Package 'wxgenR'

April 14, 2025

Type Package

Title A Stochastic Weather Generator with Seasonality

Version 1.4.4

Description A weather generator to simulate precipitation and temperature for regions with seasonality. Users input training data containing precipitation, temperature, and seasonality (up to 26 seasons). Including weather season as a training variable allows users to explore the effects of potential changes in season duration as well as average start- and end-time dates due to phenomena like climate change. Data for training should be a single time series but can originate from station data, basin averages, grid cells, etc.
Bearup, L., Gangopadhyay, S., & Mikkelson, K. (2021). ``Hydroclimate Analysis Lower Santa Cruz River Basin Study (Technical Memorandum No ENV-2020-056)." Bureau of Reclamation.
Gangopadhyay, S., Bearup, L. A., Verdin, A., Pruitt, T., Halper, E., & Shamir, E. (2019, December 1). ``A collaborative stochastic weather generator for climate impacts assessment in the Lower Santa Cruz River Basin, Arizona." Fall Meeting 2019, American Geophysical Union. <https://ui.adsabs.harvard.edu/abs/2019AGUFMGC41G1267G>.

License CC0

Copyright This software is in the public domain because it contains materials that originally came from the United States Bureau of Reclamation, an agency of the United States Department of Interior.

Encoding UTF-8

LazyData true

LazyDataCompression xz

RoxygenNote 7.3.2

Imports lubridate, msm, sm, dplyr, plyr, foreach, parallel, doParallel, stats, utils, magrittr, doRNG, mc2d, qmap, tidyr

Suggests knitr, rmarkdown, reshape2, ggpubr, moments, seas, padr, hydroGOF, SpecsVerification, ggridges, tigris, terra, stringr, mapview, xts, sf, lfstat, FedData, ggplot2

VignetteBuilder knitr

NeedsCompilation no

Depends R (>= 3.5.0)

Author Subhrendu Gangopadhyay [aut],
Lindsay Bearup [aut],
David Woodson [aut, cre],
Marketa McGuire [aut],
Andrew Verdin [aut],
Eylon Shamir [aut],
Eve Halper [aut]

Maintainer David Woodson <dwoodson@usbr.gov>

Repository CRAN

Date/Publication 2025-04-14 19:50:01 UTC

Contents

`BlacksburgVA` *Example meteorological training data for weather generator*

Description

Weather data (precipitation, temperature, and season) measured at the NWS station (GHCND:USC00440766) in Blacksburg, Virginia.

Usage

```
data(BlacksburgVA)
```

Format

A data frame.

Source

Blacksburg, VA NWS office

Examples

```
data(BlacksburgVA)
```

`BOCO_sims` *wxgenR simulation results for multiple meteorological stations in Boulder County, CO*

Description

Simulated precipitation and temperature at the following GHCN stations (data retrieved using the FedData R package): "USC00050848" "USS0005J42S"

Usage

```
data(BOCO_sims)
```

Format

A list of dataframes.

Source

GHCNd

Examples

```
data(BOCO_sims)
```

`generate_TmaxTmin` *Generate maximum and minimum daily temperature from daily average temperature*

Description

Estimate maximum and minimum temperature from ‘wx‘ simulation results of daily average temperature. This function uses quantile mapping to develop a relationship between daily average temperature and daily maximum temperature. Average and maximum temperature can be contained in your training data for example. Then, the relationship between average and maximum temperature is applied to the simulated daily average temperature output by ‘wx‘ to estimate a simulated daily maximum temperature. Daily minimum temperature is then estimated using the standard equation Tavg = 0.5(Tmax + Tmin).

Quantile mapping is trained and applied separately by calendar month. The quantile mapping function comes from the ‘qmap‘ R package, specifically using empirical quantiles.

You will likely need to post-process the ‘wx‘ function output in order to process each simulation trace since the ‘wx‘ output is in wide format by trace and this function is best applied in long format. Code examples for processing both the training data and simulated data to put into suitable format for the ‘generate_TmaxTmin‘ function are available here:

https://github.com/dwoodson-usbr/wxgenR/tree/master/vignettes

Usage

```
generate_TmaxTmin(df.train, df.sim)
```

Arguments

`df.train`	A dataframe containing the daily training data used to develop the relationship between average and maximum temperature. At a minimum should contain variables named ‘tmax‘, ‘temp‘, and ‘month‘ which represent maximum temperature, average temperature, and numeric month in which the daily observation exists, respectively.
`df.sim`	A dataframe containing the daily simulation results from the ‘wx‘ function in long format. At a minimum should contain variables named ‘sim_temp‘ and ‘month‘ which respectively represent the simulated daily average temperature from ‘wx‘ and the numeric month in which each data point resides.

