Package 'RavenR'

May 7, 2024

Type Package

Title Raven Hydrological Modelling Framework R Support and Analysis

Version 2.2.2

Date 2024-05-06

Maintainer Robert Chlumsky <rchlumsk@uwaterloo.ca>

Description Utilities for processing input and output files associated with the Raven Hydrological Modelling Framework. Includes various plotting functions, model diagnostics, reading output files into extensible time series format, and support for writing Raven input files.
The 'RavenR' package is also archived at Chlumsky et al. (2020) <doi:10.5281/zenodo.4248183>.
The Raven Hydrologic Modelling Framework method can be referenced with Craig et al. (2020) <doi:10.1016/j.envsoft.2020.104728>.

Note Package is in active development; please report errors and suggestions to rchlumsk@uwaterloo.ca.

BugReports https://github.com/rchlumsk/RavenR/issues

Depends R (>= 3.6.0)

Imports colorspace, cowplot, crayon, DiagrammeR, dplyr, dygraphs, gdata, ggplot2, igraph, lubridate, magrittr, purrr, Rcpp, RCurl, scales, stats, stringr, tidyr, utils, visNetwork, xts, zoo

Suggests devtools, knitr, rmarkdown

URL https://github.com/rchlumsk/RavenR

License GPL-3

Encoding UTF-8

LazyData TRUE

RoxygenNote 7.3.1

VignetteBuilder knitr

LinkingTo Rcpp

NeedsCompilation yes

Author Robert Chlumsky [cre, aut] (<https://orcid.org/0000-0002-1303-5064>),
James Craig [ctb, aut] (<https://orcid.org/0000-0003-2715-7166>),
Leland Scantlebury [ctb, aut],
Simon Lin [ctb, aut],
Sarah Grass [ctb, aut],
Genevieve Brown [ctb, aut],
Rezgar Arabzadeh [ctb, aut]

Repository CRAN

Date/Publication 2024-05-07 03:30:02 UTC

R topics documented:

`cmax` *cmax*

Description

Applies the base::max function across columns.

Usage

```
cmax(x, na.rm = FALSE)
```

Arguments

`x`	object to apply the max function to
`na.rm`	whether to remove na values from the calculation

Details

It applies the base::max function over columns, which is advantageous for calculating the max within a column rather than the max of the whole data frame. The default base::max will not work properly for data frames and other structures in applying over columns or different periods.

This function was included for usage with the apply.<period> and `rvn_apply_wyearly` function, as the base::max function does not work properly across columns.

Value

x with the max value in each column determined

See Also

`rvn_apply_wyearly` where this function can be applied for the water year, and the xts functions such as `apply.yearly` and `apply.monthly`

Examples

```
data(rvn_hydrograph_data)
cmax(rvn_hydrograph_data$hyd$Sub43_obs, na.rm=TRUE)

rvn_apply_wyearly(rvn_hydrograph_data$hyd, cmax, na.rm=TRUE)
```

---

hhmmss2dec *Convert hours, minutes, seconds to decimal hours*

Description

Converts string format HH:MM:SS to decimal hours

Usage

```
hhmmss2dec(x)
```

Arguments

x	input as character, format HH:MM:SS

Value

time in decimal hours

Examples

```
# return hour:minutes:seconds to decimal hours
hhmmss2dec("02:35:58")
```

---

rvn_annual_peak *Annual Peak Comparison*

Description

rvn_annual_peak creates a plot of the annual observed and simulated peaks, based on the water year.

Usage

```
rvn_annual_peak(
  sim,
  obs,
  mm = 9,
  dd = 30,
  add_line = TRUE,
  add_r2 = FALSE,
  add_eqn = FALSE
)
```

Arguments

sim	time series object of simulated flows
obs	time series object of observed flows
mm	month of water year ending (default 9)
dd	day of water year ending (default 30)
add_line	optionally adds a 1:1 line to the plot for reference (default TRUE)
add_r2	optionally computes the R2 and adds to plot (default FALSE)
add_eqn	optionally adds the equation for a linear regression line through the origin (default FALSE)

Details

Creates a scatterplot of the annual observed and simulated peaks, calculated for each available water year of data within the two series provided. The default start of the water year is October 1st, but may be adjusted through function parameters. Note that the calculation uses the peak magnitude of simulated and observed series in each water year, without considering the timing of the events in each series.

The sim and obs should be of time series (xts) format and are assumed to be of the same length and time period. The flow series are assumed to be daily flows with units of m3/s.

The R2 diagnostic is calculated for a fit with no intercept, consistent with the provided 1:1 line (in a perfect fit the points are identical, and intercept is automatically zero).

Note that a plot title is purposely omitted in order to allow the automatic generation of plot titles.

Value

returns a list with peak data in a data frame, and a ggplot object

df_peak	data frame of the calculated peaks
p1	ggplot object with plotted annual peaks

See Also

rvn_annual_volume to create a scatterplot of annual flow volumes rvn_annual_peak_event to consider the timing of peak events.

Examples

```
# load sample hydrograph data, two years worth of sim/obs
data(rvn_hydrograph_data)
sim <- rvn_hydrograph_data$hyd$Sub36
obs <- rvn_hydrograph_data$hyd$Sub36_obs

# create a plot of annual peaks with default options
peak1 <- rvn_annual_peak(sim, obs)
peak1$df_peak
peak1$p1

# plot with r2 and regression equation
peak_df <- rvn_annual_peak(sim, obs, add_r2=TRUE, add_eqn=TRUE)
peak_df$p1
```

---

rvn_annual_peak_error *Annual Peak Errors*

Description

rvn_annual_peak_error creates a plot of the annual observed and simulated peak percent errors, based on the water year.

Usage

```
rvn_annual_peak_error(
  sim,
  obs,
  mm = 9,
  dd = 30,
  add_line = TRUE,
  add_labels = TRUE
)
```

Arguments

sim	time series object of simulated flows
obs	time series object of observed flows
mm	month of water year (default 9)
dd	day of water year (default 30)
add_line	optionally adds a 1:1 line to the plot for reference (default TRUE)
add_labels	optionally adds labels for overpredict/underpredict on right side axis (default TRUE)

Details

Creates a plot of the percent errors in simulated peaks for each water year. The peaks are calculated as the magnitude of the largest event in each water year. Note that the rvn_annual_peak_error function is first used to obtain the peaks in each year, then the percent errors are calculated.

The percent errors are calculated as (QPsim-QPobs)/QPobs*100, where QP is the peak flow event.

The sim and obs should be of time series (xts) format and are assumed to be of the same length and time period. The flow series are assumed to be daily flows with units of m3/s.

The add_labels will add the labels of 'overprediction' and 'underprediction' to the right hand side axis if set to TRUE. This is useful in interpreting the plots.

Note that a plot title is purposely omitted in order to allow the automatic generation of plot titles.

Value

returns a list with peak errors in a data frame, and a ggplot object

`df_peak_error`	data frame of the calculated peak errors
`p1`	ggplot object with plotted annual peak errors

See Also

`rvn_annual_peak_event` to consider the timing of peak events `rvn_annual_peak_event_error` to calculate errors in peak events.

Examples

```
system.file("extdata","run1_Hydrographs.csv", package="RavenR") %>%
rvn_hyd_read(.) %>%
rvn_hyd_extract(subs="Sub36",.) ->
hyd_data

sim <- hyd_data$sim
obs <- hyd_data$obs

# create a plot of annual peak errors with default options
peak1 <- rvn_annual_peak_error(sim, obs)
peak1$df_peak_error
peak1$p1

# plot directly and without labels
rvn_annual_peak_error(sim, obs, add_line=TRUE, add_labels=FALSE)
```

`rvn_annual_peak_event` *Annual Peak Event Comparison*

Description

rvn_annual_peak_event creates a plot of the annual observed and simulated peaks, based on the water year.

Usage

```
rvn_annual_peak_event(
  sim,
  obs,
  mm = 9,
  dd = 30,
  add_line = TRUE,
  add_r2 = FALSE,
  add_eqn = FALSE
)
```

Arguments

`sim`	time series object of simulated flows
`obs`	time series object of observed flows
`mm`	month of water year (default 9)
`dd`	day of water year (default 30)
`add_line`	optionally adds a 1:1 line to the plot for reference (default `TRUE`)
`add_r2`	optionally computes the R2 and adds to plot (default `FALSE`)
`add_eqn`	optionally adds the equation for a linear regression line through the origin (default `FALSE`)

Details

Creates a scatterplot of the annual observed and simulated peaks, calculated for each available water year of data within the two series provided; note that the difference between this and the annual.peak function is that here the peak event simulated for the same day as the peak event in observed data is used, instead of the largest recorded simulated event. In some sense this captures the timing of the event, i.e. the peak event must be simulated on the same day as the observed peak to be captured well.

The sim and obs should be of time series (xts) format and are assumed to be of the same length and time period. The flow series are assumed to be daily flows with units of m3/s.

The R2 diagnostic is calculated for a fit with no intercept (in a perfect fit the points are identical, and intercept is automatically zero).

Note that a plot title is purposely omitted in order to allow the automatic generation of plot titles.

Value

returns a list with peak data in a data frame, and a ggplot object

df_peak_event	data frame of the calculated peak events
p1	ggplot object with plotted annual peaks

See Also

rvn_annual_peak to create a scatterplot of annual peaks (consider the magnitude of peaks only)

Examples

```
# load sample hydrograph data, two years worth of sim/obs
data(rvn_hydrograph_data)
sim <- rvn_hydrograph_data$hyd$Sub36
obs <- rvn_hydrograph_data$hyd$Sub36_obs

# create a plot of annual peak events with default options
peak1 <- rvn_annual_peak_event(sim, obs)
peak1$df_peak_event
peak1$p1
```

---

rvn_annual_peak_event_error

Annual Peak Event Errors

Description

rvn_annual_peak_event_error creates a plot of the annual observed and simulated peak event errors.

Usage

```
rvn_annual_peak_event_error(
  sim,
  obs,
  mm = 9,
  dd = 30,
  add_line = TRUE,
  add_labels = TRUE
)
```

Arguments

sim	time series object of simulated flows
obs	time series object of observed flows
mm	month of water year (default 9)
dd	day of water year (default 30)
add_line	optionally adds a 1:1 line to the plot for reference (default TRUE)
add_labels	optionally adds labels for overpredict/underpredict on right side axis (default TRUE)

Details

Creates a plot of the percent errors in simulated peak events for each water year. The peaks are calculated as using flows from the same day as the peak event in the observed series, i.e. the timing of the peak is considered here. Note that the rvn_annual_peak_event function is first used to obtain the peaks in each year, then the percent errors are calculated.

The percent errors are calculated as (QP_sim-QP_obs)/QP_obs*100, where QP is the peak flow event.

The sim and obs should be of time series (xts) format and are assumed to be of the same length and time period. The flow series are assumed to be daily flows with units of m3/s.

The add_labels will add the labels of 'overprediction' and 'underprediction' to the right hand side axis if set to TRUE. This is useful in interpreting the plots.

Note that a plot title is purposely omitted in order to allow the automatic generation of plot titles.

Value

returns a list with peak event error data in a data frame, and a ggplot object

df_peak_event_error	data frame of the calculated peak event errors
p1	ggplot object with plotted annual peak event errors

See Also

rvn_annual_peak to consider just the magnitude of each year's peak rvn_annual_peak_error to calculate errors in peaks

Examples

```
# load sample hydrograph data, two years worth of sim/obs
data(rvn_hydrograph_data)
sim <- rvn_hydrograph_data$hyd$Sub36
obs <- rvn_hydrograph_data$hyd$Sub36_obs

# create a plot of peak annual errors with default options
peak1 <- rvn_annual_peak_event_error(sim, obs)
peak1$df_peak_event_error
peak1$p1
```

`rvn_annual_peak_timing_error` *Annual Peak Timing Errors*

Description

rvn_annual_peak_timing_error creates a plot of the annual observed and simulated peak timing errors, based on the water year.

Usage

```
rvn_annual_peak_timing_error(
  sim,
  obs,
  mm = 9,
  dd = 30,
  add_line = TRUE,
  add_labels = TRUE
)
```

Arguments

`sim`	time series object of simulated flows
`obs`	time series object of observed flows
`mm`	month of water year (default 9)
`dd`	day of water year (default 30)
`add_line`	optionally adds a 1:1 line to the plot for reference (default `TRUE`)
`add_labels`	optionally adds labels for early peak/late peaks on right side axis (default `TRUE`)

Details

Creates a plot of the peak timing errors in simulated peaks for each water year. The difference in days between the simulated peak and observed peak are plotted (and/or returned in the data frame) for the water year. This diagnostic is useful in determining how accurate the timing of peak predictions is. Note that a large error in the number of days between simulated and observed peaks indicates that the model predicted a larger event at a different time of year, i.e. overestimated a different event or underestimated the actual peak event, relative to the observed flow series.

The sim and obs should be of time series (xts) format and are assumed to be of the same length and time period. The flow series are assumed to be daily flows with units of m3/s. Note that a plot title is purposely omitted in order to allow the automatic generation of plot titles.

The add_labels will add the labels of 'early peak' and 'late peak' to the right hand side axis if set to TRUE. This is useful in interpreting the plots. Note that values in this metric of less than zero indicate an early prediction of the peak, and positive values mean a late prediction of the peak (since the values are calculated as day index of simulated peak - day index of observed peak).

Value

returns a list with peak timing errors in a data frame, and a ggplot object

`df_peak_timing_error`
: data frame of the calculated peak timing errors

`p1`
: ggplot object with plotted annual peak errors

See Also

`rvn_annual_peak_event` to consider the timing of peak events `rvn_annual_peak_event_error` to calculate errors in peak events

Examples

```
# load sample hydrograph data, two years worth of sim/obs
data(rvn_hydrograph_data)
sim <- rvn_hydrograph_data$hyd$Sub36
obs <- rvn_hydrograph_data$hyd$Sub36_obs

# create a plot of peak timing errors with defaults
peak1 <- rvn_annual_peak_timing_error(sim, obs, add_line=TRUE)
peak1$df_peak_timing_error
peak1$p1

# plot directly and without labels
rvn_annual_peak_timing_error(sim, obs, add_line=TRUE, add_labels=FALSE)
```

---

rvn_annual_quantiles *Calculates Yearly Median, Upper and Lower Quantiles of Flow*

Description

Calculate the quantiles for each day of the year based on the supplied time series.

Usage

```
rvn_annual_quantiles(
  hgdata,
  prd = NULL,
  Qlower = 0.1,
  Qupper = 0.9,
  water_year = TRUE,
  mm = 9
)
```

Arguments

hgdata	Time series object of observed or simulated flows
prd	time period for subset in character format "YYYY-MM-DD/YYYY-MM-DD"
Qlower	Decimal percentage of lower quantile value (default 0.1)
Qupper	Decimal percentage of upper quantile value (default 0.9)
water_year	boolean on whether to sort quantiles by water year start date (default TRUE)
mm	month of water year ending (default 9)

Value

qdat	Time series object of monthly median and quantile values

Author(s)

Leland Scantlebury, <leland@scantle.com>

Examples

```
system.file("extdata","run1_Hydrographs.csv", package="RavenR") %>%
rvn_hyd_read(.) %>%
rvn_hyd_extract(subs="Sub36",.) ->
hyd_data

# Calculate quantiles for the simulated hydrograph
qdat <- rvn_annual_quantiles(hyd_data$sim)
head(qdat)
```

---

rvn_annual_quantiles_plot

Plot of Annual Median, Upper and Lower Quantiles of Flow

Description

Creates a plot of the annual flow quantiles provided by the rvn_annual_quantiles function.

Usage

```
rvn_annual_quantiles_plot(
  qdat,
  mediancolor = "black",
  ribboncolor = "grey60",
  ribbonalpha = 0.5,
  explot = NULL
)
```

Arguments

qdat	Time series object generated by rvn_annual_quantiles
mediancolor	Color for the median line
ribboncolor	Color for the lower/upper quantile ribbon
ribbonalpha	Transparency of lower/upper quantile ribbon
explot	Existing ggplot object to which median line and quantile ribbon should be added

Value

p1 ggplot object of quantiles plot

Author(s)

Leland Scantlebury, <leland@scantle.com>

Examples

```
system.file("extdata","run1_Hydrographs.csv", package="RavenR") %>%
rvn_hyd_read(.) %>%
rvn_hyd_extract(subs="Sub36",.) ->
hyd_data

# Calculate quantiles for the simulated hydrograph
qdat <- rvn_annual_quantiles(hyd_data$sim)
head(qdat)

# Plot
p <- rvn_annual_quantiles_plot(qdat)
p  # view plot

# Add a second hydrograph to compare
qdat_sim <- rvn_annual_quantiles(hyd_data$sim)

p1 <- rvn_annual_quantiles_plot(qdat_sim, mediancolor = 'blue', ribboncolor = 'red', explot = p)
p1 # view plot
```

---

rvn_annual_volume *Annual Volume Comparison*

Description

Creates a plot of the annual observed and simulated volumes.

Usage

```
rvn_annual_volume(
  sim,
  obs,
  mm = 9,
  dd = 30,
  add_line = TRUE,
  add_r2 = FALSE,
  add_eqn = FALSE,
  add_labels = FALSE
)
```

Arguments

sim	time series object of simulated flows
obs	time series object of observed flows
mm	month of water year ending (default 9)
dd	day of water year ending (default 30)
add_line	optionally adds a 1:1 line to the plot for reference (default TRUE)
add_r2	optionally computes the R2 and adds to plot (default FALSE)
add_eqn	optionally adds the equation for a linear regression line through the origin (default FALSE)
add_labels	optionally adds year-ending labels to each point on plot using geom_text (default FALSE)

Details

Creates a scatterplot of the annual observed and simulated volumes, calculated for each available water year of data within the two series provided. The sim and obs should be of time series (xts) format and are assumed to be of the same length and time period. Note that missing values in the observed series will impact the volume estimation, and it is recommended that the NA values are filled in prior to use of this function.

The R2 diagnostic is calculated for a fit with no intercept (in a perfect fit the points are identical, and intercept is automatically zero).

Note that a plot title is purposely omitted in order to allow the automatic generation of plot titles.

Value

returns a list with annual volume data in a data frame, and a ggplot object

df_volume	data frame of the calculated annual volumes
p1	ggplot object with plotted annual volumes

See Also

rvn_flow_scatterplot to create a scatterplot of flow values

Examples

```
# load sample hydrograph data, two years worth of sim/obs
data(rvn_hydrograph_data)
sim <- rvn_hydrograph_data$hyd$Sub36
obs <- rvn_hydrograph_data$hyd$Sub36_obs

# create a plot of the annual volumes with defaults
rvn_annual_volume(sim,obs)

# create a plot of the annual volumes with r2
rvn_annual_volume(sim,obs,add_r2=TRUE, add_eqn=TRUE)

# create a plot of the annual volumes with year-ending labels
rvn_annual_volume(sim,obs, add_labels=TRUE)

# calculate annual volumes for different water years (e.g. ending Oct 31)
vv <- rvn_annual_volume(sim, obs, mm=10, dd=31)
vv$df.volume
vv$p1
```

---

rvn_apply_wyearly *Apply function for water year*

Description

rvn_apply_wyearly calculates a function FUN for the periods defined by the water year, similar to other functions of the form apply.<time period>, for example apply.daily, apply.monthly, etc.

Usage

```
rvn_apply_wyearly(x, FUN, ..., mm = 9, dd = 30)
```

Arguments

x	xts vector to calculate FUN for
FUN	the function to be applied
...	optional arguments to FUN
mm	month of water year ending (default 9)
dd	day of water year ending (default 30)

Details

The default water year start is October 1st, but may be adjusted with the mm and dd arguments. The values for mm and dd indicate the end of the water year period (i.e. mm=9 and dd=30 indicates a new water year on Oct 1).

Note that if using FUN=mean, please use FUN=colMeans instead.

See Also

rvn_wyear_indices for obtaining endpoints in the water year

Examples

```
# use sample forcing data (or use forcings_read to read in ForcingFunctions.csv)
data(rvn_forcing_data)

# apply mean (with colMeans) as FUN to daily average temperature
rvn_apply_wyearly(rvn_forcing_data$forcings$temp_daily_ave,colMeans,na.rm=TRUE)

# apply mean as FUN to all forcings
rvn_apply_wyearly(rvn_forcing_data$forcings,colMeans,na.rm=TRUE)

# apply maximum via RavenR::cmax as FUN to all forcings (takes the max in each column)
## note that the base::max will not work properly here
rvn_apply_wyearly(rvn_forcing_data$forcings,cmax,na.rm=TRUE)

# apply to Australian water year (July 1)
rvn_apply_wyearly(rvn_forcing_data$forcings,cmax,na.rm=TRUE, mm=6, dd=30)
```

---

rvn_apply_wyearly_which_max_xts
which.max over water year periods

Description

Applies the which.max function within each water year period, and returns the corresponding max values and dates in an xts format.

