Package ‘CSHShydRology’

July 16, 2024

Type    Package
Title   Canadian Hydrological Analyses
Version 1.4.2
Date 2024-07-12
Author Kevin Shook [cre, aut],
       Paul Whitfield [aut],
       Robert Chlumsky [aut],
       Daniel Moore [aut],
       Martin Durocher [aut],
       Matthew Lemieux [ctb],
       Jason Chiang [ctb],
       Joel Trubilowicz [ctb],
       SJ Kim [ctb]
Maintainer Kevin Shook <kevin.shook@usask.ca>
Description A collection of user-submitted functions to aid in the analysis of hydrological data, particularly for users in Canada. The functions focus on the use of Canadian data sets, and are suited to Canadian hydrology, such as the important cold region hydrological processes and will work with Canadian hydrological models. The functions are grouped into several themes, currently including Statistical hydrology, Basic data manipulations, Visualization, and Spatial hydrology. Functions developed by the Floodnet project are also included. CSHHydRology has been developed with the assistance of the Canadian Society for Hydrological Sciences (CSHS) which is an affiliated society of the Canadian Water Resources Association (CWRA). As of version 1.2.6, functions now fail gracefully when attempting to download data from a url which is unavailable.
License AGPL-3
URL https://github.com/CSHS-hydRology/CSHShydRology
Depends R (>= 4.0.0)
Imports fields, Kendall, lubridate, plotrix, timeDate, stringr, ggplot2, ggspatial, stats, raster, sf, dplyr, magrittr, httr, tidyhydat, whitebox, datasets, circular
Suggests knitr, testthat, rmarkdown, readr
VignetteBuilder knitr
LazyData true
Encoding UTF-8
RoxygenNote 7.3.2
NeedsCompilation no
Repository CRAN
Date/Publication 2024-07-16 09:00:10 UTC

Contents

CSHShydRology-package ........................................ 3
Basic_data_manipulation_functions ............................. 5
CAN01AD002 .................................................. 5
CAN05AA008 .................................................. 6
ch_axis_doy .................................................. 6
ch_binned_MannWhitney ........................................ 7
ch_booth_plot ............................................... 9
ch_catchment_hyps .......................................... 10
ch_checkcatchment ......................................... 12
ch_checkchannels ......................................... 14
ch_circ_mean_reg ........................................ 16
ch_clear_wd ................................................ 17
ch_color_gradient ........................................ 18
ch_col_gradient .......................................... 19
ch_col_transparent .................................... 20
ch_contours ............................................... 21
ch_create_wd ............................................. 22
ch_cut_block ............................................. 23
ch_date_subset .......................................... 23
ch_decades_plot ......................................... 24
ch_doy .................................................... 25
ch_fdcurve ............................................... 26
ch_flow_raster .......................................... 27
ch_flow_raster_qa ..................................... 28
ch_flow_raster_trend .................................. 29
ch_get_ECDE_metadata .................................. 31
ch_get_peaks ............................................. 32
ch_get_url_data ........................................ 33
ch_get_wscstation ..................................... 35
ch_hydrograph_plot .................................... 36
ch_polar_plot .......................................... 38
ch_polar_plot_peaks .................................. 39
ch_polar_plot_prep .................................... 41
ch_qa_hydrograph ....................................... 43
ch_read_AHCCD_daily .................................. 45
ch_read_AHCCD_monthly ................................ 46
ch_read_ECDE_flows .................................. 47
Description

**CSHShydRology** is intended for the use of hydrologists, particularly those in Canada. It will contain functions which focus on the use of Canadian data sets, such as those from Environment Canada. The package will also contain functions which are suited to Canadian hydrology, such as the important cold-region hydrological processes. **CSHShydRology** will also contain functions which work with Canadian hydrological models, such as Raven, CRHM, Watflood, and MESH.

This package has been developed with the assistance of the Canadian Society for Hydrological Sciences (CSHS) [https://cshs.cwra.org/en/](https://cshs.cwra.org/en/) which is an affiliated society of the Canadian Water Resources Association (CWRA) [https://cwra.org/](https://cwra.org/).

The **CSHShydRology** will contain functions grouped into several themes, including:

- Statistical hydrology  trend detection, data screening, frequency analysis, regionalization
Basic data manipulations input/conversion/adapter functions, missing data infilling
Visualization data visualization, standardized plotting functions
Spatial hydrology basin delineation, landscape data analysis, working with GIS
Streamflow measurement analysis rating curve analysis, velocity profiles, naturalization
Network design/analysis homogeneity assessment
Ecohydrology fisheries and ecological analysis
Wrappers/unwrappers between other packages and CSHShydRology

Author(s)

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

Authors:

- Paul Whitfield <paul.h.whitfield@gmail.com>
- Robert Chlumsky <rchlumsk@uwaterloo.ca>
- Daniel Moore <dan.moore@ubc.ca>
- Martin Durocher

Other contributors:

- Matthew Lemieux [contributor]
- Jason Chiang [contributor]
- Joel Trubilowicz [contributor]
- SJ Kim [contributor]

References

To cite CSHShydRology in publications, use the command citation("CSHShydRology") to get the current version of this citation.

See Also

Useful links:

- https://github.com/CSHS-hydRology/CSHShydRology
Basic_data_manipulation_functions

Basic data manipulation functions

Description

These functions read in or convert values among formats

- **ch_read_ECDE_flows**: Reads a file of WSC daily flows from ECDaExplorer
- **ch_get_ECDE_metadata**: Reads station meta data from ECDaExplorer
- **ch_get_wscstation**: Reads station information from a data file produced by ECDE
- **ch_read_AHCCD_daily**: Reads file of daily AHCCD values
- **ch_read_AHCCD_monthly**: Reads file of monthly AHCCD values
- **ch_tidyhydat_ECDE**: Reads flows using tidyhydat and converts to ECDE format
- **ch_tidyhydat_ECDE_meta**: Reads station meta data using tidyhydat and converts to ECDE-like format

CAN01AD002 Streamflow data

Description

Daily river discharge for the station 01AD002 on St. John River at Fort Kent, New Brunswick. Data ranges from 1926 to 2014, for basin area of 14700 sq km.

Usage

CAN01AD002

Format

An object of class data.frame with 32234 rows and 2 columns.

Author(s)

Martin Durocher

Source

https://wateroffice.ec.gc.ca/
Description
A dataframe of Water Survey of Canada (WSC) daily flows for station 05AA008, CROWSNEST RIVER AT FRANK Alberta. Drainage area 403 km².

Usage
CAN05AA008

Format
A dataframe with 25252 rows and 5 columns spanning the period 1910-2013.