Value

Returns a list containing results and metadata from the multisite shuffling in ’long’ format for easy analysis and visualization.

- df.sim - Dataframe containing simulated maximum and minimum temperature generated from daily average temperature.
- qmap.monthly - List of dataframes containing fitted quantile mapping models for each calendar month.

References

Gudmundsson L (2025). qmap: Statistical transformations for post-processing climate model output. R package version 1.0-6.

Gudmundsson L, Bremnes JB, Haugen JE, Engen-Skaugen T (2012). "Technical Note: Downscaling RCM precipitation to the station scale using statistical transformations - a comparison of methods." Hydrology and Earth System Sciences, 16(9), 3383–3390. doi:10.5194/hess-16-3383-2012.

Examples

```
# Example with toy data
df.train = data.frame(
  temp = runif(10, 50, 70),
  tmax = runif(10, 60, 80),
  month = rep(1:2, each = 5)
)

df.sim = data.frame(
  sim_temp = runif(10, 50, 70),
  month = rep(1:2, each = 5)
)

result = generate_TmaxTmin(df.train, df.sim)
head(result$df.sim)
```

---

getDatesInWindow *Get dates in window*

Description

Find grouping of dates around each Julian day of year (1-366) based on the window you set. The start and end years for this function should include at least one leap year (i.e., the record should be at least 4-years in length), or else the function will return non-existing dates (February 29th during non-leap years).

Setting leapflag to true will set February 29th as NA for non-leap years.

Setting leapflag to false will remove February 29th for non-leap years (recommended).

The 'wwidth' variable is the semi-bandwidth that sets the window size to search for adjacent days. Given a value of 'wwidth', the window size will be 2*wwidth + 1. For example a 'wwidth' of 7 would give a window size of 2*7+1 = 15.

Other applications of this function might include a daily bias correction approach where it is necessary to find N adjacent days for each day of year in order to train the bias correction algorithm.

Usage

```
getDatesInWindow(syr, eyr, smo, emo, sdate, edate, wwidth, leapflag = FALSE)
```

Arguments

syr	Start year.
eyr	End year.
smo	Start month.
emo	End month.
sdate	Start date.
edate	End date.
wwidth	Window set for finding surrounding days (semi-bandwidth).
leapflag	Set index for leap years (default = F).

Value

Returns a matrix with 366 rows (one for each Julian day of year, including leap days) and nCols; where nCols = (2 x wwidth + 1) x (eyr - syr + 1). Each row is specific to a certain Julian day (e.g., day 1) and contains the preceding and antecedent dates around that Julian day based on the window length you set. The dates will be fetched for each year in the range you set between the start and ending years (inclusive of the start and end years). Matrix values are either dates formatted as 'yyyymmdd' or NA values.

Examples

```
getDatesInWindow(syr = 2000, eyr = 2005, smo = 10, emo = 09,
 sdate = 20001001, edate = 20050930, wwidth = 3, leapflag = FALSE)
```

LowerSantaCruzRiverBasinAZ

Example meteorological training data for weather generator

Description

Weather data (precipitation, temperature, and season) for the Lower Santa Cruz River Basin in Southern Arizona. Dataset was developed for the Hydroclimate Analysis within Reclamation's Lower Santa Cruz River Basin Study.

Usage

```
data(LowerSantaCruzRiverBasinAZ)
```

Format

A data frame

Source

Hydroclimate Analysis - Lower Santa Cruz River Basin Study

Examples

```
data(LowerSantaCruzRiverBasinAZ)
```

`multisite_shuffle` *Multisite shuffling of wxgenR simulation results*

Description

Runs a postprocessor on 'wx' simulation results to shuffle multiple stations' simulations such that wxgenR can be used in multisite applications. Specifically, the 'multisite_shuffle' function uses an approach developed by Iman and Conover (1982) and later applied by Clark et al. (2004) to capture the rank correlation among observed station data and introduce it to those stations' simulations. The Iman and Conover approach is implemented using the 'cornode' function from the 'mc2d' package.

Usage

```
multisite_shuffle(list.sta.wx, numbCores = NULL, aseed = NULL)
```

Arguments

`list.sta.wx`	A list containing multiple stations' observed and simulated data to be used in multisite shuffling. Each list element should be named for the station's data it holds and should contain the dataframe output from the 'wx' for that station.
`numbCores`	Enable parallel computing for multisite shuffling, set number of cores to enable (must be a positive integer greater than or equal to 2). Turned off by default; if set to 0 or 1 it will run as single thread. Use function 'detectCores()' from 'parallel' package to show the number of available cores on your machine.
`aseed`	Specify a seed for reproducibility.