Usage

```
rvn_apply_wyearly_which_max_xts(x, mm = 9, dd = 30)
```

Arguments

x	xts object
mm	month of water year ending (default 9)
dd	day of water year (default 30)

Value

xts object with max values and corresponding dates

Examples

```
data(rvn_hydrograph_data)

# obtain peak observed flows in each water year period
rvn_apply_wyearly_which_max_xts(rvn_hydrograph_data$hyd$Sub43_obs)

# will return a warning with no result if multiple columns supplied
rvn_apply_wyearly_which_max_xts(rvn_hydrograph_data$hyd)
```

---

rvn_budyko_plot *Budkyo Plot*

Description

rvn_budyko_plot creates a Budyko plot, adding supplied data points if provided.

Usage

```
rvn_budyko_plot(
  x = NULL,
  x_indices = NULL,
  limiting_labels = FALSE,
  budyko_curve = FALSE,
  mm = 9,
  dd = 30
)
```

Arguments

x	extensible time series object of PET, AET, and PRECIP (optional)
x_indices	extensible time series object of annual ARIDITY and EVAPORATION indices (optional)
limiting_labels	boolean whether to vertical line at x=1 and labels for 'Energy Limited' and 'Water Limited' to plot
budyko_curve	boolean whether to add curve to plot
mm	month of water year ending (default 9)
dd	day of water year ending (default 30)

Details

Creates a blank Budyko curve plot if no data is provided. Labels may optionally be added to the plot with `limiting_labels=TRUE` to indicate where in the curve the energy-limited and water-limited limits are. The original Budyko curve may also be added with `budyko_curve=TRUE`.

Data may be provided and plotted in the graph as well. If data is provided, it can be provided as: - x: an xts object with PRECIP, AET, and PET columns - x_indices: an xts object with indices calculated for each year, columns named ARIDITY and EVAPORATIVE

Value

p1 returns Budyko plot as ggplot object

References

Budyko, M.I. (1974), Climate and Life, Academic Press, New York.

See Also

rvn_watershedmeb_read for reading in the WatershedMassEnergyBalance.csv file, and rvn_apply_wyearly to apply functions over the water year.

Examples

```
# return blank Budyko plot
rvn_budyko_plot()

# return blank plot with labels and curve added
rvn_budyko_plot(limiting_labels=TRUE, budyko_curve=TRUE)

# plot sample data on Budyko plot (two years of data)
wstor <- system.file("extdata","run1_WatershedStorage.csv", package="RavenR") %>%
           rvn_watershed_read()
ff <- system.file("extdata","run1_ForcingFunctions.csv", package="RavenR") %>%
           rvn_forcings_read()

library(xts)
precip <- ff$forcings$rain+ff$forcings$snow
pet <- ff$forcings$PET
aet <- diff.xts(x=wstor$watershed_storage$Cum..Losses.to.Atmosphere..mm.,
          k=1, na.pad=TRUE)
aet[1] <- 0

x <- merge.xts(precip,pet,aet)
names(x) <- c("precip","pet","aet")
rvn_budyko_plot(x=x, budyko_curve=TRUE)
```

---

rvn_calc_runoff_coeff *Generate runoff coefficients upstream of gauges*

Description

Uses the rvh, custom precipitation, and hydrograph information to determine runoff coefficients.

Usage

```
rvn_calc_runoff_coeff(
  rvhfile,
  custfile = "PRECIP_Daily_Average_BySubbasin.csv",
  hydfile = "Hydrographs.csv",
  correct = FALSE
)
```

Arguments

rvhfile	file path to Raven rvh file
custfile	file path to Raven-generated custom output precip-by-subbasin file
hydfile	file path to Raven-generated hydrographs file
correct	(optional) if TRUE, tries to correct runoff coefficient for missing data (assumes missing~0 flow)

Details

Reads model.rvh file and daily avg subbasin precip file (usually PRECIP_Daily_Average_BySubbasin.csv) and generates data frame describing runoff coefficients of gauged basins and observation data coverage. Uses precipitation from entire model run history. Only determines runoff coefficient from available data - prone to overestimation with poor observation coverage.

Value

data frame with runoff coefficients of gauged basins

Author(s)

James R. Craig, University of Waterloo

See Also

rvn_rvh_read for reading and processing Raven rvh file

Examples

```
myrvh <- system.file("extdata","Nith.rvh", package="RavenR")
mycust <- system.file("extdata", "run1_PRECIP_Daily_Average_BySubbasin.csv", package="RavenR")
myhyd <- system.file("extdata", "run1_Hydrographs.csv", package="RavenR")

rcs <- rvn_calc_runoff_coeff(myrvh, mycust, myhyd, correct=TRUE)
rcs

# create a bar plot
runcoefs <- subset(rcs,select=c(runoff_coeff_sim,runoff_coeff_obs))

bp <- barplot(t(as.matrix(runcoefs)),
 main="Runoff Coefficient Comparison (w/ rough data coverage correction)",
```

```
ylab = "Runoff coeff", ylim=c(0,1),beside=TRUE,
col=c("blue","deepskyblue"),legend.text=c("sim","obs"),las=2)
```

---

rvn_csv_read *Read in generic Raven output csv files*

Description

Reads in output csv files produced by Raven.

Usage

```
rvn_csv_read(ff = NA, tzone = "UTC", xtsformat = TRUE)
```

Arguments

ff	full file path to the csv file
tzone	string indicating the timezone of the data in ff
xtsformat	boolean whether to return in xts format (if date and/or hour found)

Details

Expects a full file path to the Raven output file.

The timezone is provided by the tzone argument as "UTC" by default, and should be adjusted by the user to the local time zone as needed, based on the model run.

Value

data frame (as xts if set with `xtsformat`) read from the file

See Also

rvn_hyd_read for reading Hydrographs output files

Examples

```
# create full file path
ff <- system.file("extdata","ReservoirStages.csv", package="RavenR")

# read in the Reservoir file with the generic call
myres <- rvn_csv_read(ff)

# view contents
head(myres)
```

`rvn_cum_plot_flow` *Cumulative Plot of model flows*

Description

rvn_cum_plot_flow creates a cumulative flow plot of the simulated flows; optionally includes an observed and/or inflow series as well. Useful in diagnotic analysis of model outputs.

Usage

```
rvn_cum_plot_flow(sim = NULL, obs = NULL, inflow = NULL, mm = 9, dd = 30)
```

Arguments

`sim`	time series object of simulated flows
`obs`	optionally supply an inflow series to plot as well
`inflow`	optionally supply an inflow series to plot as well
`mm`	month of water year ending (default 9)
`dd`	day of water year ending (default 30)

Details

Plots the simulated series in all cases, and will include the observed and inflow plots if they are supplied.

The sim and obs should be of time series (xts) format. The flow series are assumed to be daily flows with units of m3/s.

Note that a plot title is purposely omitted in order to allow the automatic generation of plot titles.

Note that the cumsum function does not have an na.rm=T argument, thus if there are any NA values in the water year of data for any provided series, the values beyond an NA value will be calculated as NA. It is up to the user to handle NA values appropriately fill in or replace NA values based on the type of data supplied. For flow series, linear interpolation for small periods of missing values may be appropriate.

Value

`TRUE`	return `TRUE` if the function is executed properly

See Also

`rvn_flow_scatterplot` for creating flow scatterplots

`rvn_cum_plot_flow` for creating generic cumulative function plotting

Examples

```
# load sample hydrograph data, two years worth of sim/obs
data(rvn_hydrograph_data)
sim <- rvn_hydrograph_data$hyd$Sub36
obs <- rvn_hydrograph_data$hyd$Sub36_obs

# plot cumulative flow for sim and obs
rvn_cum_plot_flow(sim,obs)

# plot cumulative flows for specific period
prd <- "2003-10-01/2004-10-01"
rvn_cum_plot_flow(sim[prd],obs[prd])
```

---

rvn_custom_data *Custom Output Data from Raven*

Description

A dataset formatted to the xts package, read in by the rvn_custom_read function. The dataset contains average SNOW for each HRU in the Nith river model, available for download in the Raven Tutorials (linked below).

Note that this data set cannot be used with rvn_custom_output_plot as the file name information is not available in this data format. Please refer to the example in the plotting function to use the sample data file directly, which includes the filename information.

The Nith River model can be downloaded from the Raven Tutorials (tutorial #2) https://raven.uwaterloo.ca/Downloads.html

Usage

```
rvn_custom_data
```

Format

A data frame with 730 rows, containg data for 32 HRUs from 2002-10-01 to 2004-09-29

See Also

rvn_custom_read for reading in custom output files

rvn_custom_output_plot for plotting custom output

Examples

```
# Preview data
head(rvn_custom_data)
```

rvn_custom_output_plot *Plot Raven Custom Output*

Description

rvn_custom_output_plot is used to plot the custom output from Raven

Usage

```
rvn_custom_output_plot(cust, IDs = NULL, prd = NULL)
```

Arguments

cust	custom output object from custom.read
IDs	(optional) array of HRU IDs, subbasin IDs, HRU Group names/IDs to include in plots
prd	(optional) period to use in plotting

Details

The custom output should be first read in using the rvn_custom_read function. Note that in this case the plot title is included, generated from the information in the filename. This plot title may be changed with ggplot2 commands.

Value

TRUE	return TRUE if the function is executed properly

See Also

rvn_custom_output_plot for plotting custom output

Examples

```
# read in custom output from sample data
ff <- system.file("extdata/run1_SNOW_Daily_Average_ByHRU.csv", package="RavenR")
mycustomdata <- rvn_custom_read(ff)

# plot custom data (first 10 HRUs)
rvn_custom_output_plot(mycustomdata, IDs=seq(1,10), prd="2002-10-01/2003-06-01")
```

rvn_custom_read *Read Raven Custom Output files*

Description

rvn_custom_read is used to read any Raven custom output file

Usage

```
rvn_custom_read(ff = NA, no_runname = FALSE, tzone = "UTC")
```

Arguments

ff	full file path to the custom output file
no_runname	boolean for whether a runName is supplied, important for parsing the filename
tzone	string indicating the timezone of the data in ff

Details

rvn_custom_read parses the filename and predicts the file format accordingly, so it is important to use the unmodified file names for this function. The use (or not) of a runname is accounted for.

The returned object is a time series object (xts format), which can be used to easily plot the time series data. The options of the custom output are included in the rav.obj attributes.

The timezone is provided by the tzone argument as "UTC" by default, and should be adjusted by the user to the local time zone as needed, based on the model run.

Value

custom_out	data frame with the custom output data stored as xts object

See Also

rvn_custom_output_plot for plotting custom output

Examples

```
# find sample rvh file for Nith subwatershed
ff <- system.file("extdata","run1_SNOW_Daily_Average_ByHRU.csv", package="RavenR")

# extract and plot custom data
mycustomdata <- rvn_custom_read(ff)
summary(mycustomdata[,1:5])
plot(mycustomdata[,5],main='Daily Average SNOW - HRU 5')
```

`rvn_df_to_Raven_table` *Sets up tables for writing to Raven input files*

Description

Sets up tables for writing to Raven input files

Usage

```
rvn_df_to_Raven_table(attributes, units, df, id_col = TRUE, parameters = FALSE)
```

Arguments

`attributes`	array of strings containing attribute/parameter names
`units`	array of strings with the corresponding units
`df`	data frame of values corresponding to attributes/parameters
`id_col`	True/False of whether an numeric id column is the first column in the table and, in common Raven fashion, does not have a corresponding attribute (default: True)
`parameters`	bool, when adding attribues/parameter tag, should ':Parameters' be used instead of ':Attributes'?

Value

`outdf`	data.frame object

Author(s)

Leland Scantlebury, <leland@scantle.com>

Examples

```
soil_classes <- data.frame('Attributes' = c('DEFAULT','ALTERNATIVE'),
                           'SAND'      = c(0.4316, 0.3000),
                           'CLAY'      = c(0.1684, 0.4000),
                           'SILT'      = c(0.4000, 0.3000),
                           'ORGANIC'   = c(0.0000, 0.0000))
attributes <- c('%SAND','%CLAY','%SILT','%ORGANIC')
units <-  rep('none',4)
soil_classes <- rvn_df_to_Raven_table(attributes, units, soil_classes)
print(soil_classes)
```

`rvn_dist_lonlat` *Calculate distance from long/lat*

Description

Calculates distance between points based on a set of long/lat coordinates.

Usage

```
rvn_dist_lonlat(p1, p2, method = "haversine", r = 6378137)
```

Arguments

`p1`	longitude/latitude of point(s); can be a vector of two numbers, or a matrix of 2 columns (long/lat).
`p2`	second point in same format as `p1`
`method`	calculation method as either `haversine` (default) or `vincentysphere`
`r`	radius of the Earth in metres (default 6378137)

Details

Calculates distance in metres based on the longitude and latitude of two or more sets of points.

The function uses either the Haversine or Vincenty Sphere methods to calculate the distances.

Value

a vector of calculated distances (length of vector based on input)

Note

Function is based on modifications from the geosphere package scripts for `distHaversine` and `distVincentySphere`.

Examples

```
# calculate distance from Engineering 2 (p1) to Graduate House (p2) at the University of Waterloo
p1 <- c(-80.5402891965711,43.47088594350457)
p2 <- c(-80.54096577853629,43.46976096704924)
rvn_dist_lonlat(p1, p2)

# distance from University of Waterloo to Windsor
p2 <- c(-83.02099905916948,42.283371378771555)
rvn_dist_lonlat(p1, p2)
```

rvn_download *Downloads Raven*

Description

Downloads Raven executable from the Raven webpage.

Usage

```
rvn_download(version = NA, NetCDF = FALSE, check = FALSE, copy_path = NULL)
```

Arguments

version	(optional) Character: The version of Raven to be downloaded. If not provided, the latest version will be downloaded.
NetCDF	(logical) whether to download the NetCDF-enabled version of Raven (default FALSE)
check	(logical) if TRUE, function will only check whether 'Raven.exe' has been downloaded to the RavenR folder
copy_path	(character) path to an existing 'Raven.exe' file. If provided, this file will be copied to the appropriate folder and the download will be skipped.

Details

Files are downloaded from the Raven webpage (https://raven.uwaterloo.ca/Downloads.html) and placed in a temporary directory while the executable is extracted. This helps prevent any unwanted files being saved on your system outside of temporary directories if the function is interrupted. The executable is placed in the RavenR/extdata directory, wherever RavenR is installed on your system.

Note that if you are not on a Windows operating system, you may need to compile Raven for your system rather than

Any existing Raven.exe file in your RavenR/extdata folder will be overwritten with a newly downloaded executable w with this function.

The 'copy_path' argument may be useful for non-Windows users that are not able to use the precompiled Raven downloads. If the path to an existing Raven.exe file (i.e. one compiled locally) is provided, this will copy the specified file to the appropriate location and skip the download, which will allow the user to use the locally compiled Raven.exe with the 'rvn_run' function.

Reinstalling the RavenR package removes the previous files in the RavenR/ folder on your system, so this command will need to be re-run if RavenR is re-installed.

Once downloaded, the Raven.exe file can be found with `system.file("extdata", "Raven.exe", package="RavenR")`

Value

Returns TRUE if executed successfully

See Also

rvn_run

Examples

```
# check if Raven.exe has previously been downloaded
rvn_download(check=TRUE)

## Not run:
## NOT RUN (downloads executable)

# download latest without netcdf support
rvn_download()

# download specific version with netcdf support
rvn_download(version="3.0.4",NetCDF=TRUE)

# find file path to Raven.exe
system.file("extdata", "Raven.exe", package="RavenR")

## End(Not run)
```

`rvn_exhaustive_mb_read` *Read in Raven Exhaustive Mass Balance file*

Description

rvn_exhaustive_mb_read is used to read in the ExhaustiveMassBalance.csv file produced by the modelling Framework Raven.

Usage

```
rvn_exhaustive_mb_read(ff = NA, join_categories = TRUE, tzone = NULL)
```

Arguments

ff	full file path to the ExhaustiveMassBalance.csv file
join_categories	boolean whether add to the category tag as a column name prefix in exhaustivemb output (default TRUE)
tzone	string indicating the timezone of the data in ff

Details

Expects a full file path to the ExhaustiveMassBalance.csv file, then reads in the file using read.csv. The main advantage of this function is renaming the columns to nicer names and extracting the units into something that is much easier to read.

ff is the full file path of the ExhaustiveMassBalance.csv file. If the file is located in the current working directory, then simply the name of the file is sufficient.

tzone is a string indicating the timezone of the supplied file. The timezone provided is coded into the resulting data frame using the as.POSIXct function. If no timezone is provided, this is left as an empty string, and is determined by the function as the current time zone.

Value

`exhaustivemb`	data frame from the file with standardized names
`units`	vector corresponding to units of each column
`categories`	vector corresponding to the storage category of each column

See Also

`rvn_hyd_read` for reading in the Hydrographs.csv file, and `rvn_exhaustive_mb_read` for reading in the WatershedMassEnergyBalance.csv file

Examples

```
# Read in exhaustive MB file, create plot
ff <- system.file("extdata","run1_ExhaustiveMassBalance.csv", package="RavenR")
embd <- rvn_exhaustive_mb_read(ff)

# Preview data
head(embd$exhaustive_mb)

# Plot data
plot(embd$exhaustive_mb$SURFACE_WATER.Infiltration,
     main="Cumulative Surface Water Infiltration")
```

---

rvn_fdc_plot *Plots summary of watershed forcing functions*

Description

rvn_fdc_plot generation a flow duration curve plot.

Usage

```
rvn_fdc_plot(sim = NULL, obs = NULL, prd = NULL, seasonal = FALSE)
```

Arguments

sim	simulated hydrograph xts time series
obs	(optional) observed hydrograph xts time series
prd	(optional) time period over which the plot is generated
seasonal	(optional) boolean whether to add the winter and summer FDC

Details

Creates a flow duration curve using the rvn_hyd_extract obejct for a given basin. The hydrograph object passed should be the output from the rvn_hyd_extract function, which has attributes for sim and obs; if the obs is NULL, only the sim FDC will be plotted.

If the seasonal argument is included, the winter and summer FDC lines will be included on the plot as well.

See Also

rvn_hyd_read for reading in the Hydrographs.csv file, and rvn_hyd_extract for extracting basin flow information from a rvn_hyd_read object

Examples

```
# load sample hydrograph data, two years worth of sim/obs
ff <- system.file("extdata/run1_Hydrographs.csv", package="RavenR")
run1 <- rvn_hyd_read(ff)
sim <- run1$hyd$Sub36
obs <- run1$hyd$Sub36_obs

# create FDC plot, sim only
rvn_fdc_plot(sim)

 # create seasonal FDC plot with sim and obs data
rvn_fdc_plot(sim,obs,seasonal=TRUE)
```

---

rvn_flow_residuals *Residuals of model flows*

Description

rvn_flow_residuals creates a residuals time series for flow values. Useful in diagnotic analysis of model outputs.

Usage

```
rvn_flow_residuals(
  sim = NULL,
  obs = NULL,
  ma_smooth = 3,
  add_line = FALSE,
  winter_shading = FALSE,
  wsdates = c(12, 1, 3, 31)
)
```

Arguments

sim	time series object of simulated flows
obs	time series object of observed flows
ma_smooth	optional length of rolling average to smooth residuals with (default 3)
add_line	optionally adds a horizontal line to the plot for reference (default FALSE)
winter_shading	optionally adds a light blue shading to winter months (default FALSE)
wsdates	integer vector of winter shading period dates (see details)

Details

Creates a residuals time series plot for flow values, with the option to smooth out the values using the rollmean function in zoo. The winter months are optionally shaded in the time series; winter period is defined as December 1st to March 31st.

The residuals are calculated as sim - obs.

The sim and obs should be of time series (xts) format. The flow series are assumed to be daily flows with units of m3/s.

Note that a plot title is purposely omitted in order to allow the automatic generation of plot titles.

The winter_shading argument will add a transparent grey shading for the specified period by wsdates in each year that is plotted.

wsdates is formatted as c(winter start month, winter start day, winter end month, winter end day).

