Package 'CRHMr'

October 30, 2017

Type Package

Title Pre- and post- processing for CRHM

Version 2.5.5

Date 2017-10-30

Author Kevin Shook, Centre for Hydrology, University of Saskatchewan

Maintainer Kevin Shook <kevin.shook@usask.ca>

Description The package is pronounced 'Krimmer' in honour of my wife's family. The CRHMr functions are used to create time series of forcing meteorological data for the Cold Regions Hydrological Modelling (CRHM) platform. Functions are also provided to read and aggregate CRHM output.

Depends R (>= 3.1)

Imports ggplot2(>= 2.0.0), lubridate(>= 1.3), plyr, reshape2, scales, signal, stringr(>= 1.0), zoo

Suggests HYDAT, RSQLite, EcoHydRology

License GPL-3

LazyData true

URL https://github.com/CentreForHydrology/CRHMr

RoxygenNote 6.0.1

NeedsCompilation no

R topics documented:

CRHMr-package *Contains functions to perform pre- and post- processing on data used with the Cold Regions Hydrological Modelling (CRHM) platform*

Description

All data in **CRHMr**, whether model input or output must be stored in the standard type of data frame. The file input functions (`readObsFile` `readExportFile` `readOutputFile`) will automatically create standard **CRHMr** data frames. If you are reading data from another type of file using standard R functions, then you will have to force it to be in the standard format.
The first column of the data frame is labelled `datetime`, and as is the date and time stored as a `POSIXct` value, with the correctly-specified time zone. The only exception is for daily CRHM data or for aggregated values which use a daily or longer time step. In these cases the first column will have the appropriate name.
Because CRHM allows there to be many values of a variable, each variable will have a trailing number such as `.1`. This is added automatically when importing CRHM data. Most of the functions allow you to select a variable by its column number. In all cases, the column number does *NOT* include the datetime column.
To make your research more reproducible, each function writes information about what it did, including the date and time it was run, to a log file (`CRHMr.log`) in the default directory. It is suggested that you also run the function `user` when you first start to use **CRHMr** as it writes information about your computer to the log file. This may be helpful when trying to figure out bugs. Please send the output of your log file (including the `user` output) when reporting bugs.
The package contains functions to do the following:

1. Read in CRHM data into a **CRHMr** data frame. This includes data from .obs files (observation data used by CRHM), and model run outputs, either output automatically, or manually exported.
2. Manipulate obs data. Includes functions to plot the values, to find missing values, or datetimes, and to infill gaps by interpolation (linea or spline) and by imputation from other datasets.
3. Convert Ea values to RH and vice-versa. Interpolation and imputation require Ea values, as RH values depend on the air temperature. For safety, **CRHMr** functions do not permit both Ea and RH values in a data frame, as it would be impossible to know which was correct.
4. Write a data frame data to a CRHM obs file.
5. Execute CRHM. This allows CRHM to be run automatically, which is very useful for doing sensitivity analyses. There are functions to prepare a CRHM model to be executed, including setting the run start and end dates, and to run CRHM from inside R.
6. Examine output from CRHM runs. Includes functions to read in model output and to aggregate, summarize and plot the values.

References

To cite **CRHMr** in publications, use the command `citation('CRHMr')` to get the current version of the citation.
The CRHM program is described in:
Pomeroy, John W, D M Gray, T Brown, N Hedstrom, W L Quinton, R J Granger, and S K Carey.

2007. "The Cold Regions Hydrological Model : A Platform for Basing Process Representation and Model Structure on Physical Evidence". Hydrological Processes 21 (19): 2650-2567.
The CRHM model may be downloaded from http://www.usask.ca/hydrology/CRHM.php.

`aggDataframe` *Aggregates a data frame to a shorter time period*

Description

CRHM data (observations and outputs) are generally produced at hourly time steps. This function aggregates CRHM data to daily, monthly or yearly values. The data can be aggregated by their maxima, minima, means, sums and any combination of these statistics.

Usage

```
aggDataframe(CRHMdataframe, columns = 1, period = "annual",
  funs = c("mean"), AggFilename = "", startMonth = 10,
  useSecondYear = TRUE, logfile = "")
```

Arguments

`CRHMdataframe`	Required. A valid **CRHMr** data frame.
`columns`	The columns to be aggregated, not including the datetime. The default is the first column. This can be a vector, i.e. c(1,2,3).
`period`	The period of aggregation. Must be one of 'hourly', 'daily', 'monthly', 'yearly' (or 'annual') or 'hydro'. Default is 'yearly'. Multiple-hour aggregation is not yet supported.
`funs`	A character vector containing the function(s) for aggregation. The default is 'mean', but can also include 'min', 'max' and 'sum'. The function(s) will be applied to all of the specified columns
`AggFilename`	Optional. File name for the aggregated data.
`startMonth`	Optional. Starting month, to be used when aggregating by hydrological year.
`useSecondYear`	Optional. Logical. Should the hydrological year be based on the first or second calendar year. In other words would January 1, 2015 be the hydrological year 2014 or 2015? The default is `TRUE` (i.e., the hydrological year would be 2015). Note that the Campbell Scientific program SPLIT uses the first calendar year (i.e., the hydrological year would be 2014). To emulate this program, set `useSecondYear` to be `FALSE`.
`logfile`	Optional. Name of the file to be used for logging the action. Normally not used.

Value

Returns a data frame with the aggregated values.

Note

The period of aggregation must be smaller than the time step of the CRHM data. This function does NOT remove NA values before aggregation. Use na.omit or one of the infilling functions (interpolate or impute) if you want to remove missing values.

Author(s)

Kevin Shook

See Also

interpolate impute yearlyPeaks hydroYear

Examples

```
Badlake.t.monthly <- aggDataframe(BadLake7376, period='monthly',
columns=1, funs=c('mean'))
Badlake.radiation.daily <- aggDataframe(BadLake7376, period='daily',
columns=c(6,7,8), funs=c('mean'))
```

---

appendObs *Appends two CRHM obs data frames*

Description

This function joins two data frames of CRHM obs. The data frames must have the same number of columns, and the variables must be in the same order. The usual reason is for joining two data frames of differing periods. The primary and secondary data frames may be of any time periods. Where there are two values for a given datetime, the primary values are used. Rows of missing values at the beginning and end of the time series are deleted.

Usage

```
appendObs(primaryObs, secondaryObs, trim = TRUE, quiet = TRUE,
  logfile = "")
```

Arguments

primaryObs	Required. The primary **CRHMr** data frame of obs values.
secondaryObs	Required. The secondary obs data frame. Note that both data frames must have the same time intervals.
trim	Optional. If set to TRUE (the default) then rows missing all values at the beginnings and/or ends of the obs data frames will be omitted.
quiet	Optional. Suppresses display of messages, except for errors. If you are calling this function in an R script, you will usually leave quiet=TRUE (i.e. the default). If you are working interactively, you will probably want to set quiet=FALSE.
logfile	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If successful, returns a data frame of values combined from the primaryObs and secondaryObs data frames. If unsuccessful, returns the value FALSE.

Note

In addition to the usual notation in the logfile, this function also writes a separate file which summarises the new data frame. The summaries are also printed to the screen, if quiet=FALSE, the logfile also contains a complete listing of the source of each value in the infilled data frame. Each value is listed as being 'primary' (from the primaryObs data frame), 'secondary' (derived from the secondaryObs data frame) or 'NA' (missing).

Author(s)

Kevin Shook

Examples

```
## Not run:
broadview <- appendObs(broad2855, broad2856)
## End(Not run)
```

---

as.data.frame.list *Convert a list of vectors to a data frame.*

Description

This function will convert a list of vectors to a data frame. This function will handle three different types of lists of vectors. First, if all the elements in the list are named vectors, the resulting data frame will have have a number of columns equal to the number of unique names across all vectors. In cases where some vectors do not have names in other vectors, those values will be filled with NA.

Usage

```
## S3 method for class 'list'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

x	a list to convert to a data frame.
row.names	a vector equal to length(x) corresponding to the row names. If NULL, the row names will be set to names(x).
optional	not used.
...	other parameters passed to data.frame.

Details

The second case is when all the vectors are of the same length. In this case, the resulting data frame is equivalent to applying `rbind` across all elements.

The third case handled is when there are varying vector lengths and not all the vectors are named. This condition should be avoided. However, the function will attempt to convert this list to a data frame. The resulting data frame will have a number of columns equal to the length of the longest vector. For vectors with length less than this will fill the row with `NA`s. Note that this function will print a warning if this condition occurs.

Value

a data frame.

Author(s)

Jason Bryer «jason@bryer.org»

References

http://stackoverflow.com/questions/4227223/r-list-to-data-frame

Examples

```
## Not run:
test1 <- list( c(a='a',b='b',c='c'), c(a='d',b='e',c='f'))
as.data.frame(test1)

test2 <- list( c('a','b','c'), c(a='d',b='e',c='f'))
as.data.frame(test2)

test3 <- list('Row1'=c(a='a',b='b',c='c'), 'Row2'=c(var1='d',var2='e',var3='f'))
as.data.frame(test3)

test4 <- list('Row1'=letters[1:5], 'Row2'=letters[1:7], 'Row3'=letters[8:14])
as.data.frame(test4)

test5 <- list(letters[1:10], letters[11:20])
as.data.frame(test5)

test6 <- list(list(letters), letters)
as.data.frame(test6)
## End(Not run)
```

assembleObs *Assembles two CRHM obs data frames*

Description

This function joins two data frames of CRHM obs. The data frames can have different columns (it is assumed that they will), but should have some dates in common. Both data frames must have the same time step.

Usage

```
assembleObs(obs1, obs2, quiet = TRUE, logfile = "")
```

Arguments

obs1	Required. The first **CRHMr** data frame of obs values.
obs2	Required. The second obs data frame.
quiet	Optional. Suppresses display of messages, except for errors. If you are calling this function in an R script, you will usually leave quiet=TRUE (i.e. the default). If you are working interactively, you will probably want to set quiet=FALSE.
logfile	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If successful, returns the merges values of both data frames. Note that where the datetimes in the data frames are not the same, the merged values will be set to NA_real_. If unsuccessful, returns the value FALSE.

Author(s)

Kevin Shook

Examples

```
## Not run:
MSC.trh <- assembleObs(MSC.t, MSC.rh)
## End(Not run)
```

automatePrj *Automates a CRHM .prj file*

Description

Converts a CRHM .prj file to be able to be run automatically. It adds the settings Auto_Run, Auto_Exit and Log_All to the end of the .prj file, if they are required.

Usage

```
automatePrj(inputPrjFile = "", outputPrjFile = "", quiet = TRUE,
  logfile = "")
```

Arguments

inputPrjFile	Required. Name of the the CRHM .prj to be converted.
outputPrjFile	Optional. If omitted, the input .prj file will be overwritten.
quiet	Optional. Suppresses display of messages, except for errors. If you are calling this function in an R script, you will usually leave quiet=TRUE (i.e. the default). If you are working interactively, you will probably want to set quiet=FALSE.
logfile	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If successful, returns TRUE. If unsuccessful, returns FALSE.

Author(s)

Kevin Shook

See Also

runCRHM

Examples

```
## Not run:
result <- automatePrj('Bad Lake 1974-1975.prj','Bad Lake 1974-1975 auto.prj')
## End(Not run)
```

BadLake7376 *Observed meteorological data for Bad Lake*

Description

A data frame containing meteorological data from Bad Lake, Saskatchewan, from January 1, 1973 to January 1, 1976. This data was collected by the Division of Hydrology at the University of Saskatchewan.

Usage

```
BadLake7376
```

Format

A data frame with 26280 rows and 9 variables (including the datetime):

datetime date and time as a POSIXct object

t.1 air temperature (C)

rh.1 relative humidity (percent)

u.1 wind speed (m/s)

SunAct.1 Actual daily sunshine hours, repeated for each hour of the day

ppt.1 interval (hourly) precipitation (mm)

Qsi.1 Incoming shortwave radiation (W/m^2)

Qso.1 Outgoing shortwave radiation (W/m^2)

Qn.1 Net radiation (W/m^2)

Source

This data is also distributed with the example files of the CRHM program. CRHM can be obtained from http://www.usask.ca/hydrology/CRHM.php.

basinWaterBalancePlot *Stacked plot of yearly water balance variables*

Description

Creates a stacked bar plot of all water balance components computed for a basin. Basin inputs are plotted as positive values, outputs are plotted as negative.

Usage

```
basinWaterBalancePlot(yearlyWater, negCols = "")
```

Arguments

yearlyWater	Required. A data frame of water balance components. The first column must be the year. Note that this function does not allow you to select columns - all columns will be plotted.
negCols	Optional. Columns to be plotted as negative values. If not specified (the default) the columns will be guessed from their names.

Value

If successful returns a **ggplot2** object showing stacked bars of the water balance components for each year. If unsuccessful returns FALSE.

Examples

```
## Not run:
# get daily water balance from CRHM output
daily <- simpleDailyWater(Bol84, prjFile = 'Bologna1984_02.prj',
basinMean = TRUE, summarize = TRUE)

# get yearly values
daily.datetime <- dateToDatetime(daily, timezone='Etc/GMT+7')
yearly <- aggDataframe(daily.datetime, period='hydro', funs='sum')

# plot
p <- basinWaterBalancePlot(yearly)

## End(Not run)
```

---

changeEatoRH *Changes Ea values to RH in a CRHM data frame*

Description

This function converts values of Ea to RH. Note that the specified obs data frame must contain both Ea and air temperatures. For safety, **CRHMr** does not permit values of both RH and Ea in a data frame. The names of the variables containing air temperature and ea values must be of the form t.x and ea.x, respectively, where x is an number, even if the column numbers are specified.

Usage

```
changeEatoRH(obs, t.cols = 1, ea.cols = 1, quiet = TRUE, logfile = "")
```

Arguments

obs	Required. A **CRHMr** data frame of observations.

t.cols	Optional. A vector containing the column numbers (not including the datetime) holding the air temperatures. If no columns are specified then the locations of the temperatures are guessed from the column names. The air temperatures must be in °C.
ea.cols	Optional. A vector containing the column numbers (not including the datetime) holding the vapour pressures. If no columns are specified then the locations of the vapour pressures are guessed from the column name. The ea values must be in kPa.
quiet	Optional. Suppresses display of messages, except for errors. If you are calling this function in an R script, you will usually leave quiet=TRUE (i.e. the default). If you are working interactively, you will probably want to set quiet=FALSE.
logfile	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If successful, the function returns the original data frame with the ea columns converted to RH values in perceent. If unsuccessful, it returns the value FALSE.

Author(s)

Kevin Shook

See Also

changeRHtoEa

Examples

```
# First, convert RH values to Ea
BadLake7376.ea <- changeRHtoEa(BadLake7376)
# now convert Ea values back to RH
BadLake7376.rh <- changeEatoRH(BadLake7376.ea)
```

---

changeRHtoEa *Changes RH values to Ea in a CRHM dataframe*

Description

This function converts values of RH to Ea. **CRHMr** does not allow interpolation or imputation of RH values, so you must convert RH values to Ea before infilling. Note that the specified obs dataframe must contain both RH and air temperatures. For safety, **CRHMr** does not permit values of both RH and Ea in a data frame. The names of the variables containing air temperature and RH values must be of the form t.x and rh.x, respectively, where x is an number, even if the column numbers are specified.

Usage

```
changeRHtoEa(obs, t.cols = 1, rh.cols = 1, quiet = TRUE, logfile = "")
```

Arguments

obs	Required. A **CRHMr** data frame of observations.
t.cols	Optional. A vector containing the column numbers (not including the datetime) holding the air temperatures. If no columns are specified then the locations of the temperatures are guessed from the column names. The air temperatures must be in °C.
rh.cols	Optional. A vector containing the column numbers (not including the datetime) holding the relative humidities. If no columns are specified then the locations of the RH values are guessed from the column names.
quiet	Optional. Suppresses display of messages, except for errors. If you are calling this function in an R script, you will usually leave quiet=TRUE (i.e. the default). If you are working interactively, you will probably want to set quiet=FALSE.
logfile	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If successful, the function returns the original data frame with the RH columns converted to ea values in kPa. If unsuccessful, it returns the value FALSE.

Author(s)

Kevin Shook

See Also

changeEatoRH

Examples

```
BadLake7376.ea <- changeRHtoEa(BadLake7376)
```

---

convertDPtoRH *Converts dew point temperature and ambient temperature to RH*

Description

This is an internal **CRHMr** function and should normally not need to be called directly. Note that it does *NO* data checking.

Usage

```
convertDPtoRH(t, td)
```

Arguments

t	Required. Air temperature in °C.
td	Required. Dew point temperature in °C.

Value

Returns the RH as a percentage.

Author(s)

Kevin Shook

References

Lawrence, M. 2005. The relationship between relative humidity and the dewpoint temperature in moist air. American Meteorological Society: 225-233. http://andrew.rsmas.miami.edu/bmcnoldy/Humidity.htm

Examples

```
convertDPtoRH(10, 9)
```

`createObsDataframe` *Creates an empty data fame for CRHM obs*

Description

Creates a data fame to hold observations. All values are initially set to `NA_real_`. Once the data fame has been created, it may be filled from other data fames by imputation.

Usage