Value

Returns a list containing results and metadata from the multisite shuffling in 'long' format for easy analysis and visualization.

- shuffledResultsOnly - Dataframe containing shuffled simulations for all traces and stations ('sim_prcp' and 'sim_temp').
- shuffledResultsAndObs - Dataframe containing shuffled simulations as well as corresponding observations/training data ('prcp' and 'temp' are the observed data).
- shuffledAndUnshuffled - Dataframe containing shuffled simulations, unshuffled simulations, observations. Unshuffled vs and shuffled are indicated by the 'Tag' variable.

References

Iman, Ronald & Conover, William. (1982). A Distribution-Free Approach to Inducing Rank Correlation Among Input Variates. Communications in Statistics-simulation and Computation - COMMUN STATIST-SIMULAT COMPUT. 11. 311-334. 10.1080/03610918208812265.

Clark, M., Gangopadhyay, S., Hay, L., Rajagopalan, B., & Wilby, R. (2004). The Schaake Shuffle: A Method for Reconstructing Space–Time Variability in Forecasted Precipitation and Temperature Fields. Journal of Hydrometeorology, 5(1), 243-262. https://doi.org/10.1175/1525-7541(2004)005<0243:TSSAMF>2.0.CO;

R. Pouillot, M.-L. Delignette-Muller (2010), Evaluating variability and uncertainty in microbial quantitative risk assessment using two R packages. International Journal of Food Microbiology. 142(3):330-40

Examples

```
# Simulated example with two stations

data(BOCO_sims)

ms = multisite_shuffle(BOCO_sims, numbCores = 2, aseed = 123)

print(ms)
```

`repan` *Random variates from the Epanechnikov kernel*

Description

Simulate outside the historical envelope using randomly generated values from the Epanechnikov kernel (via acceptance-rejection sampling).

For more details on the Epanechnikov kernel and its use in a weather generator, see Rajagopalan et al. (1996).

Usage

```
repan(nsim)
```

Arguments

`nsim`	Number of simulations.

Value

Returns a vector of random variates sampled from the Epanechnikov kernel. ‘nsim‘ number of samples are returned.

References

Rajagopalan, B., Lall, U., & Tarboton, D. G. (1996). Nonhomogeneous Markov Model for Daily Precipitation. Journal of Hydrologic Engineering, 1(1), 33–40. https://doi.org/10.1061/(ASCE)1084-0699(1996)1:1(33)

Examples

```
repan(nsim = 10)

#simulate and plot density and distribution function
oldpar = par(mfrow=c(1,3), mar=c(2,2.5,2,1),
             oma=c(2,2,0,0), mgp=c(2,1,0), cex.axis=0.8)

par(mfrow=c(1,2))
nsim=1e5
x <- sort(repan(nsim));y=0.75*(1-x^2)
plot(x,y,xlab="x",ylab="f(x)",type="l",lwd=2)
grid()
title (main="Epanechnikov PDF",cex.main=0.8)
F=rank(x)/(nsim+1)
plot(x,F,ylab="F(x)",type="l",lwd=2)
grid()
title (main="Epanechnikov CDF",cex.main=0.8)

dev.off()

par(oldpar)
```

---

selectState *Select transition state*

Description

Function selects and returns the transition state given a uniform random number between 0 and 1 and the cumulative probability vector of the state sequence.

Usage

```
selectState(uni, wt)
```

Arguments

uni	Uniform random number between 0 and 1.
wt	Cumulative probability vector of states.

Value

Returns an object containing the transition state(s) based on the given cumulative probability vector and random numbers.

Examples

```
rand = runif(1)

print(rand)

selectState(uni = rand, wt = c(0.25, .55, 0.85, 1))
```

`spellLengths` *Spell length calculation*

Description

Function to calculate the length (duration in years) of wet or dry periods.

Usage

```
spellLengths(s)
```

Arguments

`s`	A binary vector of 0 dry and 1 wet only.

Value

Returns a list object containing a vector of dry spell lengths and a vector of wet spell lengths.

