Package 'RavenR'

November 19, 2020

Type Package

Title Raven Hydrological Modelling Framework R Support and Analysis

Version 2.0.1

Date 2020-11-08

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 xts format, and support for writing Raven input files (rvt, rvh, rvc, etc.).

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

Depends R (>= 4.0.0)

Imports colorspace, cowplot, deldir, dplyr, dygraphs, gdata, ggplot2, grDevices, graphics, igraph, lubridate, magrittr, methods, ncdf4, purrr, reshape2, rgdal, rgeos, scales, sf, sp, stats, utils, xts, zoo

Suggests devtools, magick, knitr, weathercan, tidyhydat

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

License GPL-3

Encoding UTF-8

LazyData TRUE

RoxygenNote 7.1.1

NeedsCompilation no

Author Robert Chlumsky [cre, aut], James Craig [ctb], Genevieve Brown [ctb], Leland Scantlebury [ctb], Sarah Grass [ctb], Simon Lin [ctb]

Maintainer Robert Chlumsky <rchlumsk@uwaterloo.ca>

Repository CRAN

Date/Publication 2020-11-19 08:50:05 UTC

R topics documented:

RavenR-package . . . 3
cmax . . . 4
Nith_era5_sample . . . 5
rvn_annual_peak . . . 5
rvn_annual_peak_error . . . 7
rvn_annual_peak_event . . . 8
rvn_annual_peak_event_error . . . 10
rvn_annual_peak_timing_error . . . 11
rvn_annual_quantiles . . . 13
rvn_annual_quantiles_plot . . . 14
rvn_annual_volume . . . 15
rvn_apply_wyearly . . . 17
rvn_apply_wyearly_which_max_xts . . . 18
rvn_calc_runoff_coeff . . . 19
rvn_col_transparent . . . 20
rvn_cum_plot_flow . . . 21
rvn_custom_data . . . 22
rvn_custom_output_plot . . . 23
rvn_custom_read . . . 24
rvn_df_to_Raven_table . . . 25
rvn_exhaustive_mb_read . . . 26
rvn_fdc_plot . . . 27
rvn_flow_residuals . . . 28
rvn_flow_scatterplot . . . 30
rvn_flow_spaghetti . . . 31
rvn_forcings_plot . . . 32
rvn_forcings_read . . . 33
rvn_forcing_data . . . 34
rvn_gen_gridweights . . . 35
rvn_gen_obsweights . . . 37
rvn_get_prd . . . 39
rvn_hydrograph_data . . . 40
rvn_hyd_dygraph . . . 41
rvn_hyd_extract . . . 42
rvn_hyd_plot . . . 43
rvn_hyd_read . . . 45
rvn_iscolour . . . 46
rvn_monthly_vbias . . . 47
rvn_month_names . . . 48
rvn_netcdf_to_gridshp . . . 49
rvn_num_days . . . 50
rvn_num_days_month . . . 51
rvn_res_dygraph . . . 52
rvn_res_extract . . . 53
rvn_res_plot . . . 54
rvn_res_read . . . 56

rvn_rvc_res . . . 57
rvn_rvh_blankHRUdf . . . 58
rvn_rvh_blankSBdf . . . 59
rvn_rvh_cleanhrus . . . 59
rvn_rvh_overwrite . . . 61
rvn_rvh_read . . . 63
rvn_rvi_connections . . . 65
rvn_rvi_process_plot . . . 66
rvn_rvi_read . . . 67
rvn_rvt_ECmet . . . 68
rvn_rvt_flow . . . 70
rvn_rvt_obsfile . . . 72
rvn_rvt_obsweights . . . 73
rvn_rvt_read . . . 74
rvn_rvt_tidyhydat . . . 75
rvn_rvt_write . . . 77
rvn_rvt_wsc . . . 78
rvn_stringpad . . . 80
rvn_subbasin_map . . . 81
rvn_subbasin_network_plot . . . 82
rvn_substrLeft . . . 83
rvn_substrMLeft . . . 84
rvn_substrMRight . . . 85
rvn_substrRight . . . 85
rvn_theme_RavenR . . . 86
rvn_tidyhydat_sample . . . 87
rvn_ts_infill . . . 87
rvn_watershedmeb_read . . . 88
rvn_watershed_data . . . 89
rvn_watershed_read . . . 91
rvn_which_max_xts . . . 92
rvn_write_Raven_header . . . 93
rvn_write_Raven_label . . . 94
rvn_write_Raven_newfile . . . 95
rvn_write_Raven_table . . . 96
rvn_wyear_indices . . . 97

Index **99**

RavenR-package *RavenR*

Description

Contains functions to perform pre- and post-processing of RavenR files.