```
createObsDataframe(start.date, end.date, timestep = 1, variables = c("t",
  "rh", "u", "p"), reps = 1, timezone = "", logfile = "")
```

Arguments

`start.date`	Required. Starting date, in the format 'Y-m-d', 'd/m/Y', 'd-m-Y', 'd B Y', 'd b Y', 'd-B-Y', 'd-b-Y', 'B d, Y', or 'b d, Y', where 'Y' = year, 'm' = month number, 'B' = capitalized month name, 'b' = lower case month name, 'd' = day number.
`end.date`	Required. Ending date,, in the format 'd/m/Y', 'd-m-Y', 'd B Y', 'd b Y', 'd-B-Y', 'd-b-Y', 'B d, Y', or 'b d, Y', where 'Y' = year, 'm' = month number, 'B' = capitalized month name, 'b' = lower case month name, 'd' = day number.
`timestep`	Optional. Time interval between values in hours. The default value is 1 hour. The interval can be less than 1 hour.
`variables`	Optional. vector containing the names of variables to be created. The default values are 't', 'rh', 'u', and 'p'.
`reps`	Optional. The number of repetitions for each value. The default is 1.

timezone	Required. The name of the timezone of the data as a character string. This should be the timezone of your data, but omitting daylight savings time. Note that the timezone code is specific to your OS. To avoid problems, you should use a timezone without daylight savings time. Under Linux, you can use 'CST' and 'MST' for Central Standard or Mountain Standard time, respectively. Under Windows or OSX, you can use 'etc/GMT+6' or 'etc/GMT+7' for Central Standard and Mountain Standard time. DO NOT use 'America/Regina' as the time zone, as it includes historical changes between standard and daylight savings time.
logfile	Optional. Name of the file to be used for logging the action. Normally not used.

Details

The names for the variables in the data fame are derived from the specified variables, and the number of reps. For example if the default variable names are used, with 2 reps, then the variables created will be t.1, t.2, rh.1, rh.2, u.1, u.2, p.1, and p.2.

Value

If successful, the function returns a CRHM data fame with the specified number of each variable, with values set to be NA. If unsuccessful, returns the value FALSE.

Author(s)

Kevin Shook

See Also

impute

Examples

```
vars <- c('t','rh','u')
stoon <- createObsDataframe('1/1/1960', '31/12/1960', variables=vars, reps=2, timezone='etc/GMT+6')
```

CRHM_summary *Summarizes a* **CRHMr** *data frame*

Description

Summarizes the values in a **CRHMr** data frame. This is an internal **CRHMr** function and should *never* need to be called directly.

Usage

```
CRHM_summary(CRHMdataframe)
```

Arguments

CRHMdataframe Required. Name of the **CRHMr** data frame to be summarized.

Value

If successful, returns a data frame containing the number of rows, complete rows, starting and ending datetimes, and the variables in the specified data frame. If unsuccessful, returns the value FALSE.

Author(s)

Kevin Shook

Examples

```
## Not run:
summary <- CRHMsummary(BadLake7376)
## End(Not run)
```

CRHM_vars *CRHM variables*

Description

A data frame containing information about all of the variables in CRHM that are likely to be output from a model run.

Usage

```
CRHM_vars
```

Format

A data frame with 674 rows and 5 variables:

name name of the variable

type type of variable (its dimensions)

description description of the variable

units variable units

end_of_day TRUE if the value is only output at the end of the day (00:00), otherwise FALSE

Source

The variable information was abstacted from the CRHM c++ source code.

`cumsumDataframe` *Cumulative sums for a data frame*

Description

Finds the cumulative sum of all columns in a data frame. Note that all columns must be numeric - date and datetime variables cannot be summed.

Usage

```
cumsumDataframe(df)
```

Arguments

`df`	Required. Data frame to be summed.

Value

Returns a dataframe containing the cumulative sums of all columns in the original data frame.

Note

This function is used by other **CRHMr** functions, so it does *NO* parameter testing.

Author(s)

Kevin Shook

See Also

`cumulDailyWater`

Examples

```
## Not run:
cumul <- cumsumDataframe(modelOutput)
## End(Not run)
```

cumulativeDischargePlot

Plots cumulative CRHM and/or HYDAT flows

Description

Creates a **ggplot** object of annual cumulative flows from data frames of CRHM and/or HYDAT data. The HYDAT flows can be obtained using the **HYDAT** package created by David Hutchinson. The HYDAT data is truncated so that it only includes the range of the CRHM data. If more than a single year of data is specified, then the plot will be faceted by year.

Usage

```
cumulativeDischargePlot(CRHMflows = NULL, CRHMflowsLabel = "",
  CRHMflowCol = 1, HYDATflows = NULL, HYDATflowsLabel = "",
  facetCols = 3, quiet = TRUE)
```

Arguments

CRHMflows	Optional. Optional. A data frame of CRHM modelled flows. The flows must be in m^3/s.
CRHMflowsLabel	Optional. Labels for the CRHM data. If not specified, and CRHM data are plotted, then the name(s) of the CRHM variable(s) will be used.
CRHMflowCol	Optional. Column containing the flowrates, not including the datetime. Default is 1.
HYDATflows	Optional. Data frame containing WSC daily flows. The data frame is the same as is returned by the function DailyHydrometric in the package **HYDAT** developed by David Hutchinson. The data frame has the columns STATION_NUMBER, DATE, VALUE and FLAG. The DATE must be an R date.
HYDATflowsLabel	Optional. Labels for the daily flows. If not specified, then the value in STATION_NUMBER will be used, followed by 'daily'.
facetCols	Optional. Number of columns to wrap the facets (if they are used). Setting a value less than 1 will cause an error.
quiet	Optional. Suppresses display of messages, except for errors. If you are calling this function in an R script, you will usually leave quiet=TRUE (i.e. the default). If you are working interactively, you will probably want to set quiet=FALSE.

Value

If successful, returns a **ggplot2** object containing the cumulative discharge plot. If unsuccessful, the value FALSE will be returned.

Author(s)

Kevin Shook

See Also

hydrograph

Examples

```
## Not run:
p <- cumulativeDischargePlot(crhm, 'CRHM', 1, dailyflows, 'HYDAT', facetCols=4, quiet=FALSE)
## End(Not run)
```

---

cumulDailyWater *Cumulative water fluxes*

Description

Accumulates water fluxes by year or water year. The values are cumulative for each day.

Usage

```
cumulDailyWater(dailyWaterVals, waterYear = FALSE, logfile = "")
```

Arguments

dailyWaterVals	Required. CRHM daily water fluxes as calculated by simpleDailyWater.
waterYear	Optional. If FALSE the fluxes are accumlated by calendar year. If TRUE the fluxes are accumulated by the water year, based on the first month of the model output.
logfile	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If successful, returns a data frame containing the date and the cumulative values of each variable. If unsuccessful, returns FALSE.

Note

The output from this function can be plotted using simpleRibbonPlot.

Author(s)

Kevin Shook

See Also

simpleDailyWater simpleRibbonPlot

Examples

```
## Not run:
waterYearCumul <- cumulDailyWater(CRHM_daily, waterYear=TRUE)

## End(Not run)
```

---

datetimeToDate *Converts an* R *datetime to a date*

Description

This is used when a **CRHMr** date is stored as a datetime (POSIXct) variable.

Usage

```
datetimeToDate(dataframe, logfile = "")
```

Arguments

dataframe	Required. A data frame. The first name of the column must be a POSIXct variable, which can be converted to a standard R date.
logfile	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If successful, returns a data frame where the first column is a standard **CRHMr** date. If unsuccessful, returns the value FALSE.

Author(s)

Kevin Shook

See Also

dateToDatetime

Examples

```
Badlake.radiation.daily <- aggDataframe(BadLake7376, period='daily',
columns=c(6,7,8), funs=c('mean'))
BadLake.datetime <- dateToDatetime(Badlake.radiation.daily, timezone='CST')
BadLake.date <- datetimeToDate(BadLake.datetime)
```

dateToDatetime *Converts an* R *date to a datetime*

Description

The aggDataframe function produces non-standard data frames, which have dates rather than datetimes as their first column. This function will convert a data frame with dates in the first column to have datetimes with the time being set to a constant value.

Usage

```
dateToDatetime(dataframe, hour = 0, timezone = "", logfile = "")
```

Arguments

dataframe	Required. A data frame. The first name of the column must be 'date', and it must be standard R date.
hour	Optional. The hour to be used for the datetime values. Default is 0.
timezone	Required. The name of the timezone of the data as a character string. This should be the timezone of your data, but omitting daylight savings time. Note that the timezone code is specific to your OS. To avoid problems, you should use a timezone without daylight savings time. Under Linux, you can use 'CST' and 'MST' for Central Standard or Mountain Standard time, respectively. Under Windows or OSX, you can use 'etc/GMT+6' or 'etc/GMT+7' for Central Standard and Mountain Standard time. DO NOT use 'America/Regina' as the time zone, as it includes historical changes between standard and daylight savings time.
logfile	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If successful, returns a data frame where the first column is a standard **CRHMr** datetime. If unsuccessful, returns the value FALSE.

Author(s)

Kevin Shook

See Also

aggDataframe datetimeToDate

Examples

```
Badlake.radiation.daily <- aggDataframe(BadLake7376, period='daily',
columns=c(6,7,8), funs=c('mean'))
BadLake.datetime <- dateToDatetime(Badlake.radiation.daily, timezone='CST')
```

deDupe *Removes duplicated datetimes in obs data frame*

Description

Removes duplicated datetime values. Many time series, especially from Environment Canada, may contain duplicated datetimes. This function replaces the duplicated values. It is important to use this function before interpolating or imputing values, and especially before writing the data frame to an obs file.

Usage

```
deDupe(obs, action = "mean", quiet = TRUE, logfile = "")
```

Arguments

obs	Required. A **CRHMr** data frame containing the obs values.
action	Optional. The action used to replace the duplicate values. Must be one of 'min', 'max', 'mean', 'skip', 'delete', 'split' or 'second'. Default is 'mean'.
quiet	Optional. Suppresses display of messages, except for errors. If you are calling this function in an R script, you will usually leave quiet=TRUE (i.e. the default). If you are working interactively, you will probably want to set quiet=FALSE.
logfile	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If there are no duplicates, returns 'No duplicates'. If duplicates exist, and are are successfully removed, the de-duplicated data frame is returned. If the de-duplication is unsuccessful, then the value FALSE is returned.

Note

If action='min', 'max', or 'mean', the action function is applied to all of the values for each duplicate datetime. If action='skip' or 'delete', then the values of the duplicate datetimes are deleted. If action='split' then the original values are kept, and the duplicate values are written to an obs file. The name of the obs file is the name of the obs variable followed by '_dupes.obs'. If action='second', then the second duplicate values are used. This can be useful when dealing with duplicates caused by daylight savings time.

Author(s)

Kevin Shook

See Also

findDupes

Examples

```
BadLake.deduped <- deDupe(BadLake7376, action='mean')
```

`deleteSpikes` *Deletes spikes in an obs data frame*

Description

Finds spikes, and sets their values to be NA_real_.

Usage

```
deleteSpikes(obs, colnum = 1, threshold = 0, logfile = "")
```

Arguments

`obs`	Required. A **CRHMr** obs data frame.
`colnum`	Optional. The number of the column to test for spikes, not including the date-time.
`threshold`	Required. The threshold for the *change* in the observed values. The threshold is actually a rate, i.e. the change per unit time. So if you are looking at air temperature, and the threshold is set to 5, then any change in temperature of +/- 5 degrees in one time interval will be considered to be a spike.
`logfile`	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If successful, returns a data frame consiting of the datetime and the original obs values, where all of the spike values have been set to be `NA_real_`. If unsuccessful, returns the value `FALSE`.

Author(s)

Kevin Shook

See Also

`findSpikes`

Examples

```
BadLake <- BadLake7376
# finds all windspeeds which change by more than 10 m/s per interval,
# and sets their values to NA_real_
BadLake$u.nospikes <- deleteSpikes(BadLake7376, 3, 10)
```

distributeInst *Distribute instantaneous values*

Description

Distributes instantaneous values to a shorter time interval. The missing datetimes are inserted and then the values are interpolated. This function is typically used to downscale obs values such as t, ea, and u.

Usage

```
distributeInst(obs, obsCols = "all", timeStep = 0,
  interpolationMethod = "linear", maxLength = 5, quiet = TRUE,
  logfile = "")
```

Arguments

obs	Required. The **CRHMr** data frame of obs values.
obsCols	Optional. A vector containing the columns to be imputed in the obs data frame, not including the datetime. The default 'all' specifies all columns.
timeStep	Required. The time step (in hours) for the interpolated values. This value must be smaller than the time step in the original time series.
interpolationMethod	Optional. A vector containing the methods to be used for interpolation for each of the variables. Currently supported methods are 'linear' and 'spline'. The default is to use linear interpolation. If fewer methods than columns are specified, the methods are recycled.
maxLength	Optional. The maximum gap length to be interpolated. Defaults to 5 time steps.
quiet	Optional. Suppresses display of messages, except for errors. If you are calling this function in an R script, you will usually leave quiet=TRUE (i.e. the default). If you are working interactively, you will probably want to set quiet=FALSE.
logfile	Optional. Name of the file to be used for logging the action. Normally not use

Value

If successful, returns a dataframe of the selected columns interpolated to the specified time step. If unsuccessful, returns FALSE.

Author(s)

Kevin Shook

See Also

distributeMean

Examples

```
## Not run:
hourly_vals <- distributeInst(vegreville, c(1,2,3), timeStep = 1)
## End(Not run)
```

`distributeMean` *Distribute mean values*

Description

Distributes mean values to a shorter time interval. The missing datetimes are inserted and then the values are repeated. This function is typically used to downscale obs values such as t, ea, and u.

Usage

```
distributeMean(obs, obsCols = "all", timeStep = 0, maxLength = 5,
  quiet = TRUE, logfile = "")
```

Arguments

`obs`	Required. The **CRHMr** data frame of obs values.
`obsCols`	Optional. A vector containing the columns to be imputed in the obs data frame, not including the datetime. The default '`all`' specifies all columns.
`timeStep`	Required. The time step (in hours) for the interpolated values. This value must be smaller than the time step in the original time series.
`maxLength`	Optional. The maximum gap length to be interpolated. Defaults to 5 time steps.
`quiet`	Optional. Suppresses display of messages, except for errors. If you are calling this function in an R script, you will usually leave `quiet=TRUE` (i.e. the default). If you are working interactively, you will probably want to set `quiet=FALSE`.
`logfile`	Optional. Name of the file to be used for logging the action. Normally not use

Value

If successful, returns a dataframe of the selected columns interpolated to the specified time step. If unsuccessful, returns `FALSE`.

Author(s)

Kevin Shook

See Also

`distributeInst`

Examples

```
## Not run:
hourly_vals <- distributeMean(vegreville, c(1,2,3), timeStep = 1)
## End(Not run)
```

`distributeP` *Distributes total precipitation over intervals*

Description

Distributes total precipitation (often daily) evenly over all of the applicable intervals. The function has 2 uses. The primary use is to distribute total precipitation when the precipitation time step is greater than the time step of the obs dataframe. For example, there may be 3-hour total precipitation within an obs dataframe holding hourly values for all other variables. The secondary use is to disaggregate precipitation to a shorter time step. Normally this can be done inside CRHM by putting the data in a separate obs file. However, if you want to do this in this function, you must specify the new time step.

Usage

```
distributeP(obs, p.cols = 0, timestep = 0, quiet = TRUE, logfile = "")
```

Arguments

`obs`	Required. The **CRHMr** data frame of obs values.
`p.cols`	Optional. The number of the column(s) to be used. If not specified, the column will be guessed from the variable name. Note that the variable name MUST include the letter 'p', even if the number is specified.
`timestep`	Optional. The new time step in hours. If zero (the default) the time step is not used. The time step is only used when disaggregating precipitation to a shorter time step. Note that the specified time step must be shorter than that used in the original data frame.
`quiet`	Optional. Suppresses display of messages, except for errors. If you are calling this function in an R script, you will usually leave `quiet=TRUE` (i.e. the default). If you are working interactively, you will probably want to set `quiet=FALSE`.
`logfile`	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If sucessful, returns a modified version of the obs data frame. If no time step is specified, the missing precipitation values in the specified data frame are replaced by average precipitation values. If a time step is specified, a new data frame containing ONLY the datetime and the distributed precipitation will be returned. If unsuccessful, returns the value `FALSE`.

Note

This function is potentially destructive. If you have a missing precipitation total, then the next precipitation total will be distributed over a longer time period. For example, you may have 3-hour precipitation data, normally at 03:00, 06:00, 09:00 etc., to be distributed hourly. If the 06:00 data are missing, then the 09:00 precipitation will be distributed over the interval from 04:00 to 09:00.

Author(s)

Kevin Shook

See Also

`impute` `interpolate` `distributeInst` `distributeQli` `distributeQsi`

Examples

```
## Not run:
distributed <- distributeP(obs, 3)
## End(Not run)
```

---

distributeQli *Distributes incoming longwave radiation to shorter time interval*

Description

Downscales incoming longwave radiation to shorter time intervals, based on the air temperatures, using the Stefan-Boltzmann law $Q_{li} = \epsilon \sigma T^4$. The atmospheric emissivity (ϵ) is calculated from the longwave radiation and the temperature at the original time intervals. The air temperatures are interpolated to the shorter intervals (if required), and are then used to calculate the incoming longwave, using the longer-interval emissivities. Finally, the downscaled longwave radiation is adjusted so that its mean values over the original time intervals are the same as the original values.

Usage

