simulations

data("egScalPerformance")
data("egScalSummary")
data("egClimData")

plotPerformanceSpaceMulti(egScalPerformance, egScalSummary, perfThreshMin = c(NA, 0.80),
perfThreshMax = c(30, NA), climData = egClimData)

plotScenarios

Creates summary plots of the biases in the scenarios

Description

plotScenarios uses a simulation performed using the function generateScenarios as input and creates heatmaps that show the biases in the simulated attributes with respect to the specified target values of the attributes. The plots show the magnitude (absolute value) of the mean biases, and the standard deviation of biases across replicates. The heatmaps can be used to evaluate how well the simulated attributes match the specified targets. The biases are in units of percentage for attributes
of variables like precipitation, and in units of degrees K for attributes of temperature. The function
creates two heatmaps that show:

- magnitude of the mean biases across all the replicates
- standard deviation of biases across all the replicates

Usage

```r
plotScenarios(
  sim,
  simName = NULL,
  writeToFile = FALSE,
  fileName = "plotScenarios.pdf",
  colMapRange = "default"
)
```

Arguments

- `sim`: a list; contains a stochastic simulation or the summary of a stochastic simulation
  created using the function `generateScenarios`.
- `simName`: a string; defaults to `NULL`. User-specified name of the simulation that will be used
  as the heading in the saved pdf file to identify the simulation later. If `simName` is `NULL`, a random
  name will be assigned for the simulation.
- `writeToFile`: logical; defaults to `FALSE`. Specifies whether the plots should be saved to a pdf
  file. If set to true, the heatmaps will be saved to a pdf file that would also contain
  summary pages that show the attributes, models, and optimisation settings used
  to create `sim`.
- `fileName`: a string; defaults to "plotScenarios.pdf". Specifies the name of the pdf file
  to be written, if the file exists it will be overwritten.
- `colMapRange`: a string; may be set to "default" or "full". The argument specifies whether
  the colormap of the heatmap should span the full range of the data. If set to
  "default", the colourmap limits of attributes that are in units of percentage is
  set to 0% to 10%, and the colourmap limits of the attributes of temperature is
  set to 0 degrees K to 1 degrees K. If set to "full", the colourmap limits are set
  to the minimum and maximum values in the data.

Details

The argument `sim` may be a full stochastic simulation generated using the function `generateScenarios`
or the summary of the stochastic simulation generated using `getSimSummary`.

Value

The function returns two R plots showing the biases in the targets of the scenarios generated
using the function `generateScenarios`. The figures may be saved to a pdf file by setting the
`writeToFile` argument to `TRUE`. 
## runSystemModel

**runSystemModel**

*Runs a system model and outputs the system performance*

---

**Description**

`runSystemModel` runs a system model and outputs the system performance. It takes time series of hydroclimatic variables generated using the function `generateScenarios` as input to a `systemModel` and collates the system performance for all the targets and replicates in the scenarios.

**Usage**

```r
runSystemModel(sim, systemModel, systemArgs, metrics)
```

### Examples

```r
## Not run:
# the examples are not run since the run times are too long for CRAN
# create an exposure space
attPerturb <- c("P_ann_tot_m", "P_ann_nWet_m", "P_ann_R10_m")
attHold <- c("P_Feb_tot_m", "P_SON_dWet_m", "P_JJA_avgWSD_m", "P_MAM_tot_m",
            "P_DJF_avgDSD_m", "Temp_ann_rng_m", "Temp_ann_avg_m")
attPerturbType = "regGrid"
attPerturbSamp = c(2, 1, 1)
attPerturbMin = c(0.9, 1, 1)
attPerturbMax = c(1.1, 1, 1)
expSpace <- createExpSpace(attPerturb = attPerturb, 
                           attPerturbType = attPerturbType, 
                           attPerturbSamp = attPerturbSamp, 
                           attPerturbMin = attPerturbMin, 
                           attPerturbMax = attPerturbMax, 
                           attHold = attHold, 
                           attTargetsFile = NULL)

# load example data available in foreSIGHT
data(tankDat)
# perform stochastic simulation
sim <- generateScenarios(reference = tank_obs, 
                          expSpace = expSpace, 
                          simLengthNyrs = 30, 
                          numReplicates = 2)

# plots heatmaps showing biases in simulated targets
plotScenarios(sim)
# to save the figures to a pdf file set writeToFile = TRUE
# using an example stochastic simulation summary provided with the package
data("egSimSummary")
plotScenarios(egSimSummary)

## End(Not run)
```