Details

This package provides a number of types of useful functions, including:

- reading in **Raven output files** (including ForcingFunctions, Hydrographs, etc.)
- processing some file types into **Raven format**
- flow-based **diagnostics** and plot functions
- other useful plotting functions (e.g. forcings.plot, flow duration curve, dyGraphs, network plots)
- useful utilities for time series and hydrologic functions (e.g. rvn_apply_wyearly)

See Also

The package GitHub page and James R. Craig's research page for software downloads, including the Raven web site

---

`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)
```

Nith_era5_sample *ERA5 Sample Dataset for Nith Watershed*

Description

ERA5 dataset in netcdf format, obtained as a sample dataset from the Global Water Futures (GWF) Cuizinart portal.

Details

Additional metadata on the ERA5 dataset may be found on the GWF metadata page.

See Also

rvn_netcdf_to_gridshp to generate a grid shapefile from a netcdf file

Examples

```
# access the file from the RavenR installation
system.file("extdata/Nith_era5_sample.nc", package="RavenR")
```

---

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

This function 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

This function 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

This function 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

This function 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

This function 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`	booolean 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

# Pull out a specific hydrograph
hgdata <- hyd_data$sim

# Calculate quantiles
qdat <- rvn_annual_quantiles(hgdata)
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
qdat <- rvn_annual_quantiles(hyd_data$obs)
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

rvn_annual_volume 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

This function 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

This is a function especially helpful to hydrology data or results. 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).

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

# apply mean as FUN to all forcings
rvn_apply_wyearly(rvn_forcing_data$forcings,mean,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

rvn_apply_wyearly_which_max_xts 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_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_col_transparent *Add Transparency to Colours*

Description

rvn_col_transparent is used to adjust colour codes to introduce transparency

Usage

```
rvn_col_transparent(colour, trans)
```

Arguments

colour	time series containing columns you wish to reseasonalize. xts object
trans	integer describing the degree of transparency, from ~200 (slightly transparent) to <10 (very transparent)

Details

Note that this function is not required for ggplot objects, as transparency can be added with the 'alpha' parameter.

Value

res	returned updated colour code with transparency

See Also

See original code on post in Stack Overflow plot points transparent in R

rvn_iscolour for checking validity of colour codes

Examples

```
# plot randomly distributed data
plot(rnorm(20),col='black')

# create a transparent blue colour for plotting
mycol <- rvn_col_transparent('red',100)

# plot more random points in transparent red colour
points(rnorm(20),col=mycol)
```

`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

This function will plot 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) http://www.civil.uwaterloo.ca/jrcraig/Raven/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 = NULL)
```

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 otpions of the custom output are included in the rav.obj attributes.

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	Dataframe 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_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

This function expects a full file path to the ExhaustiveMassBalance.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 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

This function 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

This function 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

This function 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

This function 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

This function 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
```

`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 = NULL)
```

Arguments

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

Details

This function 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.

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) http://www.civil.uwaterloo.ca/jrcraig/Raven/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_gen_gridweights` *Generate Raven grid weights from shapefile*

Description

Generates a Raven grid weights file given an HRU shapefile and a grid shapefile.

Usage

```
rvn_gen_gridweights(
  HRUshpfile,
  Gridshpfile,
  ValidHRUIDs,
  HRUIDcol = "HRU_ID",
  gridIDcol = "cellID",
  outfile = "GridWeights.rvt"
)
```

Arguments

HRUshpfile	polygon shapefile of HRUs with data column containing HRU IDs (.shp extension expected)
Gridshpfile	polygon shapefile of grid cells with data column containing cell IDs (.shp extension expected)
ValidHRUIDs	a vector of valid HRU IDs in the model
HRUIDcol	the name of the HRUshpfile polygon which contains the HRU IDs
gridIDcol	the name of the Gridshpfile polygon which contains the cell IDs
outfile	optional name of output Raven gaugeweights file

Details

Generate grid weights file GaugeWeights.rvt given an HRU shapefile with HRU ID column HRUIDcol (default 'HRU_ID') and a grid shapefile with ID column gridIDcol (default: 'cellID') weights are determined by the areal overlap of grid cell g and HRU k, i.e., wt[k][g]=area[k][g]/area[k] where wt[k][g] is the weight of cell g in HRU k, area[k] is the total area of HRU k, and area[k][g] is the area of HRU k that is within cell g.

By definition, the grid domain has to completely cover the HRU domain such that the sum of wt[k][g] for any k, over all g, is 1.0

Not a tonne of QA/QC is currently included - can fail due to bad netCDF file or inappropriate UTM zone; uses rgdal and rgeos libraries, and the accuracy of the gIntersection() routine leaves something to be desired. The shapefiles should be in the same projection, which is checked for in this function.