```
distributeQli(QliObs, QliColnum = 1, tObs, tColnum = 1, timeStep = 1,
  interpolationMethod = "linear", maxlength = 0, quiet = TRUE,
  logfile = "")
```

Arguments

`QliObs`	Required. A **CRHMr** obs data frame containing incoming longwave radiation in in W/m^2.
`QliColnum`	Optional. The number of the columns (not including the datetime) containing the incoming longwave values. Default is `1`.
`tObs`	Required. A **CRHMr** obs data frame containing air temperatures. The values may be in K or °C (the function will detect and adjust).

tColnum	Optional. The number of the columns (not including the datetime) containing the air temperature values. Default is 1.
timeStep	Optional. The number of hours to be used as the downscaled time step. Default is 1.
interpolationMethod	Optional. The method to be used for interpolating the air temperatures (if required). Currently supported methods are 'linear' and 'spline'. The default is to use linear interpolation.
maxlength	Optional. The maximum gap length to be interpolated. If not specified (the default), then the maximum gap length permitted is twice the time interval of the air temperatures.
quiet	Optional. Suppresses display of messages, except for errors. If you are calling this function in an R script, you will usually leave quiet=TRUE (i.e. the default). If you are working interactively, you will probably want to set quiet=FALSE.
logfile	Optional. Name of the file to be used for logging the action. Normally not used

Value

If successful, returns an obs data frame containing the downscaled incoming longwave radiation (Qli) in W/m^2. If unsuccessful, returns the value FALSE.

Author(s)

Kevin Shook

See Also

emissivity longwave interpolate

Examples

```
## Not run:
hourlyLongwave <- distributeQli(LWobs, 1, tobs, 1)
## End(Not run)
```

---

distributeQsi *Distributes incoming shortwave radiation to shorter time interval*

Description

Downscales incoming shortwave radiation to shorter time intervals, based on the extra-terrestrial radiation. The mean atmospheric transmittance is computed from the ratio of the incoming shortwave to the extra-terrestrial radiation over the original time intervals. The extra-terrestrial radiation over the shorter time intervals is then multiplied by the mean transmittance values to calculate the shorter interval incoming shortwave radiation.

Usage

```
distributeQsi(QsiObs, QsiColnum = 1, latitude = "", sunTimeOffset = 2,
  timeStep = 1, solarMethod = "simpleMaxSolar", details = FALSE,
  quiet = TRUE, logfile = "")
```

Arguments

QsiObs	Required. A **CRHMr** obs data frame containing incoming shortwave radiation in in W/m^2.
QsiColnum	Optional. The number of the columns (not including the datetime) containing the incoming shortwave values. Default is 1.
latitude	Required. The latitude for which values are to be calculated.
sunTimeOffset	Optional. The offset (in hours) is added to the solar time to convert it to local time. The default value '2' shifts the daily peak to occur at 2pm.
timeStep	Optional. The number of hours to be used as the downscaled time step. Default is 1.
solarMethod	The method to be used for calculating the extra-terrestrial radiation. The default method is 'simpleMaxSolar'. Note that this method is only valid for latitudes between 49 and 55°N. The other supported method is 'PotSolarInst', which requires the package **EcoHydRology** to be installed.
details	Optional. If TRUE then the calculated atmospheric transmittance and extra-terrestrial radiation (So_h_W) will be returned along with the downscaled Qsi. If FALSE (the default) then only the Qsi values and the datetime are returned.
quiet	Optional. Suppresses display of messages, except for errors. If you are calling this function in an R script, you will usually leave quiet=TRUE (i.e. the default). If you are working interactively, you will probably want to set quiet=FALSE.
logfile	Optional. Name of the file to be used for logging the action. Normally not used

Value

If successful, returns an obs data frame containing the downscaled incoming shortwave radiation (Qsi) in W/m^2 and, optionally, the transmittance and extra-terrestrial solar radiation (So_h_W). If unsuccessful, returns the value FALSE.

Author(s)

Kevin Shook

See Also

simpleMaxSolar distributeQli

Examples

```
## Not run:
hourlyShortwave <- distributeQsi(QsiObs=SWobs, QsiColnum=1, latitude=51.1, timeStep=1)
## End(Not run)
```

doubleMass *Calculates double mass values*

Description

Calculates cumulative sums for two variables and optionally returns a plot. Only values coinciding in time are used for the cumulative sums.

Usage

```
doubleMass(primaryObs, primaryCol = 1, secondaryObs, secondaryCol = 1,
  plot = TRUE, logfile = "")
```

Arguments

primaryObs	Required. The primary **CRHMr** data frame of obs values.
primaryCol	Optional. The column number to be used, not including the datetime. If not specified, defaults to the first column.
secondaryObs	Required. The secondary **CRHMr** data frame of obs values.
secondaryCol	Optional. The column number to be used, not including the datetime. If not specified, defaults to the first column.
plot	Optional. Should a plot be created? Default is FALSE.
logfile	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If plot = TRUE, then a **ggplot2** object will be returned, with the primaryObs values plotted on the x-axis and the secondaryObs values plotted on the y-axis. Otherwise, a dataframe is returned with the cumulative values ov each of the variables.

Author(s)

Kevin Shook

Examples

```
## Not run:
p <- doubleMass(primaryObs=CamrosePPT, secondaryObs=WetaskiwinPPT, plot=TRUE)
## End(Not run)
```

emissivity *Calculates atmospheric emissivity*

Description

Calculates atmospheric emissivity from specified longwave radiation and air temperatures. This is an internal **CRHMr** function and should *never* need to be called directly, as it does *NO* data checking.

Usage

```
emissivity(longwave, tK)
```

Arguments

longwave	Required. Longwave radiation in W/m^2.
tK	Required. Air temperatures in K.

Value

Returns the atmospheric emissivity.

Author(s)

Kevin Shook

See Also

distributeQli longwave

Examples

```
emissivity(100, 293)
```

extendObs *Extends an obs data frame*

Description

This function extends a **CRHMr** obs data frame to include the specified dates. All added values are set to be NA_real_. The purpose of this function is to create locations for data that can be infilled or imputed from another data set.

Usage

```
extendObs(obs, startDate = "", endDate = "", timezone = "",
  quiet = TRUE, logfile = "")
```

Arguments

obs	Required. The **CRHMr** obs data frame to be extended.
startDate	Optional. The starting date for the obs data frame. Can either be a year as a number (e.g. 1995) or a date in 'Y-m-d' format, i.e. '1995-06-01'. Either the startDate or the endDate, or both, must be specified.
endDate	Optional. The ending date for the obs data frame. Can either be a year as a number (e.g. 1995) or a date in 'Y-m-d' format, i.e. '1995-07-01'.
timezone	Required. The name of the timezone of the data as a character string. This should be the timezone of your data, but omitting daylight savings time. Note that the timezone code is specific to your OS. To avoid problems, you should use a timezone without daylight savings time. Under Linux, you can use 'CST' and 'MST' for Central Standard or Mountain Standard time, respectively. Under Windows or OSX, you can use 'etc/GMT+6' or 'etc/GMT+7' for Central Standard and Mountain Standard time. DO NOT use 'America/Regina' as the time zone, as it includes historical changes between standard and daylight savings time.
quiet	Optional. Suppresses display of messages, except for errors. If you are calling this function in an R script, you will usually leave quiet=TRUE (i.e. the default). If you are working interactively, you will probably want to set quiet=FALSE.
logfile	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If successful, returns a modified version of the obs data frame. The missing values in the specified data frame are replaced by NA_real_ values. If unsuccessful, returns the value FALSE.

Author(s)

Kevin Shook

Examples

```
BadLake7378 <- extendObs(BadLake7376, endDate='1978-01-01', timezone='CST')
```

---

findDupes *Finds duplicated datetimes in obs dataframe*

Description

Finds duplicate datetime values. All of the duplicate values are written to a .csv file. Many time series, especially from Environment Canada, may contain duplicate datetimes.

Usage

```
findDupes(obs, dupefile = "", quiet = TRUE, logfile = "")
```

Arguments

obs	Required. A **CRHMr** data frame containing obs values.
dupefile	Optional. The name of the output file. If omitted the dupe file will be the name of the obs data frame, followed by '_dupedatetimes.csv'.
quiet	Optional. Suppresses display of messages, except for errors. If you are calling this function in an R script, you will usually leave quiet=TRUE (i.e. the default). If you are working interactively, you will probably want to set quiet=FALSE.
logfile	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If there are duplicate datetimes, returns TRUE. If there are no duplicates, returns FALSE.

Note

If quiet=FALSE, the function gives a list of the duplicate datetimes. Use this function before removing duplicates and/or interpolation and imputation. Most importantly, use this function before writing values to an obs file. Duplicate datetime values in an obs file will prevent CRHM from executing.

Author(s)

Kevin Shook

See Also

findGaps interpolate impute writeObsFile

Examples

```
findDupes(BadLake7376, quiet=FALSE)
```

---

findGaps *Finds gaps in obs variables*

Description

Finds successive missing values in an obs data frame. This function is useful to show where data are missing, before you use the interpolate or impute functions. All of the gaps in each variable are written to a .csv file.

Usage

```
findGaps(obs, gapfile = "", minlength = 1, quiet = TRUE, logfile = "")
```

Arguments

obs	Required. A **CRHMr** data frame containing obs values.
gapfile	Optional. The name of the output file. If omitted the output file will be the name of the obs data frame, followed by '_gaps.csv'.
minlength	The minimum gap length included in the analysis (in time steps). Default is 1 time step.
quiet	Optional. Suppresses display of messages, except for errors. If you are calling this function in an R script, you will usually leave quiet=TRUE (i.e. the default). If you are working interactively, you will probably want to set quiet=FALSE.
logfile	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If successful, and there are gaps, returns TRUE. If successful, and there are no gaps, returns ''No gaps'. If unsuccessful, returns FALSE.

Note

If quiet=FALSE, the functions gives a summary of the gaps (number of gaps, their total length) for each year.

Author(s)

Kevin Shook

See Also

insertMissing interpolate impute

Examples

```
findGaps(BadLake7376, quiet=FALSE)
```

---

findSpikes *Finds spikes in an obs data frame*

Description

Finds spikes (short bursts of very large or small values) in the specified time variable.

Usage

```
findSpikes(obs, colnum = 1, threshold = 0, logfile = "")
```

Arguments

obs	Required. A **CRHMr** obs data frame.
colnum	Optional. The number of the column to test for spikes, not including the datetime.
threshold	Required. The threshold for the *change* in the observed values. The threshold is actually a rate, i.e. the change per unit time. So if you are looking at air temperature, and the threshold is set to 5, then any change in temperature of +/- 5 degrees in one time interval will be considered to be a spike.
logfile	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If successful and there are no spikes, returns 0. If there are spikes, returns their datetime values. If unsuccessful returns FALSE.

Note

If quiet=FALSE, the function displays a list of the datetimes of the spikes.

Author(s)

Kevin Shook

See Also

deleteSpikes findGaps findDupes

Examples

```
# check for spikes in wind speed - look for changes greater than 10 m/s per interval
findSpikes(BadLake7376,3, 10)
```

GMToffset *Finds offset in hours from UTC for given time zone.*

Description

Finds offset in hours from UTC for given time zone.

Usage

```
GMToffset(timezone)
```

Arguments

timezone	Required. A string containing the time zone. Works with designated time zones such as 'CST' and 'MST' for Central Standard or Mountain Standard time or offset time zones such as 'etc/GMT+6' or 'etc/GMT+7' for Central Standard and Mountain Standard time. Does NOT work with location time zones like 'America/Regina'.

Value

Returns the offset in hours as a numeric value.

Note

This code was based on the function `GMTOffsetFromTz` in the package **oce**, but works for more time zones and returns a negative value in the western hemisphere.

Author(s)

Kevin Shook

Examples

```
GMToffset('MST')
GMToffset('etc/GMT+6')
```

---

hruGroupWaterSummary *Summarize daily CRHM water outputs by HRU group*

Description

Calculates daily totals of water storages and fluxes for groups of HRUs, for models *without* subbasins.

Usage

```
hruGroupWaterSummary(dailyWater, vars = "all", prjFile, HRUgroups, basinMean,
  quiet = TRUE, logfile = "")
```

Arguments

`dailyWater`	Required. The output from the `simpleDailyWater` command.
`vars`	Optional. Variable column numbers to be used (not including the datetime). The default value 'all' selects all columns.
`prjFile`	Required. The CRHM model .prj file.
`HRUgroups`	Required. This can be either a vector or list. If a vector, it contains strings used to group the selected HRU names in the .prj file. If a list, it contains vectors of the HRU numbers to be grouped. See the examples for both usages.
`basinMean`	Required. This refers to how the daily water values were calculated by `simpleDailyWater`. If `TRUE` then the aggregated water depths/volumes are computed over the whole basin. If `FALSE` then the the aggregated water depths/volumes are computed over area of the HRU groups.
`quiet`	Optional. Suppresses display of messages, except for errors. The default is TRUE. If you are calling this function in an R script, you will usually leave quiet=TRUE (i.e. the default). If you are working interactively, you will probably want to set quiet=FALSE.
`logfile`	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If successful returns a data frame containing each variable summarized by each HRU group. The names of the variables are in the form: `hruGroup.variable.group_total` or `hruGroup.variable.basin_total`. If unsuccessful, returns the value FALSE.

Examples

```
## Not run:
# First, calculate daily water values. In these examples, the water is
# in mm per HRU

#' b <- readOutputFile('Bologna1984_30y_output.txt', timezone='MST')
# get daily values without aggregation; values calculated w.r.t the HRU area
dailyHRU <- simpleDailyWater(b, prjFile='Bologna1984_02.prj', basinMean=FALSE)

# summarize by groups of HRUs with similar names
daily_group <- hruGroupSummary(daily, prjFile = 'Bologna1984_02.prj',
HRUgroups=c('firn','ice','bl'), basinMean=FALSE)

# summarize by named groups of HRUs with specified numbers
groups <- list(group1=c(1,2,3), group2=c(4,5,6))
daily_group <- hruGroupSummary(daily, prjFile = 'Bologna1984_02.prj',
HRUgroups=groups, basinMean=FALSE)

## End(Not run)
```

---

hydrograph *Plots hydrograph of CRHM output and/or WSC daily and/or peak flows*

Description

Creates a **ggplot** hydrograph from any of CRHM flows, WSC daily flows and/or WSC peak flows.

Usage

```
hydrograph(CRHMflows = NULL, CRHMflowsLabels = "", CRHMcols = NULL,
  CRHMdaily = FALSE, WSCdailyFlows = NULL, WSCdailyFlowsLabel = "",
  WSCpeakFlows = NULL, WSCpeakFlowsLabel = "",
  forceMissingPeakTimes = FALSE, commonTime = FALSE, fakeDates = FALSE,
  quiet = TRUE)
```

Arguments

`CRHMflows` Optional. A data frame of CRHM modelled flows. The flows must be in m^3/s.

`CRHMflowsLabels`

Optional. Labels for the CRHM data. If not specified, and CRHM data are plotted, then the name(s) of the CRHM variables will be used.

CRHMcols Required. Column(s) containing the flowrates. As always, the numbers do not include the datetime.

CRHMdaily Optional. Should CRHM flows be plotted as daily values? Default is FALSE.

WSCdailyFlows Optional. Dataframe containing WSC daily flows. The data frame is the same as is returned by the function DailyHydrometricData in the **HYDAT** package developed by David Hutchinson. The datframe has the columns station_number, date, value and flag. The date must be an R date.

WSCdailyFlowsLabel
Optional. Labels for the daily flows. If not specified, then the value in station_number will be used, followed by 'daily'.

WSCpeakFlows Optional. Dataframe containing WSC annual peak flows for a single station. The data frame is the same as is returned by the function AnnualPeakData in the **HYDAT** package developed by David Hutchinson. The data frame has the columns station_number, data_type, year, peak_code, precision_code, month, day, hour, minute, time_zone, peak, and symbol.

WSCpeakFlowsLabel
Optional. Labels for the annual peak flows. If not specified, then the value in station_number will be used, followed by 'annual peak'.

forceMissingPeakTimes
Optional. Some peaks may be missing their time of day and/or time zone. If TRUE, the missing peak times will be set to noon on the date of record. The missing time zone will be set to that of the other values or the user's time zone if there are none specified. If FALSE (the default value) peaks missing times and/or timezones will be deleted from the plot.

commonTime Optional. If set to TRUE then the hydrographs will only plotted over their common time range. Default is FALSE.

fakeDates Optional. If set to TRUE then all dates have their year replaced with 2000, and the actual year is added as a variable in the plotted data. This allows the plot to be faceted by year, as shown in the examples.

quiet Optional. Suppresses display of messages, except for errors. If you are calling this function in an R script, you will usually leave quiet=TRUE (i.e. the default). If you are working interactively, you will probably want to set quiet=FALSE.

Value

If successful, returns a **ggplot2** object. If unsuccessful, returns FALSE.

Note

The CRHM flows are plotted as lines, the daily flows are plotted as steps, and the annual peaks are plotted as points.

Author(s)

Kevin Shook

Examples

```
## Not run:
p <- hydrograph(BadLakeModel,'CRHM Bad Lake model',1, dailyFlows, '', peakFlows, '')
# once the ggplot graph has been returned, it can easily be modified:
mintime <- as.POSIXct(as.Date('1975-04-01', format='%Y-%m-%d'), tz='etc/GMT+6')
maxtime <- as.POSIXct(as.Date('1975-04-30', format='%Y-%m-%d'), tz='etc/GMT+6')
p <- p + xlim(mintime, maxtime) + ylim(0, 4)