---

**See Also**

- `createExpSpace`
- `generateScenarios`
- `getSimSummary`

---

**Description**

`runSystemModel` uses time series of hydroclimatic variables generated using the function `generateScenarios` as input to a `systemModel` and collates the system performance for all the targets and replicates in the scenarios.

**Usage**

```r
runSystemModel(sim, systemModel, systemArgs, metrics)
```
Arguments

**sim**
- list; a simulation containing the scenarios generated using the function `generateScenarios`.

**systemModel**
- a function; The function runs the system model using climate data in a `data.frame` as input. The function is expected to be created by the user for specific system models. `tankWrapper` is an example system model function available in this package. `runSystemModel` calls the function `systemModel` with two arguments:
  - `data`: `data.frame`; the climate data in a `data.frame` with columns named `year`, `month`, `day`, `variable_name1`, `variable_name2`.
  - `systemArgs`: list; containing the other arguments required by the system model. `systemModel` unpacks the arguments from the list and uses them as required.
  - `metrics`: string vector; containing the names of the performance metrics that the system model returns. It is recommended that the names also contain the units of the metric. See `viewTankMetrics()` for examples.

**systemArgs**
- a list; containing the input arguments to `systemModel`.

**metrics**
- a string vector; the names of the performance metrics the `systemModel` function returns.

Details

The `runSystemModel` function code is structured to be simple and may be used as an example to create scripts that use scenarios generated using `generateScenarios` to run system models in other programming languages. Type `runSystemModel` to view the function code. The function `tankWrapper` in this package may be used as an example to create user defined functions for the `systemModel` argument. Refer to `tankWrapper` to understand how the `systemModel` is expected to use `systemArgs` and return the calculated performance metrics. The `systemModel` function is expected to return a named list of performance metrics. The elements of the vector should correspond to `metrics`.

Value

The function returns a list containing the performance metrics calculated by the `systemModel`. Each element of the list corresponds to a performance metric and is named using the `metrics` argument. Each element contains performance values calculated at all the target points in the exposure space in a matrix with `nrow` corresponding to the targets and `ncol` corresponding to the replicates.

See Also

`tankWrapper`, `generateScenarios`

Examples

```r
# Example using tankWrapper as the systemModel
# ==============================================================
## Not run:
# create an exposure space
attPerturb <- c("P_ann_tot_m", "P_ann_nWet_m")
attHold <- c("P_Feb_tot_m", "P_SON_dyWet_m", "P_JJA_avgWSD_m", "P_MAM_tot_m",
```
tankPerformance

A function to calculate difference performance from simulated tank behaviour

Description

A function to calculate difference performance from simulated tank behaviour

Usage

tankPerformance(data=NULL,  
roofArea=50,  
nPeople=1,  
firstFlush=1,  
tankVol=3000,  
write.file=TRUE,  
fname="tankperformance.csv")
Arguments

data    A dataframe of observed climate data in the form Year Month Day P Temp.
roofArea    roof area in m2
nPeople    number of people using water
tankVol    tank volume in L
firstFlush    first flush depth over roof in mm
write.file    logical. write output tank timeseries to file T/F?
fnam    string indicating name of file

Description

tankWrapper is a wrapper function for a rainwater tank system model in foreSIGHT. This function is used in examples in function help files and vignettes. This function may also be used as an example to create wrapper functions for other system models with scenarios generated using foreSIGHT in R or other programming languages.