Details
Variables:
- **ID**: StationID
- **PARAM**: Parameter 1=Flow, 2=Level
- **Date**: R date
- **Flow**: Daily flow in m³/s
- **SYM**: Water Survey FLags A, B, D, E

Source
Water Survey of Canada

ch_axis_doy
Generates the x axis beginning on specified day of year

Description
Generates an axis for day of year or day of water year; used by ch_regime_plot. Obtaining the day of water year needs to be done separately.

Usage
ch_axis_doy(wyear = 1)

Arguments
- **wyear**: Month of beginning of water year, wyear = 1 (the default) for calendar year, wyear = 10 to start October 1.
ch_binned_MannWhitney

Value

Plots a water year axis on a standard R plot

Author(s)

Paul Whitfield

See Also

ch_regime_plot

Examples

a <- seq(1, 365)
b <- runif(365)
plot(a, b, type = "p", xlab = "", xaxt = "n")
ch_axis_doy(wyear = 10) # starts in October

ch_binned_MannWhitney  Compares two time periods of data using Mann-Whitney test

Description

Compares two time periods of data using the Mann-Whitney test. Data are binned based upon a bin size, and data are extracted for two time periods and tests for change between two such periods result can be passed to ch_polar_plot or ch_decades_plot for visualization.

Usage

ch_binned_MannWhitney(
  DF,
  step,
  range1,
  range2,
  ptest = 0.05,
  variable = "discharge",
  metadata = NULL
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DF</td>
<td>A data frame of hydrometric data from ch_read_ECDE_flows</td>
</tr>
<tr>
<td>step</td>
<td>An integer indicating the degree of smoothing eg. 1, 5, 11.</td>
</tr>
<tr>
<td>range1</td>
<td>The first and last year of first period, as c(first, last)</td>
</tr>
<tr>
<td>range2</td>
<td>The first and last year of second period, as c(first, last)</td>
</tr>
<tr>
<td>ptest</td>
<td>The significance level default is 0.05.</td>
</tr>
<tr>
<td>variable</td>
<td>Name of variable. Default is ‘discharge’</td>
</tr>
<tr>
<td>metadata</td>
<td>dataframe of station metadata, default is HYDAT_list</td>
</tr>
</tbody>
</table>
Value

Returns a list containing:

- StationID: ID of station
- Station_lname: Name of station
- bin_width: Smoothing time step
- range1: First range of years
- range2: Second range of years
- p_used: p_value
- fail: TRUE if test failed due to missing values
- bin_method: method used for binning
- test_method: Mann-Whitney U-statistic
- series: a data frame containing:
  - period: period numbers i.e. 1:365/step
  - period1: median values for each bin in period 1
  - period2: median values for each bin in period 2
  - mwu: Mann-Whitney U-statistic for each bin between the two periods
  - prob: probability of U-statistic for each period
  - code: significance codes for each bin

Author(s)

Paul Whitfield

References


See Also

ch_polar_plot ch_polar_plot_prep ch_decades_plot

Examples

data(HYDAT_list)
data(CAN05AA008)
# first example fails due to missing data in both periods
range1 <- c(1960,1969)
range2 <- c(1990,1999)
b_MW <- ch_binned_MannWhitney(CAN05AA008, step = 5, range1, range2, ptest = 0.05)

range1 <- c(1970,1979)
range2 <- c(1990,1999)
b_MW <- ch_binned_MannWhitney(CAN05AA008, step = 5, range1, range2, ptest = 0.05)
ch_booth_plot

Create Booth plot of peaks over a threshold

Description

A Booth plot is a plot of peaks over threshold flood events with duration on the horizontal and either magnitude (default) or volume on the vertical axis.

Usage

ch_booth_plot(events, threshold, title, type = "mag", colour1 = 1, colour2 = 1)

Arguments

- events: A data frame of POT events from the function ch_get_peaks
- threshold: The threshold used by ch_get_peaks
- title: Plot title
- type: The plot type, either 'mag' (magnitude, the default) or 'vol' (volume)
- colour1: A vector of length 12 with line colours of rings or symbols. Defaults to those used by Booth.
- colour2: A vector of length 12 with fill colours of rings or symbols. Defaults to those used by Booth.

Value

No value is returned; a standard R graphic is created.

Author(s)

Paul Whitfield

References


See Also

ch_get_peaks
Examples

```r
threshold <- 0.1 * max(CAN05AA008$Flow) # arbitrary threshold
peaks <- ch_get_peaks(CAN05AA008, threshold)
events <- peaks$POTevents
ch_booth_plot(events, threshold, title = "05AA008", type='mag')
ch_booth_plot(events, threshold, title = "05AA008", type='vol')
```

---

**ch_catchment_hyps**  
*Catchment hypsometry*

---

**Description**

Finds the hypsometric curve, which is the total fraction of the area below vs. elevation, for a given basin.

**Usage**

```r
ch_catchment_hyps(
  catchment,
  dem,
  z_levels = NULL,
  n_levels = 10,
  zmin = NULL,
  zmax = NULL,
  quantiles = NULL,
  hypso_plot = FALSE,
  z_units = "m",
  col = "red",
  type = "o",
  xlab = "Fraction of catchment below given elevation",
  ylab = paste0("Elevation (", z_units, ")"),
  add_grid = FALSE,
  ...
)
```

**Arguments**

- `catchment`: A *sf* object containing the catchment divide.
- `z_levels`: Vector of elevation levels for the hypsometry. If specified, then no other elevation parameters are required. Default is NULL.
- `n_levels`: If specified, sets number of elevation intervals. Can be used with `zmin` and `zmax`. Default is NULL.
- `zmin`: Minimum elevation for hypsometry. If not specified, minimum catchment elevation is used. Default is NULL.
ch_catchment_hyps

- **zmax**
  - Maximum elevation for hypsometry. If not specified, maximum catchment elevation is used. Default is NULL.

- **quantiles**
  - Vector of elevation quantiles. Default is NULL.

- **hypso_plot**
  - if TRUE the hypsometric curve is plotted. Default is NULL.

- **z_units**
  - Elevation units for plot. Default is ‘m’.

- **col**
  - Colour for plot. Default is ‘red’.

- **type**
  - Type of plot. Default is ‘o’ (lines with overplotted points).

- **xlab**
  - Plot x-axis label.

- **ylab**
  - Plot y-axis label.

- **add_grid**
  - If TRUE, a grid is added to the plot. Default is FALSE

- **...**
  - Other parameters for the graph

**Details**

The elevations may be passed as a vector of elevations, or of elevation quantiles, or as minimum and maximum elevations and the number of elevation intervals. A plot of the curve may also be created.

**Value**

Returns a data frame of elevations and catchment fractions below.