# re-plot with fake dates
p2 <- hydrograph(BadLakeModel,'CRHM Bad Lake model',1, dailyFlows, '', peakFlows, '',
commonTime=TRUE, fakeDates=TRUE)
p2 <- p2 + facet_wrap(~year, scales='free_y', ncol=8)
# set axis limits to be the specified date range, and set labels to be the month names
mintime <- as.POSIXct(as.Date('2000-03-01', format='%Y-%m-%d'), tz='etc/GMT+6')
maxtime <- as.POSIXct(as.Date('2000-06-30', format='%Y-%m-%d'), tz='etc/GMT+6')
p2 <- p2 + scale_x_datetime(date_breaks = "1 month",
limits=c(mintime, maxtime), date_labels = "%b")

## End(Not run)
```

---

hydroYear *Returns a vector of the hydrological year for the input data frame*

Description

Calculates the hydrological year, based on the starting month.

Usage

```
hydroYear(CRHMdata, startMonth = 10, useSecondYear = TRUE, logfile = "")
```

Arguments

CRHMdata	Required. A **CRHMr** data frame.
startMonth	Optional. The starting month for the hydrological year. Default is 10 (October).
useSecondYear	Optional. Logical. Determines if the hydrological year is based on the first or second calendar year. In other words would the hydrological year for January 1, 2015 be 2014 or 2015? The default is TRUE (i.e., the hydrological year would be 2015). Note that the Campbell Scientific program SPLIT uses the first calendar year (i.e., the hydrological year would be 2014). To emulate this program, set useSecondYear to be FALSE.
logfile	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If successful, returns the hydrological year as a vector. If unsuccessful, returns the value FALSE.

Author(s)

Kevin Shook. Inspired by a function by Chris Marsh.

See Also

aggDataframe

Examples

```
# create a new data frame, then add the hydro year
BadLake <- BadLake7376
BadLake$hydroyear <- hydroYear(BadLake, startMonth=9)
```

---

impute *Imputes CRHM obs values*

Description

This function fills in missing (NA) values in a **CRHMr** obs data frame by imputation. The primary values are the data that will be infilled. The secondary values are those used to fill in the gaps. The secondary values are adjusted using the specified multipliers (regression slopes) and offsets (regression intercepts).

Usage

```
impute(primaryObs, primaryCols = 1, secondaryObs, secondaryCols = 1,
  multipliers = 1, offsets = 0, quiet = TRUE, logfile = "")
```

Arguments

primaryObs	Required. The primary **CRHMr** data frame of obs values.
primaryCols	Optional. A vector containing the columns to be imputed in the primary data frame, not including the datetime. If not specified, defaults to the first column.
secondaryObs	Required. The secondary CRHM obs data frame.
secondaryCols	Optional. A vector containing the columns to be imputed in the secondary data frame, not including the datetime. If not specified, defaults to the first column.
multipliers	Optional. A vector of multipliers applied to secondary observations. These may be obtained using the regress function. The default value is 1.
offsets	Optional. A vector of offsets added to secondary observations. These may be obtained using the regress function. The default value is 0.
quiet	Optional. Suppresses display of messages, except for errors. If you are calling this function in an R script, you will usually leave quiet=TRUE (i.e. the default). If you are working interactively, you will probably want to set quiet=FALSE.
logfile	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If successful returns a modified version of the primaryObs data frame. The missing values in the primary data frame are replaced by corresponding values from the secondary data frame, after adjustment using the specified multipliers and offsets. If unsuccessful, returns the value FALSE.

Note

In addition to the usual notation in the logfile, this function also writes a separate logfile which summarizes the primaryObs data frame, and the new infilled data frame. The summaries are also printed to the screen, if quiet=FALSE. The logfile also contains a complete listing of the source of each value in the infilled data frame. Each value is listed as being 'original' (unmodified from the primaryObs data frame), 'imputed' (derived from the adjusted values of the secondaryObs data frame) or 'NA' (missing).

Author(s)

Kevin Shook

See Also

interpolate regress distributeP

Examples

```
## Not run:
v.filled <- impute(veg, c(1,2,3), st,
c(2,3,1), c(0.895,0.708,1.1209), c(-0.8128, 0.0607, 9.005))
## End(Not run)
```

---

insertMissing *Inserts missing datetimes*

Description

Many time series have missing rows. This function finds missing rows, and adds rows with the appropriate datetime values and NA_real_ values for all variables. Once the missing values have been added, you can use interpolate or impute to infill them.

Usage

```
insertMissing(obs, quiet = TRUE, logfile = "")
```

Arguments

obs	Required. A **CRHMr** data frame containing obs values.
quiet	Optional. Suppresses display of messages, except for errors. If you are calling this function in an R script, you will usually leave quiet=TRUE (i.e. the default). If you are working interactively, you will probably want to set quiet=FALSE.
logfile	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If successful returns a modified version of the obs data frame with missing datetime values inserted. All of the missing variables in the inserted rows are replaced by NA_real_ values. If unsuccessful, returns the value FALSE.

Note

This function should be used before doing interpolation or imputation of variables. Note that this function is also called by the function findGaps.

Author(s)

Kevin Shook

See Also

findGaps interpolate impute

Examples

```
BadLake.withmissing <- insertMissing(BadLake7376, quiet=FALSE)
```

---

interpolate *Fills missing obs values by interpolation*

Description

Missing (NA) values in a **CRHMr** obs data frame are filled by linear or spline interpolation. The user can set the maximum gap length (in time steps) allowed to be filled. Note that this function will *NOT* interpolate RH or precipitation values.

Usage

```
interpolate(obs, varcols = 1, methods = "linear", maxlength = 5,
  quiet = TRUE, logfile = "")
```

Arguments

obs	Required. The **CRHMr** data frame of obs values.
varcols	Optional. A vector containing the columns to be imputed in the obs data frame, not including the datetime. If not specified, defaults to the first column.
methods	Optional. A vector containing the methods to be used for interpolation for each of the variables. Currently supported methods are 'linear' and 'spline'. The default is to use linear interpolation. If fewer methods than columns are specified, the methods are recycled.
maxlength	Optional. The maximum gap length to be interpolated. Defaults to 5 time steps.

quiet	Optional. Suppresses display of messages, except for errors. If you are calling this function in an R script, you will usually leave quiet=TRUE (i.e. the default). If you are working interactively, you will probably want to set quiet=FALSE.
logfile	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If successful, returns a modified version of the obs data frame. The missing values in the specified data frame are replaced by interpolated values. If unsuccessful, returns the value FALSE.

Note

In addition to the usual notation in the logfile, this function also writes a separate logfile which summarises the original obs data frame, and the new infilled data frame. The summaries are also printed to the screen, if quiet=FALSE. The logfile also contains a complete listing of the source of each value in the infilled data frame. Each value is listed as being 'original' (unmodified from the primary obsframe), 'linear interpolation' (infilled by linear interpolation), 'spline interpolation' (infilled by spline interpolation) or 'NA' (missing).

Author(s)

Kevin Shook

See Also

impute distributeP

Examples

```
## Not run:
v.filled <- interpolate(v, c(1,3))
## End(Not run)
```

---

keepGood *Keeps rows where there are some obs values*

Description

Removes rows where all values are missing, keeps the remainder.

Usage

```
keepGood(obs, quiet = TRUE, logfile = "")
```

Arguments

obs	Required. A **CRHMr** obs dataframe.
quiet	Optional. Suppresses display of messages, except for errors. If you are calling this function in an R script, you will usually leave quiet=TRUE (i.e. the default). If you are working interactively, you will probably want to set quiet=FALSE.
logfile	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If successful, returns the dataframe containing all rows with any non-missing values. If unsuccessful (including if there are only missing values in the original data), returns FALSE.

Author(s)

Kevin Shook

See Also

trimObs

Examples

```
BadLake.good <- keepGood(BadLake7376)
```

---

logAction *Writes a comment to the* **CRHMr** *log file*

Description

This is an internal **CRHMr** function and should normally not need to be called directly. This function is called by almost all **CRHMr** functions, and writes the comment, and the date and time to the logfile.

Usage

```
logAction(comment = "", logfile = "")
```

Arguments

comment	Required. Comment string to be written. This normally includes the function name, as well as the obs file being processed.
logfile	Optional. Name of the file to be used for logging the action. Normally not used. If not specified, then 'CRHMr.log' in the current working directory will be used.

Value

If successful, returns TRUE. If unsuccessful, returns FALSE.

Author(s)

Kevin Shook

Examples

```
## Not run:
 comment <- paste('tMinMaxToHourly dataframe:', obsName, sep='')
 result <- logAction(comment)
## End(Not run)
```

`longwave` *Calculates incoming longwave radiation*

Description

Calculates the incoming longwave radiation from the specified emissivity and air temperatures. This is an internal **CRHMr** function and should *never* need to be called directly, as it does *NO* data checking.

Usage

```
longwave(em, tK)
```

Arguments

`em`	Required. Emissivity (0-1).
`tK`	Required. Air temperatures in K.

Value

Returns the longwave radiation in W/m^2.

Author(s)

Kevin Shook

See Also

`distributeQli` `emissivity`

Examples

```
longwave(0.2, 293)
```

makeRegular *Makes the datetimes of an obs data frame fit the time step exactly*

Description

Makes the datetimes of an obs data frame fit the time step exactly

Usage

```
makeRegular(obs, timezone = "", quiet = TRUE, logfile = "")
```

Arguments

obs	Required. A **CRHMr** obs data frame.
timezone	Required. Timezone to be assigned to data.
quiet	Optional. Suppresses display of messages, except for errors. If you are calling this function in an R script, you will usually leave quiet=TRUE (i.e. the default). If you are working interactively, you will probably want to set quiet=FALSE.
logfile	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If successful, returns a modified version of the obs data frame containing the adjusted datetime and all of the variables. If unsuccessful, returns FALSE.

Author(s)

Kevin Shook

Examples

```
regular <- makeRegular(BadLake7376, timezone='CST')
```

maxObs *Sets maximum values for an obs data frame*

Description

Tests values in a **CRHMr** obs data frame to see if they exceed maximum thresholds. Values exceeding the thresholds can be set to either the maximum allowable value or to NA_real_, which is useful for infilling or imputing values.

Usage

```
maxObs(obs, varcols = "", maxvals = "", actions = "max", quiet = TRUE,
  logfile = "")
```

Arguments

obs	Required. The **CRHMr** obs data frame.
varcols	Optional. A vector containing the columns to be imputed in the obs data frame, not including the datetime. If not specified, defaults to all columns.
maxvals	Optional. A vector containing the maximum permissible values for each of the specified columns. If omitted, the default values are used.
actions	Optional. A vector containing the methods to be used for replacing values that exceed the maximum threshold values. Currently supported actions are 'max' (using the threshold) and 'NA' (inserting NA_real_ values). The default is 'max'. If fewer actions are specified than variables, then the actions are recycled.
quiet	Optional. Suppresses display of messages, except for errors. If you are calling this function in an R script, you will usually leave quiet=TRUE (i.e. the default). If you are working interactively, you will probably want to set quiet=FALSE.
logfile	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If successful, returns a modified version of the obs data frame. The values exceeding the specified max values are replaced by either NA_real_ or the threshold max values. If unsuccessful, returns the value FALSE.

Note

The default threshold values are

t	40 C
ea	3.5 kPa
rh	100 percent
ppt	100 mm
p	20 mm
u	10 m/s
SunAct	18 hr
qsi	1262 W/m^2
qso	1000 W/m^2
qn	250 W/m^2

Author(s)

Kevin Shook

See Also

minObs

Examples

```
# use all of the default values
```

```
bad.max <- maxObs(BadLake7376)
summary(bad.max)
# use specified columns with default max values, replace with 'NA' values
bad.max2 <- maxObs(BadLake7376, varcols=c(1,2), actions='NA')
# use specfied columns with specified max values and actions
bad.max3 <- maxObs(BadLake7376, maxvals=c(35, 10), varcols=c(1,3), actions=c('NA', 'max'))
summary(bad.max3)
```

`minObs` *Sets minimum values for an obs data frame*

Description

Tests values in a **CRHMr** obs data frame to see if they exceed minimum thresholds. Values exceeding the thresholds can be set to either the minimum allowable value or to `NA_real`, which is useful for infilling or imputing values.

Usage

```
minObs(obs, varcols = "", minvals = "", actions = "min", quiet = TRUE,
  logfile = "")
```

Arguments

`obs`	Required. The **CRHMr** obs data frame.
`varcols`	Optional. A vector containing the columns to be imputed in the obs data frame, not including the datetime. If not specified, defaults to all columns.
`minvals`	Optional. A vector containing the minimum permissible values for each of the specified columns. If omitted, the default values are used.
`actions`	Optional. A vector containing the methods to be used for replacing values that exceed the minimum threshold values. Currently supported actions are '`min`' (using the threshold) and '`NA`' (using `NA_real_` values). The default is '`min`'. If fewer actions are specified than variables, then the actions are recycled.
`quiet`	Optional. Suppresses display of messages, except for errors. If you are calling this function in an R script, you will usually leave `quiet=TRUE` (i.e. the default). If you are working interactively, you will probably want to set `quiet=FALSE`.
`logfile`	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If successful, returns a modified version of the obs data frame. The values smaller than the specified `min` values are replaced by either `NA_real_` or the threshold `min` values. If unsuccessful, returns the value FALSE.

Note

The default threshold values are

t	-40 C
ea	0.01 kPa
rh	0.5 percent
ppt	0 mm
p	0 mm
u	0 m/s
SunAct	0 hr
qsi	0 W/m^2
qso	0 W/m^2
qn	-60 W/m^2

Author(s)

Kevin Shook

See Also

maxObs

Examples

```
# use all of the default values
bad.min <- minObs(BadLake7376)
# use specified columns with default min values, replace with 'NA' values
bad.min2 <- minObs(BadLake7376, varcols=c(1,2), actions='NA')
# use specfied columns with specified min values and actions
bad.min3 <- minObs(BadLake7376, minvals=c(-30, 22), varcols=c(1,2), actions=c('min', 'NA'))
```

---

monthlyPrecipTotals *Returns mean monthly precipitation totals*

Description

Calculates mean total precipitation for each calendar month in the observation data frame. Optionally plots the values. Can work with p and/or ppt values.

Usage

```
monthlyPrecipTotals(obs, precipCols = 0, plot = TRUE, logfile = "")
```

Arguments

obs	Required. A **CRHMr** data frame containing p and/or ppt values..
precipCols	Optional. A vector containing the columns to be analyzed, not including the datetime column. If not specified, all precipitation columns are used.
plot	Optional. If TRUE, which is the default, a **ggplot2** object of the monthly mean precipitations is returned. If FALSE, the values are returned as a data frame.
logfile	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If successful, returns either a data frame or a **ggplot2** object of the monthly mean precipitations. If unsuccessful, returns FALSE.

Author(s)

Kevin Shook

See Also

aggDataframe monthlyQQplot

Examples

```
## Not run:
marmot_monthly_plot <- monthlyPrecipTotals(marmot)
marmot_monthly_means <- monthlyPrecipTotals(marmot, plot=FALSE)
## End(Not run)
```

---

monthlyQQplot *Quantile-quantile plot by month*

Description

Does QQ plots of monthly values of a single variable type in two data frames.

Usage

```
monthlyQQplot(primaryCRHM, primaryCol = 1, secondaryCRHM, secondaryCol = 1,
  samePeriod = TRUE, logfile = "")
```

Arguments

primaryCRHM	Required. The primary **CRHMr** data frame. Quantiles of this data will be plotted on the X axis.
primaryCol	Optional. The column in the primary data frame, not including the datetime. If not specified, defaults to the first column.
secondaryCRHM	Required. The secondary **CRHMr** data frame. Quantiles of this data will be plotted on the Y axis.
secondaryCol	Optional. The column in the secondary data frame, not including the datetime. If not specified, defaults to the first column.
samePeriod	Optional. Logical. Should the same period of time be used for both variables? Default is TRUE.
logfile	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If successful, returns a **ggplot2** object of faceted monthly QQ plots (3 rows x 4 columns) of the specified variables. If unsuccessful, returns FALSE.

Author(s)

Kevin Shook

See Also

monthlyPrecipTotals

Examples

```
BadLake <- BadLake7376
BadLake$year <- as.numeric(format(BadLake7376$datetime, format='%Y'))
badlake73 <- subset(BadLake, year==1973)
badlake75 <- subset(BadLake, year==1975)
p <- monthlyQQplot(badlake73, 1, badlake75, 1, samePeriod=FALSE)
print(p)
```

---

nancumsum *Finds cumulative sum of non-missing values in an array*

Description

Finds cumulative sum of non-missing values in an array

Usage

```
nancumsum(x)
```

Arguments

x	Required. An array or vector of numeric values

Value

Returns an array or vector with cumulative sums of each column.

Author(s)

Kevin Shook. The code is copied from a MATLAB program nancumsum.m written by Alan Barr.

See Also

PcpFiltPosTh

Examples

```
a <- nancumsum(BadLake7376[,2])
```

---

parseNums *Parses a string containing numbers*

Description

This is an internal **CRHMr** function and should normally not need to be called directly. It is used for reading data from messy files such as CRHM projects.

Usage

```
parseNums(numString)
```

Arguments

numString	Required. A character string containing numbers separated by any number of spaces.

Value

Returns a numeric vector.

Author(s)

Kevin Shook

Examples

```
parseNums(' 1  2 3   4     5 ')
```

---

parseText *Parses a string containing several sub-strings*

Description

This is an internal **CRHMr** function and should normally not need to be called directly. It is used for reading data from messy files such as CRHM projects.

Usage