Usage

tankWrapper(data, systemArgs, metrics)

Arguments

data    data.frame; contains observed daily precipitation and temperature to be used to run the rain water tank system model in a data.frame with columns named year month day P Temp. Note that the first three columns of the data.frame contain the year, month, and day of observation. The columns have to be named as specified. Please refer data provided with the package that may be loaded using data(tankDat) for an example of the expected format of data.
systemArgs    a list; contains the input arguments to the rain water tank system model. The valid fields in the list are:
    • roofArea : numeric; the roof area in sq.m
    • nPeople : integer; number of people using water
    • tankVol : numeric; volume of the tank in L
    • firstFlush : numeric; first flush depth over roof in mm
    • write.file : logical; indicates whether output is to be written to file
    • fnam : string; name of the output file
metrics    string vector; the metrics of performance of the system model to be reported. The valid strings may be viewed using the function viewTankMetrics()
Value

The function returns a list containing the calculated values of the performance metrics specified in
metrics after running the system model.

See Also

runSystemModel, viewTankMetrics

Examples

# view available performance metrics
viewTankMetrics()
# load example climate data to run the system model
data(tankDat)
systemArgs <- list(roofArea = 205, nPeople = 1, tankVol = 2400,
firstFlush = 2.0, write.file = FALSE)
tankWrapper(tank_obs, systemArgs,
metrics = c("average daily deficit (L)", "reliability (fraction)"))

------------------------
tank_obs Observations for demo tank model examples and vignette
------------------------

Description

Dataset of observations for tank model examples

Format

A dataframe of observed climate data in the form Year Month Day P Temp.

------------------------
viewAttributeDef Prints the definition of an attribute
------------------------

Description

viewAttributeDef prints the short definition of a valid attribute

Usage

viewAttributeDef(attribute)

Arguments

attribute A string: the name of the attribute.
See Also

viewAttributes, createExpSpace

Examples

# To view the definition of any valid attribute
viewAttributeDef("P_ann_tot_m")
# To view the valid attributes
viewAttributes()

viewAttributes                  Prints the names of valid attributes

Description

viewAttributes() prints the names of valid attributes that may be used to create an exposure space.

Usage

viewAttributes()

Details

The function does not take any input arguments. The valid attributes that may be specified as attPerturb or attHold to create an exposure space using the function createExpSpace are printed.

See Also

createExpSpace, viewAttributeDef

Examples

# To view the valid attributes.
viewAttributes()

# To view the definition of any valid attribute
viewAttributeDef("P_ann_tot_m")
viewDefaultOptimArgs  \hspace{1cm} \textit{Prints the default optimisation arguments}

\textbf{Description}

\begin{quote}
\texttt{viewDefaultOptimArgs()} prints the default values of optimisation arguments (\texttt{optimisationArguments}) used by \texttt{generateScenarios}.
\end{quote}

\textbf{Usage}

\begin{quote}
\texttt{viewDefaultOptimArgs()}
\end{quote}

\textbf{Details}

The function does not take any input arguments. This a helper function that prints the default values of the optimisation arguments. The user may specify alternate values of these arguments in fields named according to the corresponding argument name nested under \texttt{optimisationArguments} in a JSON file to use as the \texttt{controlFile} input to the \texttt{generateScenarios} function.

\textbf{See Also}

\begin{quote}
\texttt{writeControlFile}
\end{quote}

\textbf{Examples}

\begin{quote}
# To view the default optimisation arguments
\texttt{viewDefaultOptimArgs()}
\end{quote}

\begin{table}
\centering
\begin{tabular}{l l}
\hline
\textbf{viewModelParameters} & \textit{Prints the names and bounds of the parameters of the stochastic models} \\
\hline
\end{tabular}
\end{table}

\textbf{Description}

\begin{quote}
\texttt{viewModelParameters()} prints the names of the parameters of the stochastic model and its default minimum and maximum bounds. The stochastic model is specified using the function arguments.
\end{quote}