Examples

```
#use 0 for dry and 1 for wet years
spells = c(0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0)

spellLengths(spells)
```

`stationData` *Meteorological training data for weather generator from nine stations*

Description

* Weather data (precipitation, temperature) from nine stations within the Global Historical Climatology Network (GHCN): + USC00162534 (DONALDSONVILLE 4 SW LA, US) + USC00050372 (ASPEN 1 SW CO, US) + USC00080236 (ARCHBOLD BIO STATION FL, US) + USC00440766 (BLACKSBURG NWS VA, US) + USW00014606 (BANGOR INTL. AIRPORT ME, US) + USW00094240 (QUILLAYUTE AIRPORT WA, US) + USC00346386 (NORMAN 3 SSE OK, US) + USC00028795 (TUCSON 17 NW AZ, US) + USC00472314 (EAGLE RIVER WI, US)

Usage

```
data(stationData)
```

Format

A data frame.

Source

Global Historical Climatology Network

Examples

```
data(stationData)
```

`writeSim` *Write simulations to file*

Description

Write simulation results to .csv files (one .csv file is generated for each trace). Inputs include the weather simulations stored in the list object output from the 'wx()' function as well as the 'nsim' and 'nrealz' variables that were inputs to the 'wx()' function.

A debug flag allows for more detailed reports (debug = TRUE), but setting 'debug = FALSE' is generally recommended for more concise output. Keeping 'debug = FALSE' will also include a simulation time stamp (year, month, day) beginning in year 1.

This function will write the .csv files to your working directory.

Leap years may be included in the simulated weather if they are included in your training data, so non-leap years include a row of 'NA' values at the end of the calendar year as a book-keeping measure so that the total number of rows in each trace is the same.

Usage

```
writeSim(wxOutput, nsim, nrealz, path = NULL, debug = FALSE)
```

Arguments

`wxOutput`	Weather simulations output from 'wx()' function.
`nsim`	Number of simulation years.
`nrealz`	Number of realizations (ensemble size).
`path`	Specified path to where simulation output shall be written. Defaults to current working directory (path = NULL). Specified path should be a character string of the folder location ending with '/'.

debug — Option to include additional variables in the .csv file outputs for debugging and advanced analysis. Includes sampling date, etc. Default = FALSE (off). If debug is off, the weather simulations will have a simulation year time stamp (beginning in year 1) as well as month and day time stamps.

Value

No return value, called to write simulation results to file.

Examples

```
z = wx(trainingData = LowerSantaCruzRiverBasinAZ,
 eyr = 1990, nsim = 5, nrealz = 5, aseed = 23,
  wwidth = 3, unitSystem = "U.S. Customary",
   ekflag = TRUE, awinFlag = TRUE, tempPerturb = TRUE,
    pcpOccFlag = FALSE, numbCores = NULL)


writeSim(wxOutput = z, nsim = 5, nrealz = 5, path = paste0(tempdir(), "/"), debug = FALSE)
```

---

wx *Runs weather generator*

Description

Runs the weather generator based on user inputs.

Your input/training data MUST have the following variables, in this order: year, month, day, prcp, temp, season. These variables are case sensitive and must be spelled as specified here.

Your training data must be a temporally complete time series (i.e., the time series includes all expected timestamps, even if data is missing). The training data is expected by default to start at the beginning of the calendar year (January 1) but custom year definitions (e.g., water years) can be set using the 'smo' and 'emo' arguments to define start and end months, respectively.

Use starting- and ending- years to subset your input data if desired; otherwise starting and ending dates will default to the beginning and end of your dataset.

Using 'ekflag = T' will generate simulations outside of the historical envelope via an Epanechnikov kernel. For more details on the Epanechnikov kernel and its use in a weather generator, see Rajagopalan et al. (1996).

Leap years may be included in the simulated weather if they are included in your training data, so non-leap years include a row of 'NA' values at the end of the calendar year as a book-keeping measure so that the total number of rows in each trace is the same.

The weather generator can handle missing precipitation and temperature data if it is marked as 'NA' in your training data. It will set 'NA' precipitation values to 0 and pass along 'NA' temperature values if that date is sampled for the simulations. Consider replacing any missing data with monthly or daily averages to avoid 'NA' values in your simulated weather.