```
parseText(textString)
```

Arguments

textString	Required. A character string containing strings separated by any number of spaces.

Value

Returns a character vector.

Author(s)

Kevin Shook

Examples

```
parseText(' red  green    blue      black')
```

PcpFiltPosTh *Cleans up a cumulative precipitation time series*

Description

Cleans up a cumulative precipitation time series by transferring changes below a specified threshold to neighbouring periods, and eliminating large negative changes associated with gauge servicing (bucket emptying). The filtering is done using a brute-force algorithm that identifies small or negative changes (below dPcpTh) then transfers them to neighbouring positive changes thus aggregating all changes to values above dPcpTh. The transfers are made in ascending order, starting with the lowest (most negative). The cumulative total remains unchanged. This description (and most of the comments in the code) were written by Alan Barr.

Usage

```
PcpFiltPosTh(Pcp, dPcpTh = 0.1, dpServicingTh = -100, quiet = TRUE)
```

Arguments

Pcp	Required. Measured cumulative precipitation (mm).
dPcpTh	Optional. Minimum interval precipitation (mm). Default is 0.1 mm.
dpServicingTh	Optional. Threshold for change in storage due to servicing (mm). Default is -100 mm.
quiet	Optional. Suppresses display of messages, except for errors. If set quiet=FALSE, the output will pause for 5 seconds ever 100 iterations, which may be useful for debugging your parameters. If you are calling this function in an R script, you will usually leave quiet=TRUE (i.e. the default), which causes fewer parameters to be output, without pausing, every 1000 iterations.

Value

If unsuccessful, returns the value FALSE. If successful, returns a time series of cleaned values. Writes parameter values to screen.

Author(s)

Kevin Shook. The code is copied from a MATLAB program PcpFiltPosTh.m written by Alan Barr, 28 Aug 2012.

Examples

```
## Not run:
testp <- PcpFiltPosTh(dl2$precip, dPcpTh = 0.05, dpServicingTh = -50)
## End(Not run)
```

---

phaseCorrect *Corrects precipitation for undercatch and calculates phase*

Description

Applies an iterative solution to calculate the psychrometric hydrometeor temperature. The hydrometeor temperature is then used to partition the raw precipitation into rainfall and snowfall. The snowfall is adjusted for gauge undercatch, if selected.

Usage

```
phaseCorrect(obs, Tcol = 1, RHcol = 2, Ucol = 3, Pcol = 4,
  RH_type = 1, shield = 1, quiet = TRUE, logfile = "")
```

Arguments

obs	Required. A **CRHMr** data frame of observations.
Tcol	Required. Column number containing the air temperature (not including the datetime). Default is column 1. The values must be in °C.
RHcol	Required. Column number containing the RH (not including the datetime). Default is column 2. The values must be as percentages.
Ucol	Required. Column number containing the wind speeds (not including the datetime). Default is column 3. The values must be in m/s.
Pcol	Required. Column number containing the precipitaion (not including the datetime). Default is column 4. The values must be in mm.
RH_type	Optional. Set to 1 if RH is relative to water, any othe value if relative to ice. Default is 1.
shield	Optional. Set to 1 to use Alter shield undercatch correction from MacDonald and Pomeroy (2007). Set to 2 to use Alter shield undercatch correction from Smith. Set to any other value for unadjusted precipitation. Default is 1.

quiet	Optional. Suppresses display of messages, except for errors. If you are calling this function in an R script, you will usually leave quiet=TRUE (i.e. the default). If you are working interactively, you will probably want to set quiet=FALSE.
logfile	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If unsuccessful returns FALSE. If successful, returns the original dataframe with additional columns for the rain, snow, total precipitation and rain ratio.

Author(s)

Phillip Harder

References

Harder, P., and J. Pomeroy (2013), "Estimating precipitation phase using a psychrometric energy balance method", Hydrol. Process., 27(13), 1901-1914, doi:10.1002/hyp.9799.

Macdonald, J., and J. Pomeroy (2007), "Gauge Undercatch of Two Common Snowfall Gauges in a Prairie Environment", Proc. 64th East. Snow Conf. St. John's, Canada., 119-126.

Examples

```
corrected <- phaseCorrect(BadLake7376, Tcol=1, RHcol=2, Ucol=3, Pcol=5)
```

---

plotObs — *Creates a* **ggplot** *object from an obs data frame*

Description

This function creates a **ggplot2** object from an obs data frame. If the specified variables are all of single type (e.g. temperature), it returns a single-pane plot. If the specified variables are of more than one type (e.g. temperature, precipitation), then the plot is faceted. The function returns an object that you can modify, or save with the standard **ggplot2** commands.

Usage

```
plotObs(obs, varcols = c(0), startDate = "", endDate = "",
  plotType = "lines")
```

Arguments

obs	Required. The **CRHMr** obs data frame to be plotted.
varcols	Optional. A vector containing the numbers of the columns (not including the datatime) to be plotted. If not specified, all variables are plotted.

startDate	Optional. The starting date for the plot. Can either be a year as a number (e.g. 1995) or a date in 'Y-m-d' format, i.e. '1995-06-01'. If not specified, the first datetime value in the obs data is used.
endDate	Optional. The ending date for the plot. Can either be a year as a number (e.g. 1995) or a date in 'Y-m-d' format, i.e. '1995-12-31'. If not specified, the first datetime value in the obs data is used.
plotType	Optional. Defines the type of the plot. Either 'lines' or 'points'. Default is 'lines'. If you want to show the extent of data, use 'points' as lines will NOT be drawn when points are separated by NA_real_ values.

Details

Units are automatically specified for obs variables of the following types: t, rh, u, p, ea, and solar radiation.

Value

If successful returns a **ggplot2** object. If unsuccessful, returns the value FALSE.

Note

Note that missing (NA_real_) values in the obs data frame will result in a warning message.

Author(s)

Kevin Shook

See Also

summariseObsFiles

Examples

```
p <- plotObs(BadLake7376)
print(p)
```

---

plotPrecipsByYear *Creates a* **ggplot2** *object of cumulative precips by year*

Description

Calculates cumulative precipitations and creates a **ggplot2** faceted by year. Note that this function would work for any other cumulative variable, if the Y-axis label is changed.

Usage

```
plotPrecipsByYear(obs = "", obsNames = "", precipCols = 1,
  commonDates = TRUE)
```

Arguments

obs	Required. Either a **CRHMr** obs data frame or a list of data frames.
obsNames	Optional. A vector containing names for the data frames. If there are more data frames than names, the data will be named obs 1, obs 2 etc.
precipCols	Optional. Column(s) containing precipitation. Can be a single value or a vector. Values will be recycled if necessary. Default is 1.
commonDates	Optional. If TRUE (the default) then the range of dates will be restricted to the common first and last dates of the datasets.

Value

If successful, returns a faceted **ggplot2** object of cumulative precipitation vs time. If unsuccessful, returns FALSE.

Author(s)

Kevin Shook.

See Also

plotTempsByYear

Examples

```
## Not run:  l <- list(meas, WRF)
names <- c("Measured", "WRF")
p <- plotPrecipsByYear(l, names)
library(ggplot2)
p <- p + scale_colour_manual(values=c("red", "blue")
## End(Not run)
```

---

plotTempsByYear *Creates a* **ggplot2** *object of daily mean air temps by year*

Description

Aggregates air temeratures to mean daily values and creates a **ggplot2** faceted by year. Note that this function would work for any other non-cumulative variable, if the Y-axis label is changed.

Usage

```
plotTempsByYear(obs = "", obsNames = "", tempCols = 1,
  commonYears = TRUE)
```

Arguments

obs	Required. Either a **CRHMr** obs data frame or a list of data frames.
obsNames	Optional. A vector containing names for the data frames. If there are more data frames than names, the data will be named obs 1, obs 2 etc.
tempCols	Optional. Column(s) containing air temperatures. Can be a single value or a vector. Values will be recycled if necessary. Default is 1.
commonYears	Optional. If TRUE (the default) then the range of years will be restricted to the common first and last years.

Value

If successful, returns a faceted **ggplot2** object of mean daily air temperatures vs time. If unsuccessful, returns FALSE.

Author(s)

Kevin Shook.

See Also

plotPrecipsByYear

Examples

```
## Not run:  l <- list(meas, WRF)
names <- c("Measured", "WRF")
p <- plotTempsByYear(l, names)
library(ggplot2)
p <- p + scale_colour_manual(values=c("red", "blue")
## End(Not run)
```

---

qair2rh *qair2rh*

Description

From Bolton 1980 The computation of Equivalent Potential Temperature http://www.eol.ucar.edu/projects/ceop/dm/documents/refdata_report/eqns.html.

Usage

```
qair2rh(qair, temp, press = 1013.25)
```

Arguments

qair	specific humidity, dimensionless (e.g. kg/kg) ratio of water mass / total air mass
temp	degrees C
press	pressure in mb

Details

Convert specific humidity to relative humidity

Value

rh relative humidity, ratio of actual water mixing ratio to saturation mixing ratio

Author(s)

David LeBauer https://github.com/PecanProject/pecan/blob/master/modules/data.atmosphere/R/metutils.R. Modified by Kevin Shook for air temps < 0 °C and to work for vectors or scalars.

Examples

```
qair2rh(0.01, 10)
```

---

qqplotValues *Calculates quantile values for a Quantile-Quantile plot*

Description

The built-in `qqplot` function does not work with **ggplot2**. This just calls the qqplot function to caclulate the quantiles without plotting them.

Usage

```
qqplotValues(x, y)
```

Arguments

x	Required. A numeric vector.
y	Required. A numeric vector.

Value

Returns a dataframe with the quantiles of x and y.

Author(s)

Kevin Shook

Examples

```
quantiles <- qqplotValues(runif(20), runif(50))
```

`readCampbell` *Reads in Campbell datalogger data*

Description

Reads a datafile produced by Campbell Scientific dataloggers into a **CRHMr** data frame. Both old (multiple table, no header) and new (single table, with header) data files can be read in.

Usage

```
readCampbell(campbellFile, missingValue = -6999, timezone = "",
  quiet = TRUE, logfile = "")
```

Arguments

`campbellFile`	Required. The name of the file containing the Campbell Scientific datalogger output.
`missingValue`	Optional. Default is -6999. All values less than or equal to this will be set to be `NA_real_` .
`timezone`	Required. The name of the timezone of the data as a character string. This should be the timezone of your data, but omitting daylight savings time. Note that the timezone code is specific to your OS. To avoid problems, you should use a timezone without daylight savings time. Under Linux, you can use '`CST`' and '`MST`' for Central Standard or Mountain Standard time, respectively. Under Windows or OSX, you can use '`etc/GMT+6`' or '`etc/GMT+7`' for Central Standard and Mountain Standard time. DO NOT use '`America/Regina`' as the time zone, as it includes historical changes between standard and daylight savings time.
`quiet`	Optional. Suppresses display of messages, except for errors. If you are calling this function in an R script, you will usually leave `quiet=TRUE` (i.e. the default). If you are working interactively, you will probably want to set `quiet=FALSE`.
`logfile`	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If unsuccessful, returns `FALSE`. If successful, returns a **CRHMr** data frame. If there are two tables in the campbellFile, then they are returned as a list.

Note

If the campbellFile does not contain headers, then the variable names will be assigned by `R`. In any case, you will probably want to rename the variables using the command `names()`.

Author(s)

Kevin Shook

See Also

readObsFile readExportFile readOutputFile

Examples

```
## Not run:
# read in an old format Campbell file
oldformatdata <- readCampbell('OldDataLogger.dat', timezone='MST')
# now extract the data frames from the returned list
obs1 <- oldfomatdata[[1]]
obs2 <- oldformatdata[[2]]
# read in a new format Campbell file
obs3 <- readCampbell('BowHut_Fifteen.dat', timezone='MST')
## End(Not run)
```

---

readCLASSfile *Reads forcing data from CLASS formatted file*

Description

Reads data from files created for CLASS into a stanard **CRHMr** obs data frame.

Usage

```
readCLASSfile(CLASSfile = "", timezone = "", convertNames = TRUE,
  quiet = TRUE, logfile = "")
```

Arguments

CLASSfile	Required. File of meteorological data in CLASS format.
timezone	Required. The name of the timezone of the data as a character string. This should be the timezone of your data, but omitting daylight savings time. Note that the timezone code is specific to your OS. To avoid problems, you should use a timezone without daylight savings time. Under Linux, you can use 'CST' and 'MST' for Central Standard or Mountain Standard time, respectively. Under Windows or OSX, you can use 'etc/GMT+6' or 'etc/GMT+7' for Central Standard and Mountain Standard time. DO NOT use 'America/Regina' as the time zone, as it includes historical changes between standard and daylight savings time.
convertNames	Optional. If TRUE, then the CLASS variables will be converted to Qsi.1, Qli.1, p.1, t.1, ea.1, u.1, and AirPressure.1. Otherwise, the CLASS names SWin, LWin, Prec, T, qa, U, and P will be used.
quiet	Optional. Suppresses display of messages, except for errors. If you are calling this function in an R script, you will usually leave quiet=TRUE (i.e. the default). If you are working interactively, you will probably want to set quiet=FALSE.
logfile	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If successful, returns a **CRHMr** data frame. If unsuccessful, returns the value FALSE.

Author(s)

Kevin Shook, with assistance from Andrew Ireson.

Examples

```
## Not run:
class <- readCLASSfile('driv.met', timezone='Etc/GMT+7')
## End(Not run)
```

---

readDebugFile *Reads in CRHM debug file*

Description

Reads in CRHM debug file

Usage

```
readDebugFile(debugFile = "", returnData = "all")
```

Arguments

debugFile	Required. CRHM debug file name.
returnData	Optional. The data to be returned. If 'all' (the default) then all sections will be returned as a list of data frames. Other options are 'header', 'initalconditions', 'finalconditions', 'basinsummary', and 'groupsummary'. Note that if 'groupsummary' is specified, and there are no groups, then an error will result.

Value

If successful, returns a list of data frames, or a single specified data frame. If unsuccessful, returns FALSE.

Examples

```
## Not run:
allDebug <- readDebug("07-08_debug.txt")
# show header
View(allDebug$header)
# show group summary
View(allDebug$groupSummary)
## End(Not run)
```

readExportFile *Reads exported CRHM output into a data frame*

Description

Reads output from a CRHM model, that has been manually exported, into a **CRHMr** data frame. CRHM model output can be exported in several formats. This function is intended to read in all of them. Note that because of the way that R reads in files, the variable names will have appended periods, e.g. `t.1`.

Usage

```
readExportFile(exportFile, timezone = "", quiet = TRUE, logfile = "")
```

Arguments

`exportFile`	Required. The name of the CRHM export file to be read in.
`timezone`	Required. The name of the timezone of the data as a character string. This should be the timezone of your data, but omitting daylight savings time. Note that the timezone code is specific to your OS. To avoid problems, you should use a timezone without daylight savings time. Under Linux, you can use 'CST' and 'MST' for Central Standard or Mountain Standard time, respectively. Under Windows or OSX, you can use 'etc/GMT+6' or 'etc/GMT+7' for Central Standard and Mountain Standard time. DO NOT use 'America/Regina' as the time zone, as it includes historical changes between standard and daylight savings time.
`quiet`	Optional. Suppresses display of messages, except for errors. If you are calling this function in an R script, you will usually leave `quiet=TRUE` (i.e. the default). If you are working interactively, you will probably want to set `quiet=FALSE`.
`logfile`	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If successful, returns a **CRHMr** data frame. If unsuccessful, returns the value `FALSE`.

Author(s)

Kevin Shook

See Also

`readOutputFile` `readObsFile`

Examples

```
## Not run:
brandon <- readExportFile('Brandon_export.txt', 'etc/GMT+6')
## End(Not run)
```

`readObsFile` *Reads a CRHM obs fileinto a dataframe*

Description

Reads a file of CRHM observation data into a **CRHMr** data frame. Note that because of the way that R reads in files, the variable names will have appended periods, e.g. `t.1`.

Usage

```
readObsFile(obsFile = "", timezone = "", quiet = TRUE, logfile = "")
```

Arguments

`obsFile`	Required. Name of the CRHM obs file to be read in.
`timezone`	Required. The name of the timezone of the data as a character string. This should be the timezone of your data, but omitting daylight savings time. Note that the timezone code is specific to your OS. To avoid problems, you should use a timezone without daylight savings time. Under Linux, you can use '`CST`' and '`MST`' for Central Standard or Mountain Standard time, respectively. Under Windows or OSX, you can use '`etc/GMT+6`' or '`etc/GMT+7`' for Central Standard and Mountain Standard time. DO NOT use '`America/Regina`' as the time zone, as it includes historical changes between standard and daylight savings time.
`quiet`	Optional. Suppresses display of messages, except for errors. If you are calling this function in an R script, you will usually leave `quiet=TRUE` (i.e. the default). If you are working interactively, you will probably want to set `quiet=FALSE`.
`logfile`	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If successful, returns a **CRHMr** data frame. If unsuccessful, returns the value `FALSE`.

Author(s)

Kevin Shook

See Also

`readExportFile` `readOutputFile`

Examples

```
# output example obs data to a file
BadLake7376.obs <- writeObsFile(BadLake7376, 'BadLake7376.obs')
# now read data back in
BadLake <- readObsFile('BadLake7376.obs', 'etc/GMT+6')
```

`readOutputFile` *Reads CRHM model output into a data frame*

Description

Reads a file containing output from a CRHM model into a **CRHMr** data frame. Note that because of the way that R reads in files, the variable names will have appended periods, e.g. `t.1`. This version reads in both old CRHM output (without units) and new CRHM output which contains the variable units in the second line.

Usage