Value

TRUE	returns TRUE if executed properly. Also output GaugeWeights.rvt file

Author(s)

James R. Craig, University of Waterloo, 2019

See Also

rvn_netcdf_to_gridshp for converting netcdf files to grid shapefile format

Examples

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

# adjust HRU shapefile to one per subbasin for demonstration
rvh$HRUtable <- rvh$HRUtable[c(1,6,15,25),]
rvh$HRUtable$Area <- rvh$SBtable$Area
rvh$HRUtable$ID <- rvh$HRUtable$SBID

# define HRU shapefile (use subbasin shapefile for example)
```

```
HRUshpfile <- system.file("extdata","Nith_shapefile_sample.shp",package = "RavenR")

# write grid shapefile from netcdf file
nithnc <- system.file("extdata/Nith_era5_sample.nc", package="RavenR")
Gridshpfile <- file.path(tempdir(), "Nith_gridcells.shp")
myshp <- rvn_netcdf_to_gridshp(ncfile=nithnc, projID=26917, outshp=Gridshpfile)

# generate .rvt file of grid weights
ValidHRUIDs <- rvh$HRUtable$ID
tfout <- file.path(tempdir(), "Nith_GridWeights.rvt")
rvn_gen_gridweights(HRUshpfile, Gridshpfile, ValidHRUIDs,
gridIDcol = 'GridIDs', HRUIDcol = "subID", outfile = tfout)
```

`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

wtsreturns numeric vector of weights

Author(s)

James R. Craig, University of Waterloo

See Also

rvn_rvt_obsweights 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)
tf1 <- file.path(tempdir(), "myobservations.rvt")
rvn_rvt_obsfile(tf1, flows, 36,
  typestr = "HYDROGRAPH")

# 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)

# write observation weights to rvt file
tf2 <- file.path(tempdir(), "myobservations_wts.rvt")
rvn_rvt_obsweights(tf2, wts2, 36, typestr="HYDROGRAPH")
```

rvn_get_prd *Check period input*

Description

rvn_get_prd is a robust function used to check 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) http://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 = "", 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
figheight	height of figure, in pixels

Value

res 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

# 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)
```

Arguments

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

Details

rvn_hyd_extract is used to extract 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 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 a list with sim, obs, and inflow time series

`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_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

This function 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

TRUE	return TRUE if the function is executed properly

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
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 = NULL)
```

Arguments

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

Details

This function 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. If no timezone is provided, this is left as an empty string, and is determined by the function as the current time zone.

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_iscolour` *Check Validity of Colour Representation*

Description

rvn_iscolour checks whether a string or string vector contains valid colour representations (in text or hexadecimal form). Useful in error checking colour arguments for functions.

Usage

```
rvn_iscolour(x)
```

Arguments

x	string or string vector of colour representations to test

Value

y	vector of TRUE or FALSE indicating whether the colour is valid

See Also

See original code on post in Stack Overflow Check if character string is a valid color representation

`rvn_col_transparent` for creating transparent colour codes

Examples

```
rvn_iscolour(c(NA, "black", "blackk", "1", "#00", "#000000"))
#   <NA>   black  blackk       1     #00 #000000
#   TRUE    TRUE   FALSE    TRUE   FALSE    TRUE
```

`rvn_monthly_vbias` *Monthly Volume Bias*

Description

rvn_monthly_vbias 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
)
```

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)

Details

This function 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

rvn_month_names is used to 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_netcdf_to_gridshp` *Generate grid overlay from netCDF file*

Description

Takes the latitude-longitude cell coordinates from a netCDF file (assumed to be named 'lat' and 'long') generates an estimate of the grid polygons associated with each netCDF cell and exports this to a shapefile (outshp)

Usage

```
rvn_netcdf_to_gridshp(ncfile, projID = NULL, outshp = NULL)
```

Arguments

`ncfile`	netCDF file with latitude and longitude variables
`projID`	projected coordinate system ID (EPSG numeric code) to project shapefile to (default NULL, optional)
`outshp`	name of output shapefile (DEFAULT NULL, optional)

Details

projID should be provided as an integer value referring to a valid EPSG coordinate system. If projID is left NULL, the output shapefile will be left in lat/long coordinates and WGS84 projection. Note that the function can fail due to bad netCDF file or inappropriate coordinate system ID. A list of IDs for projected coordinate system may be found on the Spatial Reference webpage or on the ESRI webpage.

If the outshp is NULL, the shapefile object is returned by the function and nothing is written to file. If outshp is provided, the shapefile is also written to file. The outshp should be supplied as the file name (with or without .shp extension).

This function uses the sf::st_write function to write the shapefile, if outshp is provided.

Additional metadata on the sample netcdf file may be found with '?Nith_era5_sample'.

Value

`shapefile`	returns sf shapefile object; will also write a shapefile to the outshp if provided

Author(s)

James R. Craig, University of Waterloo, 2019

See Also

`rvn_gen_gridweights` for generating a shapefile of gridweights

Examples