**Author(s)**

Dan Moore

**Examples**

```r
# Note: example not tested automatically as it is very slow to execute due to the downloading
library(raster)
library(magrittr)
# change the following line to specify a directory to hold the data
dir_name <- tempdir(check = FALSE)
# create directory to store data sets
if (!dir.exists(dir_name)) {
  dir.create(dir_name, recursive = TRUE)
}
# get 25-m dem
dem_fn <- file.path(dir_name, "gs_dem25.tif")
dem_upc <- ch_get_url_data(dem_url, dem_fn)
dem_upc

# get catchment boundaries
cb_fn <- file.path(dir_name, "gs_catchments.GeoJSON")
cb <- ch_get_url_data(cb_url, cb_fn)
```
# quick check plot - all catchments
raster::plot(dem_upc)
plot(cb, add = TRUE, col = NA)

# subset 240 catchment
cb_240 <- cb %>% dplyr::filter(wsc_name == "240")
plot(cb_240, col = NA)

## test function

# test different combinations of arguments
ch_catchment_hyps(cb_240, dem_upc, quantiles = seq(0, 1, 0.1))
ch_catchment_hyps(cb_240, dem_upc, z_levels = seq(1600, 2050, 50))
ch_catchment_hyps(cb_240, dem_upc, n_levels = 6)
ch_catchment_hyps(cb_240, dem_upc)
ch_catchment_hyps(cb_240, dem_upc, zmin = 1600, zmax = 2050)
ch_catchment_hyps(cb_240, dem_upc, zmin = 1600, zmax = 2050, n_levels = 6)

# generate a graph
ch_catchment_hyps(cb_240, dem_upc, hypso_plot = TRUE)
ch_catchment_hyps(cb_240, dem_upc, hypso_plot = TRUE, col = "blue", type = "l", ylim = c(1500, 2200))
ch_catchment_hyps(cb_240, dem_upc, hypso_plot = TRUE, add_grid = TRUE, quantiles = seq(0, 1, 0.1))
ch_catchment_hyps(cb_240, dem_upc, hypso_plot = TRUE, ylab = expression("z ("*10"^{-3} ~ "km")"))

# extract specific quantiles (e.g., median and 90%)
ch_catchment_hyps(cb_240, dem_upc, quantiles = c(0.5, 0.9))

---

### ch_checkcatchment: Check Catchments

**Description**

Generates a simple map to allow a visual assessment of the catchment boundaries relative to the elevation contours.

**Usage**

```r
ch_checkcatchment(
  dem,
  catchment,
  outlet,
  outlet_label = NULL,
  main_label = "",
  bbox_type = "catchment",
  channel_vec = NULL,
)```

cb_colour = "red",
pp_colour = "red",
channel_colour = "blue",
contour_colour = "grey",
plot_na = TRUE,
plot_scale = TRUE,
na_location = "tr",
scale_location = "bl"
)

Arguments

dem raster DEM that catchments were generated from.
catchment Catchment polygon (sf object).
outlet Location of catchment outlet (sf object).
outlet_label Character label for outlet.
main_label Main label for catchment plot.
bbox_type type of bounding box. If ‘catchment’, then the contours are bounded by the catchment, otherwise they are plotted to the extent of the DEM.
channel_vec Vectors of the channels will be plotted if specified.
channel_colour Colour for channel. Default is "blue".
pp_colour Colour for catchment pour points. Default is "red".
contour_colour Colour for contours Default is "grey".
plot_na If TRUE (the default) a north arrow is added to the plot.
plot_scale If TRUE (the default) a scale bar is added to the plot.
na_location Location for the north arrow. Default is ‘tr’, i.e. top-right.
scale_location Location for the scale bar. Default is ‘bl’, i.e. bottom-left.

Details
Also generates a table summarizing the catchments, including the coordinates of the outlet point and the catchment area.

Value
TRUE. A map of the catchments is also plotted and the catchment parameters are printed.

Author(s)
Dan Moore and Kevin Shook

See Also
ch_checkchannels
Examples

# Only proceed if Whitebox executable is installed
library(whitebox)
if (check_whitebox_binary()){
  library(raster)
  test_raster <- ch_volcano_raster()
  dem_raster_file <- tempfile(fileext = "tif")
  no_sink_raster_file <- tempfile("no_sinks", fileext = "tif")

  # write test raster to file
  writeRaster(test_raster, dem_raster_file, format = "GTiff")

  # remove sinks
  removed_sinks <- ch_wbt_removesinks(dem_raster_file, no_sink_raster_file, method = "fill")

  # get flow accumulations
  flow_acc_file <- tempfile("flow_acc", fileext = "tif")
  flow_acc <- ch_wbt_flow_accumulation(no_sink_raster_file, flow_acc_file)

  # get pour points
  pourpoint_file <- tempfile("volcano_pourpoints", fileext = "shp")
  pourpoints <- ch_volcano_pourpoints(pourpoint_file)
  snapped_pourpoint_file <- tempfile("snapped_pourpoints", fileext = "shp")
  snapped_pourpoints <- ch_wbt_pourpoints(pourpoints, flow_acc_file, pourpoint_file, snapped_pourpoint_file, snap_dist = 10)

  # get flow directions
  flow_dir_file <- tempfile("flow_dir", fileext = "tif")
  flow_dir <- ch_wbt_flow_direction(no_sink_raster_file, flow_dir_file)
  fn_catchment_ras <- tempfile("catchment", fileext = "tif")
  fn_catchment_vec <- tempfile("catchment", fileext = "shp")
  catchments <- ch_wbt_catchment(snapped_pourpoint_file, flow_dir_file, fn_catchment_ras, fn_catchment_vec)

  # check results
  ch_checkcatchment(test_raster, catchments, snapped_pourpoints)
} else {
  message("Examples not run as Whitebox executable not found")
}

---

ch_checkchannels  Check Channels

Description

Generates a map of the generated channel network layer.
Usage

`ch_checkchannels(
  dem,
  channels,
  outlet = NULL,
  main_label = "",
  channel_colour = "blue",
  pp_colour = "red",
  contour_colour = "grey"
)
``

Arguments

dem     raster DEM that catchments were generated from
channels  channel polyline (or channels list from `ch_wbt_channels`) (sf object)
outlet  location of catchment outlet (sf object)
main_label  Main label for channel plot.
channel_colour  Colour for channel. Default is "blue".
pp_colour  Colour for catchment pour points. Default is "red".
contour_colour  Colour for contours Default is "grey".

Details

Generates a simple map of the drainage network plotted over the contours to allow a visual assessment.

Value

check_map     a `ggplot` object of a map with channel layer

Author(s)

Dan Moore

See Also

`ch_checkcatchment`

Examples

# Only proceed if Whitebox executable is installed
library(whitebox)
if (check_whitebox_binary()){
  library(raster)
  test_raster <- ch_volcano_raster()
  dem_raster_file <- tempfile(fileext = c(".tif"))
  no_sink_raster_file <- tempfile("no_sinks", fileext = c(".tif"))
}
# write test raster to file
writeRaster(test_raster, dem_raster_file, format = "GTiff")

# remove sinks
removed_sinks <- ch_wbt_removesinks(dem_raster_file, no_sink_raster_file, method = "fill")

# get flow accumulations
flow_acc_file <- tempfile("flow_acc", fileext = c(".tif"))
flow_acc <- ch_wbt_flow_accumulation(no_sink_raster_file, flow_acc_file)

# get flow directions
flow_dir_file <- tempfile("flow_dir", fileext = c(".tif"))
flow_dir <- ch_wbt_flow_direction(no_sink_raster_file, flow_dir_file)
channel_raster_file <- tempfile("channels", fileext = c(".tif"))
channel_vector_file <- tempfile("channels", fileext = c(".shp"))
channels <- ch_wbt_channels(flow_acc_file, flow_dir_file, channel_raster_file, channel_vector_file, 1)

# get pour points
pourpoint_file <- tempfile("volcano_pourpoints", fileext = ".shp")
pourpoints <- ch_volcano_pourpoints(pourpoint_file)
snapped_pourpoint_file <- tempfile("snapped_pourpoints", fileext = ".shp")

ch_checkchannels(test_raster, channels, snapped_pourpoints)
}

} else {
  message("Examples not run as Whitebox executable not found")
}

---

**ch_circ_mean_reg**  
Calculates the circular mean, median, and regularity

**Description**

Calculate the circular mean, median, and regularity using a year of 365 days. Days of year are converted to degrees internally, results are returned as positive days of year

**Usage**

ch_circ_mean_reg(dataframe)

**Arguments**

- **dataframe**: a dataframe of day year of event; can be amax or pot.

**Value**

Returns a list of the following statistics

- **n**: number of samples
mean    circular mean of array
median  circular median of array
rho     regularity or mean resultant length

References

See Also
ch_sh_get_amax

Examples