Value

resids	residual time series

See Also

rvn_flow_scatterplot to create a scatterplot of flow values

Examples

```
# load sample hydrograph data, two years worth of sim/obs
ff <- system.file("extdata/run1_Hydrographs.csv", package="RavenR")
run1 <- rvn_hyd_read(ff)
```

```
sim <- run1$hyd$Sub36
obs <- run1$hyd$Sub36_obs

# default with moving average smoothing shading of winter months
rvn_flow_residuals(sim,obs)$plot

# plot with more smoothing than the default 3
rvn_flow_residuals(sim, obs, ma_smooth=10)$plot

# with zero line and winter shading
rvn_flow_residuals(sim,obs, add_line=TRUE, winter_shading = TRUE)$plot

# change winter shading to Nov 1 - April 30
rvn_flow_residuals(sim,obs, add_line=TRUE,
  winter_shading = TRUE, wsdates=c(11,1,4,30))$plot
```

`rvn_flow_scatterplot` *Scatterplot of model flows*

Description

rvn_flow_scatterplot creates a scatterplot of the simulated and observed flows. Useful in diagnotic analysis of model outputs.

Usage

```
rvn_flow_scatterplot(
  sim,
  obs,
  add_line = TRUE,
  add_r2 = FALSE,
  add_eqn = FALSE
)
```

Arguments

`sim`	time series object of simulated flows
`obs`	time series object of observed flows
`add_line`	optionally adds a 1:1 line to the plot for reference (default `TRUE`)
`add_r2`	optionally computes the R2 and adds to plot (default `FALSE`)
`add_eqn`	optionally adds the equation for a linear regression line (default `FALSE`)

Details

Creates a scatterplot of flows.

The sim and obs should be of time series (xts) format. The flow series are assumed to be daily flows with units of m3/s.

The R2 diagnostic is calculated for a fit with no intercept (in a perfect fit the points are identical, and intercept is automatically zero). The R2 is calculated with the NA values removed

Note that a plot title is purposely omitted in order to allow the automatic generation of plot titles.

Value

TRUE	return TRUE if the function is executed properly

See Also

rvn_forcings_read for reading in the ForcingFunctions file

Examples

```
# load sample hydrograph data, two years worth of sim/obs
data(rvn_hydrograph_data)
sim <- rvn_hydrograph_data$hyd$Sub36
obs <- rvn_hydrograph_data$hyd$Sub36_ob

# plot the flow scatterplot, produce an R2 metric
rvn_flow_scatterplot(sim,obs,add_r2=TRUE)

# plot again with a regression equation
rvn_flow_scatterplot(sim,obs,add_r2=TRUE,add_eqn=TRUE)
```

---

rvn_flow_spaghetti *Flow Spaghetti Plot*

Description

rvn_flow_spaghetti creates a spaghetti plot of the flow series provided.

Usage

```
rvn_flow_spaghetti(flow)
```

Arguments

flow	time series object of simulated flows

Details

Creates a spaghetti plot of the annual flow series in each year of data provided. The flows are plotted for each water year of data available, set as October 1st.

Note that the plotting to the day of year is approximate in order to simplify the plotting of leap years and non-leap years. The years are plotted including day 366 and starting on day 274, regardless of whether it is a leap year or not. This is likely without consequence in seeing the trends between water years, however the user is warned of this deficiency.

The flow series provided should be of time series (xts) format.

Note that a plot title is purposely omitted in order to allow the automatic generation of plot titles.

Value

TRUE	return TRUE if the function is executed properly

See Also

rvn_flow_scatterplot to create a scatterplot of flow values

Examples

```
# load sample hydrograph data, two years worth of sim/obs
data(rvn_hydrograph_data)

# create spaghetti plot of simulated flows
rvn_flow_spaghetti(rvn_hydrograph_data$hyd$Sub36)

# create spaghetti plot of observed flows
rvn_flow_spaghetti(rvn_hydrograph_data$hyd$Sub36_obs)
```

---

rvn_forcings_plot *Plots summary of watershed forcing functions*

Description

rvn_forcings_plot generates a set of 5 plots (precip,temperature,PET,radiation, and potential melt), which summarize the watershed-averaged forcings. Returns a list with the individual plots.

Usage

```
rvn_forcings_plot(forcings, prd = NULL)
```

Arguments

forcings	forcings attribute from forcings.read function
prd	(optional) time period over which the plots are generated

Details

Creates multiple plots from a ForcingFunctions.csv file structure generating using RavenR's forcings.read function

Value

forcing_plots list of ggplot objects of individual forcing plots and the combined plot

See Also

rvn_forcings_read for the function used to read in the forcings function data

Examples

```
# read in sample forcings data
data("rvn_forcing_data")
fdata <- rvn_forcing_data$forcings

# plot forcings data
p1 <- rvn_forcings_plot(fdata)
p1$Precipitation
p1$AllForcings

# plot subset of forcing data for 2002-2003 water year
prd <- "2002-10-01/2003-09-30"
rvn_forcings_plot(fdata,prd)$AllForcings

# add Legend back to plot (using ggplot2::theme)
library(ggplot2)
rvn_forcings_plot(fdata,prd)$Temperature+
theme(legend.position='top')
```

---

rvn_forcings_read *Read in Raven ForcingFunctions file*

Description

rvn_forcings_read is used to read in the ForcingFunctions.csv file produced by the modelling Framework Raven.

Usage

```
rvn_forcings_read(ff = NA, tzone = "UTC")
```

Arguments

ff	full file path to the ForcingFunctions.csv file
tzone	string indicating the timezone of the data in ff

Details

Expects a full file path to the ForcingFunctions.csv file, then reads in the file using read.csv. The main advantage of this functon is renaming the columns to nicer names and extracting the units into something much easier to read.

ff is the full file path of the ForcingFunctions.csv file. If the file is located in the current working directory, then simply the name of the file is sufficient.

The timezone is provided by the tzone argument as "UTC" by default, and should be adjusted by the user to the local time zone as needed, based on the model run.

Value

`forcings`	data frame from the file with standardized names
`units`	vector corresponding to units of each column

See Also

`rvn_hyd_read` for reading in the Hydrographs.csv file

Examples

```
# read in sample forcings data
ff <- system.file("extdata","run1_ForcingFunctions.csv", package="RavenR")
myforcings <- rvn_forcings_read(ff)

# check data (first 5 columns for brevity)
head(myforcings$forcings[,1:5])
summary(myforcings$forcings[,1:5])
```

`rvn_forcing_data` *Forcings Data from Raven*

Description

A dataset formatted to the xts package, read in by the forcings.read function. The dataset contains the typical columns from the Raven outputted ForcingFunctions.csv file, available for download in the Raven Tutorials (linked below).

Usage

```
rvn_forcing_data
```

Format

rvn_forcing_data is a data frame with two object

forcings various forcing functions and related output from Raven model

units units associated with each variable in forcings

rvn_forcing_data$forcings is an xts (time series) object with 731 rows and 21 variables, containing data from 2002-10-01 to 2004-09-30. The details of each forcing function can be found in the Raven Manual

- day_angle
- rain
- snow
- temp
- temp_daily_min
- temp_daily_max
- temp_daily_ave
- temp_monthly_min
- temp_monthly_max
- air_dens
- air_pres
- rel_hum
- cloud_cover
- ET_radiation
- SW_radiation
- LW_radiation
- wind_vel
- PET
- OW_PET
- daily_correction
- potential_melt

The Nith River model can be downloaded from the Raven Tutorials (tutorial #2) https://raven.uwaterloo.ca/Downloads.html

See Also

rvn_forcings_read for reading in forcing functions output files

rvn_forcings_plot for plotting forcing functions in a convenient way

rvn_fortify_xts *Fortify xts object to specific format*

Description

Applies the fortify function to an xts object and updates the Index character column to a date column called 'Date'.

Usage

```
rvn_fortify_xts(x)
```

Arguments

x	xts formatted object to fortify to tibble

Details

Useful in preparing data to plotting or other tidy-style analysis. This function is used internally in many RavenR plotting functions.

Value

tibble format of the xts data

Examples

```
ff <- system.file("extdata","run1_Hydrographs.csv", package="RavenR")
hyd <- rvn_hyd_read(ff)$hyd
hyd_fortified <- rvn_fortify_xts(hyd)
head(hyd_fortified)
```

rvn_gen_obsweights *Create weights time series for calibration/diagnostic evaluation*

Description

Creates an observation data weights time series based upon dates stored in an xts time series and criterion given by the user

Usage

```
rvn_gen_obsweights(
  ts,
  criterion = "BETWEEN",
  startdate = "1785-10-05",
  enddate = "2500-01-01"
)
```

Arguments

ts	xts time series
criterion	criterion used to determine weighted vs. non-weighted dates one of 'BEFORE', 'AFTER', 'BETWEEN', 'BETWEEN_CYCLIC'
startdate	Date indicating start of time period (for 'BETWEEN' or 'AFTER' criterion) or start of annual cyclic period (for 'BETWEEN_CYCLIC'). In the latter case, only the julian day of the startdate matters.
enddate	Date indicating end of time period (for 'BETWEEN' or 'BEFORE' criterion) or end of annual cyclic period (for 'BETWEEN_CYCLIC'). In the latter case, only the julian day of the enddate matters.

Details

for criterion ='BEFORE', all timestamps prior to the enddate have a weight of 1, 0 otherwise
for criterion ='AFTER', all timestamps after the startdate have a weight of 1, 0 otherwise
for criterion ='BETWEEN', all timestamps after the startdate and before the enddate have a weight of 1, 0 otherwise
for criterion ='BETWEEN_CYCLIC', all julian days after the startdate and before the enddate have a weight of 1, 0 otherwise; if startdate is more than enddate, then the opposite is true, e.g, for startdate="2002-11-01" and enddate "2002-01-31", only November, December and January timestamps have a weight of 1

Value

returns numeric vector of weights

Author(s)

James R. Craig, University of Waterloo

See Also

rvn_rvt_write to write the weights to an rvt file

Examples

```
# locate hydrograph sample csv data from RavenR package
ff <- system.file("extdata","run1_Hydrographs.csv", package="RavenR")

# read in Raven Hydrographs file, store into mydata
mydata <- rvn_hyd_read(ff, tzone="EST")

# generate rvt file using just observations from Subbasin ID 36
flows <- rvn_ts_infill(mydata$hyd$Sub36_obs)

# weight March-October flows:
wts <- rvn_gen_obsweights(flows,criterion = "BETWEEN_CYCLIC",
  startdate="2000-03-01", enddate="2003-11-01")
```

```
# and only after March 2003:
wts2 <- rvn_gen_obsweights(flows,criterion = "AFTER",
  startdate="2003-03-01")
wts2 <- wts2*wts # product merges weights

# show weights over time
plot(wts2)
```

`rvn_get_prd` *Check period input*

Description

Checks a period argument either as a character or against an xts object.

Usage

```
rvn_get_prd(x = NULL, prd = NULL)
```

Arguments

`x`	xts object
`prd`	period argument in format YYYY-MM-DD/YYYY-MM-DD as a character

Details

The function may take some combination of an xts object, a character string or both.

If a character is provided, the consistency of the character string against the YYYY-MM-DD/YYYY-MM-DD format is checked. If an xts object is provided, the period for that xts object is returned. If both are provided to the function, both checks are made and the consistency of the character period against the xts object is performed. In any case, a period character string is returned.

Value

prd argument with warnings provided if needed

See Also

`rvn_theme_RavenR` provides a theme for the RavenR package

Examples

```
data(rvn_hydrograph_data)

# check if string is a valid prd argument
rvn_get_prd(prd="2000-10-01/2002-09-30")
# rvn_get_prd(prd="2000-10-01/2002-24-30") # returns error

# get full valid prd argument for xts object
rvn_get_prd(rvn_hydrograph_data$hyd$Sub43_obs)

# check prd argument against xts object
rvn_get_prd(rvn_hydrograph_data$hyd$Sub43_obs, "2020-01-01/2020-02-01")
# rvn_get_prd(rvn_hydrograph_data$hyd$Sub43_obs, "2002-24-01/2020-02-01") # returns error
# rvn_get_prd(rvn_hydrograph_data$hyd$Sub43_obs, "20-24-01/2020-02-01")  # returns error
```

`rvn_hydrograph_data` *Hydrograph Data from Raven*

Description

A dataset formatted to the xts package, read in by the hyd.read function. The dataset contains the typical columns from the Raven outputted Hydrographs.csv file, available for download in the Raven Tutorials (linked below).

Usage

```
rvn_hydrograph_data
```

Format

rvn_hydrograph_data is a data frame with two object

hyd simulated and observed flows from Raven model

units units associated with each variable in hyd

obs.flag flag for each column indicating whether it is observed data or not

rvn_hydrograph_data$hyd is an xts (time series) object with 731 rows and 5 variables, with data from 2002-10-01 to 2004-09-30. More details on the Hydrographs.csv output can be found in the Raven manual.

precip - precipitation time series

Sub36 - outflows from subbasin 36 in Nith model

Sub36_obs - observed outflows from subbasin 36

Sub43 - outflows from subbasin 43 in Nith model

Sub43_obs - observed outflows from subbasin 43

The Nith River model can be downloaded from the Raven Tutorials (tutorial #2) https://raven.uwaterloo.ca/Downloads.html

See Also

rvn_custom_read for reading in custom output files

rvn_custom_output_plot for plotting custom output

---

rvn_hyd_dygraph *Read in Raven Hydrograph file*

Description

rvn_hyd_dygraph plots modeled vs observed hydrographs when supplied with hydrograph data structure read using rvn_hyd_read

Usage

```
rvn_hyd_dygraph(
  hy,
  timezone = "UTC",
  basins = "",
  main = NULL,
  figheight = 400
)
```

Arguments

hy	hydrograph data structure generated by rvn_hyd_read routine
timezone	data timezone; defaults to UTC
basins	list of subbasin names from hydrograph file. Each subbasin creates separate dygraph plots
main	dygraph title to override the default (applied to all dygraphs)
figheight	height of figure, in pixels

Value

a list of plot handles to dygraph plots

Examples

```
# read in RavenR sample hydrographs data
hy <- rvn_hydrograph_data

# view contents for subbasin 36 as dyGraph
dyplots <- rvn_hyd_dygraph(hy,basins="Sub36")
dyplots

rvn_hyd_dygraph(hy,basins="Sub36", main="test title")
```

```
# view contents for all basins in hydrograph data
rvn_hyd_dygraph(hy)
```

`rvn_hyd_extract` *Extract function for Raven Hydrograph object*

Description

rvn_hyd_extract is used for extracting data from the Raven hydrograph object. Works for objects passed from rvn_hyd_read function (which reads the Hydrographs.csv file produced by the modelling framework Raven).

Usage

```
rvn_hyd_extract(subs = NA, hyd = NA, prd = NULL, rename_cols = TRUE)
```

Arguments

`subs`	column name for plotting/extracting
`hyd`	full hydrograph data frame (including units) produced by rvn_hyd_read
`prd`	time period for plotting, as string. See details
`rename_cols`	boolean for whether to rename columns to generic terms (sim, obs, etc.) or leave column names as they appear in hyd

Details

Extracts the modelled and observed data from a Raven hydrograph object by name reference. It is also easy to create plots of modelled and observed data using this function. The simulated and observed files are outputted regardless of whether a plot is created, for the specified period.

The subs input is the name of the column desired for use; the most common use of this will be for subbasin outflows, where the names will be of the form "subXX", for example "sub24".

The hyd object is the full hydrograph object (hyd and units in one data frame) created by the rvn_hyd_read function. Both the hyd and units are required, since the units are placed onto the plots if one is created. This is useful to at least see the units of the plotted variable, even if the plot is later modified.

The prd input is used to specify a period for the plot and/or the data output. The period should be specified as a string start and end date, of the format "YYYY-MM-DD/YYYY-MM-DD", for example, "2006-10-01/2010-10-01". If no period is supplied, the entire time series will be used.

Value

returns an xts object with sim, obs, inflow, and obs_inflow time series (if available)

sim	model simulation for specified column and period
obs	observed data for specified column and period
inflow	inflow simulation for specified column and period
obs_inflow	observed inflow simulation for specified column and period

See Also

rvn_hyd_read for reading in the Hydrographs.csv file and creating the object required in this function. rvn_hyd_plot for conveniently plotting the output object contents onto the same figure.

Examples

```
# read in hydrograph sample csv data from RavenR package
ff <- system.file("extdata","run1_Hydrographs.csv", package="RavenR")

# read in Raven Hydrographs file, store into myhyd
myhyd <- rvn_hyd_read(ff)

# no plot or observed data, specified period
flow_36 <- rvn_hyd_extract(subs="Sub36",myhyd)

attributes(flow_36)

# extract simulated and observed flows
sim <- flow_36$sim
obs <- flow_36$obs

# extract precipitation forcings
myprecip <- rvn_hyd_extract(subs="precip",hyd=myhyd)
myprecip <- myprecip$sim

# plot all components using rvn_hyd_plot
rvn_hyd_plot(sim,obs,precip=myprecip)
```

---

rvn_hyd_plot *Create Hydrograph Plot*

Description

rvn_hyd_plot creates a hydrograph plot object for the supplied flow series, or equivalently a stage plot for reservoir stages.

Usage

```
rvn_hyd_plot(
  sim = NULL,
  obs = NULL,
  inflow = NULL,
  precip = NULL,
  prd = NULL,
  winter_shading = FALSE,
  wsdates = c(12, 1, 3, 31)
)
```

Arguments

sim	time series object of simulated flows
obs	time series object of observed flows
inflow	time series object of inflows to subbasin
precip	time series object of precipitation
prd	period to use in plotting
winter_shading	optionally adds shading for winter months (default FALSE)
wsdates	integer vector of winter shading period dates (see details)

Details

Creates a hydrograph plot using the supplied time series; any series not supplied will not be plotted. If the precip time series is supplied, the secondary y axis will be used to plot the precip time series.

The function assumes that the supplied time series have the same length and duration in time. If this is not true, then the defined period or period calculated from the first available flow series will be used to determine the plotting limits in time. If the data is used directly from Raven output, this is not a concern. The supplied time series should be in xts format, which again can be obtained directly by using the hyd.extract function.

The winter_shading argument will add a transparent grey shading for the specified period by wsdates in each year that is plotted.

wsdates is formatted as c(winter start month, winter start day, winter end month, winter end day).

Note that a plot title is purposely omitted in order to allow the automatic generation of plot titles.

Value

p1	returns ggplot plot object

See Also

rvn_flow_spaghetti to create a spaghetti plot of annual flow series

rvn_hyd_extract to extract time series from Raven objects

Examples

```
# load sample hydrograph data, two years worth of sim/obs
ff <- system.file("extdata","run1_Hydrographs.csv", package="RavenR")
run1 <- rvn_hyd_read(ff)
sim <- run1$hyd$Sub36
obs <- run1$hyd$Sub36_obs
precip <- run1$hyd$precip

# create a nice hydrograph
rvn_hyd_plot(sim,obs)

# create a hydrograph with precip as well;
rvn_hyd_plot(sim,obs,precip=precip)

# create a hydrograph with precip as well for a specific subperiod
prd <- "2003-10-01/2004-10-01"
rvn_hyd_plot(sim,obs,precip=precip,prd=prd)

# add the winter shading
rvn_hyd_plot(sim,obs,precip=precip,prd=prd, winter_shading=TRUE)

# change winter shading dates (Nov 1st to April 15th)
rvn_hyd_plot(sim,obs,precip=precip,prd=prd, winter_shading=TRUE, wsdates=c(11,1,4,15))
```

---

rvn_hyd_read *Read in Raven Hydrograph file*

Description

rvn_hyd_read is used to read in the Hydrographs.csv file produced by the modelling Framework Raven.

Usage

```
rvn_hyd_read(ff = NA, tzone = "UTC")
```

Arguments

ff	full file path to the Hydrographs.csv file
tzone	string indicating the timezone of the data in ff (default "UTC")

Details

Expects a full file path to the Hydrographs.csv file, then reads in the file using fread. The main advantage of this function is renaming the columns to nicer names and extracting the units into something much easier to read.

This function is also built to support the rvn_hyd_extract function, which uses the object created here for extracting by reference to the columns named here, for example sub24.

ff is the full file path of the Hydrographs.csv file. If the file is located in the current working directory, then simply the name of the file is sufficient.

tzone is a string indicating the timezone of the supplied Hydrographs file. The timezone provided is coded into the resulting hyd data frame using the as.POSIXct function. The timezone is provided as "UTC" by default, and should be adjusted by the user to the local time zone as needed, based on the model run.