\textbf{Usage}

\begin{quote}
\texttt{viewModelParameters(variable, modelType, modelParameterVariation)}
\end{quote}

\textbf{Arguments}

\begin{itemize}
\item \texttt{variable} \hspace{1cm} A string: the name of the variable. Type \texttt{viewModels()} to view valid variable names.
\item \texttt{modelType} \hspace{1cm} A string: the model type. Use \texttt{viewModels} to view the valid values.
\item \texttt{modelParameterVariation} \hspace{1cm} A string: the parameter variation. Use \texttt{viewModels} to view the valid values.
\end{itemize}
The available stochastic models can be viewed using the function viewModels(). This function prints the default ranges of the parameters of the stochastic model specified the stochastic model of interest.

**See Also**

viewModels, writeControlFile

**Examples**

```r
viewModelParameters("P", "wgen", "annual")
viewModelParameters("P", "wgen", "harmonic")
```

---

**Description**

viewModels prints the stochastic model options available for the different hydroclimatic variables in foreSIGHT. These options may be used to create an controlFile for input to function generateScenarios.

**Usage**

```r
viewModels(variable = NULL, compatibleAtts = FALSE)
```

**Arguments**

- `variable` String; the variable name. Type `viewModels()` without arguments to view the valid variable names.
- `compatibleAtts` TRUE/FALSE (default is FALSE). Whether the attributes compatible with each model should be printed.

**See Also**

writeControlFile, generateScenarios

**Examples**

```r
# To view the valid variable names use the function without arguments
viewModels()

# Examples to view the model options available for different variables
viewModels("P")
viewModels("Temp")
viewModels("Radn")
viewModels("PET")
```
**viewTankMetrics**

Prints the names of the performance metrics of the rain water tank system model

**Description**

viewTankMetrics prints the names of the performance metrics available in the example rain water tank system model. T

**Usage**

viewTankMetrics()

**Details**

This is a helper function that does not take any input arguments. The user may specify one or more of the metric names as the metric argument of tankWrapper to select the performance metrics from the tank system model to select the performance metrics.

**See Also**

tankWrapper

**Examples**

viewTankMetrics()

---

**viewVariables**

Prints the names of and units of valid variables

**Description**

viewVariables() prints the names of valid variables and their units in the package. The user should input these variable in the same units.

**Usage**

viewVariables()

**Details**

The function does not take any input arguments.

**See Also**

generateScenarios
Examples

```r
# To view the valid variables
viewVariables()
```

---

**Description**

`writeControlFile()` writes a sample `controlFile.json` file. The `controlFile.json` file is used to specify alternate model and optimisation options and used as an input to the function `generateScenarios`. The user may use the sample file created by this function as a guide to create an "controlFile.json" file for their application.

**Usage**

```r
writeControlFile(
  jsonfile = "sample_controlFile.json",
  basic = TRUE,
  nml = NULL
)
```

**Arguments**

- `jsonfile` string; to specify the name of the json file to be written. The default name of the sample file is "sample_controlFile.json". The file will be written to the working directory of the user.

- `basic` logical (TRUE/FALSE); used to specify whether a "basic" or "advanced" sample file is to be written. The default is TRUE. A "basic" controlFile does not contain `modelParameterBounds`, and is sufficient for most applications.

- `nml` list; the namelist to be written to the json file, as an R list. This argument may be used to create a JSON file using an controlFile from an existing simulation. If this argument is set to NULL, the function writes the default model/optimisation options defined in the package to the json file.

**Details**

The function may be used without any input arguments to write a "basic" sample controlFile.

**Value**

A json file. The file may be used as an example to create an "controlFile.json" file for input to `generateScenarios`. An "controlFile.json" file may contain any subset of the fields listed below. The user may delete the unused fields from the file. The exception cases where it is mandatory to specify two fields together in `controlFile` are detailed as part of the list below.
writeControlFile

- **modelType**: a list by variable. Each element of the list is a string specifying the type of stochastic model. If `modelType` is specified for a variable in `controlFile`, `modelParameterVariation` should also be specified. This is because these two fields together define the stochastic model. Use `viewModels()` to view the valid options for `modelType` by variable.