```r
data(CAN05AA008)
am <- ch_sh_get_amax(CAN05AA008)
m_r <- ch_circ_mean_reg(am)
```

Refer to the 'ch_circular_stats' package documentation for more details.

### Description

Empties and removes a working directory.

### Usage

```r
ch_clear_wd(wd, do_check = TRUE)
```

### Arguments

- **wd**: working directory file path
- **do_check**: If TRUE, the default, the user is asked to confirm the deletion of the working directory. If TRUE, the directory is deleted without confirmation.

### Details

The data for raster layers read in as Whitebox files are held on disk rather than in memory.

### Value

- **result**: returns TRUE upon successful execution
Author(s)

Dan Moore

See Also

ch_create_wd to create working directory

Examples

# not tested as deleting all files in the directory cannot be tested in CRAN

# create an empty working directory
my_wd <- tempdir()
ch_create_wd(my_wd) # confirm creation

# clear the working directory
ch_clear_wd(my_wd)

---

ch_color_gradient ch_color_gradient

Description

set colour gradient

Usage

ch_color_gradient(
  x,
  colors = c("darkred", "red", "white", "green", "darkgreen"),
  colsteps = 100,
  climits = NULL
)

Arguments

  x array of variable
  colors an array of colours to form the desired gradient. Default is ("darkred", "red", "white", "green", "darkgreen")
  colsteps number of steps to be used in gradient, default is 100.
  climits provide specific limits for common scaling

Value

  vector of colors
ch_col_gradient

Author(s)

Paul Whitfield

Examples

```r
cxin <- c(0, 1, 1, 3, 4, 5, 10)
cxout <- ch_color_gradient(cxin)
# [1] "#8B0000" "#B50000" "#B50000" "#FF2B2B" "#FF9292"
# [6] "#FFF9F9" "#006400"
```

ch_col_gradient

*Creates a colour gradient*

Description

Creates a colour gradient for plotting.

Usage

```r
ch_col_gradient(
  x, 
  colors = c("darkred", "red", "white", "blue", "darkblue"),
  colsteps = 100,
  climits = NULL
)
```

Arguments

- **x** Vector of values used for gradient.
- **colors** Vector of colours to form a gradient. Default is `c("darkred", "red", "white", "blue", "darkblue")`.
- **colsteps** The number of steps in the gradient. Default is 100.
- **climits** Sets specific limits for common scaling.

Value

- **res** returned array of colour codes

Author(s)

modified by Paul Whitfield
Examples

plot(rnorm(20), col='black')

# create a red blue colour gradient for plotting
mycol <- ch_col_gradient(rnorm(20), colsteps = 100)

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

ch_col_transparent      Add Transparency to plot colours

Description

Adds transparency to a colour based on an integer between 0 and 255, with 0 being fully transparent and 255 being opaque. Based on function rvn_col_transparent in package RavenR.

Usage

ch_col_transparent(colour, trans)

Arguments

colour    colour that is to be made transparent, or an array of colours
trans     an integer (or array of integers) describing the degree of transparency, 0 to 255. Must be the same length as colour. Values < 10 (very transparent), values > 200 (solid colour).

Value

res    returned updated colour code with transparency

Author(s)

Rob Chlumsky; Paul Whitfield

See Also

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

Examples

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

# create a transparent blue colour for plotting
mycol <- ch_col_transparent('blue', 100)
```r
# plot more random points in transparent blue colour
points(rnorm(20), col = mycol)

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

# create two transparent colour for plotting
mycol <- ch_col_transparent(c('green', 'red'), c(100, 200))

# plot more random points in transparent colours
points(rnorm(20), col = mycol[2])
```

---

**ch_contours**

*Create Contours*

**Description**

Creates contour lines from a DEM.

**Usage**