```
# get location for sample netcdf file
?Nith_era5_sample
ncfile <- system.file("extdata/Nith_era5_sample.nc", package="RavenR")

# produce shapefile in lat/long
myshp <- rvn_netcdf_to_gridshp(ncfile)
class(myshp)
sf::st_crs(myshp)$input
plot(myshp$geometry)

# write shapefile to file in UTM coordinates
projID <- 26917 # NAD83 UTM Zone 17N, appropriate UTM zone for Nith watershed
outshp <- file.path(tempdir(), "Nith_gridcells.shp")
myshp <- rvn_netcdf_to_gridshp(ncfile, projID, outshp)
sf::st_crs(myshp)$input
```

---

rvn_num_days *Number of Days between two dates*

Description

rvn_num_days is used to 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

rvn_num_days_month is used to calculate 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

rvn_res_extract is used to extract 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

rvn_res_plot creates a reservoir stage plot for the supplied flow series, or equivalently a stage plot for reservoir stages.

Usage

```
rvn_res_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 stage
`obs`	time series object of observed stage
`inflow`	time series object of inflow 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

This function 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 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 rvn_res_extract function.

The winter_shading argument will add a transparent cyan 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

TRUE	return TRUE if the function is executed properly

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

rvn_res_read is used to read in the ReservoirStages.csv file produced by the modelling Framework Raven.

Usage

```
rvn_res_read(ff = NA, tzone = NULL)
```

Arguments

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

Details

This function 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.

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_rvc_res *Create initial conditions file for Reservoirs*

Description

rvn_rvc_res is used to 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

This function is used to write 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_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.

Usage

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

Arguments

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

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,
  ProtectedHRUList = 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.
ProtectedHRUList	list of HRU IDs that are sacrosanct (not to be removed)

Details

rvh.clean 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)

Value

hru_tablecleaned 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)
rvh$HRUtable <- rvn_rvh_cleanhrus(rvh$HRUtable,rvh$SBtable,area_tol = 0.05, merge=TRUE)
```

`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,
  HRUtable,
  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', 'Terrain', 'Aquifer', '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.

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_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

This function does not like comma-delimited tables 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 t0o 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.

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, Terrain, Aquifer, 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_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_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

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

Value

Returns a dataframe of all of the process connections Includes the following data columns: process type, algorithm, 'from' compartment, 'to' compartment, and conditional.

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_plot` 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"))

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

conn <- rvn_rvi_connections(rvi)
head(conn)
```

`rvn_rvi_process_plot` *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_plot(connections, pdfout = NULL)
```

Arguments

`connections`	a dataframe of from-to connections generated using rvn_rvi_connections()
`pdfout`	name of pdf file to save the network plot to, if null no PDF is generated

Details

This function uses the output from the `rvn_rvi_connections` function to generate the plot.

Value

p1returns ggplot object. Also generates a .pdf file in working directory if pdfplot argument is not NULL.

Note

tries to follow basic network structure, accomodates unrecognized state variables on LHS of plot

Author(s)

James R. Craig, University of Waterloo

See Also

`rvn_rvi_connections` to generate connections table from an rvi object

See also the Raven page

Examples

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

rvn_rvi_process_plot(conn)
```

`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 including a lone item:

`HydProcTable`	a data table of hydrologic processes. Includes the following data columns: process type, algorithm, 'from' compartment, 'to' compartment, and condition

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_rvt_ECmet` *EC Climate Station File Conversion*

Description

rvn_rvt_ECmet converts Environment Canada historical meteorological data for a given station into the .rvt format usable in Raven.

Usage

```
rvn_rvt_ECmet(
  metdata,
  filename = NULL,
  prd = NULL,
  stnName = NULL,
  forcing_set = 1,
  prefix = "met_",
  write_redirect = FALSE,
  write_stndata = FALSE,
  rd_file = "met_redirects.rvt",
  stndata_file = "met_stndata.rvt"
)
```

Arguments

`metdata`	EC meteorological data from one or more stations (e.g., from weathercan::weather_dl())
`filename`	specified name of file to write to (optional)
`prd`	(optional) data period to use in .rvt file
`stnName`	(optional) station name to use (instead of name in file)
`forcing_set`	(optional) specifies the set of forcings to print to file
`prefix`	(optional) prefixes the file name (default: "met_")
`write_redirect`	(optional) write the :RedirectToFile commands in a separate .rvt file
`write_stndata`	(optional) write the gauge data to a separate .rvt file
`rd_file`	(optional) name of the redirect file created (if write_redirect = TRUE)
`stndata_file`	(optional) name of the station data file created (if write_stndata = TRUE)

Details

The function prints in the :MultiData format; the particular set of forcings to print can be set with the forcing_set command. The data should be downloaded with missing days included. The download website is linked below.