Usage

```
wx(
  trainingData,
  syr = NULL,
  eyr = NULL,
  smo = NULL,
  emo = NULL,
  nsim,
  nrealz,
  aseed,
  wwidth,
  unitSystem,
  ekflag,
  awinFlag = FALSE,
  tempPerturb,
  pcpOccFlag = FALSE,
  traceThreshold = 0.005,
  numbCores = NULL,
  returnTempModel = F
)
```

Arguments

trainingData	Either a matrix, dataframe, or path to a .csv file with the following variables is required: year, month, day, prcp (daily precipitation), temp (daily temperature), and season (1, 2, ..., N, for N seasons - up to 26 seasons will work but seasons need to be defined in a meaningful way). Units must be either U.S. Customary (inches, degrees F) or metric (mm, degrees C) and must be specified with the 'unitSystem' input variable. Input data can be station-based, basin averages, grid cells, etc. Input data MUST have these variables: year, month, day, prcp, temp, season.
syr	Optional: subset training data to specific start year (defaults to beginning of training data). Subset will begin on the first day available in 'syr'.
eyr	Optional: subset training data to specific end year (defaults to end of training data). Subset will end on the last day available in 'eyr'.
smo	Training data start month (you can also use to subset your training data).
emo	Training data end month (you can also use to subset your training data).
nsim	Number of simulation years.
nrealz	Number of realizations or traces (i.e., ensemble size).
aseed	Specify a seed for reproducibility.

`wwidth` Set the sampling window for each day of year, a lower value for ‘wwidth‘ will sample fewer surrounding days (lower variability) and a higher value will sample more days (higher variability). Typical setting of ‘wwidth‘ is between 2 and 15, resulting in a daily sampling window of 5 days and 31 days, respectively. Can either be a single number for a uniform window width through the year, or a vector of window widths specific to each season in the training data. In the case of variable window widths, the number of window widths should be equal to the number of seasons.

`unitSystem` Specify the unit system of your training data. Input a string that is either "U.S. Customary" or "Metric". U.S. Customary corresponds to inches and degrees Fahrenheit, while Metric corresponds to millimeter and degrees Celsius. If Metric is specified, units will automatically be converted to U.S. Customary for weather simulation, then re-converted to Metric for results output.

`ekflag` Simulate outside historical envelope using an Epanechnikov kernel? (T/F)

`awinFlag` Set to T or TRUE if you would like to see the results of the adaptive window width. If only one or zero precipitation values (>0.01 inches) are found within the initial window width you set from a day where precipitation occurred, it will be iteratively increased until two or more precipitation values are found. By default, the results are not shown.

`tempPerturb` Set to T or TRUE if you would like to add random noise to the temperature simulations based on a normal distribution fit on the training data.

`pcpOccFlag` Set to TRUE to use precipitation occurrence as a variable in the temperature simulation model or set to FALSE to omit precipitation occurrence as a variable. Simulated daily temperature uses concurrent daily precipitation occurrence as a variable if enabled. By default, this is turned off.

`traceThreshold` Threshold for determining whether precipitation depth is considered a trace amount or not. Precipitation depths below this value will be considered trace amounts and will not be used for simulation. A default value of 0.005-inches is used based on National Weather Service guidance. If using a custom trace depth, ensure that it is in the same unit system as your training data and specified by the ‘unitSystem‘ flag.

`numbCores` Enable parallel computing for precipitation simulation, set number of cores to enable (must be a positive integer greater than or equal to 2). Turned off by default; if set to 0 or 1 it will run as single thread. Use function ’detectCores()’ from ’parallel’ package to show the number of available cores on your machine.

`returnTempModel` Optional flag to return the fitted linear model for daily temperature simulation along with simulation results. Enable by setting TRUE (FALSE by default).

Value

Returns a list containing both inputs to the weather generator as well as outputs.

- dat.d - User inputs to weather generator, saved for future use.
- simyr1 - The years sampled for each trace.
- X - The simulated daily dry/wet sequences for each trace (0 = dry, 1 = wet).

- Xseas - The simulated season by day for each trace.
- Xpdate - If precipitation was simulated to occur on a given day, this is the date from which historical precipitation is sampled.
- Xpamt - The simulated daily precipitation depth.
- Xtemp - The simulated daily mean temperature.

Examples

```
data(LowerSantaCruzRiverBasinAZ)

head(LowerSantaCruzRiverBasinAZ)

#No input for `syr` because we want the training period to begin at the beginning of the data
#record (1970), but set `eyr` = 1990 because we want to subset training period to end in 1990.

wx(trainingData = LowerSantaCruzRiverBasinAZ,
 eyr = 1990, nsim = 3, nrealz = 3, aseed = 23,
  wwidth = 3, unitSystem = "U.S. Customary",
   ekflag = TRUE, awinFlag = TRUE, tempPerturb = TRUE,
    pcpOccFlag = FALSE, numbCores = NULL)
```

---

wxgenR wxgenR *package*

Description

wxgenR: A weather generator with seasonality

Details

This package provides functions for weather generation that incorporate seasonality.

Index