```r
ch_contours(dem, zmin = NULL, zmax = NULL, n_levels = 10, z_levels = NULL)
```

**Arguments**

- `dem` Raster object of your dem in the desired projection (note: should have had sinks removed).
- `zmin` Minimum elevation value for contours. If not specified, minimum value `dem` is used.
- `zmax` Maximum elevation value for contours. If not specified, maximum value `dem` is used.
- `n_levels` Number of contour lines. Default is 10.
- `z_levels` Levels at which to plot contours. If specified, overrides `zmin`, `zmax` and `n_levels`.

**Details**

Generates contour lines from a DEM, which are returned as an sf object. The user can either provide a vector of elevation values by specifying the `z_levels` argument, or by supplying the minimum and maximum elevations (`zmin` and `zmax`) and the number of contour lines (`n_levels`).

**Value**

- `contours_sf` sf object containing contours
ch_create_wd

Author(s)
Dan Moore

Examples
# use volcano DEM
dem <- ch_volcano_raster()
# generate contours
contours <- ch_contours(dem)

# plot contours map
plot(contours)

ch_create_wd  Create working directory

Description
Creates a working directory.

Usage
ch_create_wd(wd)

Arguments
wd  name of a directory in which to store files created by WhiteboxTools functions

Value
TRUE  returns TRUE upon successful execution

Author(s)
Dan Moore

See Also
ch_clear_wd to clear the working directory

Examples
# not tested automatically as will return a warning
ch_create_wd(tempdir())
ch_cut_block

Extracts a specified time period from a longer record

Description
The function could also be used to get the same period of time from several station for comparison.

Usage
ch_cut_block(DF, st_date, end_date)

Arguments
- **DF**: A daily streamflow data frame as from ch_read_ECDE_flows
- **st_date**: starting date format is %Y/%m/%d
- **end_date**: ending date format is %Y/%m/%d

Value
Returns a portion of the original dataframe.

Author(s)
Paul Whitfield

Examples
```r
data(CAN05AA008)
subset <- ch_cut_block(CAN05AA008, "2000/01/01", "2010/12/31")
```

ch_date_subset

Subsets dates by string

Description
Subsets a data frame by a specified date range, provided as a string by the prd argument. This function is meant to emulate the subsetting capability of the xts package.

Usage
ch_date_subset(df, prd)

Arguments
- **df**: data frame of time series data; includes a variable called Date
- **prd**: date range as string formatted as ‘YYYY-MM-DD/YYYY-MM-DD’
**ch_decades_plot**

Plots output from ch_binned_MannWhitney for decades

Description

Creates a simple plot comparing two decades from the output of ch_binned_MannWhitney.

Usage

`ch_decades_plot(mplot)`

Arguments

- `mplot` List output by the function ch_binned_MannWhitney

Value

A standard R graphic is created.

Author(s)

Paul Whitfield

See Also

ch_decades_plot
ch_doys

Examples

range1 <- c(1970, 1979)
range2 <- c(1990, 1999)
b_MW <- ch_binned_MannWhitney(CAN05AA008, step = 5, range1, range2, ptest = 0.05)
ch_decades_plot(b_MW)

ch_doys

Days of year and water year

Description

Converts an array of dates into a dataframe with date, year, month, doy, wyear, dowy.
The day of water year is computed from the first of the specified water year month.

Usage

ch_doys(Date, water_yr = 10)

Arguments

Date an array of R dates, as produced by as.Date()
water_yr the month starting the water year, default is 10 (October). If a value of 1 is specified, the 10 will be used.

Details

Converts a date array into a data frame with years, wateryears, and days of year and of water year.

Value

Returns a dataframe with date information:
Date in Date format
year numeric calendar year
month number calendar month
doy numeric day of year
wyear numeric water year starting on day 1 of selected month
dwy numeric day of water year

Author(s)

Paul Whitfield, Kevin Shook

Examples

dd <- seq.Date(as.Date("2010-01-01"), as.Date("2018-01-01"), by = 1)
output <- ch_doys(dd, water_yr=10)
head(output)
Description

A flow duration curve is a plot of flow magnitude against exceedance probability. The plot may contain the Gustard Curves (default) or they can be omitted. The default is for curves to be plotted against probability, but an option is to plot against the normalized exceedance probability. In that case, the x axis represents a normal distribution.

Usage

```
ch_fdcurve(DF, normal = FALSE, gust = TRUE, metadata = NULL)
```

Arguments

- `DF`: a dataframe of daily flows from `ch_read_ECDE_flows`
- `normal`: If `normal = TRUE` then exceedance probability is normalized. Default is FALSE.
- `gust`: If TRUE (the default), adds the curves from Gustard et al. 1992 are added.
- `metadata`: dataframe of metadata, defaults to `HYDAT_list`.

Details

Create a Flow Duration Curve based upon Observations.

Value

Plots the flow duration curve and returns a data frame containing:

- exceedance probability
- probability
- flow
- d=flow values

Author(s)

Paul Whitfield

References


Examples

```r
data(HYDAT_list)
data(CAN05AA008)
# plot with Gustard 1992 curves
test <- ch_fdcurve(CAN05AA008, normal = FALSE, gust = TRUE)
# plot with normalized exceedance probability
test <- ch_fdcurve(CAN05AA008, normal = TRUE, gust = FALSE)
```

ch_flow_raster

**Raster plot of daily streamflows**

Description

Produces a raster plot: years by day of year, showing magnitude of flow. This produces a plot showing the flow data in colours, showing different context than in a hydrograph. High flows are in warm colours.

Usage

```r
ch_flow_raster(
  DF,
  rastercolours = c("lightblue", "cyan", "blue", "slateblue", "orange", "red"),
  metadata = NULL
)
```

Arguments

- **DF** A data frame of daily flow data as read by ch_read_ECDE_flows.
- **rastercolours** A vector of colours used for flow magnitudes (default c("lightblue", "cyan", "blue", "slateblue", "orange", "red").
- **metadata** A dataframe of station metadata, defaults to HYDAT_list.

Value

No value is returned; a standard R graphic is created.

Author(s)

Paul Whitfield

See Also

ch_read_ECDE_flows
ch_flow_raster_trend ch_flow_raster_qa
ch_flow_raster_qa

Examples

ch_flow_raster(CAN05AA008)

Description

Raster plot with WSC quality flags. This produces a plot showing the flow data in grayscale overlain by the Water Survey of Canada quality flags. Colours are consistent with ECDataExplorer. Raster layout lets the use see the flags in a different context than in a hydrograph.

Usage

ch_flow_raster_qa(DF, metadata = NULL)

Arguments

DF   dataframe of daily streamflow read by ch_read_ECDE_flows
metadata   dataframe of metadata or defaults to "HYDAT_list"

Value

Produces a raster plot: years against day of year, showing the data flags:

A   (Partial) in green
B   (Backwater) in cyan
D   (Dry) in yellow
E   (Estimated) in red

Returns TRUE if executed properly; a standard R graphic is created.

Author(s)

Paul Whitfield

See Also

ch_read_ECDE_flows
ch_flow_raster_trend ch_flow_raster

Examples

data(HYDAT_list)
data(CAN05AA008)
qaplot <- ch_flow_raster_qa(CAN05AA008)
**ch_flow_raster_trend**  
*Raster plot and simple trends of observed streamflows by periods*

**Description**

Creates a raster plot plus trend plots for day of year, which are binned by a number of days (step), and the max, min, and median annual discharge across years. The plot contains four panels based upon binned data.

**Usage**