The function will write to name generated from the station name, otherwise the .rvt filename may be specified with the filename argument (full path to the filename, including .rvt extension).

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.

stnName can be supplied to overwrite the station name that is otherwise obtained from the Station Name field in the climate file. Spaces in raw Station Names will be replaced with underscores.

prefix can be used to add a prefix to the .rvt file name, ("met_" by default) which may be useful in organizing multiple climate data files.

forcing_set specifies the set of forcings to print to .rvt file. Currently there are only two available sets. A value of 1 prints total precipitation, and 2 splits the precipitation into rainfall and snowfall. In some cases the EC data provides only total precipitation, which is a good check to make for the particular climate station before printing rvt files. Both sets currently print max and min daily temperature. Future extensions to this function may provide more options for forcing sets.

write_redirect will print out the :RedirectToFile commands in a separate file, met_redirects.rvt. These commands can be copied into the main model's .rvt file to redirect to the produced time series files. The function will append to the file if it already exists, meaning that this works for iterations of this function.

write_stndata wil print out the gauge metadata to file (met_stndata.rvt) 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.

perform.qc is currently under construction and is not yet available; setting to TRUE will result in an warning.

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

* checking that all supplied files are for the same climate station * ensuring; the timestep (data resolution) is the same in each file * automatically; combining time series and ensuring there are no gaps in the data supplied (i.e. time gaps, not missing values); and * check for missing data and issuing a warning that post-processing will be required

Note: Data quality is not assessed in this package, such as consistency between minimum and maximum temperatures. Subdaily data is not currenty supported.

Note: this function is designated to use data from the weathercan package. The weathercan package is external to RavenR and is not an explicit dependent package of RavenR.

Value

TRUE	return TRUE if the function is executed properly

See Also

rvn_rvt_wsc to convert WSC flow gauge data to Raven format

Download EC climate data from EC Historical Data

Examples

```
# Download data using weathercan weather_dl
library(weathercan)
kam <- weather_dl(station_ids = 51423,
                  start = "2016-10-01", end = "2019-09-30", interval="day")

# basic use, override filename to temporary file
# default forcing_set (PRECIP, MAX TEMP, MIN TEMP)
rvn_rvt_ECmet(metdata = kam, forcing_set = 1,
  filename = file.path(tempdir(), "rvn_rvt_ECmetfile1.rvt"))

# use the second forcing set instead
# default forcing_set (PRECIP, MAX TEMP, MIN TEMP)
rvn_rvt_ECmet(metdata = kam, forcing_set = 2,
  filename = file.path(tempdir(), "rvn_rvt_ECmetfile2.rvt"))
```

---

rvn_rvt_flow *Write Raven rvt file from Flow Series*

Description

rvn_rvt_flow generates a Raven rvt file from a flow series

Usage

```
rvn_rvt_flow(
  flow.series,
  subID,
  stnName = NULL,
  rvt_type = "ObservationData",
  prd = NULL,
  write_redirect = FALSE,
  flip_number = FALSE,
  filename = NULL
)
```

Arguments

flow.series	flows to write to file in xts format
subID	subbasin ID corresponding to the flow series

stnName	name of the station or file to write to file (used to build rvt file name, required if filename not provided)
rvt_type	type of flow-based rvt file to write, default ObservationData
prd	period to use in writing rvt file, format "YYYY-MM-DD/YYYY-MM-DD"
write_redirect	(optional) write the :RedirectToFile commands in a separate .rvt file
flip_number	(optional) put the subID first in the .rvt filename
filename	specified name of file to write to (optional)

Details

This function writes the rvt file for a given time series of flows. The supplied flows should be in the xts format. This function operates similarly to the ECflow.rvt function (linked below).

prd is used by the xts formatted-data to restrict the data reported in .rvt file 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.

write_redirect will print out the :RedirectToFile command in a separate file called, "flow_stn_redirect_text.rvt". This command 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 alphabeticized 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, otherwise the .rvt filename may be specified with the filename argument (full path to the filename, including .rvt extension).