```
readOutputFile(outputFile, timezone = "", quiet = TRUE, logfile = "")
```

Arguments

`outputFile`	Required. The name of the CRHM output file to be read in.
`timezone`	Required. The name of the timezone of the data as a character string. This should be the timezone of your data, but omitting daylight savings time. Note that the timezone code is specific to your OS. To avoid problems, you should use a timezone without daylight savings time. Under Linux, you can use '`CST`' and '`MST`' for Central Standard or Mountain Standard time, respectively. Under Windows or OSX, you can use '`etc/GMT+6`' or '`etc/GMT+7`' for Central Standard and Mountain Standard time. DO NOT use '`America/Regina`' as the time zone, as it includes historical changes between standard and daylight savings time.
`quiet`	Optional. Suppresses display of messages, except for errors. If you are calling this function in an R script, you will usually leave `quiet=TRUE` (i.e. the default). If you are working interactively, you will probably want to set `quiet=FALSE`.
`logfile`	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If successful, returns a **CRHMr** data frame. If unsuccessful, returns the value `FALSE`.

Author(s)

Kevin Shook

See Also

`readExportFile` `readObsFile` `readOutputUnits`

Examples

```
## Not run:
stoon <- readOutputFile('CRHM_output_1.txt', 'etc/GMT+6')
## End(Not run)
```

`readOutputUnits` *Reads variable units*

Description

Gets units for all variables in a specified CRHM output file. If the file is the new CRHM format, then the units are read directly from the second line of the file. If the file is in the older CRHM format, then units are found by matching the variables to those in the internal data frame `CRHM_vars`.

Usage

```
readOutputUnits(outputFile, quiet = TRUE)
```

Arguments

`outputFile`	Required. A CRHM output file.
`quiet`	Optional. Suppresses display of messages, except for errors. If you are calling this function in an R script, you will usually leave `quiet=TRUE` (i.e. the default). If you are working interactively, you will probably want to set `quiet=FALSE`.

Value

If successful, returns a **CRHMr** data frame containing the name of each variable and its units. If unsuccessful, returns the value `FALSE`.

Author(s)

Kevin Shook

See Also

`readOutputFile`

Examples

```
## Not run:
units <- readOutputFile('CRHM_output_1.txt')
## End(Not run)
```

readPrjNumVals *Reads numeric values from a CRHM project text object*

Description

Reads specified number of numeric values from a project file variable.

Usage

```
readPrjNumVals(prj, searchString, value_count)
```

Arguments

prj	Required. A CRHM file read into memory as a sequence of lines, as returned from readPrj.
searchString	Required. The identifying string.
value_count	Required. The number of values to be read in.

Value

If successful, returns a vector with the specified number of values. If unsuccessful, returns the value FALSE.

Examples

```
## Not run:
hru_area <- readPrjNumVals(Bologna,'hru_area',19)
## End(Not run)
```

readPrjOutputVariables *Reads output variables from a CRHM model .prj file*

Description

Reads output variables from a CRHM model .prj file

Usage

```
readPrjOutputVariables(prjFile, asDataframe = FALSE, logfile = "")
```

Arguments

prjFile	Required. Name of CRHM .prj file to read.
asDataframe	Optional. If FALSE (the default), returns the set of output variables as a vector. If TRUE, returns a data frame containing the module, variable and HRUs as a string. The vector formis useful if you want to delete some variables. The data frame is more useful if you want to change the variable or its HRUs.
logfile	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If successful, returns the output variables. If unsuccessful, returns FALSE.

Author(s)

Kevin Shook

See Also

setPrjOutputVariables

Examples

```
## Not run:
# read in as a vector
variables <- readPrjOutputVariables('Bad_Lake_1974-1975.prj')

# read in as a data frame
variables <- readPrjOutputVariables('Bad_Lake_1974-1975.prj', asDataframe=TRUE)
# change value
variables$HRUs[1] <- '1 2'
## End(Not run)
```

---

readPrjTextVals *Reads text values from a CRHM project text object*

Description

Reads numeric values from a project file variable. The number of values to be read in does not need to be specified.

Usage

```
readPrjTextVals(prj, searchString, valueCount)
```

Arguments

prj	Required. A CRHM file read into memory as a sequence of lines, as returned from readPrj.
searchString	Required. The identifying string.
valueCount	Required. The number of values to be read in.

Value

If successful, returns a vector with the numeric values. If unsuccessful, returns the value FALSE.

Examples

```
## Not run:
hru_names <- readPrjTextVals(Bologna,'hru_names',19)
## End(Not run)
```

---

regress *Regresses one set of CRHM observations against another*

Description

The purpose of this function is to generate the multiplier and an offset values (if required) used by the impute function. Therefore, it performs a linear (least-squares) regression of the primary dataset (the values to be produced by imputation) against the secondary dataset. The function removes missing values and merges the data sets together before doing the regression to ensure that the variables are aligned by datetime.

Usage

```
regress(primaryCRHM, primary.columns = 1, secondaryCRHM,
  secondary.columns = 1, forceOrigin = FALSE, plot = FALSE,
  quiet = TRUE, logfile = "")
```

Arguments

primaryCRHM	Required. The primary **CRHMr** data frame.
primary.columns	Optional. Optional. A vector containing the columns to be imputed in the primary data frame, not including the datetime. If not specified, defaults to the first column.
secondaryCRHM	Required. The secondary **CRHMr** data frame.
secondary.columns	Optional. A vector containing the columns to be imputed in the secondary data frame, not including the datetime. If not specified, defaults to the first column.

forceOrigin	Optional. If TRUE then the regressions will be forced through the origin. Note that in this case the values of r^2 and the intercept will *not* be calculated as they have no meaning.
plot	Optional. Set plot=TRUE if you want a plot for each regression. Default is FALSE. The function has to guess the axis units from the variable names and it may be mistaken; you can always change the labels.
quiet	Optional. Suppresses display of messages, except for errors. If you are calling this function in an R script, you will usually leave quiet=TRUE (i.e. the default). If you are working interactively, you will probably want to set quiet=FALSE.
logfile	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If successful, and plot=TRUE, returns a plot of all of the regressions. If successful, and plot=FALSE, returns a data frame containing the regression parameters. If unsuccessful, returns FALSE.

Note

If successful, exports a data frame with the columns xvals (secondary data frame), yvals (primary data frame), first date (in common data frame), last date (in common data frame), and slope for each variable to the regression file. If forceOrigin=FALSE then the parameters intercept, and r2 will also be returned. This function will not permit regression on RH values, so convert these to Ea values first.

Author(s)

Kevin Shook

See Also

impute

Examples

```
# this would be used to get the regression coefficients used to impute data from Saskatoon
# to fill gaps in the Regina data
## Not run:
reg <- regress(regina, c(1,2,3), saskatoon, c(1,2,3))
## End(Not run)
```

---

rh2vp *Converts air temperature and relative humidity to vapour pressure*

Description

This is an internal **CRHMr** function and should *never* need to be called directly, as it does *NO* data checking. Use the function changeRHtoEa instead.

Usage

```
rh2vp(airtemp, rh)
```

Arguments

airtemp	Required. Air temperature in °C.
rh	Required. Relative humidity in percent.

Value

Returns vapour pressure in kPa. If the air temperatures are scalar, returns a scalar value. If the air temperatures are a vector, returns a vector.

Author(s)

Kevin Shook

See Also

changeRHtoEa

Examples

```
rh2vp(-5, 50)
```

---

ribbonPlotAirTemps *Plots min, max and mean air temperatures for each day of the year*

Description

Creates **ggplot2** ribbon plots of the daily minimum, maximum and mean values of air temperatures over a year. The shaded area indicates the range of values in the data frame for each day.

Usage

```
ribbonPlotAirTemps(obs = "", obsNames = "", tempCols = 1, facet = TRUE,
  facetColour = "", facetCols = 2)
```

Arguments

obs	Required. Either a **CRHMr** obs data frame or a list of data frames.
obsNames	Optional. A vector containing names for the data frames. If there are more data frames than names, the data will be named obs 1, obs 2 etc.
tempCols	Optional. Column(s) containing temperatures. Can be a single value or a vector. Values will be recycled if necessary. Default is 1.
facet	Optional. If FALSE (the default), then all variables will be on a single plot. If TRUE, then each variable will be on a separate facet.

facetColour	Optional. The colour to be used for shading the ribbon plot in each facet. If not specified (the default), then each plot will be coloured separately.
facetCols	Optional. The number of columns to be used for the facets, if selected.

Value

If unsuccessful, returns FALSE. If successful, returns a **ggplot2** object.

Author(s)

Kevin Shook

See Also

plotTempsByYear

Examples

```
p <- ribbonPlotAirTemps(BadLake7376, tempCols = 1, facet=FALSE)
```

---

runCRHM *Runs a CRHM model*

Description

Runs CRHM with a specified .prj file and (optionally) .obs file(s) and .par file(s). Under OSX and Linux, CRHM requires either 1) the program Wine to run CRHM.exe or 2) a native-compiled version. This function will use Wine if it is required. Note that none of the file names or their paths can contain spaces.

Usage

```
runCRHM(CRHMfile = "", prjFile = "", obsFiles = "", parFiles = "",
  outFile = "", logfile = "")
```

Arguments

CRHMfile	Required. CRHM executable file (usually CRHM.exe.
prjFile	Required. Name of .prj file. If the file does not contain any displayed observations or variables, or if any of the settings Auto_Run, Auto_Exit or Log_All are missing, then the function will terminate with an error message.
obsFiles	Optional. Name(s) of obs file(s).
parFiles	Optional. Name(s) of parameter file(s).
outFile	Optional. Name(s) of output file(s). If not specified, the original CRHM output file name (e.g. 'CRHM_output_1.txt') is kept.
logfile	Optional. Name of the file to be used for logging the action. Normally not used.

Details

If Wine is required, the function will copy the specified .obs, and .par files to the directory of the .prj file, before execution of CRHM. After the program has finished, the copies will be deleted. Running a CRHM model requires that the .prj file has been setup to run autmatically, as shown in the function automatePrj.

Value

If successful, returns TRUE. If unsuccessful, returns FALSE.

Author(s)

Kevin Shook

See Also

automatePrj

Examples

```
## Not run:
result <- runCRHM('c:/CRHM/CRHM.exe', 'c:/BadLake/BadLake1975.prj',
outfile='c:/BadLake/BadLake1975Output.txt')
## End(Not run)
```

---

saturatedVP *Converts calculates saturated vapour pressure from air temperature*

Description

This is an internal **CRHMr** function and should normally not need to be called directly.

Usage

```
saturatedVP(airTemp)
```

Arguments

airTemp	Required. Air temperature in °C.

Value

If successful, returns the saturated vapour pressure in kPa. If unsuccessful, returns the value FALSE.

Author(s)

Kevin Shook

Examples

```
saturatedVP(-5)
```

`setPrjBasinName` *Sets the basin name in a CRHM model* `.prj` *file*

Description

Sets the basin name in a CRHM model `.prj` file

Usage

```
setPrjBasinName(inputPrjFile = "", basinName = "", outputPrjFile = "",
  logfile = "")
```

Arguments

`inputPrjFile`	Required. Name of the `.prj` file.
`basinName`	Required. New name for basin.
`outputPrjFile`	Optional. If omitted, the input `.prj` file will be overwritten.
`logfile`	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If successful, returns `TRUE`. If unsuccessful, returns `FALSE`.

Author(s)

Kevin Shook

See Also

`runCRHM` `setPrjHRUnames` `setPrjDates` `setPrjParameters`

Examples

```
## Not run:
result <- setPrjBasinName('c:/CRHM/Bad Lake 1974-1975.prj', 'NewBadLakemodel')
## End(Not run)
```

`setPrjDates` *Sets the model run start and end dates in a CRHM model* `.prj` *file*

Description

Sets the model run start and end dates in a CRHM model `.prj` file

Usage

```
setPrjDates(inputPrjFile = "", startDate = "", endDate = "",
  outputPrjFile = "", initialStateFile = "", finalStateFile = "",
  logfile = "")
```

Arguments

`inputPrjFile`	Required. Name of the `.prj` file.
`startDate`	Required. Model run starting date in format '`YYYY MM DD`'.
`endDate`	Required. Model run ending date in format '`YYYY MM DD`'.
`outputPrjFile`	Optional. If omitted, the input `.prj` file will be overwritten.
`initialStateFile`	Optional. If specified, the specified file will be set as the initial state file, i.e. the state variables to be read in at the beginning of the model run.
`finalStateFile`	Optional. If specified, the specified file will be set as the final state file, i.e. the state variables to be exported at the end of the model run.
`logfile`	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If successful, returns `TRUE`. If unsuccessful, returns `FALSE`.

Note

If you are using Windows, the paths for the state files must use '`\\`' instead of '`\`' for the directory delimiter. The reason is that R, like many programming languages, interprets '`\`' as an escape code. This is different from the file path set in '`inputPrjFile`', which must be an R path using the '`/`' symbol. See the example below.

Author(s)

Kevin Shook

See Also

`runCRHM`

Examples

```
## Not run:
result <- setPrjDates('c:/CRHM/Bad Lake 1974-1975.prj',
'1974 10 1', '1975 10 1', initialStateFile='c:\\CRHM\\Model_Initial_State.int',
 finalStateFile='c:\\CRHM\\Model_Final_State.int')
## End(Not run)
```

---

setPrjHRUnames *Sets the names of HRUs in a CRHM model* .prj *file*

Description

Replaces the existing names of all HRUs, in a .prj file.

Usage

```
setPrjHRUnames(inputPrjFile = "", HRUnames = "", outputPrjFile = "",
  logfile = "")
```

Arguments

inputPrjFile	Required. Name of the .prj file.
HRUnames	Required. Vector containing new HRU names. There must be at least as many values in this vector as HRU names in the orginal .prj file.
outputPrjFile	Optional. If omitted, the input .prj file will be overwritten.
logfile	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If successful, returns TRUE. If unsuccessful, returns FALSE.

Note

Currently, this function has only been tested on simple .prj files, i.e. without REW Grp parameters.

Author(s)

Kevin Shook

See Also

runCRHM setPrjParameters setPrjDates

Examples

```
## Not run:
result <- setPrjHRUnames('c:/CRHM/Bad74_Frozen.prj', c('fallow', 'stubble', 'grass'))
## End(Not run)
```

setPrjOutputVariables *Sets the output variables in a CRHM model* .prj *file*

Description

Sets the output variables in a CRHM model .prj file

Usage

```
setPrjOutputVariables(inputPrjFile = "", variables = "",
  outputPrjFile = "", logfile = "")
```

Arguments

inputPrjFile	Required. Name of the .prj file.
variables	Required. Variables to be written to output file. These may be either a character vector combining the variable module, variable name and HRUs or a dataframe with the same information in 3 separate columns. Either format can be returned by setOutputVariables. If the 'null' is specified for variables, then all variables are removed.
outputPrjFile	Optional. If omitted, the input .prj file will be overwritten.
logfile	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If successful, returns TRUE. If unsuccessful, returns FALSE.

Author(s)

Kevin Shook

See Also

readPrjOutputVariables

Examples

```
## Not run:
# read in existing variables as a vector
variables <- readPrjOutputVariables('Bad_Lake_1974-1975.prj', asDataframe=FALSE)

# delete the first set of variables and write to a new file
variables <- variables[-1]
setPrjOutputVariables('Bad_Lake_1974-1975.prj', variables, 'Bad_Lake_1974-1975_revised.prj')

# read in existing variables as a dataframe and write to new file
variables <- readPrjOutputVariables('Bad_Lake_1974-1975.prj', asDataframe=TRUE)
newVariables <- c('evap', 'evapGrangerD', '1 2 3 4 5')
```

```
variables <- rbind(variables, newVariables)
setPrjOutputVariables('Bad_Lake_1974-1975.prj', variables, 'Bad_Lake_1974-1975_revised2.prj')

# deletes all output variables and overwrites the file
result <- setPrjOutputVariables('Bad_Lake_1974-1975.prj', 'null')
## End(Not run)
```

`setPrjParameters` *Sets the parameter values in a CRHM model* `.prj` *file*

Description

Replaces the existing values of a single parameter, for all HRUs, in a `.prj` file.

Usage

```
setPrjParameters(inputPrjFile = "", paramName = "", paramVals = "",
  outputPrjFile = "", quiet = TRUE, logfile = "")
```

Arguments

`inputPrjFile`	Required. Name of the `.prj` file.
`paramName`	Required. Name of parameter to set.
`paramVals`	Required. Vector containing new parameter values. There must be at least as many values in this vector as parameter values in the orginal `.prj` file.
`outputPrjFile`	Optional. If omitted, the input `.prj` file will be overwritten.
`quiet`	Optional. Suppresses display of messages, except for errors. If you are calling this function in an R script, you will usually leave `quiet=TRUE` (i.e. the default). If you are working interactively, you will probably want to set `quiet=FALSE`.
`logfile`	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If successful, returns `TRUE`. If unsuccessful, returns `FALSE`.

Note

Currently, this function has only been tested on simple `.prj` files, i.e. without REW Grp parameters. It *might* work for complex files if the '`REW Grp`' specification is included in the parameter name.

Author(s)

Kevin Shook

See Also

`runCRHM` `setPrjDates`

Examples