```r
ch_flow_raster_trend(
  DF,
  step = 5,
  missing = FALSE,
  metadata = NULL,
  colours = c("lightblue", "cyan", "blue", "slateblue", "darkblue", "red")
)
```

**Arguments**

- **DF** - dataframe of daily flow data as read by ch_read_ECDE_flows
- **step** - a number indicating the degree of smoothing eg. 1, 5, 11.
- **missing** If FALSE years with missing data are excluded. If TRUE partial years are included.
- **metadata** a dataframe of station metadata, default is HYDAT_list.
- **colours** A vector of colours used for the raster plot. The default is c("lightblue", "cyan", "blue", "slateblue", "darkblue", "red").

**Details**

The four plots are: (1) The maximum, minimum, and median flow with a trend test for each period: red arrows indicate decreases, blue arrows indicate increases. (2) The scale bar for the colours used in the raster plot, (3) The raster plot with a colour for each period and each year where data exist, and (4) A time series plot of the minimum, median, and maximum annual bin values. If there is no trend (p > 0.05) the points are black. Decreasing trends are in red, increasing trends are in blue.

**Value**

Returns a list containing:

- **stationID** Station ID eg. 05BB001
- **missing** How missing values were used FALSE = used, TRUE = removed
- **step** number of days in a bin
- **periods** number of periods in a year
- **period** period numbers i.e. 1:365/step
bins  values for each period in each year
med_period  median for each period
max_period  maximum for each period
min_period  minimum for each period
tau_period  Kendalls Tau for each period
prob_period  probability of Tau for each period
year  years spanning the data
median_year  median bin for each year
max_year  maximum bin for each year
min_year  minimum bin for each year
tau_median_year  value of tau and probability for annual median
tau_maximum_year  value of tau and probability for annual maximum
tau_minimum_year  value of tau and probability for annual minimum

Author(s)
Paul Whitfield

References

See Also
ch_flow_raster

Examples

data(CAN05AA008)
mplot <- ch_flow_raster_trend(CAN05AA008, step=5)
ch_get_ECDE_metadata

**Description**

Reads the file that is generated from ECDE 'save favourite stations' to capture the ECDE metadata. The dataframe returned contains 20 fields from ECDE.

**Usage**

ch_get_ECDE_metadata(filename, writefile = NULL)

**Arguments**

- **filename**: The name of the ECDE file, 'FavHydatStations.tb0'.
- **writefile**: Default is NULL, but if it is a filename e.g. 'filename.csv' then the dataframe is saved to a csv file.

**Value**

Returns a dataframe consisting of:

- **Station**
- **StationID**
- **StationName**
- **Station Name**
- **HYDStatus**
- **Active or Discontinued**
- **Prov**
- **Province**
- **Latitude**
- **Longitude**
- **DrainageArea**
- **km²**
- **Years**
- **Number of years with data**
- **From**
- **Start Year**
- **To**
- **End Year**
- **Reg.**
- **Regulated?**
- **Flow**
- **If TRUE/Yes flow data exists**
- **Level**
- **If TRUE/Yes level data exists**
- **Sed**
- **If TRUE/Yes sediment data exists**
- **OperSched**
- **Operations current - Continuous or Seasonal**
- **RealTime**
- **If TRUE/Yes real time data is available**
- **RHBN**
- **If TRUE/Yes the stations is in the reference hydrologic basin network**
- **Region**
- **Name of regional office operating station**
- **Datum**
- **Elevation datum**
- **Operator**
- **Operator or provider of the data**
ch_get_peaks

Extracts peak flows over a threshold

Description

This function is development code being shared as is. It is expected that the user will be interested in the data frame returned for POT analysis and for plotting (i.e. ch_booth_plot).

This function retrieves peaks greater than or equal to the prescribed threshold. It returns a data frame of peak characteristics suitable for subsequent analysis.

The portion under development is returns a list of the flows during an event with the values of the four preceding days and three subsequent days. If the peak is a single point the fragment is nine points long; if the events is longer the fragment contains all days above the threshold and eight additional days.

Usage

ch_get_peaks(dataframe, threshold)

Arguments

dataframe : a data frame of streamflow data containing columns named ‘Date’ and ‘Flow’
threshold : a value for the threshold. Values above the threshold are tested for peaks.

Value

Returns a list containing:

POTevents : a dataframe containing details of the events
events : a vector with the value 0 when the flow is below the threshold and 1 when above.
event_num : a vector with the value 0 when the flow is below a threshold or the index of the events when the threshold was exceeded. i.e. 1,2,3, etc
st_date : start date of events
case : a list of the daily flows in each individual event (see details for more information)

Examples

## Not run:
# Don't run this example as it requires an ECDE file
filename <- "FavHydatStations.tb0"       # dummy file name (not supplied)
meta0 <- ch_get_ECDE_metadata(filename)
meta1 <- ch_get_ECDE_metadata(filename, writefile="study52_metadata.csv")

## End(Not run)
The POTevents data frame contains five columns:

- **st_date**: starting date of event
- **max_date**: date of maximum in the event
- **max**: maximum discharge during event
- **volume**: flow volume during the event
- **duration**: length of the event in days

The case list contains the flows during an event and also for four preceding and subsequent days. Each event will have a length between nine to n days in length. Note: in rare cases where the event is in progress when data becomes available the event might be shorter than nine days long.

**Author(s)**

Paul Whitfield

**References**


**See Also**

- ch_booth_plot

**Examples**

```r
CAN05AA008 <- CAN05AA008
treshold <- 0.5*max(CAN05AA008$Flow)  # arbitrary threshold
my_peaks <- ch_get_peaks(CAN05AA008, threshold)
str(my_peaks)
```

**ch_get_url_data**

*Gets remote data sets*

**Description**

Accesses data sets, via a url the first time, saves them locally, then accesses them locally after the first time the script is executed.

**Usage**

```r
ch_get_url_data(gd_url, gd_filename, quiet = FALSE)
```
ch_get_url_data

Arguments

- **gd_url**: url for accessing data set
- **gd_filename**: name of file on local drive, including full path
- **quiet**: Optional. If FALSE (the default) error/warning messages are printed if the data cannot be found.

Value

Returns a data frame (from a .csv file), a raster object (from a .tif file), or an sf object (from a GeoJSON file).

Author(s)

Dan Moore

Examples

# Example not tested automatically as multiple large data files are downloaded which is slow

# Tested using files in the Upper Penticton Creek
# zenodo repository https://zenodo.org/record/4781469
library(ggplot2)
library(raster)

# create directory to store data sets
dir_name <- tempdir(check = FALSE)
if (!dir.exists(dir_name)) {
  dir.create(dir_name)
}

# test with soil moisture data in csv format
sm_fn <- file.path(dir_name, "sm_data.csv")
sm_url <- "https://zenodo.org/record/4781469/files/sm_data.csv"
sm_data <- ch_get_url_data(sm_url, sm_fn)
head(sm_data)

# test with tif/tiff file containing a dem
ra_fn <- file.path(dir_name, "gs_dem25.tif")
ra_data <- ch_get_url_data(ra_url, ra_fn)
plot(ra_data)

# test with GeoJSON
gs_fn <- file.path(dir_name, "gs_soilmaps.GeoJSON")
gs_data <- ch_get_url_data(gs_url, gs_fn)
ggplot(gs_data) +
  geom_sf(aes(fill = new_key)) +
  labs(fill = "Soil class",
       x = "UTM Easting (m)",
       y = "UTM Northing (m)"
)
ch_get_wscstation