Value

TRUE	returns TRUE if the function executed successfully

See Also

rvn_rvt_wsc to create an rvt file from Water Survey Canada (WSC) data

Examples

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

# write out observation flow file to temporary file
rvn_rvt_flow(obs,subID=36,
  filename = file.path(tempdir(), "Nith_obs.rvt"))
```

`rvn_rvt_obsfile` *Create Raven observation data (rvt) file*

Description

Creates an observation data file named filename from a continuous (gap-free) xts time series ts

Usage

```
rvn_rvt_obsfile(filename, ts, SBID, typestr = "HYDROGRAPH", units = "m3/s")
```

Arguments

`filename`	observation data file to be created, with .rvt extension
`ts`	xts time series with single data column
`SBID`	Subbasin ID for hydrographs and reservoir stages or HRU ID for observations of state variables (e.g., snow depth)
`typestr`	Raven-recognized data type string: 'HYDROGRAPH', 'RESERVOIR_STAGE', 'RESERVOIR_INFLOW', 'RESERVOIR_NET_INFLOW', or a state variable name (e.g., SOIL[0] or SNOW)
`units`	units of the data (should be consistent with Raven units; neither Raven nor this routine checks or corrects

Value

TRUE returns TRUE if function runs properly

Author(s)

James R. Craig, University of Waterloo

See Also

`rvn_ts_infill` for infilling time series, and `rvn_rvt_obsweights` to write an rvt observation weights 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)
tf <- file.path(tempdir(), "run1_Hydrographs.rvt")
```

```
rvn_rvt_obsfile(tf, flows, 36, typestr = "HYDROGRAPH")
readLines(tf) %>% head()
```

`rvn_rvt_obsweights` *Create Raven observation data weight (rvt) file*

Description

Writes observation weights generated from `rvn_gen_obsweights` to a Raven rvt format.

Usage

```
rvn_rvt_obsweights(
  filename = "_obsweights.rvt",
  wts,
  SBID = 1,
  typestr = "HYDROGRAPH"
)
```

Arguments

`filename`	observation data file to be created, with .rvt extension
`wts`	xts time series with single data column of observation weights
`SBID`	Subbasin ID for hydrographs and reservoir stages or HRU ID for observations of state variables (e.g., snow depth)
`typestr`	Raven-recognized data type string: 'HYDROGRAPH', 'RESERVOIR_STAGE', 'RESERVOIR_INFLOW', 'RESERVOIR_NET_INFLOW', or a state variable name (e.g., SOIL[0] or SNOW)

Details

Any NA values in the weights are converted to the flag -1.2345, used in Raven as NA values.

Value

TRUE returns TRUE if function runs properly

Author(s)

James R. Craig, University of Waterloo

See Also

`rvn_rvt_obsfile` `rvn_gen_obsweights`

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)
rvn_rvt_obsfile(filename=file.path(tempdir(),"run1_Hydrographs.rvt"),
  flows, 36, typestr = "HYDROGRAPH")

# 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)

# write observation weights to rvt file
rvn_rvt_obsweights(filename=file.path(tempdir(), "run1_Hydrographs_obsweights.rvt"),
  wts2, 36, typestr="HYDROGRAPH")
```

`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.

Usage

```
rvn_rvt_read(filename)
```

Arguments

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

Details

It supports :MultiData, :Data, :ObservedData, :BasinInflowHydrograph, and most of the other :Data-like time series commands. It does NOT support the master .rvt file with :Gauge or :Gridded-Forcing commands

Value

Returns an xts time series with at least one dataset (multiple for :MultiData files)

Author(s)

James R. Craig, University of Waterloo

Examples

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

This function 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 the .csv file, multiple observation files are created

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 alphabeticized 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.

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

rvn_rvt_write generates a Raven rvt file from a time series

Usage

```
rvn_rvt_write(
  ts,
  params,
  units,
  dates = NULL,
  prd = NULL,
  tt = "00:00:00",
  dt = 1,
  ff = "raven_rvt_write.rvt"
)
```

Arguments

ts	time series to write in xts or dataframe format
params	the full string expression for the parameters line to write to file
units	the full string expression for the units line to write to file
dates	vector of date objects passed, necessary only if ts is not xts
prd	period to use in writing rvt file, format "YYYY-MM-DD/YYYY-MM-DD"
tt	initial start time to file
dt	time interval to write to file
ff	filename to write to without .rvt extension (added automatically)

Details

This function writes the rvt file for a given time series dataset. The function will write out the entirety of the columns provided in the given xts object. Please ensure that the parameters supplied in the params and units objects match the xts object supplied.

Value

flag	returns TRUE if the function executed successfully

See Also

rvn_rvt_wsc to create an rvt file from Water Survey Canada (WSC) data

Examples

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

# write time series to rvt file using data from subbasin 36
rvn_rvt_write(mydata$hyd$Sub36, params = "HYDROGRAPH", units = "m3/s",
  ff = file.path(tempdir(), 'raven_rvt_write'))
```

---

rvn_rvt_wsc *EC Streamgauge File Conversion*

Description

rvn_rvt_wsc converts Environment Canada historical streamgauge data, downloaded from the Water Survey Canada, into .rvt format files usable in Raven.