- **modelParameterVariation**: a list by variable. Each element of the list is a string specifying the type of the parameter variation (annual, seasonal, harmonic etc.) of the stochastic model. If `modelParameterVariation` is specified for a variable in `controlFile`, `modelType` should also be specified. This is because these two fields together define the stochastic model. Use `viewModels()` to view valid options for `modelParameterVariation` by variable.

- **modelParameterBounds**: a nested list by variable. Each element is a list containing the bounds of the parameters of the chosen stochastic model. This field exists to provide an option to overwrite the default bounds of the parameters of the stochastic model. Careful consideration is recommended prior to setting `modelParameterBounds` in the `controlFile` to overwrite the defaults provided in the package.

- **optimisationArguments**: a list. Contains the optimisation options used by function `ga` from the `ga` package. Brief definitions are given below.
  - `pcrossover` a value of probability of crossover. Defaults to 0.8.
  - `pmutation` a value of probability of mutation. Defaults to 0.1.
  - `maxiter` a value of the maximum number of generations. Defaults to 50.
  - `maxFitness` a value of the stopping criteria. Defaults to -0.001.
  - `popSize` a value of the population size. Defaults to 500.
  - `run` a value of an alternative stopping criteria, consecutive runs without improvement in fitness. Defaults to 20.
  - `seed` a value of the random seed. Defaults to NULL.
  - `parallel` specifies if parallel computing should be used. Defaults to False. Can be set to the number of desired cores, or `TRUE`, where it will detect the number of available cores and run.
  - `keepBest` specifies if the optimisation should keep the best solution in each generation. Defaults to `TRUE`.
  - `suggestions` suggestions for starting values of parameters for optimisation.

- **penaltyAttributes**: a character vector of climate attributes to place specific focus on during targeting via the use of a penalty function during the optimisation process. The `penaltyAttributes` should belong to `attPerturb` or `attHold` that are specified in the exposure space used as input to `generateScenarios`. If `penaltyAttributes` are specified in the `controlFile`, `penaltyWeights` should also be specified.

- **penaltyWeights**: a numeric vector; the length of the vector should be equal to the length of `penaltyAttributes`. `penaltyWeights` are the multipliers of the corresponding `penaltyAttributes` used during the optimisation.

See Also

generateScenarios, viewModels, viewDefaultOptimArgs

Examples

```r
## Not run:
```
# To write a sample controlFile
writeControlFile()

# To write an advanced sample controlFile
writeControlFile(jsonfile = "sample_controlFile_advanced.json", basic = FALSE)

## End(Not run)
Index

* datasets
  climdata, 4
egClimData, 7
egScalPerformance, 7
egScalSummary, 8
egSimOATPerformance, 8
egSimOATS Summary, 9
egSimPerformance, 9
egSimPerformance_systemB, 10
egSimSummary, 10
tank_obs, 33

* functions
  modCalibrator, 15
  modSimulator, 16
  calculateAttributes, 3
climdata, 4
createExpSpace, 4
egClimData, 7
egScalPerformance, 7
egScalSummary, 8
egSimOATPerformance, 8
egSimOATS Summary, 9
egSimPerformance, 9
egSimPerformance_systemB, 10
egSimSummary, 10
generateScenario, 11
generateScenarios, 12
getSimSummary, 14

modCalibrator, 15
modSimulator, 16

plotExpSpace, 17
plotOptions, 19
plotPerformanceOAT, 21
plotPerformanceSpace, 22
plotPerformanceSpaceMulti, 25
plotScenarios, 27
runSystemModel, 29
tank_obs, 33
tankPerformance, 31
tankWrapper, 32
viewAttributeDef, 33
viewAttributes, 34
viewDefaultOptimArgs, 35
viewModelParameters, 35
viewModels, 36
viewTankMetrics, 37
viewVariables, 37
writeControlFile, 38