```
y = "UTM Northing (m)" +
coord_sf(datum = 32611) +
theme_bw()
```

---

**ch_get_wscstation**  
*Reads station information from a data file produced by ECDE*

---

**Description**

Retrieves station information for an individual Water Survey of Canada site, based on stationID; adds a text string at position 21 that combines key elements for a title.

**Usage**

```
ch_get_wscstation(stnID, metadata = NULL)
```

**Arguments**

- **stnID**: A Water Survey of Canada station number
- **metadata**: a data frame of station information from ECDataExplorer. The data frame ‘HYDAT_list’ is supplied with this package.

**Value**

Returns a line from a data frame with 21 variables

- **Station**  
  StationID
- **StationName**  
  Station Name
- **HYDStatus**  
  Active or Discontinued
- **Prov**  
  Province
- **Latitude**
- **Longitude**
- **DrainageArea**: Area in km²
- **Years**: # of years with data
- **From**: Start Year
- **To**: End Year
- **Reg.**: Regulated or natural
- **Flow**: if TRUE/Yes flow data is available
- **Level**: if TRUE/Yes water level data is available
- **Sed**: if TRUE/Yes sediment data is available
- **OperSched**: Current operation schedule- Continuous or Seasonal
- **RealTime**: if TRUE/Yes real itme data exists
RHBN if TRUE/Yes is in the reference hydrologic basin network
Region WSC Region
Datum Datum used
Operator Agency responsible for collecting data
Station_lname Added field combining StationID, StationName, Province and if station is RHBN an * is added

Author(s)
Paul Whitfield

Examples

data("HYDAT_list")
s_info <- ch_get_wscstation("0580001", metadata = HYDAT_list)
title <- s_info[21]
print(title)

ch_hydrograph_plot Hydrograph plot

Description
Creates a hydrograph plot for simulated, observed, and inflow hydrograph series, including precipitation if provided. The secondary y axis will be used to plot the precip time series.

Usage

ch_hydrograph_plot(
  flows = NULL,
  precip = NULL,
  prd = NULL,
  winter_shading = FALSE,
  winter_colour = "cyan",
  range_mult_flow = NULL,
  range_mult_precip = 1.5,
  flow_labels = NULL,
  ylabel = NULL,
  precip_label = "Precipitation [mm]",
  leg_pos = NULL,
  leg_box = NULL,
  zero_axis = TRUE
)
ch_hydrograph_plot

Arguments

flows  
data frame of flows to plot
precip  
data frame of precipitation values to plot
prd  
period to use in plotting
winter_shading  
optionally adds a transparent cyan shading for the December 1st to March 31st period in each year that is plotted. Default is FALSE.
winter_colour  
colour to use in winter shading polygons
range_mult_flow  
range multiplier for max value in hydrograph. This is useful in preventing overlap if precip is also plotted. This value should not be less than 1.0, otherwise the values will be cutoff in the plot.
range_mult_precip  
range multiplier for max value in precipitation plot (default 1.5)
flow_labels  
string vector of labels for flow values
ylabel  
text label for y-axis of the plot (default 'Flow [m^3/s]')
precip_label  
text label for precipitation y-axis (default 'Precipitation [mm]')
leg_pos  
string specifying legend placement on plot e.g. 'topleft', 'right', etc., and is consistent with the legend function options. If NULL, the function will place the legend left, if precip added, on the topleft otherwise).
leg_box  
boolean on whether to put legend in an opaque white box or not. If NULL (the default), the function will automatically not use a white box and leave the background of the legend transparent.
zero_axis  
fixes the y axis to start exactly at zero (default TRUE). By default, R will plot the values with a small buffer for presentation. Be warned that if this option is set to TRUE, the minimum value is set to zero without checking if any flow values are less than zero. This option should not be used for reservoir stage plotting, since most reservoir stage is typically reported as an elevation.

Details

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. The supplied time series should be in xts format. Note that a plot title is purposely omitted in order to allow the automatic generation of plot titles.

Value

Returns TRUE if the function is executed properly.

Author(s)

Robert Chlumsky
Examples

# example with synthetic random data
dd <- seq.Date(as.Date("2010-10-01"), as.Date("2013-09-30"), by = 1)
x <- abs(rnorm(length(dd)))
y <- abs(rnorm(length(dd))) * x
df <- data.frame("Date" = dd, x, y)
myprd <- "2011-10-01/2012-09-30"

precip <- data.frame("Date" = dd, "precip" = abs(rnorm(length(dd))) * 10)

# basic hydrograph plot
ch_hydrograph_plot(flows = df, winter_shading = FALSE)

# with different labels and winter shading
ch_hydrograph_plot(flows = df, winter_shading = TRUE,
                   flow_labels = c("simulated", "observed"))

# add precipitation, increase the plot ranges to separate flows and precip, and add a legend box
ch_hydrograph_plot(flows = df, precip = precip, range_mult_flow = 1.7,
                   range_mult_precip = 2, leg_box = TRUE)

ch_polar_plot

Polar plot of daily streamflows

Description

Produces a polar plot similar to that used in Whitfield and Cannon, 2000. It uses output from the function ch_binned_MannWhitney or a data structure created using the function ch_polar_plot_prep.

Usage

ch_polar_plot(
  bmw,
  lcol1 = c("black", "gray50"),
  lcol2 = c("black", "gray50"),
  lfill = c("yellow", "green"),
  lsig = c("red", "blue")
)

Arguments

bmw output from ch_binned_MannWhitney
lcol1 line colour, default is c("black", "gray50")
lcol2 point colour, default is c("black", "gray50")
lfill fill colour, default is c("yellow", "green")
lsig significance symbol colour, default is c("red", "blue")
ch_polar_plot_peaks

Value

No value is returned; a standard R graphic is created.

Author(s)

Paul Whitfield

References


See Also

ch_binned_MannWhitney ch_polar_plot_prep

Examples

```r
range1 <- c(1970,1979)
range2 <- c(1990,1999)
b_MW <- ch_binned_MannWhitney(CAN05AA008, step = 5, range1, range2, ptest <- 0.05)
ch_polar_plot(b_MW)
```

ch_polar_plot_peaks  Polar / circular plots of peak flows

Description

Polar / circular plots of peak flows. Creates a polar plot of flow peaks in one of several different forms. Basic plot has shading for nival and pluvial centroids.

Usage