Value

hyd	data frame from the file with standardized names

See Also

rvn_hyd_extract for extraction tools related to the rvn_hyd_read output file

Examples

```
# read in hydrograph sample csv data from RavenR package
ff <- system.file("extdata","run1_Hydrographs.csv", package="RavenR")

# read in Raven Hydrographs file, store into myhyd
myhyd <- rvn_hyd_read(ff)

# view contents
head(myhyd$hyd)
myhyd$units
```

---

rvn_met_interpolate *Interpolate meteorological data using IDW*

Description

Interpolates/infills missing meteorological data by using an inverse-distance weighting scheme to infill using data from nearby meteorological stations; issues where maximum temperature is less than minimum temperature can also be resolved.

Usage

```
rvn_met_interpolate(
  weather_data = NULL,
  cc = c("max_temp", "min_temp", "total_precip"),
  key_stn_ids = NULL,
  ppexp = 2,
  fix_interp_temp = TRUE,
  fix_base_temp = FALSE,
```

```
  min_dist = 100
)
```

Arguments

weather_data	data frame of input meteorological data from multiple stations
cc	columns from weather_data to infill missing values in
key_stn_ids	station IDs in which to perform the interpolation
ppexp	exponent to use in inverse distance weighting calculation (default 2)
fix_interp_temp	function will swap interpolated min and max temp if appropriate
fix_base_temp	function will swap any min and max temp if appropriate
min_dist	minimum distance used in IDW to avoid issues with stations of different IDs but exact same coordinates (default 100m)

Details

This function takes a meteorological data set with multiple station data in one data frame and interpolates the missing values for key stations, for the specified meteorological variables to interpolate.

The format of the weather_data input is consistent with that from the `weathercan::weather_dl` function, which is the recommended tool to gather this input (see the examples).

This function does not guarantee to infill all missing values, since this depends on the availability of data at other locations when it lacks at a given station, although a warning is issued if missing values remain following the interpolation. If this is encountered, it may be prudent to consider including additional stations in the available data for interpolation and/or applying alternative interpolation schemes in conjunction with `rvn_met_interpolate`.

This function does not (currently) perform checks for the quality of the supplied or infilled data, such as checking for maximum temperature less than minimum temperature, unreasonable precipitation values, etc.

The key_stn_ids indicates which stations should have their missing values interpolated. It is likely that the user will require more data to perform a proper interpolation than the user cares to have gauge records at, i.e. some stations are only provided for the purposes of infilling missing data at other stations. Since the interpolation of all stations provided can become computationally expensive, the user can specify which stations they want to interpolate data for with the key_stn_ids parameter. Station IDs that are not within the key_stn_ids (if not NULL) will still be used, but not themselves infilled.

fix_interp_temp is a boolean for the function to swap any max_temp and min_temp fields where the maximum temperature is less than the minimum temperature, a common fix in messy environmental data. This function will only apply if the columns `max_temp` and `min_temp` are present, and will only apply to the interpolated fields. By default this is enabled, and automatic fixes to interpolated data will be done; supplied data where the max temp is less than min temp will not be fixed unless fix_base_temp is enabled.

The distance calculation, estimating the distance between stations, is performed using the `rvn_dist_lonlat` function, which is based on the `geosphere` package. The min_dist is included to provide a minimum distance used in IDW, and avoid issues with stations of different names and IDs but the same coordinates. This can occur when stations upgrade or are reinstated at a later time in the same location, which would otherwise result in a div/0 error in the IDW calculation.

Value

new_wd infilled meteorological data set

See Also

rvn_rvt_write_met for writing meteorological data sets to rvt format.

Examples

```
## Not run:
## NOT RUN (downloads data + long runtime)

# example to create infilled data sets

library(weathercan)
stn <- weathercan::stations_search(name="Glen allan", interval = "day")
dl_stn <- stn
all_stns <- weathercan::stations_search(coords=c(stn$lat, stn$lon), dist=40,
          interval="day", starts_latest = 2002,
          ends_earliest = 2010)

# download data with weathercan::weather_dl()
weather_data <- weather_dl(station_ids = all_stns$station_id,
                           start = "2002-10-01",
                           end = "2005-10-01",
                           interval="day")

# define first three as key stations of interest for infilling
dl_stn <- all_stns[1:3,]

# confirm missing data in key columns
key_cols <- c("min_temp","max_temp","total_precip")
length(which(is.na(weather_data[,key_cols])))

# perform interpolation for key stns (3) using all stations downloaded (5)
new_wd <- rvn_met_interpolate(weather_data = weather_data,
                              key_stn_ids = dl_stn$station_id,
                              cc = key_cols)

# no warning - confirm no missing values in key columns
length(which(is.na(new_wd[,key_cols])))

## End(Not run)
```

---

rvn_met_recordplot *EC Climate Gauge Record Overlap Visualization*

Description

This function plots the length of Environment Canada climate station records, accessed via the **weathercan** package, to identify periods in which multiple station records overlap.

Usage

```
rvn_met_recordplot(
  metadata = NULL,
  stndata = NULL,
  variables = NULL,
  colorby = NULL
)
```

Arguments

`metadata`	tibble of the station meta-data from `weathercan::stations_search()`
`stndata`	tibble of the station data from `weathercan::weather_dl()`. Used in conjunction with `variables` argument.
`variables`	if using `weathercan::weather_dl()`, column names for variables of interest (currently only accepts 1 per call)
`colorby`	column name by which to color station records. Set to 'elev' (elevation) by default. Can be set to "dist" (distance from coordinates of interest) if supplying `weathercan::stations_search` results.

Details

Accepts outputs from either the `stations_search()` or `weather_dl()` functions from the **weathercan** package and extracts the start and end dates of the record from each station for plotting.

Outputs from `stations_search()` indicate when data collection at a station generally began but do not contain information for specific climate variables and thus should only be used for a "first look". Plots created with station metadata do not refer to specific climate variables.

Station records are plotted chronologically on a timeline, and can be colored according to either the station's elevation (default, works for both types of inputs) or the station's distance from a point of interest (works only when supplying `stations_search()` results as metadata input).

The timeline plot is accompanied by a bar plot counting the number of stations with available data year by year.

Large differences in elevation between stations may point towards consideration for the effect of lapse rates on climate forcings driving a model response.

Value

returns a 2x1 plot object containing 2 ggplot objects

`top:`	A chronological horizontal bar plot depicting each station's record period
`bottom:`	A vertical bar plot depicting the number of station records available each year

Examples

```
# load metadata from RavenR sample data
data(rvn_weathercan_metadata_sample)

## code that would be used to download metadata using weathercan
# library(weathercan)
#
# metadata = stations_search(coords=c(50.109,-120.787),
#    dist=150, # EC stations 150 km of Merritt, BC
#   interval='day'
# )
# metadata = metadata[metadata$start>=2000,] # subset stations with recent data
# metadata = metadata[1:3,] # take only the first 3 stations for brevity

# plot line colours by station elevation
rvn_met_recordplot(metadata=rvn_weathercan_metadata_sample, colorby='elev')

# plot line colours by distance to specified co-ordinates
rvn_met_recordplot(metadata=rvn_weathercan_metadata_sample, colorby='distance')

## load sample weathercan::weather_dl() with single station
data(rvn_weathercan_sample)

# compare records for a specific variable
rvn_met_recordplot(stndata=rvn_weathercan_sample, variables = "total_precip")
```

---

rvn_monthly_vbias *Monthly Volume Bias*

Description

Creates a plot of the monthly volume biases in the simulated flow series.

Usage

```
rvn_monthly_vbias(
  sim,
  obs,
  add_line = TRUE,
  normalize = TRUE,
  add_labels = TRUE,
  incomplete_month = FALSE
)
```

Arguments

sim	time series object of simulated flows

obs	time series object of observed flows
add_line	optionally adds a horizontal line to the plot for reference (default TRUE)
normalize	option to normalize the biases and report as percent error (default TRUE)
add_labels	optionally adds labels for early peak/late peaks on right side axis (default TRUE)
incomplete_month	whether to include months with missing days in the summation (default FALSE)

Details

Calculates the monthly volume biases and optionally creates a plot of them. The monthly volume biases are averaged across all years of data. If normalized, the biases are calculated as:

(Vi_sim - Vi_obs)/Vi_obs*100

to be expressed as a percent error.

The sim and obs should be of time series (xts) format and are assumed to be of the same length and time period. The flow series are assumed to be daily flows with units of m3/s. Note that a plot title is purposely omitted in order to allow the automatic generation of plot titles.

The add_labels will add the labels of 'overestimated' and 'underestimated' to the right hand side axis if set to TRUE. This is useful in interpreting the plots. Note that the biases are calculated as sim_Volume - obs_Volume, which means that negative values mean the volume is underestimated, and positive values mean the volume is overestimated.

Value

mvbias	monthly volume biases

See Also

rvn_annual_volume to create a scatterplot of annual flow volumes

Examples

```
# load sample hydrograph data, two years worth of sim/obs
data(rvn_hydrograph_data)
sim <- rvn_hydrograph_data$hyd$Sub36
obs <- rvn_hydrograph_data$hyd$Sub36_obs

# check the monthly volume bias; normalizes by default
rvn_monthly_vbias(sim, obs)

# check unnormalzied monthly volume biases; see the larger volumes in certain periods
rvn_monthly_vbias(sim,obs,normalize = FALSE)
```

rvn_month_names *Months in the Year vector*

Description

Return a character vector of months in the year

Usage

```
rvn_month_names(short = TRUE)
```

Arguments

short	boolean to return shortened form of months

Value

character array of month names

See Also

rvn_num_days for calculating the number of days in a month

Examples

```
rvn_month_names()
rvn_month_names(FALSE)
```

rvn_num_days *Number of Days between two dates*

Description

Calculate the number of days between two provided dates.

Usage

```
rvn_num_days(date1, date2)
```

Arguments

date1	first day, date format
date2	second day, date format

Details

This method handles leap years if they exist between the specified dates.

Value

`int`	number of days between the two days

See Also

`rvn_num_days_month` for calculating the number of days in a month

Examples

```
rvn_num_days(as.Date("2017-02-05"),as.Date("2017-02-12"))
# 7
```

`rvn_num_days_month` *Number of Days in Month*

Description

Calculates the number of days in the month

Usage

```
rvn_num_days_month(date)
```

Arguments

`date`	object in date format

Details

This method includes leap years if they exist in the specified month.

Value

`int`	number of days in the month

See Also

`rvn_num_days` for calculating the number of days between two dates

See original code on post in Stack Overflow the number of days in a month

Examples

```
rvn_num_days_month(as.Date("2016-02-05"))
# 29

rvn_num_days_month(as.Date("2017-01-17"))
# 31
```

`rvn_res_dygraph` *Plot Raven reservoir/lake stage time series using dygraph*

Description

rvn_res_dygraph plots modeled vs observed stage plots when supplied with reservoir stage data structure read using `rvn_res_read`

Usage

```
rvn_res_dygraph(resdata, timezone = "UTC", basins = "", figheight = 400)
```

Arguments

`resdata`	reservoir stage time series data structure generated by `rvn_res_read` routine
`timezone`	data timezone; defaults to UTC
`basins`	list of subbasin names from reservoir file. Each subbasin creates separate dygraph plots
`figheight`	height of figure, in pixels

Value

a list of plot handles to dygraph plots

See Also

`rvn_hyd_dygraph` to generate dygraphs for hydrograph series

Examples

```
# read in Raven Reservoir Stages file
ff <- system.file("extdata","ReservoirStages.csv", package="RavenR")
resdata <- rvn_res_read(ff)

# view contents for all subbasins as dyGraph
dyplots <- rvn_res_dygraph(resdata)
dyplots[[1]]
dyplots[[2]]
```

rvn_res_extract *Extract function for Raven Reservoir object*

Description

rvn_res_extract is used for extracting data from the Raven reservoir object. Works for objects from rvn_res_read function (for reading in the ReservoirStages.csv file).

Usage

```
rvn_res_extract(subs = NA, res = NA, prd = NULL)
```

Arguments

subs	column name for plotting/extracting
res	full reservoir data frame (including units) produced by res.read
prd	time period for plotting, as string. See details

Details

Extracts the modelled and observed data from a Raven reservoir object by name reference. It is also easy to create plots of modelled and observed data using this function. The simulated and observed files are outputted regardless of whether a plot is created, for the specified period.

The subs input is the name of the column desired for use; the most common use of this will be for subbasins, where the names will be of the form "subXX", for example "sub24".

The res object is the full reservoir object (res and units in one data frame) created by the res.read function. Both the res and units are required, since the units are placed onto the plots if one is created. This is useful to at least see the units of the plotted variable, even if the plot is later modified.

The prd input is used to specify a period for the plot and/or the data output. The period should be specified as a string start and end date, of the format "YYYY-MM-DD/YYYY-MM-DD", for example, "2006-10-01/2010-10-01". If no period is supplied, the entire time series will be used.

Value

sim	model simulation for specified column and period
obs	observed data for specified column and period
inflow	inflow simulation for specified column and period

See Also

rvn_res_read for reading in the Reservoirs.csv file and creating the object required in this function.
rvn_res_plot for plotting the extracted stage time series

Examples

```
ff <- system.file("extdata","ReservoirStages.csv", package="RavenR")

# Read in Raven Reservoirs file, store into myres
myres <- rvn_res_read(ff)

# Extract stage using this function
stage36 <- rvn_res_extract(subs="sub36",res=myres,prd="2002-10-01/2003-10-01")
summary(stage36)
summary(stage36$sim)

# Example for precipitation
precip <- rvn_res_extract(subs="precip",res=myres)
```

---

rvn_res_plot *Plot Reservoir Stage*

Description

Creates a reservoir stage plot for the supplied stage series.

Usage

```
rvn_res_plot(
  sim = NULL,
  obs = NULL,
  precip = NULL,
  prd = NULL,
  winter_shading = FALSE,
  wsdates = c(12, 1, 3, 31)
)
```

Arguments

sim	time series object of simulated stage
obs	time series object of observed stage
precip	time series object of precipitation
prd	period to use in plotting
winter_shading	optionally adds shading for winter months (default FALSE)
wsdates	integer vector of winter shading period dates (see details)

Details

Creates a reservoir stage plot using the supplied time series; any series not supplied will not be plotted. If the precip time series is supplied, the secondary y axis will be used to plot the precip time series.

The function assumes that the supplied time series have the same length and duration in time. If this is not true, then the defined period or period calculated from the first available stage series will be used to determine the plotting limits in time. If the data is used directly from Raven output, this is not a concern. The supplied time series should be in xts format, which again can be obtained directly by using the rvn_res_extract function.

The winter_shading argument will add a transparent grey shading for the December 1st to March 31st period in each year that is plotted (or other period specified by wsdates).

wsdates is formatted as c(winter start month, winter start day, winter end month, winter end day).

Value

p1	returns ggplot plot object

See Also

rvn_hyd_read for reading in the Hydrographs.csv file, and rvn_res_extract to extract time series from Raven objects

Examples

```
# read in sample reservoir file
ff <- system.file("extdata","ReservoirStages.csv", package="RavenR")
rvn_res_read(ff) %>%
rvn_res_extract(subs="sub36", res=.) -> mystage
sim <- mystage$sim
obs <- mystage$obs
precip <- rvn_res_read(ff)$res$precip

# create a nice reservoir stage plot
rvn_res_plot(sim,obs)

# create a reservoir stage plot with precip as well
rvn_res_plot(sim,obs,precip=precip)

# create a reservoir stage plot with precip as well for a specific subperiod
prd <- "2003-10-01/2005-10-01"
rvn_res_plot(sim,obs,precip=precip,prd=prd)

# add winter shading
rvn_res_plot(sim,obs,precip=precip, winter_shading=TRUE)
```

`rvn_res_read` *Read in Raven ReservoirStages file*

Description

Reads in the ReservoirStages.csv file produced by Raven.

Usage

```
rvn_res_read(ff = NA, tzone = "UTC")
```

Arguments

`ff`	full file path to the ReservoirStages.csv file
`tzone`	string indicating the timezone of the data in ff

Details

Expects a full file path to the ReservoirStages.csv file, then reads in the file using read.csv. The main advantage of this function is renaming the columns to nicer names and extracting the units into something much easier to read.

This function is also built to support the `rvn_res_extract` function, which uses the object created here for extracting by reference to the columns named here, for example sub24.

ff is the full file path of the ReservoirStages.csv file. If the file is located in the current working directory, then simply the name of the file is sufficient.

The timezone is provided by the tzone argument as "UTC" by default, and should be adjusted by the user to the local time zone as needed, based on the model run.

Value

`res`	data frame from the file with standardized names

See Also

`rvn_res_extract` for extraction tools related to the rvn_res_read output file

Examples

```
# create full file path
ff <- system.file("extdata","ReservoirStages.csv", package="RavenR")

# read in the Reservoir file
myres <- rvn_res_read(ff)

# view contents
head(myres$res)
myres$units
```

rvn_run *Run Raven Executable*

Description

Invokes shell to execute a Raven model.

Usage

```
rvn_run(
  fileprefix = NULL,
  indir = getwd(),
  ravenexe = NULL,
  outdir = NULL,
  rvc = NULL,
  rvt = NULL,
  rvp = NULL,
  rvh = NULL,
  showoutput = FALSE,
  rvi_options = NULL,
  run_chmod = FALSE
)
```

Arguments

fileprefix	file prefix for main Raven input files.
indir	string path for Raven input files
ravenexe	file path to Raven executable
outdir	string path for Raven output files (optional)
rvc	file path to specific rvc file (optional)
rvt	file path to specific rvt file (optional)
rvp	file path to specific rvp file (optional)
rvh	file path to specific rvh file (optional)
showoutput	boolean whether to show output in console (passed to show.output.on.console within system) (default FALSE)
rvi_options	string vector of additional options to add to rvi file temporarily for run
run_chmod	runs a chmod system call to the provided executable ('chmod +x') (default FALSE)

Details

Uses the shell command to run the Raven.exe command.

If the fileprefix is not supplied, the function will detect the rvi file automatically and use that in the run (if exactly one rvi file exists in the directory).

The `indir` path must point to the main Raven input files, if they are not in the working directory.

The `ravenexe` must point to the Raven.exe file; if not supplied it will look for it in the RavenR/extdata path that is saved to when using `rvn_download`.

`rvi_options` can include a vector of any additional commands to add to the rvi file prior to execution. Any commands are added temporarily, and the rvi file is restored following the Raven run. The original rvi file is backed up as a copy in case of an interruption of the function or other error to prevent loss of data. The rvi commands provided are checked against a list of known rvi commands and a warning is issued if an unrecognized command is provided, but all commands provided are nonetheless written to the temporary rvi file as provided. All rvi commands should include the colon prefix to the command (e.g. ":SilentMode" not "SilentMode"), as this is not added automatically.

Note that this function may not work in all servers, as some more specific setups when invoking the `system` command may be required. In addition, errors may occur if the Raven.exe file does not have permission to execute. This can be rectified with the `run_chmod` parameter set to `TRUE`

Value

Returns output code from the system command when running Raven

See Also

`rvn_download`

Examples

```
## Not run:
## NOT RUN (data download + runs Raven.exe)
url<-"https://raven.uwaterloo.ca/files/RavenTutorialFiles.zip"
temploc <- tempdir()
destfile<-paste(temploc,"/RavenTutorialFiles.zip",sep="")
download.file(url,destfile)
destdir<-paste(temploc,"/RavenTutorialFiles",sep="")
dir.create(destdir)
unzip(zipfile=destfile,exdir=destdir)
file.remove(destfile)

## check that Raven.exe is downloaded
if (!rvn_download(check=TRUE)) {rvn_download()}

# Irondequoit example
rvn_run(indir=paste(destdir,"/Irond",sep=""),
        showoutput=TRUE)

# run in Silent Mode (rvi option)
rvn_run(indir=paste(destdir,"/Irond",sep=""),
        showoutput=TRUE,
        rvi_options=c(":SilentMode"))

## End(Not run)
```

`rvn_rvc_from_custom_output` *Generate RVC file from Custom Output CSVs*

Description

Creates an initial conditions rvc file that specifies the initial state using the information from the supplied custom output file.