```
## Not run:
result <- setPrjParameters('c:/CRHM/Bad74_Frozen.prj',
'fetch', c(1500, 1500, 1500))
## End(Not run)
```

`setPrjRunID` *Sets the Run ID in a CRHM model* `.prj` *file*

Description

Sets the Run ID in a CRHM model `.prj` file

Usage

```
setPrjRunID(inputPrjFile = "", RunID = 0, outputPrjFile = "",
  logfile = "")
```

Arguments

`inputPrjFile`	Required. Name of the `.prj` file.
`RunID`	Required. New RunID for basin. Must be an integer.
`outputPrjFile`	Optional. If omitted, the input `.prj` file will be overwritten.
`logfile`	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If successful, returns `TRUE`. If unsuccessful, returns `FALSE`.

Author(s)

Kevin Shook

See Also

`runCRHM` `setPrjHRUnames` `setPrjDates` `setPrjParameters`

Examples

```
## Not run:
result <- setPrjRunID('c:/CRHM/Bad Lake 1974-1975.prj', '5')
## End(Not run)
```

simpleDailyWater *Aggregates simple CRHM model output flows to daily values*

Description

Calculates daily values of the CRHM outputs of water storages and fluxes, for models *without* sub-basins, for each HRU by its area, and calculates the net fluxes for the basin and each sub-basin.

Usage

```
simpleDailyWater(CRHMoutput, vars = "all", prjFile = "", basinMean = TRUE,
  summarize = TRUE, units = "mm", quiet = TRUE, logfile = "")
```

Arguments

CRHMoutput	Required. The CRHM model output as a standard **CRHMr** dataframe (obs, export or output data)
vars	Optional. Variable column numbers to be used (not including the datetime). The default 'all' selects all columns.
prjFile	Required. The CRHM .prj file.
basinMean	Optional. If TRUE (the default) then the fluxes and storages will be averaged over the entire basin. If FALSE, then fluxes and storages are computed for each HRU's area.
summarize	Optional. If TRUE (the default), then fluxes and storages are aggregated over the basin for all HRUs. If FALSE then fluxes and storages are not aggregated and will be returned for all HRUs.
units	Optional. The units for output. The default is 'mm', but 'm3/s' can also be specified.
quiet	Optional. Suppresses display of messages, except for errors. The default is TRUE. If you are calling this function in an R script, you will usually leave quiet=TRUE (i.e. the default). If you are working interactively, you will probably want to set quiet=FALSE.
logfile	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If successful returns a data frame containing the daily values of all of the water fluxes and storages.

Author(s)

Kevin Shook

See Also

cumulDailyWater

Examples

```
## Not run:
b <- readOutputFile('Bologna1984_30y_output.txt', timezone='MST')
# get daily values without aggregation; values calculated w.r.t the HRU area
dailyHRU <- simpleDailyWater(b, prjFile='Bologna1984_02.prj', basinMean=FALSE)

# get daily values without aggregation; values calculated w.r.t the basin area
dailyHRU_per_basin <- simpleDailyWater(b, prjFile='Bologna1984_02.prj', basinMean=TRUE)

get daily basin values
daily_basin <- simpleDailyWater(b, prjFile='Bologna1984_02.prj', basinMean=TRUE, summarize=TRUE)


## End(Not run)
```

---

simpleMaxSolar *Estimates extra-terrestrial shortwave radiation*

Description

Simple estimation of extra-terrestrial radiation on a horizontal plane. Daily and sub-daily values in MJ/m^2 and W/m^2 are provided,

Usage

```
simpleMaxSolar(datetime, latitude, hoursOffset = 2, quiet = TRUE,
  logfile = "")
```

Arguments

`datetime`	Required. Can either be a vector of **CRHMr** datetimes or any dataframe with a `datetime` in column 1.
`latitude`	Required. The latitude for which values are to calculated.
`hoursOffset`	Optional. The offset (in hours) is added to the solar time to convert it to local time. The default value '2' shifts the daily peak to occur at 2pm.
`quiet`	Optional. Suppresses display of messages, except for errors. If you are calling this function in an R script, you will usually leave `quiet=TRUE` (i.e. the default). If you are working interactively, you will probably want to set `quiet=FALSE`.
`logfile`	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If successful, returns a dataframe containing these variables:

1. `datetime`
2. `So_MJ`, the daily extra-terrestrial radiation in MJ/m^2

3. So_W, the daily extra-terrestrial radiation in W/m^2
4. R_h_d, the conversion from daily to hourly fluxes
5. So_h_W, the sub-daily extra-terrestrial radiation in W/m^2
6. So_h_MJ, the sub-daily extra-terrestrial radiation in MJ/m^2

If unsuccessful, the value FALSE will be returned.

Note

This version is valid for latitudes between 49 °N and 55°N. It is not known how well it will perform outside this range.

Author(s)

Kevin Shook

References

This code is based on Shook, K., and J. Pomeroy (2011), "Synthesis of incoming shortwave radiation for hydrological simulation", Hydrol. Res., 42(6), 433, doi:10.2166/nh.2011.074.
Please cite this paper if you use this function in a publication.

Examples

```
maxSolar <- simpleMaxSolar(BadLake7376, 51.366)
```

---

simpleRibbonPlot *Simple plots of CRHM output max, min and mean values*

Description

Creates **ggplot2** ribbon plots of the daily minimum, maximum and mean values for all selected variables over a year, for models *without* sub-basins. The shaded area indicates the range of values in the data frame for each day.

Usage

```
simpleRibbonPlot(CRHMoutput, columns = "all", waterYear = FALSE,
  ylabel = "Basin total (mm)", skipLeapDay = TRUE, facet = FALSE,
  facetColour = "blue", facetCols = 3)
```

Arguments

CRHMoutput	Required. A data frame containing either instantaneous or cumulative values. The data may be either daily or hourly.
columns	The columns to be aggregated, not including the datetime. The default is the 'all', but can be a vector, i.e. c(1,2,3). Note that all variables must have the same units.
waterYear	Optional. If True, the statistics are calculated and plotted by water year, if FALSE (the default) the statistics are calculated and plotted by calendar year. If you are plotting cumulative values using cumulDailyWater, this parameter must be set to be the same as that function.
ylabel	Optional. Label to be used for the y-axis.
skipLeapDay	Optional. If TRUE (the default) then values for Feb. 29 will be omitted from the analyses and plots. If FALSE, then these values will be included, which may result in strange-looking plots.
facet	Optional. If FALSE (the default), then all variables will be on a single plot. If TRUE, then each variable will be on a separate facet.
facetColour	Optional. The colour to be used for shading the ribbon plot in each facet, if selected.
facetCols	Optional. The number of columns to be used for the facets, if selected.

Value

If unsuccessful, returns FALSE. If successful, returns a **ggplot2** object.

Author(s)

Kevin Shook

See Also

cumulDailyWater

Examples

```
## Not run:
p <- ribbonPlot(waterYearCumul, waterYear=TRUE, units='mm', facet=TRUE, facetCols=2
## End(Not run)
```

summariseObsFiles *Summarises all obs files in a directory*

Description

This function summarises all of the obs files in specified directory, optionally plotting the values in each obs file in a separate graph.

Usage

```
summariseObsFiles(file.dir = "", timezone = "",
  summaryFile = "obsSummary.csv", plot = FALSE)
```

Arguments

file.dir	Optional Directory containing .obs files. If not specified, defaults to current directory. Note that this is an R path, which uses the '/' symbol on ALL operating systems.
timezone	Required. The name of the timezone of the data as a character string. This should be the timezone of your data, but omitting daylight savings time. Note that the timezone code is specific to your OS. To avoid problems, you should use a timezone without daylight savings time. Under Linux, you can use 'CST' and 'MST' for Central Standard or Mountain Standard time, respectively. Under Windows or OSX, you can use 'etc/GMT+6' or 'etc/GMT+7' for Central Standard and Mountain Standard time. DO NOT use 'America/Regina' as the time zone, as it includes historical changes between standard and daylight savings time.
summaryFile	Optional. File to contain summary. Defaults to 'obsSummary.csv' in the same directory as the obs files.
plot	Optional. Logical. Do you want to plot graphs of each obs file? Defaults to FALSE.

Value

If successful returns no value. If unsuccessful, returns FALSE.

Author(s)

Kevin Shook

See Also

plotObs

Examples

```
## Not run:
summariseObsFiles('c:/data/BadLake/MSC', timezone='etc/GMT+6', plot=TRUE)
## End(Not run)
```

`tMinMaxToHourly` *Converts daily to hourly temperatures*

Description

Converts daily values of tmin, tmax and (if they exist) tmean to hourly values, by spline interpolation. Only single columns of the daily temperatures can be used.

Usage

```
tMinMaxToHourly(obs, tmin.col = 0, tmax.col = 0, tmean.col = 0,
  tmin.time = "07:00", tmax.time = "15:00", tmean.time = "12:00",
  timezone = "", quiet = TRUE, logfile = "")
```

Arguments

`obs`	Required. A **CRHMr** obs dataframe.
`tmin.col`	Optional. The number of the tmin column (omitting the datetime) in the obs dataframe. If omitted, the column named 'tmin' will be used. There must be a tmin column in the dataset
`tmax.col`	Optional. The number of the tmax column (omitting the datetime) in the obs dataframe. If omitted, the column named 'tmax' will be used. There must be a tmax column in the dataset.
`tmean.col`	Optional. The number of the tmax column (omitting the datetime) in the obs dataframe. If omitted, the column named 'tmean' will be searched for. This column need not be present in the dataset. If the value is set to `-1`, then the mean temperature will not be used in the interpolation.
`tmin.time`	Optional. The time of day at which the minimum air temperature occurs, specified in '`HH:MM`' format. If not specified, defaults to '`07:00`'.
`tmax.time`	Optional. The time of day at which the maximum air temperature occurs, specified in '`HH:MM`' format. If not specified, defaults to '`15:00`'.
`tmean.time`	Optional. The time of day at which the maximum air temperature occurs, specified in '`HH:MM`' format. If not specified, defaults to '`12:00`'.
`timezone`	Required. The name of the timezone of the data as a character string. This should be the timezone of your data, but omitting daylight savings time. Note that the timezone code is specific to your OS. To avoid problems, you should use a timezone without daylight savings time. Under Linux, you can use '`CST`' and '`MST`' for Central Standard or Mountain Standard time, respectively. Under Windows or OSX, you can use '`etc/GMT+6`' or '`etc/GMT+7`' for Central Standard and Mountain Standard time. DO NOT use '`America/Regina`' as the time

	zone, as it includes historical changes between standard and daylight savings time.
quiet	Optional. Suppresses display of messages, except for errors. If you are calling this function in an R script, you will usually leave quiet=TRUE (i.e. the default). If you are working interactively, you will probably want to set quiet=FALSE.
logfile	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If successful, returns an obs dataframe with the hourly air temperatures. If not successful, returns FALSE.

Note

Interpolation over intervals longer than 1 day is not allowed.

Author(s)

Kevin Shook

See Also

interpolate

Examples

```
## Not run:
MSC.hourly <- tMinMaxToHourly(MSC.daily, timezone='etc/GMT+7')
## End(Not run)
```

---

trimObs *Trims a dataframe of missing values at the beginning and end*

Description

Removes values before 01:00 on the first complete day of the data, and after 00:00 on the last complete rows. Because it is destructive, this command should *only* be used immediately before exporting observations to a CRHM .obs file. For daily obs, removes values before the first complete row and after the last complete row.

Usage

```
trimObs(obs, quiet = TRUE, logfile = "")
```

Arguments

obs	Required. A **CRHMr** obs dataframe.
quiet	Optional. Suppresses display of messages, except for errors. If you are calling this function in an R script, you will usually leave quiet=TRUE (i.e. the default). If you are working interactively, you will probably want to set quiet=FALSE.
logfile	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If successful, returns the trimmed dataframe. If unsuccessful, returns FALSE.

Author(s)

Kevin Shook

See Also

writeObsFile

Examples

```
BadLake.trimmed <- trimObs(BadLake7376)
```

---

user *Logs information about the user*

Description

Writes information about the user and the user's computer to the logfile.

Usage

```
user(logfile = "")
```

Arguments

logfile	Optional. Name of the file to be used for logging the action. Normally not used.

Value

Returns nothing.

Note

This function records information about your computer (the user, the operating system, the version of R, and all of the packages in use). It will make it easier to debug your problems and may help you to figure out why R code that used to work is no longer giving the correct answer.

Author(s)

Kevin Shook

Examples

```
user()
```

`vectorsToVelocity` *Convert wind vectors to velocity*

Description

Converts wind u and v vectors to a single wind velocity. The vectors may be in any units, i.e. km/h or m/s.

Usage

```
vectorsToVelocity(uObs, uColnum = 1, vObs, vColnum = 1, quiet = TRUE,
  logfile = "")
```

Arguments

`uObs`	Required. A **CRHMr** obs data frame containing the wind u vectors.
`uColnum`	Optional. The column number containing the u values, not including the date-time. Default is column 1.
`vObs`	Required. A **CRHMr** obs data frame containing the wind v vectors. Note that u and v must have the same units!
`vColnum`	Optional. The column number containing the u values, not including the date-time. Default is column 1.
`quiet`	Optional. Suppresses display of messages, except for errors. If you are calling this function in an R script, you will usually leave `quiet=TRUE` (i.e. the default). If you are working interactively, you will probably want to set `quiet=FALSE`.
`logfile`	Optional. Name of the file to be used for logging the action. Normally not used

Value

If successful, returns an obs data frame containing the wind speed (in the original vector units) and the wind direction in degrees from North. If unsuccessful, returns the value `FALSE`.

Author(s)

Kevin Shook

Examples

```
## Not run:
windspeed <- vectorsToVelocity(windObs, uColnum=1, uObs, vColnum=2)
## End(Not run)
```

vp2rh *Converts air temperature and vapour pressure to relative humidity*

Description

This is an internal **CRHMr** function and should *never* need to be called directly, as it does *NO* data checking. Use the function changeEatoRH instead.

Usage

```
vp2rh(airtemp, vapourPressure)
```

Arguments

airtemp	Required. Air temperature in °C.
vapourPressure	Required. Vapour pressure in kPa.

Value

Returns RH in percent.

Author(s)

Kevin Shook

See Also

changeEatoRH

Examples

```
vp2rh(-5, 0.2007355)
```

weighingGauge-methods *Processes weighing gauge precipitation*

Description

These functions clean accumulated precipitation data from a weighing gauge. The functions are wrapper for other R and **CRHMr** functions as well as functions adapted from MATLAB code by Alan Barr. The typical sequence of operation of the functions is indicated by their numbers, however it it NOT always necessary to execute any given function.

weighingGaugePlot Plots cumulative and interval precipitation

weighingGauge1 Gap fills the data

weighingGauge2 Removes spikes

weighingGauge3 Applies the Savitzky-Golay polynomial filter

weighingGauge4 Calls `PcpFiltPosTh`

weighingGauge5 Simplified reset and jitter removal

weighingGaugeInterval Deacumulates the precipitation

Author(s)

Kevin Shook

See Also

`weighingGaugePlot` `weighingGauge1` `weighingGauge2` `weighingGauge3` `weighingGauge4` `weighingGauge5` `weighingGaugeInterval`

Examples

```
## Not run:
# check data - shows jitter, missing values and a large reset
weighingGaugePlot(wg)
# see jitter and missing values only
weighingGaugePlot(wg, endDate = '2010-10-01')

# fill gaps, and check to see if all missing values have been interpolated
wg1 <- weighingGauge1(wg, maxGapLength=10)
weighingGaugePlot(wg1)

# remove the spikes, and check
wg2 <- weighingGauge2(wg1, spikeThreshold=300, maxSpikeGap=3)
weighingGaugePlot(wg2)

# apply the smoothing filter - only for use with high-frequency data
wg3 <- weighingGauge3(wg2, filterLength=5)
weighingGaugePlot(wg3)

# apply Alan Barr's filter
wg4 <- weighingGauge4(wg3, smallDropThreshold = 0.05, serviceThreshold = -50)
weighingGaugePlot(wg4)

# deaccumulate precipitation
wg5 <- weighingGaugeInterval(wg4)

# use simplified method and plot results
# note that the infilled values are used
wg6 <- weighingGauge5(wg1)
weighingGaugePlot(wg6)
wg7 <- weighingGaugeInterval(wg6)
## End(Not run)
```

weighingGauge1 *Fills gaps in weighing gauge precipitation data*

Description

This method fills missing values in accumulated precipitation data from a weighing gauge using linear interpoaltion. This function is a wrapper for functions and was suggested by Craig Smith. This function need not be called if your data does not contain any gaps. It is a good idea to use `weighingGaugePlot` to look at your data before calling this function.

Usage

```
weighingGauge1(obs, precipCol = 1, maxGapLength = 5, quiet = TRUE,
  logfile = "")
```

Arguments

`obs`	Required. A standard **CRHMr** obs file.
`precipCol`	Optional. The number of the column containing the weighing gauge cumulative precipitation data, not including the `datetime`. Default is column 1. These values are cumulative preciptiation.
`maxGapLength`	Optional. The maximum gap length included in the analysis (in time intervals). Default is 5 time intervals. Used by `interpolate`.
`quiet`	Optional. Suppresses display of messages, except for errors. If you are calling this function in an R script, you will usually leave `quiet=TRUE` (i.e. the default). If you are working interactively, you will probably want to set `quiet=FALSE`.
`logfile`	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If unsuccessful, returns `FALSE`. If sucessful, returns a modified version of the `obs` dataframe containing only the datetime and the gap-filled total precipitation values.

Note

This version does not explicitly incorporate the effects of evaporation.

Author(s)

Kevin Shook

See Also

`weighingGauge2` `weighingGauge3` `weighingGauge4` `weighingGaugePlot` `weighingGaugeInterval`

Examples