Usage

```
rvn_rvt_wsc(
  ff,
  subIDs,
  prd = NULL,
  stnNames = NULL,
  write_redirect = FALSE,
  flip_number = FALSE,
  filename = NULL
)
```

Arguments

ff	WSC flow file in csv format
subIDs	vector of subbasin IDs to correspond to the stations in ff
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
filename	specified name of file(s) to write to (optional)

Details

This function takes a single WSC flow file and converts the flow data for each station in the file into .rvt formatted files for a Raven model. If multiple stations exist in the .csv file,

The file should be downloaded in a .csv Date-Data format with missing days included. The download website is linked below. Any level data will be ignored, although can be included in the file for stations with flow and level data. If no data is found for a given station, an error will be reported.

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. If the subbasin is currently unknown, please supply a placeholder value.

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 .csv file. 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 alphabeticized 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.

Value

TRUE	return TRUE if the function is executed properly

See Also

rvn_annual_peak_event to consider the timing of peak events

Download EC streamgauge data from WSC Historical Data.

Examples

```
ff = system.file("extdata",'Daily__Oct-1-2020_08_20_52PM.csv', package="RavenR")
rvn_rvt_wsc(ff,subIDs=c(6), filename=file.path(tempdir(), "stn_02GB001.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")
```

Arguments

`string`	Text string
`width`	Number of characters total, including desired spaces
`just`	'r' for right, 'l' for left

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_subbasin_map *Plot Continuous Data Using Subbasin Shapefile*

Description

Plots Raven custom output into a subbasin map

Usage

```
rvn_subbasin_map(
  shpfilename,
  subIDcol,
  plot_date,
  cust_data = NULL,
  normalize_data = FALSE,
  invalid_stop = TRUE,
  basins_label = "subID",
  plot_invalid = FALSE
)
```

Arguments

shpfilename	filename of shapefile containing HRU polygons, with one column inidicating Raven HRU ID
subIDcol	string of subbasin ID column in shapefile
plot_date	string of date to plot in custom.data
cust_data	custom data set as read in by custom.read, for daily by_subbasin data
normalize_data	whether to normalize data by all cust_data (TRUE) or just the data for the given date (FALSE)
invalid_stop	whether to stop if invalid basins are found (TRUE) or just continue with a warning (FALSE)
basins_label	label to put on basins, one of c('None,'subID','value') to show nothing, subbasinIDs, or actual plotted values
plot_invalid	boolean indicating whether to plot invalid basins in grey (currently disabled)

Details

Does not currently support discrete data such as land use. Ability to include invalid basins in grey is currently disabled.

Value

p1	ggplot object of subbasin map

Author(s)

James R. Craig, University of Waterloo

Robert Chlumsky

Genevieve Brown

See Also

rvn_subbasin_network_plot to create network plots

Examples

```
# Raw shapefile sample data
shpfilename <- system.file("extdata","Nith_shapefile_sample.shp",package="RavenR")

# Custom Output data from Raven for Nith basin
cust_file <- system.file("extdata","run1_PRECIP_Daily_Average_BySubbasin.csv",
                         package="RavenR")
cust_data <- rvn_custom_read(cust_file)

subIDcol <- 'subID'        # attribute in shapefile with subbasin IDs
plot_date <- "2003-03-30"  # date for which to plot custom data

# Generate plot object
p1 <- rvn_subbasin_map(shpfilename,subIDcol,plot_date,cust_data, normalize=TRUE)
p1
```

---

rvn_subbasin_network_plot

Plot Raven subbasin network.

Description

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

Usage

```
rvn_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.

Value

p1ggplot 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_subbasin_network_plot(rvh$SBtable)

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

---

rvn_substrLeft *substring from the Left*

Description

rvn_substrLeft 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

rvn_substrMLeft 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

rvn_substrMRight 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

rvn_substrRight 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

rvn_theme_RavenR makes the general Raven R Theme for all ggplots

Usage

```
rvn_theme_RavenR()
```

Details

This function 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

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 inbetween 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

rvn_watershedmeb_read is used to read in the WatershedMassEnergyBalance.csv file produced by the modelling Framework Raven.

Usage

```
rvn_watershedmeb_read(ff = NA, tzone = NULL)
```

Arguments

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

Details

This function expects a full file path to the WatershedMassEnergyBalance.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. 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.

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) http://www.civil.uwaterloo.ca/jrcraig/Raven/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

rvn_watershed_read is used to read in the WatershedStorage.csv file produced by Raven.

Usage