Usage

```
rvn_rvc_from_custom_output(filename, custfiles, FUN, init_date = NULL, ...)
```

Arguments

`filename`	filepath of rvc file to be created (with .rvc extension)
`custfiles`	array of filepaths to Raven Custom Output files (must be ByHRU)
`FUN`	the aggregation function to be applied to state variables (e.g. mean, passed to sapply)
`init_date`	datetime of model start (optional, can be calculated from Custom Output files)
`...`	optional arguments passed to `rvn_rvc_write` (e.g. author, description)

Value

`TRUE`	if executed successfully

Author(s)

Leland Scantlebury

Examples

```
# Create array of custom output file(s)
custout <- c(system.file("extdata","run1_SNOW_Daily_Average_ByHRU.csv",package = "RavenR"))

# Create rvc file of mean snow for each HRU
rvn_rvc_from_custom_output(filename = file.path(tempdir(), "Blank.rvc"),
                           custfiles = custout,
                           FUN = mean)
```

rvn_rvc_res *Create initial conditions file for Reservoirs*

Description

Write an initial conditions (rvc) format file for Raven, with the calculated reservoir stages written in.

Usage

```
rvn_rvc_res(ff, initial_percent = 0, output = "initial_res_conditions.rvc")
```

Arguments

ff	full file path to the reservoir information file
initial_percent	an optional double for percentage of maximum stage to use as initial condition; default 0.0
output	file rvc lines are written to (default: initial_res_conditions.rvc)

Details

Writes an initial conditions format file for Raven with the relevant initial reservoir stages. This file can be used directly as the model rvc file, or one may copy and paste the information into a separate rvc file for use (i.e. if there is other information to be included in the model rvc file).

The supplied file in ff should be a csv file consistent with the format from the Raven-generated ReservoirStages.csv file. External observations of reservoirs may be used given that the csv file follows the same format.

The initial_percent must be between 0 and 1, so that the initial stage is not less than the dry stage or greater than the maximum.

Value

TRUE	return TRUE if the function is executed properly

See Also

rvn_res_read for reading in the ReservoirStages.csv file

Examples

```
ff <- system.file("extdata","ReservoirStages.csv", package="RavenR")

# set initial conditions at 40% capacity, view file
tf <- file.path(tempdir(), "modelname.rvc")
rvn_rvc_res(ff, initial_percent=0.4, output=tf)
readLines(tf)
```

rvn_rvc_write *Write Raven Initial Condition (rvc) file*

Description

Given initial conditions for state variables at HRUs, generates a rvc file

Usage

```
rvn_rvc_write(
  filename,
  initHRU,
  init_date,
  description = "Generated by RavenR rvn_rvc_write",
  author = NULL
)
```

Arguments

filename	filepath of rvc file to write to (with .rvc extension)
initHRU	dataframe of initial conditions for state variables (columns) for each HRU (rows). columns must be valid SV names and there must be an explicit column of HRU ids named 'HRU'.
init_date	datetime of model start
description	(optional) Description added after file header
author	(optional) Name of author, to be printed in file header.

Value

TRUE	return TRUE if the function is executed properly

Author(s)

Leland Scantlebury

See Also

rvn_rvc_res rvn_rvc_from_custom_output

Examples

```
# Create an initial condition HRU table where SOIL[0] is 0.5mm for all HRUs
# Check.names is set to FALSE to allow for brackets in the column name
initHRU <- data.frame('HRU'=1:10, 'SOIL[0]'=0.5, check.names=FALSE)
model_start = as.Date('2001-10-01')

# Generate RVC file
```

```
rvn_rvc_write(filename = file.path(tempdir(), "New.rvc"),
              initHRU = initHRU,
              init_date = model_start,
              author = 'Harry Potter')
```

---

rvn_rvh_blankHRUdf *Generate Blank Raven HRU DataFrame*

Description

Used to generate a blank HRU table that can be filled in by the user. Compatible with rvn_rvh_write.

Usage

```
rvn_rvh_blankHRUdf(nHRUs = 1, subbasinIDs = NULL)
```

Arguments

nHRUs	Number of HRUs, used to determine number of rows in table (default = 1)
subbasinIDs	Subbasins that HRUs belong to (default = all equal 1)

Details

Note that if the length of the subbasinIDs vector is greater than the number of HRUs (nHRUs) specified, this will create a table with HRUs belonging to multiple subbasins, which is not feasible. A warning will be issued that the table will need to be modified for hydrologic consistency.

Value

data.frame of blank HRU properties to be filled in by user

Author(s)

Leland Scantlebury

See Also

rvn_rvh_blankSBdf to generate blank subbasin data frame

Examples

```
HRUtable <- rvn_rvh_blankHRUdf(nHRUs = 3, subbasinIDs=c(1,1,2))
HRUtable

# fewer nHRUs than subbasinIDs specified
rvn_rvh_blankHRUdf(nHRUs = 1, subbasinIDs=c(1,2))
```

`rvn_rvh_blankSBdf` *Generate Blank Raven SubBasin DataFrame*

Description

Generates a blank data frame for Raven subbasin properties. Compatible with `rvn_rvh_write`.

Usage

```
rvn_rvh_blankSBdf(nSubBasins = 1)
```

Arguments

`nSubBasins`	Number of SubBasins in model, used to determine number of rows in table (default = 1)

Details

The subbasin names are provided as 'sub00x', where x is the basin ID. The padding is determined from the number of subbasins. The downstream IDs are generated to assume a linear downstream progression, with an outlet at the terminal subbasin ID, which can be modified after the data frame is created.

Value

data.frame of blank SubBasin properties to be filled in by user

Author(s)

Leland Scantlebury

See Also

`rvn_rvh_blankHRUdf` to generate blank HRU data frame

Examples

```
SBtable <- rvn_rvh_blankSBdf(nSubBasins = 3)
SBtable
```

rvn_rvh_cleanhrus *Clean HRU data table.*

Description

Takes rvn_rvh_read-generated HRUtable and SBTable and returns cleaned HRUtable with (hopefully) fewer HRUs

Usage

```
rvn_rvh_cleanhrus(
  HRUtab,
  SBtab,
  area_tol = 0.01,
  merge = FALSE,
  elev_tol = 50,
  slope_tol = 4,
  aspect_tol = 20,
  ProtectedHRUs = c(),
  LockedHRUs = c(),
  LockedSubbasins = c()
)
```

Arguments

HRUtab	table of HRUs generated (typically) by rvn_rvh_read
SBtab	table of Subbasins generated (typically) by rvn_rvh_read
area_tol	percentage of watershed area beneath which HRUs should be removed (e.g., default value of 0.01 would indicate anything smaller than 1 percent of watershed extent should be removed)
merge	TRUE if similar HRUs are to be merged (this can be slow for large models)
elev_tol	elevation difference (in metres) considered similar. only used if merge=TRUE
slope_tol	slope difference (in degrees) considered similar. only used if merge=TRUE
aspect_tol	slope difference (in degrees) considered similar. only used if merge=TRUE
ProtectedHRUs	vector of HRU IDs that are sacrosanct (not to be removed, but may still increase in area)
LockedHRUs	vector of HRU IDs that are locked (not to be modified)
LockedSubbasins	vector of subbasin IDs that are locked (not to be modified).

Details

rvn_rvh_cleanhrus removes HRUs in two ways:

1. it removes all HRUs smaller than the area_tol percentage of total area. Adjacent HRUs in the subbasin are expanded by the lost area to keep the same relative coverage.

2. it consolidates similar HRUs within the same subbasin (those with same land cover, vegetation, soil profile and similar slope, aspect, and elevation)

The ProtectedHRUs allows the specification of HRUs that should not be removed, even if they would otherwise be merged or removed. These HRUs may still increase in size as other HRUs are consolidated.

The LockedHRUs allows for the specification of HRUs that will not change (removed or increase in size), which may be useful for specific land types such as glaciers or water bodies. It is possible that locking HRUs may prevent the script from resizing remaining HRUs within a subbasin, in which case a warning is issued to the user that the area has changed. If this is the case, it is suggested to reduce the area threshold to try and prevent this issue, or consider simply locking some subbasins.

Note that merging can be a computationally expensive process, and for this reason is set as FALSE by default.

Value

hru_table	cleaned HRU table as a dataframe

Author(s)

James R. Craig, University of Waterloo

See Also

rvn_rvh_read for the function used to read in the HRU and Subbasin data, and rvn_rvh_write to write rvh information to file.

Examples

```
# read in example rvh file
nith <- system.file("extdata","Nith.rvh",package = "RavenR")
rvh <- rvn_rvh_read(nith)

# number of HRUs in existing configuration
nrow(rvh$HRUtable)

# clean contents (in this case, remove all HRUs covering less than 5% of the total area)
newHRUs <- rvn_rvh_cleanhrus(rvh$HRUtable,rvh$SBtable,area_tol = 0.05, merge=TRUE)

# clean contents but locking urban areas (two HRUs locked)
newHRUs <- rvn_rvh_cleanhrus(rvh$HRUtable,rvh$SBtable,area_tol = 0.05, merge=TRUE,
   LockedHRUs=rvh$HRUtable[rvh$HRUtable$LandUse=="URBAN", "ID"])
```

`rvn_rvh_overwrite` *Write/Overwrite Raven rvh file*

Description

Given an HRU and SubBasin dataframe, writes to the specified .rvh file. In the case of rvn_rvh_overwrite, just the :SubBasins-:EndSubBasins and :HRUs-:EndHRUs blocks are re-written, retaining all other content.

Usage

```
rvn_rvh_overwrite(filename, SBtable, HRUtable, basefile)

rvn_rvh_write(
  filename,
  SBtable = NULL,
  HRUtable = NULL,
  description = "Generated by RavenR rvn_rvh_write",
  author = NULL
)
```

Arguments

`filename`	filepath of rvh file to write to (with .rvh extension)
`SBtable`	Valid subbasin dataframe with required columns "SBID", "Name", "Downstream_ID", "Profile", "ReachLength", and "Gauged". Can have additional columns.
`HRUtable`	Valid HRU dataframe with required columns 'ID', 'Area', 'Elevation', 'Latitude', 'Longitude', 'SBID', 'LandUse', 'Vegetation', 'SoilProfile', 'Aquifer', 'Terrain', 'Slope', and 'Aspect'. Can have additional columns.
`basefile`	original rvh file to overwrite (only used with rvn_rvh_overwrite)
`description`	(optional) Description added after file header
`author`	(optional) Name of author, to be printed in file header.

Details

`rvn_rvh_write` is typically used to create a 'clean' .rvh file.

`rvn_rvh_overwrite` is usually used after reading an original .rvh file and processing the HRU and SubBasin tables, using (e.g., `rvn_rvh_cleanhrus`). This may also be used to preserve commands in the rvh file such as reservoir information, comments outside of the subbasin and HRU blocks, RedirectToFile commands, etc.

Note that if using the rvn_rvh_overwrite function and the filename and basefile arguments are the same, the original file will be overwritten while preserving lines outside of the subbasin and HRU blocks.

If using `rvn_rvh_write`, the SBtable or HRUtable parameters may be omitted and provided as `NULL`. In these cases, those sections will not be written in the rvh file. This may be useful in cases where one wishes to separate the SubBasins and HRUs in different files.

Value

TRUE returns TRUE if function runs properly

Functions

- rvn_rvh_overwrite(): Overwrite contents of original .rvh file

Author(s)

James R. Craig, University of Waterloo

Leland Scantlebury

See Also

rvn_rvh_read for the function used to read in the HRU and SubBasin data rvn_rvh_cleanhrus for the function used to process HRU dataframe
For generating blank SubBasin and HRU tables, see: rvn_rvh_blankSBdf and rvn_rvh_blankHRUdf

Examples

```
## Example: write a blank rvh file
## create some blank tables
sbs_data <- rvn_rvh_blankSBdf(nSubBasins = 1)
hru_data <- rvn_rvh_blankHRUdf(nHRUs = 3)

# write to rvh file
rvn_rvh_write(file.path(tempdir(), "Blank.rvh"),
              SBtable = sbs_data,
              HRUtable = hru_data,
              description = "Example output - Blank Watershed Discretization File",
              author = "Raven Development Team")

# Example: Read in an rvh, clean its contents and write back to new file
nith <- system.file("extdata","Nith.rvh",package = "RavenR")
rvh <- rvn_rvh_read(nith)

# remove HRUs covering less than 5% of the total area
rvh$HRUtable <- rvn_rvh_cleanhrus(rvh$HRUtable, rvh$SBtable, area_tol = 0.05)


# write the Subbasin and HRU tables to new file using rvn_rvh_write:
rvn_rvh_write(filename=file.path(tempdir(), "Nith_cleaned_write.rvh"),
             SBtable = rvh$SBtable,
             HRUtable = rvh$HRUtable)

# write to new file, while preserving all unedited information using rvn_rvh_overwrite:
rvn_rvh_overwrite(filename=file.path(tempdir(), "Nith_cleaned_overwrite.rvh"),
                  basefile=nith,
                  SBtable = rvh$SBtable,
                  HRUtable = rvh$HRUtable)
```

rvn_rvh_query *Queries RVH object for subbasins and HRUs of interest*

Description

Queries the RVH object for subbasins or HRUs that are upstream of, downstream of, or the opposite of those conditions, for a given subbasin ID.

Usage

```
rvn_rvh_query(rvh = NULL, subbasinID = NULL, condition = "upstream_of")
```

Arguments

rvh	rvh object as returned by rvn_rvh_read
subbasinID	subbasinID of basin of interest, as character or integer
condition	condition applied to the query

Details

Based on the definition of subbasins by their outlets in Raven, it is assumed here that 'upstream' includes the specified subbasin (i.e. everything upstream of subbasin X includes subbasin X as well), and 'downstream' of subbasin X does not include subbasin X. This is different from the default behaviour of igraph, which includes the specified subbasin in either query.

Value

rvh object in same format, but queried to condition and all features (SBtable, HRUtable, SBnetwork) updated.

Note

Raven has capabilities for creating subbasin and HRU groups that meet certain criteria as well, consider reviewing the ':PopulateSubbasinGroup', ':PopulateHRUGroup', and other commands in Section A.3.2 of the Raven User's Manual.

See Also

rvn_rvh_write to write contents of the generated (and usually modified HRU and SubBasin tables)
rvn_rvh_read to read a Raven RVH file into R

Examples

```
# load example rvh file
nith <- system.file("extdata","Nith.rvh",package = "RavenR")
rvh <- rvn_rvh_read(nith)

# plot full watershed with igraph library
plot(rvh$SBnetwork)

# query all subbasins upstream of basin 39, plot
rvh_upstream_of_39 <- rvn_rvh_query(rvh, subbasinID=39, condition="upstream_of")
plot(rvh_upstream_of_39$SBnetwork)

# query of HRUs downstream of basin 39
rvn_rvh_query(rvh, subbasinID=39, condition="downstream_of")$SBtable
```

`rvn_rvh_read` *Read Raven .rvh (watershed discretization) file*

Description

This routine reads in a valid Raven watershed discretization (.rvh) file and returns the information about HRUs and Subbasins as data tables. It also returns a subbasin igraph network object which describes stream network connectivity and adds additional HRU-derived subbasin characteristics such as total upstream area and dominant land/vegetation classes.

Usage

```
rvn_rvh_read(ff)
```

Arguments

`ff`	the filepath of the .rvh file (with .rvh extension included).

Details

The supplied file should not be comma-delimited with a trailing comma. The function also does not like tabs in the rvh file, the file should be untabified first. This function uses the igraph library to build the networks and compute the total upstream area. The .rvh file can have arbitrary contents outside of the :HRUs-:EndHRUs and :SubBasins-:EndSubBasins command blocks.

Partial rvh files may be provided to this function (i.e. with only :SubBasin or :HRUs blocks but not the other), however, some calculations and the calculation of the `SBnetwork` output will not be completed. Omitted structures (e.g. `SBtable`) will be returned as `NULL` if the section is not found in the rvh file directly. Note that this function does not look for additional files specified with `:RedirectToFile` commands.

The ff argument can be a relative path name or absolute one.

The TotalUpstreamArea is the total drainage area upstream of the given subbasin outlet. With this calculation, headerwater subbasins will have a total upstream area equal to their own subbasin area.

Value

Returns a list including:

SBtable	a data table of Subbasin characteristics indexed by Subbasin ID (SBID). Includes the following data columns from the .rvh file : SBID, Name, Downstream_ID, Profile, ReachLength, Gauged. The rvn_rvh_read() functions supplements this with additional columns: Area, Elevation, AvgLatit, AvgLongit, AvgSlope, AvgAspect, DomLU, DomLUArea, DomLUFrac, DomVeg, DomVegArea, DomVegFrac. Elevation, AvgLatit, AvgLongit, AvgSlope, and AvgAspect are the area-weighted averages from all constituent HRUs. DomLU is the dominant land use name, DomLUArea is the area (in km2) of the dominant land use and DomLUArea is the percentage of the basin covered with DomLU; same applies to DomVeg.
HRUtable	a data table of HRU characteristics, with land use and vegetation classes as factors. Contains identical information as found in the :HRUs-:EndHRUs block of the .rvh file: ID, Area, Elevation, Latitude, Longitude, SBID, LandUse,Vegetation, SoilProfile, Aquifer, Terrain, Slope, and Aspect.
SBnetwork	an igraph network graph network describing subbasin stream network connectivity, with nodes indexed by SBID.

Author(s)

James R. Craig, University of Waterloo

See Also

rvn_rvh_write to write contents of the generated (and usually modified HRU and SubBasin tables)
rvn_rvh_subbasin_network_plot to plot the subbasin network

Examples

```
# load example rvh file
nith <- system.file("extdata","Nith.rvh",package = "RavenR")
rvh <- rvn_rvh_read(nith)

# number of HRUs
nrow(rvh$HRUtable)

# total watershed area
sum(rvh$HRUtable$Area)

# sub-table of headwater basins (upstream area = subbasin area)
rvh$SBtable$SBID[rvh$SBtable$Area == rvh$SBtable$TotalUpstreamArea]

# sub-table of Urban HRUs
subset(rvh$HRUtable, LandUse == "URBAN")

# get total area upstream of subbasin containing outlet
upstr <- cumsum(rvh$SBtable$Area)
upstr[rvh$SBtable$Downstream_ID == -1]
```

```
# show upstream areas for each subbasin
rvh$SBtable[,c("SBID","TotalUpstreamArea")]

# plot network diagram using igraph library
igraph::plot.igraph(rvh$SBnetwork)
```

`rvn_rvh_subbasin_network_plot` *Basic Raven subbasin network plot*

Description

Generates a plot of the subbasin network from rvh file information.

Usage

```
rvn_rvh_subbasin_network_plot(SBtable, labeled = FALSE)
```

Arguments

`SBtable`	a valid table of Raven subbasins, obtained from `rvn_rvh_read`
`labeled`	TRUE if the nodes are labeled with the SubBasin ID, SBID

Details

Takes the information gathered from an .rvh file via the function `rvn_rvh_read` and generates a plot object of the subbasin network, where nodes are located at SubBasin lat-long centroids, and edge widths of the network correspond to contributing upstream area.

The plot is generated using the ggplot2 library, with dependencies on igraph for handling network information.

Value

`p1`	ggplot object of subbasin network plot

Author(s)

James R. Craig, University of Waterloo

Examples

```
# read in rvh file
rvh <- rvn_rvh_read(system.file("extdata","Nith.rvh", package="RavenR"))

# create network plot of watershed structure from rvh file
rvn_rvh_subbasin_network_plot(rvh$SBtable)

# include labels
rvn_rvh_subbasin_network_plot(rvh$SBtable, labeled=TRUE)
```

`rvn_rvh_subbasin_visnetwork_plot` *Plot subbasin network using visNetwork*

Description

Takes an `rvh` object generated using `rvn_rvh_read` and returns the connections information of subbasins as an interactive visNetwork graph.

Usage

```
rvn_rvh_subbasin_visnetwork_plot(rvh, groupBy = "Gauged")
```

Arguments

`rvh`	an `rvh` object, provided by `rvn_rvh_read`
`groupBy`	a character referring to one of the sub-basins attributes in the `rvh`

Value

returns visNetwork plot object

See Also

`rvn_rvh_read` to import an watershed network table from an `rvh` file.