```
## Not run:
test1 <- weighingGauge1(wg,  maxGapLength=6)
## End(Not run)
```

`weighingGauge2` *Removes spikes from weighing gauge cumulative precipitation*

Description

Removes positive and negative spikes from deaccumulated weighing gauge precipitation. The gaps left from spike removal are infilled by linear interpolation. This function is a wrapper for other functions and was suggested by Craig Smith. This function need not be called if your data does not contain any spikes. It is a good idea to use `weighingGaugePlot` to look at your data before calling this function. Note that this function will also remove gauge resets, if they are wihin the range of the `spikeThreshold`.

Usage

```
weighingGauge2(obs, precipCol = 1, spikeThreshold = 1000, maxSpikeGap = 3,
  quiet = TRUE, logfile = "")
```

Arguments

`obs`	Required. A standard **CRHMr** obs file.
`precipCol`	Optional. The number of the column containing the weighting gauge cumulative precipitation data, not including the `datetime`. Default is column 1. This function can only work on 1 column of precipitation.
`spikeThreshold`	Optional. Threshold for single-interval precipitation (mm). Any spikes (changes in cumulative precipitation) whose absoloute value is greater than the threshold will be deleted. The default value is 1000 mm.
`maxSpikeGap`	Optional. Maximum length of spikes (in time steps) to be filled. Default is 3.
`quiet`	Optional. Suppresses display of messages, except for errors. If you are calling this function in an R script, you will usually leave `quiet=TRUE` (i.e. the default). If you are working interactively, you will probably want to set `quiet=FALSE`.
`logfile`	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If unsuccessful, returns `FALSE`. If sucessful, returns a modified version of the `obs` dataframe containing only the datetime and the adjusted precipitation values.

Note

If you don't specify a large enough value for `maxSpikeGap`, then all of the values after the spike will be set to `NA`.

Author(s)

Kevin Shook

See Also

weighingGauge1 weighingGauge3 weighingGauge4 weighingGaugePlot weighingGaugeInterval

Examples

```
## Not run:
test2 <- weightingGauge2(wg, spikeThreshold = 300)
## End(Not run)
```

---

weighingGauge3 *Filters weighing gauge cumulative precipitation*

Description

Thes functions is a wrapper for the Savitzky-Golay polynomial filter function `sgolayfilt` from the package **signal**. It passes the specified cumulative precipitation data, the filter length, the filter order (set to 1) and the derivative order (set to 0) to the filter. It is a good idea to use `weighingGaugePlot` to look at your data before and after calling this function. Note that this function is only intended to be used with very high frequency (i.e. 1 minute) gauged data, and should not be used with longer-period accumulations.

Usage

```
weighingGauge3(obs, precipCol = 1, filterLength = 0, quiet = TRUE,
  logfile = "")
```

Arguments

`obs`	Required. A standard **CRHMr** obs file.
`precipCol`	Optional. The number of the column containing the weighing guage cumulative precipitation data, not including the `datetime`. Default is column 1. This function can only work on 1 column of precipitation.
`filterLength`	Required. The number of time intervals used by the Savitzky-Golay polynomial filter. Note that this must be an ODD number. The appropriate value will depend on the type of your data, and the noise present.
`quiet`	Optional. Suppresses display of messages, except for errors. If you are calling this function in an R script, you will usually leave `quiet=TRUE` (i.e. the default). If you are working interactively, you will probably want to set `quiet=FALSE`
`logfile`	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If unsuccessful, returns FALSE. If sucessful, returns a modified version of the obs dataframe containing the adjusted precipitation values. Note that the name of the precipitation variable is modified: '_sg_filtered' is appended.

Note

This function is potentially destructive, as it smooths the accumulated precipitation. You may need to set the filter length by trial-and error. You can check the effects using the function weighingGaugePlot.

Author(s)

Kevin Shook

See Also

weighingGauge1 weighingGauge2 weighingGauge4 weighingGaugePlot weighingGaugeInterval

Examples

```
## Not run:
test3 <- weighingGauge3(wg, filterLength=5)
## End(Not run)
```

---

weighingGauge4 *Runs Alan Barr's filter on weighing gauge cumulative precipitation*

Description

This function is a wrapper for Alan Barr's function PcpFiltPosTh which removes jitter from precipitation data. The called function is iterative and may be slow to execute.

Usage

```
weighingGauge4(obs, precipCol = 1, smallDropThreshold = 0.1,
  serviceThreshold = -100, serviceGapLength = 3, quiet = TRUE,
  logfile = "")
```

Arguments

obs	Required. A standard **CRHMr** obs file.
precipCol	Optional. The number of the column containing the weighing gauge cumulative percipitaion data, not including the datetime. Default is column 1. This function can only work on 1 column of precipitation.
smallDropThreshold	Optional. Minimum interval precipitation (mm). Default is 0.1.

serviceThreshold
: Optional. Threshold for removing change in storage due to servicing (mm). Note that this must be a NEGATIVE value. Default is -100.

serviceGapLength
: Optional. Maximum gap length (in time intervals) to be filled by linear interpolation after removing servicing (i.e. gauge reset) data. The default value is 3, but you may have to adjust it for very high-frequency data.

quiet
: Optional. Suppresses display of messages, except for errors. If you are calling this function in an R script, you will usually leave quiet=TRUE (i.e. the default). If you are working interactively, you will probably want to set quiet=FALSE.

logfile
: Optional. Name of the file to be used for logging the action. Normally not used.

Value

If unsuccessful, returns FALSE. If sucessful, returns a modified version of the obs dataframe containing the adjusted precipitation values. Note that the name of the precipitation variable is modified: '_PcpFiltPosT' is appended.

Note

This function is potentially destructive, as it smooths the accumulated precipitation. You may need to set the filter length by trial-and error. You can check the effects using the function weighingGaugePlot. This version does not explicitly incorporate the effects of evaporation.

Author(s)

Kevin Shook

See Also

PcpFiltPosTh weighingGauge1 weighingGauge2 weighingGauge3 weighingGaugePlot weighingGaugeInterval

Examples

```
## Not run:
test4 <- weighingGauge4(wg, smallDropThreshold = 0.05, serviceThreshold = -50)
## End(Not run)
```

---

weighingGauge5 *Simplified weighing gauge processing*

Description

This function removes gauge resets (large negative changes usually due to servicing) and uses a simple method (cumulative maxima) to remove jitter. It is a good idea to use weighingGauge1 to infill missing values before calling this function and weighingGaugePlot to look at your data before and after calling this function.

Usage

```
weighingGauge5(obs, precipCol = 1, resetThreshold = 50, quiet = TRUE,
  logfile = "")
```

Arguments

obs	Required. A standard **CRHMr** obs file.
precipCol	Optional. The number of the column containing the weighing gauge cumulative precipitation data, not including the datetime. Default is column 1. These values are cumulative preciptiation. This function can only work on 1 column of precipitation. The precipitaition cannot contain missing values - infill them using weighingGauge1 first.
resetThreshold	Optional. The minimum (absolute value) of negative values considered to be gauge resets. Default is 50 mm.
quiet	Optional. Suppresses display of messages, except for errors. If you are calling this function in an R script, you will usually leave quiet=TRUE (i.e. the default). If you are working interactively, you will probably want to set quiet=FALSE.
logfile	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If unsuccessful, returns FALSE. If sucessful, returns a modified version of the obs dataframe containing only the datetime and the modified cumulative precipitation values.

Note

This version does not explicitly incorporate the effects of evaporation.

Author(s)

Kevin Shook, based on an algorithm developed by Thai Nguyen of Alberta Environment.

See Also

weighingGauge1 weighingGaugePlot weighingGaugeInterval

Examples

```
## Not run:
wg1 <- weighingGauge1(wg,  maxGapLength=500)
wg5 <- weighingGauge5(wg1,  resetThreshold=70)
## End(Not run)
```

`weighingGaugeInterval` *Deaccumulates weighing gauge precipitation*

Description

Converts weighing gauged cumulative precipitation to interval values. The interval precipitation is checked to make sure that it does not contain any negative values. Negative precipitation will result in the function displaying an error message and returning the value `FALSE`.

Usage

```
weighingGaugeInterval(obs, precipCol = 1, quiet = TRUE, logfile = "")
```

Arguments

`obs`	Required. A standard **CRHMr** obs file.
`precipCol`	Optional. The number of the column containing the weighting gauge cumulative precipitation data, not including the `datetime`. Default is column 1.
`quiet`	Optional. Suppresses display of messages, except for errors. If you are calling this function in an R script, you will usually leave `quiet=TRUE` (i.e. the default). If you are working interactively, you will probably want to set `quiet=FALSE`.
`logfile`	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If unsuccessful, returns `FALSE`. If sucessful, returns a modified version of the `obs` dataframe containing the datetime, the cumulative and interval precipitation.

Author(s)

Kevin Shook

See Also

`weighingGauge1` `weighingGauge2` `weighingGauge3` `weighingGauge4` `weighingGauge5` `weighingGaugePlot`

Examples

```
## Not run:
test5 <- weightingGaugeStep5(wg)
## End(Not run)
```

weighingGaugePlot *Plots weighing gauge precipitation*

Description

Plots culumative precipitation data from a weighting gauge. The plot contains facets with the cumulative and interval precipitations plotted against time.

Usage

```
weighingGaugePlot(obs, precipCol = 1, startDate = "", endDate = "",
  showMissing = TRUE)
```

Arguments

obs	Required. A standard **CRHMr** obs dataframe.
precipCol	Optional. The column containing the cumulative precipitation data. Default is column 1.
startDate	Optional. The starting date for the plot. Can either be a year as a number (e.g. 1995) or a date in 'Y-m-d' format, i.e. '1995-06-01'. If not specified, the first datetime value in the obs data is used.
endDate	Optional. The ending date for the plot. Can either be a year as a number (e.g. 1995) or a date in 'Y-m-d' format, i.e. '1995-12-31'. If not specified, the first datetime value in the obs data is used.
showMissing	Optional. If set to TRUE (the default), then the position of missing values will be indicated by red points on the x-axis.

Value

If successful returns a ggplot2 object. If unsuccessful, returns the value FALSE.

Author(s)

Kevin Shook.

See Also

weighingGauge1 weighingGauge2 weighingGauge3 weighingGauge4 weighingGauge5

Examples

```
## Not run:
p <- weighingGaugePlot(test1)
## End(Not run)
```

wg *Synthetic weighing precipitation gauge data*

Description

A dataframe containing data for use with the weighingGauge functions.

Usage

```
wg
```

Format

A dataframe with 17460 rows and 2 variables (including the datetime):

datetime date and time as a POSIXct object

p cumulative precipitation (mm)

Source

This data was derived from real data provided by Craig Smith. It has been adjusted by incorporating missing values and a simulated gauge reset.

win.eol *Gets the Windows end of line characters*

Description

Finds the end of line (eol) characters required for writing Windows files, such as CRHM obs files. No parameters are required. This is an internal **CRHMr** function and should *never* need to be called directly.

Usage

```
win.eol()
```

Value

Returns the Windows end of line characters (cr and lf).

Note

This function is used to make the creation of Windows-specific files work on all platforms. CRHM requires its obs and project files to use the Windows end of line characters, which are expressed differently on UNIX-based operating systems such as Linux and OSX.

Author(s)

Kevin Shook

See Also

automatePrj setPrjDates writeObsFile runCRHM

Examples

```
windowsEndOfLine <- win.eol()
```

---

writeChangeLogFile *Writes information about changes to a* **CRHMr** *dataframe to a file*

Description

This is an internal **CRHMr** function and should normally not need to be called directly. This function writes a file containing the a summary of the files as well as the changed data values and a description of their source.

Usage

```
writeChangeLogFile(action, original.data.info, changed.data.info,
  changed.data.type, comment1 = "", comment2 = "", comment3 = "",
  comment4 = "", quiet)
```

Arguments

action	Required. A character string indicating the action taken - ususally this is the name of the function making the changes.
original.data.info	Required. Information about the original dataframe from CRHM_summary.
changed.data.info	Required. Information about the changed dataframe from CRHM_summary.
changed.data.type	Required. A vector indicating the type/source of data in the changed dataframe.
comment1	Optional. A comment about the data and/or changes.
comment2	Optional. A comment about the data and/or changes.
comment3	Optional. A comment about the data and/or changes.
comment4	Optional. A comment about the data and/or changes.
quiet	Optional. Suppresses display of messages, except for errors. If you are calling this function in an R script, you will usually leave quiet=TRUE (i.e. the default). If you are working interactively, you will probably want to set quiet=FALSE.

Value

If successful, returns TRUE. If unsucessful, returns FALSE.

Author(s)

Kevin Shook

Examples

```
## Not run:
writeChangeLogFile('appendObs', obs1.info, obs2.info, output.type,
'primary data: obs1', 'secondary data: obs2')
## End(Not run)
```

---

writeCLASSfile *Writes forcing data to a CLASS file*

Description

Writes a standard **CRHMr** data frame to a CLASS forcing data file.

Usage

```
writeCLASSfile(CLASS, CLASSfile = "", logfile = "")
```

Arguments

CLASS	Required. A **CRHMr** data frame containing the fields SWin, LWin, Prec, T, qa, U, and P, in addition to the datetime.
CLASSfile	Required. A the file to hold the CLASS data.
logfile	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If successful, returns the value TRUE. If unsuccessful, returns the value FALSE.

Examples

```
## Not run:
result <- writeCLASSfile(class, 'driv.met')
## End(Not run)
```

writeObsFile *Writes a dataframe to a .obs file*

Description

Writes a CRHM dataframe to an obs file. It will not allow ea values - they must be converted to RH values first. The CRHM filter command to convert RH values to ea is added automatically to the header of the .obs file.

Usage

```
writeObsFile(obs, obsfile = "", comment = "", quiet = TRUE,
  logfile = "")
```

Arguments

obs	Required. A **CRHMr** obs data frame.
obsfile	Required. Name for file to hold observations.
comment	Optional. A comment to be added to the header of the .obs file. If omitted, the header will consist of the name of the obs data frame followed by '`created by CRHMr function writeObsFile`'.
quiet	Optional. Suppresses display of messages, except for errors. If you are calling this function in an R script, you will usually leave `quiet=TRUE` (i.e. the default). If you are working interactively, you will probably want to set `quiet=FALSE`.
logfile	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If successful, returns `TRUE`. If unsuccessful, returns `FALSE`.

Note

If there are missing values in the obs dataframe, the last line of the header will contain the line '`$$ Missing`'.

Author(s)

Kevin Shook

See Also

`trimObs` `changeEatoRH`

Examples

```
# write sample obs dataframe to .obs file
result <- writeObsFile(BadLake7376, 'BadLake7376.obs')
```

yearlyPeaks *Finds datetimes of annual max values*

Description

Finds the annual maxima (and, optionally, minima) of the specfied columns in a CRHM dataframe and their corresponding datetimes.

Usage

```
yearlyPeaks(CRHMdata, columns = 0, minvals = FALSE, logfile = "")
```

Arguments

CRHMdata	Required. A **CRHMr** dataframe.
columns	Optional. A vector containing the columns to be analyzed, not including the datetime column. If not specified, all data columns are used.
minvals	Optional. Logical. Specifies if yearly minimum values should also be found. Default value is FALSE.
logfile	Optional. Name of the file to be used for logging the action. Normally not used.

Value

Returns a dataframe with maximum (and minimum if selected) values, and their corresponding datetimes, for each year for each selected variable.

Author(s)

Kevin Shook

See Also

aggDataframe

Examples

```
## Not run:
p <- yearlyPeaks(BrandonObs, columns=c(1,2), minvals=TRUE)
## End(Not run)
```

`zeroMissingPrecip` *Sets missing precipitation values to zero*

Description

Removes missing precipitation (`NA_real_` values (p and/or ppt) in an obs dataframe, by setting them to zero. This function is potentially dangerous, as setting the missing values to zero prevents any further imputation/interpolation of values.

Usage

```
zeroMissingPrecip(obs, precipCols = 0, quiet = TRUE, logfile = "")
```

Arguments

`obs`	Required. A **CRHMr** obs dataframe.
`precipCols`	Optional. A vector containing the numbers of the columns (excluding the date-time) to be processed. If omitted, all of the columns whose names contain 'p' or 'ppt' will be processed
`quiet`	Optional. Suppresses display of messages, except for errors. If you are calling this function in an R script, you will usually leave `quiet=TRUE` (i.e. the default). If you are working interactively, you will probably want to set `quiet=FALSE`.
`logfile`	Optional. Name of the file to be used for logging the action. Normally not used.

Value

If the action is successful, returns a dataframe with all missing precipitation values set to zero. If the action is unsuccessful, then the value `FALSE` is returned. If `quiet=FALSE`, then a summary of the returned obs data will be returned.

Note

As described, this function is potentially dangerous, and its action *CANNOT* be undone. It should only be used when there is no further hope for imputing/interpolating precipitation data. Note that its use is really only justified in arid or semi-arid climates, where precipitation is rare. Because it is so destructive, the function should only be used immediately before exporting to an obs file.

Author(s)

Kevin Shook

See Also

`writeObsFile`

Examples

```
BadLake7376.clean <- zeroMissingPrecip(BadLake7376, quiet=FALSE)
```

Index