```
rvn_watershed_read(ff = NA, tzone = NULL)
```

Arguments

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

Details

This function 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.

This function is also built to support the wshd.animate function, which uses the object created here for creating an animation of the watershed storage containers.

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.

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_which_max_xts` *which.max for xts objects*

Description

rvn_which_max_xts 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

This function is intended to act 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

rvn_wyear_indices 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 30th 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

Note that this function is meant to emulate 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]
```

Index

∗ **datasets**
rvn_custom_data, 22
rvn_forcing_data, 34
rvn_hydrograph_data, 40
rvn_tidyhydat_sample, 87
rvn_watershed_data, 89

apply.monthly, *4*
apply.yearly, *4*

cmax, 4

endpoints, *97, 98*

format, *96*

Nith_era5_sample, 5

RavenR-package, 3
rvn_annual_peak, 5, *9*, *11*
rvn_annual_peak_error, 7, *11*
rvn_annual_peak_event, *6*, *8*, 8, *12*, *79*
rvn_annual_peak_event_error, *8*, 10, *12*
rvn_annual_peak_timing_error, 11
rvn_annual_quantiles, 13, *14*
rvn_annual_quantiles_plot, 14
rvn_annual_volume, *6*, 15, *48*, *86*
rvn_apply_wyearly, *4*, 17, *98*
rvn_apply_wyearly_which_max_xts, 18, *92*
rvn_calc_runoff_coeff, 19
rvn_col_transparent, 20, *46*
rvn_cum_plot_flow, 21, *22*
rvn_custom_data, 22
rvn_custom_output_plot, *22, 23*, 23, *24*, *40*
rvn_custom_read, *22, 23*, 24, *40*
rvn_df_to_Raven_table, 25
rvn_exhaustive_mb_read, 26, *27*
rvn_fdc_plot, 27
rvn_flow_residuals, 28
rvn_flow_scatterplot, *16*, *22*, *29*, 30, *31*
rvn_flow_spaghetti, 31, *44*
rvn_forcing_data, 34
rvn_forcings_plot, 32, *35*
rvn_forcings_read, *30*, *32*, 33, *35*
rvn_gen_gridweights, 35, *49*
rvn_gen_obsweights, 37, *73*
rvn_get_prd, 39
rvn_hyd_dygraph, 41, *52*
rvn_hyd_extract, *28*, 42, *44, 45*
rvn_hyd_plot, *42*, 43
rvn_hyd_read, *27, 28*, *34*, *41, 42*, 45, *55*, *88*, *91*
rvn_hydrograph_data, 40
rvn_iscolour, *20*, 46
rvn_month_names, 48
rvn_monthly_vbias, 47
rvn_netcdf_to_gridshp, *5*, *36*, 49
rvn_num_days, *48*, 50, *51*
rvn_num_days_month, *50*, 51
rvn_res_dygraph, 52
rvn_res_extract, 53, *55, 56*
rvn_res_plot, *53*, 54
rvn_res_read, *52, 53*, 56, *57*
rvn_rvc_res, 57
rvn_rvh_blankHRUdf, 58, *59*, *62*
rvn_rvh_blankSBdf, *58*, 59, *62*
rvn_rvh_cleanhrus, 59, *62*
rvn_rvh_overwrite, 61
rvn_rvh_read, *19*, *59, 60*, *62*, 63, *82, 83*
rvn_rvh_write, *58*, *60*, *64*
rvn_rvh_write *(*rvn_rvh_overwrite*)*, 61
rvn_rvi_connections, 65, *66*
rvn_rvi_process_plot, *65*, 66
rvn_rvi_read, *65*, 67
rvn_rvt_ECmet, 68
rvn_rvt_flow, 70
rvn_rvt_obsfile, 72, *73*
rvn_rvt_obsweights, *38*, *72*, 73
rvn_rvt_read, 74
rvn_rvt_tidyhydat, 75, *87*

rvn_rvt_write, 77
rvn_rvt_wsc, *70, 71*, *78*, 78
rvn_stringpad, 80
rvn_subbasin_map, 81
rvn_subbasin_network_plot, *64*, *82*, 82
rvn_substrLeft, 83, *84–86*
rvn_substrMLeft, 84, *85, 86*
rvn_substrMRight, *83, 84*, 85, *86*
rvn_substrRight, *83–85*, 85
rvn_theme_RavenR, *39*, 86
rvn_tidyhydat_sample, 87
rvn_ts_infill, *72*, 87
rvn_watershed_data, 89
rvn_watershed_read, *89, 90*, 91
rvn_watershedmeb_read, 88, *91*
rvn_which_max_xts, 92
rvn_write_Raven_header, 93
rvn_write_Raven_label, 94
rvn_write_Raven_newfile, 95
rvn_write_Raven_table, 96
rvn_wyear_indices, *17*, 97

which.max, *92*
write.fwf, *96*