See also the Raven page

Examples

```
## additional example from tutorial files (not run)
## Not run:
  path <- dirname(tempfile())
  dir.create(paste(path,"/tmp",sep=""))
  url<-"https://raven.uwaterloo.ca/files/RavenOstrichTutorialFiles.zip"
  download.file(url,dest=paste(path,"/tmp/example.zip",sep=""))
  unzip(zipfile = paste(path,"/tmp/example.zip",sep=""),
        exdir = paste(path,"/tmp",sep=""))
```

```
  rvh<-rvn_rvh_read(paste(path,"/tmp/Demo_C4/model/LOWRL.rvh",sep=""))
  rvn_rvh_subbasin_visnetwork_plot(rvh,groupBy="Gauged")
  rvn_rvh_subbasin_visnetwork_plot(rvh,groupBy="DomLU")
  rvn_rvh_subbasin_visnetwork_plot(rvh,groupBy="Elevation")

## End(Not run)

rvh <- rvn_rvh_read(system.file("extdata","Nith.rvh", package="RavenR"))
rvn_rvh_subbasin_visnetwork_plot(rvh,groupBy="Gauged")
rvn_rvh_subbasin_visnetwork_plot(rvh,groupBy="Elevation")
```

`rvn_rvh_summarize` *Summarize RVH object*

Description

Summarizes the RVH object provided in a number of useful ways, and returns a list with the summarized information.

Usage

```
rvn_rvh_summarize(rvh = NULL, return_list = TRUE)
```

Arguments

`rvh`	rvh object as returned by `rvn_rvh_read` or `rvn_rvh_query`
`return_list`	boolean whether the to return the summary list object, if `FALSE` only `TRUE` is returned (default `TRUE`)

Details

The total subbasin area is the total area of all subbasins in the RVH object. If there are multiple outlets in the model, or additional subbasin information that is not part of the model domain, this will be counted in the total area and other summary diagnostics. Consider using `rvn_rvh_query` to isolate portions of the domain in such instances.

Information for dominant Terrain and Aquifer classes is also returned, but only if there are Terrain or Aquifer classes respectively other than '[NONE]'. The number of unique HRUs include Terrain and Aquifer classes in the consideration of unique HRUs if there are instances of classes other than '[NONE]' for these classes.

Value

Returns `TRUE` if the parameter `return_list` is `FALSE`, otherwise returns a list with multiple items:

`total_subbasin_area`	total subbasin area, computed as the sum of all areas in `rvh$SBtable$Area`
`num_subbasins`	total number of subbasins

`num_hrus` total number of HRUs

`num_unique_hrus`
: total number of unique HRUs, i.e. HRUs with unique combination of LandUse, Vegetation, Soil Profile, and Terrain/Aquifer if relevant (see details)

`hru_summary_landuse`
: a dataframe summarizing the RVH object by LandUse class

`hru_summary_vegetation`
: a dataframe summarizing the RVH object by Vegetation class

`hru_summary_soilprofile`
: a dataframe summarizing the RVH object by Soil Profile

`hru_summary_terrain`
: a dataframe summarizing the RVH object by Terrain class

`hru_summary_aquifer`
: a dataframe summarizing the RVH object by Aquifer class

`hru_summary_general`
: a dataframe summarizing the RVH object by LandUse, Vegetation, Soil Profile, and Terrain/Aquifer if relevant (see details)

`dom_landuse` dominant LandUse class by total area

`dom_landuse_ratio`
: ratio of the total area of dominant LandUse class by total subbasin area

`dom_vegetation` dominant Vegetation class by total area

`dom_vegetation_ratio`
: ratio of the total area of dominant Vegetation class by total subbasin area

`dom_soilprofile`
: dominant Soil Profile by total area

`dom_soilprofile_ratio`
: ratio of the total area of dominant Soil Profile by total subbasin area

`dom_terrain` dominant Terrain class by total area (NULL if no Terrain classes defined)

`dom_terrain_ratio`
: ratio of the total area of dominant Terrain class by total subbasin area (NA if no Terrain classes defined)

`dom_aquifer` dominant Aquifer class by total area (NULL if no Aquifer classes defined)

`dom_aquifer_ratio`
: ratio of the total area of dominant Aquifer class by total subbasin area (NA if no Aquifer classes defined)

See Also

`rvn_rvh_read` to read a Raven RVH file into R `rvn_rvh_query` to query the RVH file prior to other operations, such as summarizing)

Examples

```
# load example rvh file
nith <- system.file("extdata","Nith.rvh",package = "RavenR")
rvh <- rvn_rvh_read(nith)

# summarize rvh
rvn_rvh_summarize(rvh, return_list=FALSE)

# query of HRUs upstream of basin 39, then summarize
rvh %>%
  rvn_rvh_query(subbasinID=39, condition="upstream_of") %>%
  rvn_rvh_summarize(return_list=FALSE)
```

`rvn_rvh_write_subbasingroup` *Write a subbasin group to Raven RVH format*

Description

Writes the subbasins in a given RVH object to Raven RVH format in the specified file.

Usage

```
rvn_rvh_write_subbasingroup(
  rvh = NULL,
  sbgroup_name = NULL,
  outfile = NULL,
  subs_per_line = 30,
  overwrite = TRUE
)
```

Arguments

`rvh`	rvh object as returned by `rvn_rvh_read` or `rvn_rvh_query`
`sbgroup_name`	the name of the subbasin group to write to file
`outfile`	the output file to write the subbasin group to
`subs_per_line`	the number of subabsins to write per line in the file (default 30)
`overwrite`	if `TRUE`, will overwrite the existing `outfile` specified if it already exists (default `FALSE`)

Details

Writes the subbasin group for all subbasins in the rvh$SBtable data frame to file using the ':SubbasinGroup' command in Raven. This function is intended to be used with `rvn_rvh_query` to first subset the RVH object in order to create useful groups, though this can be done manually as well.

Value

Returns TRUE if the file is written successfully

Note

Raven has capabilities for creating subbasin and HRU groups that meet certain criteria as well, consider reviewing the ':PopulateSubbasinGroup', ':PopulateHRUGroup', and other commands in Section A.3.2 of the Raven User's Manual.

See Also

rvn_rvh_read to read a Raven RVH file into R rvn_rvh_query to query the RVH file prior to other operations, such as summarizing or writing out subbasin groups)

Examples

```
# load example rvh file
nith <- system.file("extdata","Nith.rvh",package = "RavenR")
rvh <- rvn_rvh_read(nith)

# query all subbasins upstream of basin 39
rvh_upstream_of_39 <- rvn_rvh_query(rvh, subbasinID=39, condition="upstream_of")

# temporary filename
tf <- file.path(tempdir(), 'mysubbasigroup.rvh')

# write this subbasin group to file
rvn_rvh_write_subbasingroup(rvh=rvh_upstream_of_39, outfile=tf)
```

---

rvn_rvi_commandupdate *Update command in Raven input file*

Description

Updates the provided rvi file with the command and value provided.

Usage

```
rvn_rvi_commandupdate(
  filename = NULL,
  command = NULL,
  value = NULL,
  outputfile = NA
)
```

Arguments

filename	the name of the .rvi file (with .rvi extension included), either relative to the working directory or absolute.
command	the rvi command with preceeding colon to update
value	value of the command to update
outputfile	if writing to a new file, otherwise filename is overwritten

Value

returns TRUE if run successfully

Author(s)

James R. Craig, University of Waterloo

See Also

rvn_rvi_read to read and process rvi files with RavenR

Examples

```
# load example rvi file
ff <- system.file("extdata","Nith.rvi", package="RavenR")
tf <- tempfile(fileext=".rvi")

rvn_rvi_commandupdate(filename=ff,
command=":StartDate",
value="2022-10-01 00:00:00",
outputfile=tf)
```

---

rvn_rvi_connections *Generate Hydrological process connections list*

Description

This routine reads in a hydrologic process list from rvn_rvi_read and generates the list of hydrologic process connections.

Usage

```
rvn_rvi_connections(
  rvi,
 ProcConDataFile = system.file("extdata", "RavenProcessConnections.dat", package =
    "RavenR")
)
```

Arguments

rvi	data object generated from the rvn_rvi_read routine
ProcConDataFile	(optional) path to RavenProcesConnections.dat file

Details

Relies on a valid and up-to-date RavenProcessConnections.dat file. This file is provided with the RavenR package, but may be overridden by a more recent file if provided manually.

Value

Returns a list with two items:

connections	a a dataframe of all of the process connections Includes the following data columns: process type, algorithm, 'from' compartment, 'to' compartment, and conditional
AliasTable	a table of aliases (unchanged from the supplied rvi$AliasTable)

Author(s)

James R. Craig, University of Waterloo

See Also

rvn_rvi_read to read a .rvi file and generate an rvi object, and rvn_rvi_process_ggplot or rvn_rvi_process_diagrammer to plot the process network produced in this function.

See also the Raven page

Examples

```
rvi <- rvn_rvi_read(system.file("extdata","Nith.rvi", package="RavenR"))

rvi_conn <- rvn_rvi_connections(rvi)
head(rvi_conn$connections)
head(rvi_conn$AliasTable)
```

---

rvn_rvi_getparams *Retrieve Parameter Information from RVI file Algorithms*

Description

Reads the processed rvi object and returns a data frame of parameter information for parameters associated with processes in the rvi object.

Usage

```
rvn_rvi_getparams(
  rvi,
 RavenAlgParamsFile = system.file("extdata", "RavenAlgParams.dat", package = "RavenR"),
 RavenParamsFile = system.file("extdata", "RavenParameters.dat", package = "RavenR")
)
```

Arguments

rvi	data object generated from the rvn_rvi_read routine
RavenAlgParamsFile	(optional) path to RavenAlgParams.dat file
RavenParamsFile	(optional) path to RavenParameters.dat file

Details

Uses the Raven database files in the /extdata folder to (1) associate parameters with particular algorithms, and (2) subset the database parameter list based on the information in the rvi file.

These files are stored with the RavenR package and retrieved with this function by default, but a separate link may be provided to a modified file if desired. Note that the database files held in the RavenR package are unofficial copies of those in the official Raven SVN, and any discrepancies should defer to the Raven SVN versions.

Value

Returns a data frame with parameter information containing the parameter name, parameter class, units, auto (whether the parameter may be autogenerated within Raven), default, min, and max values. Note that the flags 'xxx', -9999, and 9999 are used as missing values within the parameter data frame.

Examples

```
# sample workflow of rvn_rvi_read
rvi <- system.file("extdata","Nith.rvi", package="RavenR") %>%
rvn_rvi_read(.)

# get data frame of parameters related to processes in rvi file
rvn_rvi_getparams(rvi)
```

rvn_rvi_process_diagrammer *Plot Raven hydrologic process network using DiagrammeR*

Description

This routine takes a connections data from generated using rvn_rvi_connections() and returns the connections information as a DiagrammeR object.

Usage

```
rvn_rvi_process_diagrammer(
  rvi_conn,
  sv_omit = c("SNOW_DEPTH", "COLD_CONTENT", "PONDED_WATER/SNOW_LIQ", "NEW_SNOW",
    "SNOW_DEFICIT"),
  repel_force = 0.001,
  repel_iter = 2000,
  lbl_size = 0.5,
  lbl_height = 0.3,
  lbl_width = 1,
  pdfout = NULL
)
```

Arguments

rvi_conn	a list of connections and AliasTable, provided by rvn_rvi_connections
sv_omit	character vector of state variables to omit from the plot
repel_force	numeric value indicating the 'force' with which the repel function will move labels
repel_iter	the maximum number of iterations for the repel algorithm
lbl_size	estimated height of labels, used in repel algorithm
lbl_height	actual height of the labels (in inches)
lbl_width	relative width of the labels (multiplier)
pdfout	name of pdf file to save the network plot to, if null no PDF is generated

Details

Uses the output from the rvn_rvi_connections function to generate the plot with the DiagrammeR library.

Note that the output can be plotted using the render_graph function in the DiagrammeR library. The outputted DiagrammeR object may also have aesthetics modified with various commands from the same library, if desired, as shown in the examples. The rsvg and DiagrammeRsvg packages may be required to export to PDF with desired results, but are not explicit dependencies of RavenR.

sv_omit is used to reduce the clutter in the process plot of state variables that one may wish to omit from the plot.

The function uses the functionality from ggrepel to repel labels from one another. The degree of separation in the labels can be controlled by the repel_force and lbl_size parameters (increasing either will increase the separation between labels). The repel_force may range from approximately 1 to 1e-6. The lbl_size is a relative estimate of the label height (default 0.5), which is used in estimating the label size in the repel functionality. Providing a larger number will increase the perceived size of the label in the repel functionality and tend towards more separation between labels, and vice-versa. Both of these parameters may need to change depending on the plot size and number of labels. The lbl_height and lbl_width parameters can be changed to affect the height and relative width of the actual labels.

The basic model structure outline is followed, but unrecognized state variables are plotted on the left hand side of the plot (determined with internal RavenR function rvn_rvi_process_layout).

Value

d1 returns DiagrammeR object. Also generates a .pdf file in working directory if pdfplot argument is not NULL.

See Also

rvn_rvi_connections to generate connections table from an rvi object

rvn_rvi_process_ggplot to generate the structure plot using ggplot.

See also the Raven page. Additional details on the DiagrammeR package may be found on the Github page.

Examples

```
d1 <- rvn_rvi_read(system.file("extdata","Nith.rvi", package="RavenR")) %>%
  rvn_rvi_connections() %>%
    rvn_rvi_process_diagrammer()

# plot diagram using the DiagrammeR package
library(DiagrammeR)

d1 %>%
  render_graph()

# modify default plot attributes, plot
d1 %>%
  select_nodes() %>%
  set_node_attrs_ws(node_attr = fillcolor, value = "hotpink") %>%
  select_edges() %>%
  set_edge_attrs_ws(edge_attr = style, value = "dashed") %>%
  set_edge_attrs_ws(edge_attr = penwidth, value = 2) %>%
  render_graph()
```

rvn_rvi_process_ggplot

Plot Raven hydrologic process network

Description

This routine takes a connections data from generated using `rvn_rvi_connections` and returns the connections information as a network graph ggplot object.

Usage

```
rvn_rvi_process_ggplot(
  rvi_conn,
  sv_omit = c("SNOW_DEPTH", "COLD_CONTENT", "PONDED_WATER/SNOW_LIQ", "NEW_SNOW",
    "SNOW_DEFICIT"),
  repel_force = 0.001,
  repel_iter = 2000,
  lbl_size = 0.5,
  lbl_fill = "lightblue",
  arrow_size = 0.25,
  arrow_adj = 0.25,
  pdfout = NULL
)
```

Arguments

`rvi_conn`	a list of connections and AliasTable, provided by `rvn_rvi_connections`
`sv_omit`	character vector of state variables to omit from the plot
`repel_force`	numeric value indicating the 'force' with which the repel function will move labels
`repel_iter`	the maximum number of iterations for the repel algorithm
`lbl_size`	estimated height of labels, used in repel algorithm
`lbl_fill`	fill colour for labels (default 'lightblue')
`arrow_size`	size of plotted arrows (default 0.25)
`arrow_adj`	adjustment in line length reduction for arrows (default 0.25)
`pdfout`	name of pdf file to save the network plot to, if null no PDF is generated

Details

Uses the output from the `rvn_rvi_connections` function to generate the plot with the `ggplot2` library..

`sv_omit` is used to reduce the clutter in the process plot of state variables that one may wish to omit from the plot.

The function uses the functionality from ggrepel to repel labels from one another. The degree of separation in the labels can be controlled by the repel_force and lbl_size parameters (increasing either will increase the separation between labels). The repel_force may range from approximately 1 to 1e-6. The lbl_size is a relative estimate of the label height (default 0.5), which is used in estimating the label height in the repel functionality. Providing a larger number will increase the perceived size of the label in the repel functionality and tend towards more separation between labels, and vice-versa. Both of these parameters may need to change depending on the plot size and number of labels.

arrow_adj is the amount that each line segment is reduced in length to accommodate the arrow. Increasing this value will decrease the length of the line segment, and place the arrow further from the box. This value should generally be similar to the arrow_size parameter.

The basic model structure outline is followed, but unrecognized state variables are plotted on the left hand side of the plot (determined with internal RavenR function rvn_rvi_process_layout).

Value

returns ggplot object. Also generates a .pdf file in working directory if pdfplot argument is not NULL.

See Also

rvn_rvi_connections to generate connections table from an rvi object

rvn_rvi_process_diagrammer to generate the structure plot using DiagrammeR.

See also the Raven page

Examples

```
library(ggplot2)

p1 <- rvn_rvi_read(system.file("extdata","Nith.rvi", package="RavenR")) %>%
  rvn_rvi_connections() %>%
    rvn_rvi_process_ggplot()
p1 ## plot to screen

## change the colour of the background
p1 + theme(panel.background = element_rect(fill = 'lightgrey', colour = 'purple'))

## adjust line/arrow colours (no conditional lines shown in Nith example)
p1 + scale_colour_manual(values=c('FALSE'='purple', 'TRUE'='red'))

## adjust line/arrow types (no conditional lines shown in Nith example)
p1 + scale_linetype_manual(values=c('FALSE'='longdash', 'TRUE'='twodash'))
```

rvn_rvi_read *Read Raven .rvi (watershed discretization) file*

Description

This routine reads in a valid Raven main input (.rvi) file and returns the information about hydrological processes as a data table.

Usage

```
rvn_rvi_read(filename)
```

Arguments

filename	the name of the .rvi file (with .rvi extension included), either relative to the working directory or absolute.

Details

This function does not like tabs in the .rvi file - it should be untabified first. Comma-delimited tables with a trailing comma are also problematic. The .rvi file can have arbitrary contents outside of the :HydrologicProcesses- :EndHydrologicProcesses block and :SubBasins-:EndSubBasins command blocks.

Value

Returns a list with two items:

HydProcTable	a data table of hydrologic processes. Includes the following data columns: process type, algorithm, 'from' compartment, 'to' compartment, conditional (logical), and condition (character)
AliasTable	a table of aliases read from the rvi file, NULL if no aliases are included

Author(s)

James R. Craig, University of Waterloo

Examples

```
# sample workflow of rvn_rvi_read
rvi <- system.file("extdata","Nith.rvi", package="RavenR") %>%
rvn_rvi_read(.)

# get number of Hydrologic processes
nrow(rvi$HydProcTable)
```

`rvn_rvi_write_template` *Write Raven rvi file based on model configuration templates*

Description

Writes a Raven rvi file based on one of several template model configurations.

Usage

```
rvn_rvi_write_template(
  template_name = "UBCWM",
  filename = NULL,
  overwrite = TRUE,
  writeheader = TRUE,
  filetype = "rvi ASCII Raven",
  author = "RavenR",
  description = NULL
)
```

Arguments

`template_name`	name of the model template to be written (default 'UBCWM')
`filename`	Name of the rvi file, with extension (optional)
`overwrite`	boolean whether to overwrite file if it already exists (default `FALSE`)
`writeheader`	boolean whether to write a header to the rvi file (default `TRUE`)
`filetype`	File extension, Encoding, Raven version (e.g. "rvp ASCII Raven v3.8") (optional)
`author`	Name of file author (optional)
`description`	File Description for header (e.g., Basin or project information, R script name) (optional)

Details

Raven has the capability of emulating a number of existing model configurations, and a number of additional novel model configurations are provided which may be helpful to the user. These can be written with this function for ease of getting started with a model using Raven.

The template_name parameter should be one of "UBCWM", "HBV-EC", "HBV-Light", "GR4J", "CdnShield", "MOHYSE", "HMETS", "HYPR", "HYMOD", "SAC-SMA", "blended", or "blended_v2".

This function uses the same model template files that are provided in the Raven User's manual, Appendix D.

The `rvn_write_Raven_newfile` is used to write a header in the rvi file. Writing of a header can be disabled with `writeheader=FALSE`.

Value

TRUE	returns TRUE if executed successfully

Examples

```
# write the Canadian Shield configuration to 'mymodel.rvi'
rvn_rvi_write_template(template_name="CdnShield",
   filename=file.path(tempdir(), "mymodel.rvi"))

# write the HMETS model with some additional details in the description
rvn_rvi_write_template(template_name="HMETS",
   filename=file.path(tempdir(), "mynewmodel.rvi"),
   author="Robert Chlumsky",
   description="RVI file for the HMETS model (Martel, 2017) created by RavenR")
```

---

rvn_rvp_calib_template

Rewrite rvp file with placeholder values

Description

Rewrites a Raven rvp file with placeholder parameter values.

Usage

```
rvn_rvp_calib_template(
  rvp_file = NULL,
  rvp_outfile = NULL,
  ost_outfile = "ostIn.txt",
  params_calibration = NULL,
  overwrite = FALSE,
 RavenParamsFile = system.file("extdata", "RavenParameters.dat", package = "RavenR")
)
```

Arguments

rvp_file	path to the model *.rvp file
rvp_outfile	file path to rewritten rvp file
ost_outfile	file path to Ostrich input file
params_calibration	vector of parameters to include in the calibration
overwrite	whether to overwrite an existing template file (default FALSE)
RavenParamsFile	path to RavenParameters.dat file (default path points to file included with RavenR installation)

Details

Here, the rvp file is replaced with generic placeholder values to create a template file, which is commonly required for model calibration. Although parameters may be found in other Raven input files, this command focuses on the rvp file. Other parameters (such as gauge corrections in RVT or subbasin-level corrections in the RVH file) must be done manually.

The Raven rvp file may be generated from the `rvn_rvp_fill_template` function.

The list of parameters to be calibrated may be provided by the user (via `params_calibration` argument), or determined by RavenR. The intent of this function is to provide a functional example of an RVP template file that may be used in calibration, not a high quality calibration with clever selection of parameters to use. It is highly recommended to build from the RVP template file created using expert hydrologic modelling knowledge.

If `rvp_outfile` is not provided, Raven will attempt to write to the file prefix of the provided template file with a .rvp.tpl extension. If there is a conflict with an existing file and `overwrite==FALSE`, the function will automatically overwrite a file with the suffix "_ravenr_generated.rvp.tpl".

Similarly with `ost_outfile`, the parameter default/min/max values and other Ostrich inputs will be written based on a default template file to the ostIn.txt (or other provided file name). If this is set to NULL, the file will not be written.

The default parameter values come from the RavenParameters.dat file included with RavenR in the extdata folder. The user may provide their own file with updated values if preferred. Note that the database files held in the RavenR package are unofficial copies of those in the official Raven SVN, and any discrepancies should defer to the Raven SVN versions.

Any parameters not found in this file will be ignored and a warning provided.

If you find parameters not found by this function, please open an ticket on Github (https://github.com/rchlumsk/RavenR/issues).

Value

`TRUE` if the function executed successfully

See Also

`rvn_rvi_getparams` to get parameter ranges from rvi.

Examples

```
# write rvp from template file
rvp_tempfile <- tempfile(fileext=".rvp")
rvn_rvp_fill_template(rvi_file=system.file("extdata","Nith.rvi", package="RavenR"),
                      rvh_file=system.file("extdata","Nith.rvh", package="RavenR"),
                      rvp_template_file =system.file("extdata","nithmodel.rvp_temp.rvp",
                      package="RavenR"),
                      rvp_out=rvp_tempfile)

# setup calibration rvp template and ostin file
ost_tempfile <- tempfile(fileext=".txt")
rvptpl_tempfile <- tempfile(fileext=".rvp.tpl")
```

```
rvn_rvp_calib_template(rvp_file=rvp_tempfile,
                       rvp_outfile=rvptpl_tempfile,
                       ost_outfile=ost_tempfile)
```

`rvn_rvp_fill_template` *Rewrite template rvp file with values*

Description

Rewrites a Raven template rvp file with default parameter values.

Usage

```
rvn_rvp_fill_template(
  rvi_file = NULL,
  rvh_file = NULL,
  rvp_template_file = NULL,
  fileprefix = NULL,
  rvp_out = NULL,
  overwrite = FALSE,
  default_param_value = 0.12345,
  default_soil_thickness = 0.5,
  use_default_flag = FALSE,
  avg_annual_runoff = NULL,
  extra_commands = NULL,
 RavenParamsFile = system.file("extdata", "RavenParameters.dat", package = "RavenR")
)
```

Arguments

`rvi_file`	path to the model *.rvi file
`rvh_file`	path to the model *.rvh file
`rvp_template_file`	path to the model rvp template file (*.rvp_temp.rvp)
`fileprefix`	model name prefix for the main model files; if provided, the function will attempt to find all missing file paths in the current working directory
`rvp_out`	file path to rewritten rvp file
`overwrite`	logical for whether to rewrite the *.rvp file if it already exists (default FALSE)
`default_param_value`	default parameter value to write for any parameters not found in RavenParameters.dat
`default_soil_thickness`	default soil layer thickness (m) to provide in :SoilProfile block
`use_default_flag`	writes all soil/land use/vegetation classes with [DEFAULT] flag

`avg_annual_runoff`
: adds a line for :AvgAnnualRunoff if value provided with this parameter and not already in file

`extra_commands`
: additional commands to add to end of rvp file as character vector

`RavenParamsFile`
: path to RavenParameters.dat file (default path points to file included with RavenR installation)

Details

The Raven rvp template file is generated by Raven when the :CreateRVPTemplate command is included in the rvi file (the default when `rvn_rvi_write_template` is used to produce an rvi file). This template file displays the layout of the rvp file with required parameters based on the hydrologic processes in the rvi file, but is not immediately usable. This function uses the soil model information in the rvi file and the HRU class information in the rvh file to rewrite the template file with default parameter values so that it can be used in a model run.

If `rvp_out` is not provided, Raven will attempt to write to the file prefix of the provided template file with a .rvp extension. If there is a conflict with an existing file and `overwrite==FALSE`, the function will automatically overwrite a file with the suffix "_ravenr_generated.rvp".

The default parameter values come from the RavenParameters.dat file included with RavenR in the extdata folder. The user may provide their own file with updated values if preferred. Note that the database files held in the RavenR package are unofficial copies of those in the official Raven SVN, and any discrepancies should defer to the Raven SVN versions.

Any parameters not found in this file will be written with the value provided by `default_param_value`. The default soil thickness for the :SoilProfiles block is provided by the `default_soil_thickness` function parameter, which is applied for all soil classes.

As an alternative to specifying the three input files (rvi, rvh, rvp_temp.rvp), the `fileprefix` of the model (e.g. 'Nith' for Nith.rvi) may be provided instead. If provided, the function will attempt to find all required input files based on the provided fileprefix in the current working directory.

Additional commands can be added to the end of the rvp file with `extra_commands`, which are not quality controlled but simply appended to the rvp file. This can be useful for non-standard commands such as :RedirectToFile for channel properties rvp files that are not added automatically to base templates rvp files.

If you find parameters not found by this function, please open an ticket on Github (https://github.com/rchlumsk/RavenR/issues).

Value

`TRUE` if the function executed successfully

See Also

`rvn_rvi_getparams` to get parameter ranges from rvi.

Examples

```
### this section is not run, but illustrates how an rvp template file would be created
# -----
## Not run:
## create an rvi file and template file with Raven
rvn_rvi_write_template(modelname="HBV-EC",
   filename="nithmodel.rvi")

## download Raven.exe if not already downloaded
if (!rvn_download(check=TRUE)) {
  rvn_download()
}

## run Raven to create template file
rvn_run(fileprefix="nithmodel")

## End(Not run)
# -----

# load pre-generated template file and other model files
nithmodel_template_file <- system.file("extdata","nithmodel.rvp_temp.rvp", package="RavenR")
nith_rvi_file <- system.file("extdata","Nith.rvi", package="RavenR")
nith_rvh_file <- system.file("extdata","Nith.rvh", package="RavenR")
rvp_out_file <- file.path(tempdir(), 'nithmodel.rvp')

# rewrite template with parameter values
rvn_rvp_fill_template(rvi_file=nith_rvi_file,
                      rvh_file=nith_rvh_file,
                      rvp_template_file=nithmodel_template_file,
                      rvp_out=rvp_out_file)
```

---

rvn_rvt_mappings_data *Rvt Mappings Data*

Description

A list of lists that has information for use in the RavenR rvt-related files.

Additional information on Raven variables can be found in the Raven User's Manual, available from https://raven.uwaterloo.ca/Downloads.html

Format

`rvn_rvt_mappings_data` is a list with four lists.

rvt_mapping list: data formats for all rvt file types

rvt_data_type_mapping list: data types permitted in Raven, and associated units

rvn_met_raven_mapping list: meteorological forcing functions permitted in Raven, and associated units

rvt_met_mapping_weathercan list: mapping of weathercan variable names to Raven variables

See Also

rvn_rvt_read for reading in rvt files

rvn_rvt_write and rvn_rvt_write_met for writing data to rvt files.

---

rvn_rvt_read *Read .rvt (Raven time series) file*

Description

This routine reads in a valid Raven time series input (.rvt) file and returns the information as an xts time series and metadata data frame.

Usage

```
rvn_rvt_read(filename, tzone = "UTC")
```

Arguments

filename	path/name of the .rvt file (with .rvt extension included)
tzone	string indicating the timezone of the data provided in filename (default "UTC")

Details

All rvt data types available in Raven are supported (e.g. :MultiData, :Data, :ObservedData, :BasinInflowHydrograph). This is handled by the mappings provided in the `data("rvn_rvt_mappings_data")` function, and additional support for other rvt types can likely be included by making updates to that function with modifications to the somewhat generic read function here.

It does NOT support the master .rvt file with :Gauge or :GriddedForcing commands, only the reading of time-series based .rvt files with a single time series block within the file.

The timezone is provided by the tzone argument as "UTC" by default, and should be adjusted by the user to the local time zone as needed, based on the model run.

Value

Returns a list with two objects:

rvt_xts	xts formatted time series with data
rvt_metadata	data frame with a 'param' and 'value' column providing rvt metadata (number of points, time interval, subbasin ID, etc.)

Examples

```
# read in rvt file
system.file('extdata','GlenAllan.rvt',package="RavenR") %>%

rvn_rvt_read(.) -> rvt
plot(rvt$rvt_xts$TEMP_DAILY_MIN)
```

`rvn_rvt_tidyhydat` *EC Streamgauge File Conversion from tidyhydat*

Description

rvn_rvt_tidyhydat converts Environment Canada historical streamgauge data, accessed via the tidyhydat package, into .rvt format files usable in Raven.

Usage

```
rvn_rvt_tidyhydat(
  indata,
  rvt_type = "ObservationData",
  data_type = "HYDROGRAPH",
  subIDs,
  prd = NULL,
  stnNames = NULL,
  write_redirect = FALSE,
  flip_number = FALSE,
  rd_file = "flow_stn_redirect_text.rvt",
  filename = NULL
)
```

Arguments

`indata`	tibble of WSC flow data from tidyhydat's hy_daily_flows() function
`rvt_type`	type of rvt file to write (e.g. ObservationData, BasinInflowHydrograph, etc.)
`data_type`	Raven-syntax data type for flow data (default 'HYDROGRAPH')
`subIDs`	vector of subbasin IDs to correspond to the stations in indata
`prd`	(optional) data period to use in .rvt file
`stnNames`	(optional) character vector of alternative station names to use
`write_redirect`	(optional) write the :RedirectToFile commands in a separate .rvt file
`flip_number`	(optional) put the subID first in the .rvt filename
`rd_file`	(optional) name of the redirect file created (if `write_redirect=TRUE`)
`filename`	specified name of file(s) to write to (optional)

Details

Takes a single flow tibble generated from tidyhydat and converts the flow data for each station in the file into .rvt formatted files for a Raven model. If multiple stations exist in indata, multiple observation files are created. This function is a wrapper for `rvn_rvt_write`, with the benefit of automatically parsing the tidyhydat download for possibly multiple stations in xts formats before passing to the rvt writing function.

rvt_type is the specified rvt file type to write to (see the Raven User's Manual or the `rvn_rvt_mappings` function for more rvt types). This should be a flow-based rvt type, such as ObservationData, BasinInflowHydrograph, ReservoirExtraction, etc. Most applications of this will likely write the tidyhydat observations as ObservedData for use in model evaluation to historic records.

data_type is the type of Raven input data type, likely 'HYDROGRAPH', for the corresponding flow data. If the flow is used as a reservoir-related flow, the data_type may be RESERVOIR_INFLOW or RESERVOIR_NETINFLOW.

subIDs is required and should correspond to the subID to be used in the .rvt file for each station in the ff file, in the order in which it will be read in.

prd is used by the xts formatted-data to restrict the data reported in .rvt files, for each station, to this period. The prd should be defined in "YYYY-MM-DD/YYYY-MM-DD" string format. If the period supplied results in an empty time series (i.e. non-overlapping time periods), an error will be thrown.

stnNames is an optional character vector to replace the EC station codes found in the HYDAT database. If supplied, the vector must be of the same length as the number of stations supplied and the subIDs vector. If not supplied, the EC station codes will be used. Note that this does not impact model function, only filename readability and station recognition.

write_redirect will print out the :RedirectToFile commands in a separate file called, "flow_stn_redirect_text.rvt". These commands can be copied into the main model's .rvt file to redirect to the produced time series files.

flip_number is a useful option to place the subID first in the filename. This is often cleaner for organizing files in a folder, since the alphabetized order is not dependent on the station name, and the observed files will be in one set.

The function will write to name generated from the station name(s), otherwise the .rvt filename may be specified with the filename argument (full path to the filename, including .rvt extension). If multiple stations are provided, the filename argument may be a vector of filenames.

Note that the function uses `sort(unique(indata$STATION_NUMBER))` to determine the order of stations, thus the filenames and stnNames should correspond to the sorted vector of station numbers as well.

Note that only daily flow data is supported, as tidyhydat only allows for the download of daily (or coarser resolution) flow data. Hourly data that is obtained must be first processed into xts format and then written with `rvn_rvt_write`.

If the data is found to have an inconsistent timestep, the function will attempt to correct it by infilling missing time steps with `rvn_ts_infill`. If successful, a warning is issued to the user and the function will proceed, else an error will be raised.

Value

`TRUE`	return TRUE if the function is executed properly

Examples

```
# note: example modified to avoid using tidyhydat directly, uses saved
## tidyhydat data from RavenR package sample data
# library(tidyhydat)
stations <- c("05CB004","05CA002")

# Gather station data/info using tidyhydat functions
# hd <- tidyhydat::hy_daily_flows(station_number = stations,
#  start_date = "1996-01-01", end_date = "1997-01-01")
data(rvn_tidyhydat_sample)
hd <- rvn_tidyhydat_sample

# station_info <- hy_stations(stations)

tf1 <- file.path(tempdir(), "station1.rvt")
tf2 <- file.path(tempdir(), "station2.rvt")

# Create RVT files
rvn_rvt_tidyhydat(hd, subIDs=c(3,11),
  filename=c(tf1,tf2))
```

---

rvn_rvt_write *Write Raven rvt file from Time Series*

Description

Generates a Raven rvt file of the specified type from an xts time series.

Usage

```
rvn_rvt_write(
  x,
  filename = NULL,
  rvt_type = "ObservationData",
  data_type = "HYDROGRAPH",
  basin_ID = NULL,
  NA_value = -1.2345
)
```

Arguments

x	time series in xts format to write to file
filename	name of output file (with rvt extension)
rvt_type	type of rvt file to write (e.g. ObservationData)
data_type	type of data in x (e.g. HYDROGRAPH)
basin_ID	subbasin (or HRU) ID corresponding to the time series
NA_value	value to use for NA values in rvt file (default -1.2345 for Raven format)

Details

Writes the rvt file for a given time series dataset. The type of rvt file to write is determined by the rvt_type argument, which must match one of the supported Raven types. Note that this function does not support the writing of meteorological data, this is handled by the `rvn_rvt_write_met` function.

The format of the rvt file, including required fields to write to file, are determined from the supplied rvt_type parameter and from the mapping provided by `data("rvn_rvt_mappings_data")`. The data_type is also checked against the provided mappings to check for valid state variables and accompanying units.

If the data is found to have an inconsistent timestep, the function will attempt to correct it by infilling missing time steps with `rvn_ts_infill`. If successful, a warning is issued to the user and the function will proceed, else an error will be raised.

The timezone of the xts object is used as supplied to write to file, no assumption on changing the time zone is done in this function.

No other quality control of the data is performed here. Some rvt types, such as ObservationWeights, cannot have missing values in the data; it is the responsibility of the user to supply `x` with no missing values if required. Any missing values in `x` are written to file with the missing value code provided by `NA_value`.

`x` should be an xts time series object with multiple rows of data and a single column.

Value

`TRUE` if the function executed successfully

See Also

`rvn_rvt_read` to read in rvt data files, and `rvn_rvt_write_met` to write meteorological rvt files.

Examples

```
# load sample flow data
system.file('extdata','run1_Hydrographs.csv', package = "RavenR") %>%
rvn_hyd_read() -> mydata

# temporary filename
tf <- file.path(tempdir(), 'mydata.rvt')

# write time series to rvt file using data from subbasin 36 as observed data
rvn_rvt_write(x=mydata$hyd$Sub36,
  rvt_type = "ObservationData",
  data_type = "HYDROGRAPH",
  basin_ID = 36,
  filename = tf)
```

rvn_rvt_write_met *EC Climate Station File Conversion*

Description

Converts meteorological data for a given station into the Raven .rvt format.

Usage

```
rvn_rvt_write_met(
  metdata,
  rvt_met_mapping = NULL,
  filenames = NULL,
  met_file_prefix = "met_",
  prd = NULL,
  write_stndata = TRUE,
  filename_stndata = "met_stndata.rvt",
  NA_value = -1.2345
)
```

Arguments

metdata	EC meteorological data from one or more stations (e.g., from weathercan::weather_dl())
rvt_met_mapping	list that provides the mapping between metdata names and those used in Raven
filenames	(optional) character vector of filenames for the rvt data files, length same as number of stations in metdata
met_file_prefix	(optional) prefixes the file name (default: "met_")
prd	(optional) data period to use in .rvt file
write_stndata	(optional) write the gauge data to a separate .rvt file
filename_stndata	(optional) name of the station data file created (if write_stndata=TRUE)
NA_value	(optional) value to use for NA values in rvt file (default -1.2345 for Raven format)

Details

Writes data in either :Data or :MultiData format depending on the number of supported parameter columns provided. The data should be downloaded and prepared with missing days included, and preferably is downloaded directly using weathercan.

metdata contains all of the meteorological data to be written to file. metdata should be in a tibble or data frame format, and is required to have a date/time column (either DATE, TIME, or DATETIME, all of which have at least a Date component), the STATION_NAME, and all desired forcings to be written to file. If the columns ELEV, LAT, LON are included, the station meta data may be written

to a separate rvt file as well (see below). All supported data columns in metdata are written into rvt format, so the desired columns for the rvt file should be passed through metdata and filtered first if needed.

rvt_met_mapping is a list that maps the metdata column names to Raven variables. If `weathercan` is used then this may be left NULL, as the mapping is automatically provided. Otherwise, the user must convert all desired column names to Raven-recognized names, or provide the mapping information as a list through this parameter. An example format can be seen (the mapping used for weathercan by default) in `data("rvn_rvt_mappings_data")`.

filenames may be used to provide the specific desired filenames (with paths) for each station rvt data file being generated; this should be a character vector of length equal to the number of unique station names in the data.

Note that the function uses `sort(unique(metdata$STATION_NAME))` to determine the order of stations, thus the filenames should correspond to the sorted vector of station numbers as well.

prd is used by the xts formatted-data to restrict the data reported in .rvt files, for each station, to this period. The prd should be defined in "YYYY-MM-DD/YYYY-MM-DD" string format. If the period supplied results in an empty time series (i.e. non-overlapping time periods), an error will be thrown.

met_file_prefix can be used to add a prefix to the .rvt data file names, ("met_" by default) which may be useful in organizing multiple climate data files. This is ignored if filenames are specified.

write_stndata will print out the gauge(s) metadata to file (specified by filename_stndata parameter) in the .rvt format, which is required to include a meterological station in Raven. The function will append to the file if it already exists, meaning that this works for iterations of this function. metdata must include the columns ELEV, LAT, and LON if station data is to be written, else the meta data file will not be created.

The function has several built-in data quality checks. These include:

* checking that the time interval is consistent for each station; * ensuring that meta data is unique for each station name; and * check for missing data and issuing a warning that post-processing will be required

Data quality is not assessed in this package, such as consistency between minimum and maximum temperatures and missing data. Consider viewing the RavenR.extras package for functions to interpolate missing meteorological data and checking for min/max temperature consistency.

This function is designated to use data from the weathercan package, but may be used with any supplied data frame of meteorological information if in the correct format and other relevant information (such as rvt_met_mapping, if needed) is supplied. The weathercan package is external to RavenR and is not an explicit dependency of RavenR, although a sample weathercan data set can be viewed as `data(rvn_weathercan_sample)`.

Value

`TRUE` return `TRUE` if the function is executed properly

See Also

`rvn_rvt_write` to write non-forcing time series data to Raven rvt format.

Download Environment Canada Historical weather data from (climate.weather.gc.ca), or use the 'weathercan' package to access this data through R.

Examples

```
# note: example modified to avoid using weathercan directly, uses saved
## weathercan data from RavenR package sample data
data(rvn_weathercan_sample)
kam <- rvn_weathercan_sample

# basic use, override filename to temporary file
rvn_rvt_write_met(metdata = kam,
  filenames = file.path(tempdir(), "rvn_rvt_metfile.rvt"),
  filename_stndata = file.path(tempdir(), "met_stndata.rvt"))
```

`rvn_stringpad` *Pads string with spaces, either right or left justified*

Description

Pad string with spaces, justified on either the left or right

Usage

```
rvn_stringpad(string, width, just = "r", padstring = " ")
```

Arguments

`string`	Text string
`width`	Number of characters total, including desired spaces
`just`	'r' for right, 'l' for left
`padstring`	string to use for padding (default space character)

Value

Padded string

Author(s)

Leland Scantlebury, <leland@scantle.com>

Examples

```
# Returns "   To the right"
rvn_stringpad('To the right', 15, just='r')

# Returns "Padded    "
rvn_stringpad('Padded', 10, just='l')
```

rvn_substrLeft *substring from the Left*

Description

Returns n characters from the left side of the supplied string x.

Usage

```
rvn_substrLeft(x, n)
```

Arguments

x	a string to manipulate
n	number of characters to remove from the left side of the string

See Also

rvn_substrRight for using n characters from right side of string

rvn_substrMRight for removing n characters from the right side of a string

Examples

```
rvn_substrLeft("hello world",3)
# returns "hel"
```

rvn_substrMLeft *substring minus characters from the Left*

Description

Returns a string x with n characters removed from the left side of the string.

Usage

```
rvn_substrMLeft(x, n)
```

Arguments

x	a string to manipulate
n	number of characters to remove from the left side of the string

See Also

rvn_substrRight for using n characters from the right side of string,

rvn_substrLeft for using n characters from the left side of string

rvn_substrMRight for removing n characters from the right side of a string

Examples

```
rvn_substrMLeft("hello world",3)
# returns "lo world"
```

---

rvn_substrMRight *substring minus characters from the Right*

Description

Returns a string x with n characters removed from the right side of the string.

Usage

```
rvn_substrMRight(x, n)
```

Arguments

x	a string to manipulate
n	number of characters to remove from the right side of the string

See Also

rvn_substrRight for using n characters from the right side of string

rvn_substrLeft for using n characters from the left side of string

rvn_substrMLeft for removing n characters from the left side of a string

Examples

```
rvn_substrMRight("hello world",3)
# returns "hello wo"
```

rvn_substrRight *substring from the Right*

Description

Returns n characters from the right side of the supplied string x.

Usage

```
rvn_substrRight(x, n)
```

Arguments

x	a string to manipulate
n	number of characters to use from the right side of the string

See Also

rvn_substrLeft for using n characters from the left side of string

rvn_substrMRight for removing n characters from the right side of a string

rvn_substrMLeft for removing n characters from the left side of a string

Examples

```
rvn_substrRight("hello world",3)
# returns "rld"
```

rvn_theme_RavenR *RavenR ggplot theme*

Description

Makes the general Raven R Theme for all ggplots

Usage

```
rvn_theme_RavenR()
```

Details

Sets up the default theme for all ggplots generated using a built in Raven R function. Made by adjusting the built in theme_bw.

Value

returns a theme for use in ggplot2 figures

See Also

rvn_annual_volume to create a scatterplot of annual flow volumes.

Examples

```
# generate a basic ggplot and apply the RavenR theme
library(ggplot2)
ggplot(data=cars, aes(x=speed, y=dist))+
geom_point()+
rvn_theme_RavenR()
```

---

rvn_tidyhydat_sample *tidyhydat sample data for RavenR package*

Description

A dataset downloaded using the tidyhydat package for two stations, between 1996-01-01 and 1997-01-01. Additional details on the tidyhydat package and data formats can be found at the tidyhydat github page.

Note that this sample is provided to avoid loading the tidyhydat package and requiring the download_hydat() function in CRAN testing of examples.

Additional information on data provided by the Water Survey of Canada may be found on the WSC webpage.

Usage

```
rvn_tidyhydat_sample
```

Format

rvn_tidyhydat_sample is a tibble with 671 rows, and 5 columns.

STATION_NUMBER station number from WSC stations in HYDAT database

Date date in YYYY-MM-DD format

Parameter Code indicating the parameter provided in the given row as either flow or level data

Value the value of the parameter provided (flow or level)

Symbol additional data flags provided by WSC

Source

Water Survey of Canada (wateroffice.ec.gc.ca) via 'tidyhydat' package

See Also

rvn_rvt_tidyhydat for writing rvt files from tidyhydat data

---

rvn_ts_infill *Infill discontinuous time series with blank values*

Description

Infills missing time values from a time series based on a regular interval.

Usage

```
rvn_ts_infill(ts)
```

Arguments

ts	valid xts time series

Details

Takes xts dataset, finds minimum interval between time stamps and returns a new regular interval xts with same data content, but NA values in between known data values

Only handles data with minimum time interval of 1 day; 1,2,3,4,6,8, or 12 hrs.

Note that in default reading in of date/time data, the daylight savings timezones may be assigned to the date/time when reading in a data file with Raven using functions such as rvn_hyd_read. This function will then detect differences in the intervals and throw an error. To avoid this, the timezone may be assigned explicitly to all values with the read function and all daylight savings/endings will be ignored.

Value

ts	continuous xts time series

Author(s)

James R. Craig, University of Waterloo

Examples

```
system.file("extdata","run1_Hydrographs.csv", package="RavenR") %>%
rvn_hyd_read(., tzone="EST") -> mydata
mydata <- mydata$hyd$precip
mydata<-mydata[-seq(2,nrow(mydata),3),] # remove every 3rd day
head(mydata)

# fill back with rvn_ts_infill using NA values
rvn_ts_infill(mydata$precip) %>%
head()
```

rvn_watershedmeb_read *Read in Raven WatershedMassEnergyBalance file*

Description

Used to read in the WatershedMassEnergyBalance.csv file produced by the modelling Framework Raven.

Usage

```
rvn_watershedmeb_read(ff = NA, tzone = "UTC")
```

Arguments

ff	full file path to the WatershedMassEnergyBalance.csv file
tzone	string indicating the timezone of the data in ff

Details

Expects a full file path to the WatershedMassEnergyBalance.csv file, then reads in the file using read.csv. The main advantage of this function is renaming the columns to nicer names and extracting the units into something much easier to read. The from and to rows are also properly handled, which is not as straightforward as some of the other Raven files.

ff is the full file path of the WatershedMassEnergyBalance.csv file. If the file is located in the current working directory, then simply the name of the file is sufficient.

The timezone is provided by the tzone argument as "UTC" by default, and should be adjusted by the user to the local time zone as needed, based on the model run.

Value

watershedmeb	data frame from the file with standardized names
units	vector corresponding to units of each column
from	vector of the 'from' compartments in file
to	vector of the 'to' compartments in file

See Also

rvn_watershed_read for reading in the WatershedStorage.csv file

Examples

```
# locate RavenR Watershed Mass Energy Balance storage file
ff <- system.file("extdata","run1_WatershedMassEnergyBalance.csv", package="RavenR")

# read in file
mywshdmeb <- rvn_watershedmeb_read(ff)
```

```
# view mass energy balance time series
head(mywshdmeb$watershedmeb)

# view 'from' dataframe
mywshdmeb$from
```

`rvn_watershed_data` *Watershed Storage Data from Raven*

Description

A dataset formatted to the xts package, read in by the watershed.read function. The dataset contains the typical columns from the Raven outputted WatershedStorage.csv file, available for download in the Raven Tutorials (linked below).

Usage

```
rvn_watershed_data
```

Format

rvn_watershed_data is a data frame with two object

watershed.storage various storage variable states outputted from the Raven model

units units associated with each variable in watershed.storage

rvn_watershed_data$watershed.storage is an xts (time series) object with 731 rows and 19 variables, containing data from 2002-10-01 to 2004-09-30. The details of each watershed storage state can be found in the Raven Manual

- rainfall
- snowfall
- Channel.Storage
- Reservoir.Storage
- Rivulet.Storage
- Surface.Water
- Cum..Losses.to.Atmosphere..mm.
- Ponded.Water
- Soil.Water.0
- Soil.Water.1
- Soil.Water.2
- Snow.Melt..Liquid..mm.
- Snow

- Canopy
- Canopy.Snow
- Total
- Cum..Inputs..mm.
- Cum..Outflow..mm.
- MB.Error

The Nith River model can be downloaded from the Raven Tutorials (tutorial #2) https://raven.uwaterloo.ca/Downloads.html

See Also

rvn_watershed_read for reading in watershed storage output files

Examples

```
# View data
head(rvn_watershed_data$watershed.storage)
# Also has units
rvn_watershed_data$units
```

---

rvn_watershed_read *Read in Raven WatershedStorage file*

Description

Read in the WatershedStorage.csv file produced by Raven.

Usage

```
rvn_watershed_read(ff = NA, tzone = "UTC")
```

Arguments

ff	full file path to the WatershedStorage.csv file
tzone	string indicating the timezone of the data in ff

Details

Expects a full file path to the WatershedStorage.csv file, then reads in the file using read.csv. The main advantage of this functon is renaming the columns to nicer names and extracting the units into something much easier to read.

ff is the full file path of the WatershedStorage.csv file. If the file is located in the current working directory, then simply the name of the file is sufficient.

The timezone is provided by the tzone argument as "UTC" by default, and should be adjusted by the user to the local time zone as needed, based on the model run.

Value

`watershed_storage` data frame from the file with standardized names

`units` vector corresponding to units of each column

See Also

`rvn_hyd_read` for reading in the Hydrographs.csv file `rvn_watershedmeb_read` for reading in the WatershedMassEnergyBalance.csv file

Examples

```
# locate in RavenR Watershed storage file
ff <- system.file("extdata","run1_WatershedStorage.csv", package="RavenR")

# create full file path and read in file
mywshd <- rvn_watershed_read(ff)

# check data
head(mywshd$watershed_storage)
```

---

rvn_weathercan_metadata_sample

weathercan sample metadata for RavenR package

Description

A dataset downloaded using the **weathercan** package function `stations_search` for stations within 150 km of Merritt, BC. Additional details on the weathercan package and data formats can be found at the weathercan github page.

Note that this sample is provided to avoid loading the **weathercan** package while compiling and testing the package.

Additional information on data provided by Environment Canada can be found on the Historical Data portal.

Usage

```
rvn_weathercan_metadata_sample
```

Format

rvn_weathercan_sample is a tibble with 3 rows, and 17 columns.

Source

Historical Climate Data from Environment Canada (climate.weather.gc.ca) via 'weathercan' package

See Also

rvn_met_recordplot for checking the record of meteorological data. This function example also has the original code used to create the rvn_weathercan_metadata_sample data set.

---

rvn_weathercan_sample *weathercan sample data for RavenR package*

Description

A dataset downloaded using the weathercan package for the 'KAMLOOPS A' station (station id 51423), between 2016-10-01 and 2019-09-30. Additional details on the weathercan package and data formats can be found at the weathercan github page.

Note that this sample is provided to avoid loading the weathercan package while compiling and testing the package.

Additional information on data provided by Environment Canada can be found on the Historical Data portal.

Usage

```
rvn_weathercan_sample
```

Format

rvn_weathercan_sample is a tibble with 1095 rows, and 37 columns.

Source

Historical Climate Data from Environment Canada (climate.weather.gc.ca) via 'weathercan' package

See Also

rvn_rvt_write_met for writing rvt files from meteorological data (including weathercan data).

rvn_which_max_xts *which.max for xts objects*

Description

Applies the which.max function and returns an xts object with the maximum value and associated date.

Usage

```
rvn_which_max_xts(x)
```

Arguments

x	xts object to apply which.max to

Details

Acts as the which.max function, applicable to xts objects and returning values in an xts format.

Note that when deploying the rvn_apply_wyearly function, the dates are overwritten and the dates of the water year ending periods are displayed rather than the event dates. In order to obtain the corresponding dates when using the rvn_apply_wyearly function, please use rvn_apply_wyearly_which_max_xts.

Value

xts object with max value and corresponding date

See Also

which.max base which.max function rvn_apply_wyearly_which_max_xts for using apply_wyearly with the rvn_which_max_xts function

Examples

```
data(rvn_hydrograph_data)

# obtain the peak observed flow and the corresponding date
rvn_which_max_xts(rvn_hydrograph_data$hyd$Sub43_obs)

# note that the usual rvn_apply_wyearly does not provide the correct dates with this function
rvn_apply_wyearly(rvn_hydrograph_data$hyd$Sub43_obs, rvn_which_max_xts)
```

rvn_write_Raven_header *Write common Raven file header*

Description

Writes the common Raven file header to file. All lines are Appended.

Usage

```
rvn_write_Raven_header(
  filename,
  filetype,
  author = NULL,
  creationDate = TRUE,
  textlen = 40
)
```

Arguments

filename	Name of the file, with extension
filetype	File extension, Encoding, Raven version (e.g. "rvp ASCII Raven 2.9.1")
author	Name of file author (optional)
creationDate	Bool of whether creation date should be added to header. (default TRUE)
textlen	Length of lines (default: 40, used to right-align text)

Value

TRUE	returns TRUE if executed successfully

Author(s)

Leland Scantlebury, <leland@scantle.com>

Examples

```
tf <- file.path(tempdir(), 'HogwartsBasin.rvp')
rvn_write_Raven_header(filename = tf,
                   filetype = 'rvp ASCII Raven 2.9.1',
                   author   = 'Harry Potter')

# view file
readLines(tf)
```

rvn_write_Raven_label *Writes common Raven labeled line to file, with optional value (appends)*

Description

Writes common Raven labeled line to file, with optional value (appends)

Usage

```
rvn_write_Raven_label(
  label,
  filename,
  value = NULL,
  digits = NULL,
  indent_level = 0
)
```

Arguments

label	character, (e.g. "SoilClasses")
filename	character, file name/path to write to, with extension
value	numeric or character, corresponding value written after label (optional)
digits	Number of digits to round value to (optional)
indent_level	Adds two spaces before label for every one level (default = 0)

Value

TRUE	returns TRUE if executed successfully

Author(s)

Leland Scantlebury, <leland@scantle.com>

Examples

```
tf <- file.path(tempdir(), "Hogwarts.rvi")

# Numeric example
rvn_write_Raven_label('Duration', filename=tf, value=365)

# Hydrologic Processes
rvn_write_Raven_label('HydrologicProcesses', tf)

# String example, with indent
rvn_write_Raven_label('SnowBalance', filename = tf,
```

```
                      value = paste('SNOBAL_HMETS', 'MULTIPLE', 'MULTIPLE'),
                      indent_level = 1)

# Preview file
readLines(tf)
```

`rvn_write_Raven_newfile` *Opens/Creates a new file, writes common file header.*

Description

Opens/Creates a new file, writes common file header.

Usage

```
rvn_write_Raven_newfile(
  filename,
  description,
  filetype,
  author = NULL,
  creationDate = TRUE,
  linelen = 74,
  textlen = 40
)
```

Arguments

`filename`	Name of the file, with extension
`description`	File Description (e.g., Basin or project information, R script name)
`filetype`	File extension, Encoding, Raven version (e.g. "rvp ASCII Raven 2.9.1")
`author`	Name of file author (optional)
`creationDate`	Bool of whether creation date should be added to header. (default `TRUE`)
`linelen`	length (width) of header, in text characters (default: 74)
`textlen`	Length of textlines (default: 40, used to right-align text)

Value

`TRUE`	returns `TRUE` if executed successfully

Author(s)

Leland Scantlebury, <leland@scantle.com>

Examples

```
tf <- file.path(tempdir(), "HogwartsBasin.rvp")
rvn_write_Raven_newfile(filename = tf,
                description = "Hogwarts River Basin RVP File Generated by HP_FileGen.R",
                filetype = "rvp ASCII Raven 2.9.1",
                author = 'Harry Potter')

# view file
readLines(tf)
```

---

rvn_write_Raven_table *Writes a nicely formatted tables of Raven attributes/parameters*

Description

Writes a nicely formatted tables of Raven attributes/parameters

Usage

```
rvn_write_Raven_table(
  attributes,
  units,
  df,
  filename,
  id_col = TRUE,
  justify = "right",
  sep = ", ",
  ...
)
```

Arguments

attributes	array of strings containing attribute/parameter names
units	array of strings with the corresponding units
df	Dataframe of values corresponding to attributes/parameters
filename	Name of the file, with extension, to append the table to
id_col	TRUE/FALSE of whether an numeric id column is the first column in the table and, in common Raven fashion, does not have a corresponding attribute (default TRUE)
justify	alignment of character columns (default 'right'). See format
sep	character(s) used to seperate columns (default ', ')
...	Extra arguments for write.fwf

Value

TRUE	returns TRUE if executed successfully

Author(s)

Leland Scantlebury, <leland@scantle.com>

Examples

```
soil_classes <- data.frame('Attributes' = c('DEFAULT','ALTERNATIVE'),
                           'SAND'      = c(0.4316, 0.3000),
                           'CLAY'      = c(0.1684, 0.4000),
                           'SILT'      = c(0.4000, 0.3000),
                           'ORGANIC'   = c(0.0000, 0.0000))
attributes <- c('%SAND','%CLAY','%SILT','%ORGANIC')
units <-  rep('none',4)

tf <- file.path(tempdir(), "Hogwarts.rvp")
rvn_write_Raven_table(tf, attributes = attributes, units = units, df = soil_classes)

# view file
readLines(tf)
```

---

rvn_wyear_indices *Water Year Indices*

Description

Returns the indices of the provided time series for the start/end of the water year. The month/day of the water year defaults to September 30 for an October 1 water year cycle. However, this may be supplied as other values, for example as June 30 for a July 1 water year (i.e. the Australian water year). This function is useful in supplying endpoints for water year evaluations.

Usage

```
rvn_wyear_indices(x, mm = 9, dd = 30)
```

Arguments

x	xts object or Date/POSITXtc series to obtain indices for
mm	month of water year (default 9)
dd	day of water year (default 30)

Details

Emulates the endpoints function for a water year period. The first and last points are included in all supplied endpoints, which may introduce partial periods to the analysis.

Value

ep	array of indices corresponding to start/end of water years

See Also

rvn_apply_wyearly to apply functions over the water year endpoints workhorse function for generating endpoints in xts for other periods

Examples

```
# read in sample forcings data
data(rvn_forcing_data)

# get the indices of the water year for October 1 (the default)
rvn_wyear_indices(rvn_forcing_data$forcings)
rvn_wyear_indices(rvn_forcing_data$forcings) %>%
rvn_forcing_data$forcings[., 1:2]

# get the indices of the start of the water year for July 1
## note that the last period is the last index, and not a complete water year period
rvn_wyear_indices(rvn_forcing_data$forcings, mm=6, dd=30) %>%
rvn_forcing_data$forcings[., 1:2]
```

---

rvn_xts_plot *Create plot from xts data*

Description

Generic function for plotting data from an xts format using the ggplot2 function.

Usage

```
rvn_xts_plot(
  x = NULL,
  prd = NULL,
  winter_shading = FALSE,
  wsdates = c(12, 1, 3, 31)
)
```

Arguments

x	time series object (xts) of data to plot
prd	period to use in plotting
winter_shading	optionally adds shading for winter months (default FALSE)
wsdates	integer vector of winter shading period dates (see details)

Details

Creates a plot using the supplied time series in xts format. The xts object is converted to a tibble using the rvn_fortify_xts function, and all columns are plotted by their respective names.

The winter_shading argument will add a transparent grey shading for the specified period by wsdates in each year that is plotted. Note that by changing the wsdates parameter values, the shading can be applied to any time of year.

wsdates is formatted as c(winter start month, winter start day, winter end month, winter end day). By default, wsdates is set to generate shading for December 1st to March 31st.

Note that a plot title is purposely omitted in order to allow the automatic generation of plot titles.

Value

p1	returns ggplot plot object

See Also

rvn_hyd_plot to create a hydrograph plot

rvn_hyd_extract to extract time series from Raven objects

Examples

```
# load sample hydrograph data, two years worth of sim/obs
ff <- system.file("extdata","run1_Hydrographs.csv", package="RavenR")
run1 <- rvn_hyd_read(ff)

# create a hydrograph with the generic xts plotting function
rvn_xts_plot(run1$hyd[,2:5])

# add shading for the month of August
rvn_xts_plot(run1$hyd[,2:5], winter_shading=TRUE, wsdates=c(8,1,8,31))
```

%notin% *%notin% operator*

Description

Syntax for the opposite of %in%

Usage

```
x %notin% table
```

Arguments

x	values to be matched
table	the values to be matched against

Examples

```
seq(1,5,1) %notin% seq(3,7,1)
```

Index