```r
ch_polar_plot_peaks(
  title = NA,
  direction = NULL,
  regularity = NULL,
  days = NULL,
  shading = FALSE,
  shade = 35,
  pt_col = "darkblue",
  in_pch = NULL,
  in_cex = NULL,
  in_col = NULL,
  ```
in_detail = NULL,
labels = NULL,
label_pos = NULL,
out_pch = 16,
out_cex = 0.8,
...)

Arguments

title a title to be added to the plot
direction a value or array of mean/median direction, circular mean or median of points
from ch_circ_mean_reg (optional)
regularity a value or array of regularity from ch_circ_mean_reg (optional).
days an array of days of year to be plotted on perimeter (optional).
shading if TRUE adds shading and labels for nival and pluvial regimes default = FALSE
shade percentage of shading, default is 35.
pt_col colour used for points for events. default = "darkblue". If pt_col is an array it is
used to colour the individual points of days
in_pch a value or an array of symbols to be used for centroids. To be in color, must be
one of 21 to 25 to get a symbol with border, elsewise a red symbol is plotted.
in_cex an array of symbol sizes
in_col an array of colors, either numbers or names to apply to centroid points (optional,
default is "red")
[4] size
labels an array of labels to be placed beside points with direction and regularity (optional)
label_pos an array of positions indicating when label be placed (1, 2, 3, or 4 - below, left,
above, right)(optional - default is below)
out_pch symbols for points on outside of circle
out_cex point size for symbol
... other plot options

Value

Creates a circular plot of peak flows.

Note

points inside the plot
in_pch, in_col, and in_cex will normally be of the same length and that would be the maximum
index of in_detail
points on the outside
Author(s)
Paul Whitfield

References

Examples

```r
# base plot
ch_polar_plot_peaks()

# base plot with area shading
ch_polar_plot_peaks(shading = TRUE)

# plot of annual maximum series
data(CAN05AA008)
am <- ch_sh_get_amax(CAN05AA008)
ch_polar_plot_peaks(days = am$doy, title = "05AA008")

# remove partial years
am <- am[am$days >= 365,]
ch_polar_plot_peaks(days = am$doy, title = "05AA008")

# plot the centroid
m_r <- ch_circ_mean_reg(am)
ch_polar_plot_peaks(direction = m_r$mean, regularity = m_r$regularity, title = "05AA008")

# plot peaks and centroid
ch_polar_plot_peaks(days = am$doy, direction = m_r$mean, regularity = m_r$regularity, title = "05AA008")
```

ch_polar_plot_prep

Creates a data structure to be passed to ch_polar_plot

Description
Could be used to move data from a different type of analysis different to the ch_binned_MannWhitney function which uses flows. The two series need to be of the same length and their length is related to the step size. For examples, for five day periods there will be 73 periods.
ch_polar_plot_prep

Usage

ch_polar_plot_prep(
    station,
    plot_title,
    step,
    x0,
    x1,
    stat,
    prob,
    test_s,
    variable = "discharge",
    bin_method = "unstated",
    test_method = "unstated",
    lline1 = "Period 1",
    lline2 = "Period 2",
    pvalue = 0.05
)  

Arguments

station  
Typically a station number

plot_title  
Polar plot title - usually a station name

step  
The number of days binned

x0  
Time series of length n for a single seasonal cycle

x1  
Time series of length n for a single seasonal cycle

stat  
Time series of length n for statistical test value for each bin

prob  
Time series of length n of probability of test value

test_s  
Vector with values of -1, 0, 1 for significance, -1 negative, 1 positive, 0 not significant

variable  
Name of variable plotted. Default is 'discharge'

bin_method  
Default is 'unstated'

test_method  
Default is 'unstated'

lline1  
Names of first period, default is 'Period 1'

lline2  
Names of second period, default is 'Period 2'

pvalue  
Value of p used. Default is 0.05

Value

Returns a list containing:

StationID  
ID of station

Station_lname  
Name of station

variable  
Name of variable

bin_width  
Smoothing time step in days
**ch_qa_hydrograph**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>range1</td>
<td>First range of years</td>
</tr>
<tr>
<td>range2</td>
<td>Second range of years</td>
</tr>
<tr>
<td>p_used</td>
<td>p_value</td>
</tr>
<tr>
<td>fail</td>
<td>TRUE if test failed due to missing values</td>
</tr>
<tr>
<td>bin_method</td>
<td>Method used for binning</td>
</tr>
<tr>
<td>test_method</td>
<td>Mann-Whitney U</td>
</tr>
<tr>
<td>series</td>
<td>A data frame containing six columns</td>
</tr>
<tr>
<td>period</td>
<td>period numbers i.e. 1:365/step</td>
</tr>
<tr>
<td>period1</td>
<td>median values for each bin in period 1</td>
</tr>
<tr>
<td>period2</td>
<td>median values for each bin in period 2</td>
</tr>
<tr>
<td>mwu</td>
<td>Mann Whitney U-statistic for each bin between the two periods</td>
</tr>
<tr>
<td>prob</td>
<td>probability of U for each period</td>
</tr>
<tr>
<td>code</td>
<td>significance codes for each bin</td>
</tr>
</tbody>
</table>

The series data frame contains

**Author(s)**

Paul Whitfield

**References**


**See Also**

ch_binned_MannWhitney ch_polar_plot

---

**ch_qa_hydrograph**  
**Plots a hydrograph with the data quality symbols and returns a report on qa symbols and missing data.**

**Description**

Plots a hydrograph of a WSC daily data file read from from ECDataExplorer (ECDE). The hydrograph shows individual days with data quality symbols [SYM] in colour and counts cases of each and reports them in the legend. The colours and symbols are those produced by ECDataExplorer.

There is an option to provide start and end dates to show only part of the time period for which data exists and the plot is annotated to indicate this. Counts of missing observations is also provided in the legend.
ch_qa_hydrograph

Usage

ch_qa_hydrograph(
  DF,
  st_date = NULL,
  end_date = NULL,
  cts = TRUE,
  rescale = FALSE,
  sym_col = c("black", "green", "cyan", "yellow", "red", "white"),
  metadata = NULL
)

Arguments

DF          Data frame retrieved from ECDataExplorer as returned by the function ch_read_ECDE_flows.
st_date     Optional start date in the form 'yyyy-mm-dd'. Default is NULL.
end_date    Optional end date in the form 'yyyy-mm-dd'. Default is NULL.
cts         If TRUE (the default) shows the counts of SYM in the legend. If FALSE the counts are omitted as in ECDE.
rescale     If FALSE (the default), the y-axis scaling is determined by the time period. If TRUE then determined by the whole dataset.
sym_col     Colours used for SYM; default is those used in ECDE ("black", "green", "cyan", "yellow", "red", "white"). The final "white" can be changed to highlight missing data points.
metadata    a dataframe of station metadata, default is HYDAT_list.

Value

Produces a plot and returns a list that contains:

  station name or title used

  st_date        starting date
  end_date       ending date
  n              the number of data points
  sym_count      summary of the SYM counts
  missing        number of missing data

Author(s)

Paul Whitfield

Examples

m_test <- ch_qa_hydrograph(CAN05AA008)
m_test <- ch_qa_hydrograph(CAN05AA008, st_date="1980-01-01", end_date="1999-12-31")
ch_read_AHCCD_daily

Reads AHCCD daily file

Description
This program reads an Adjusted and Homogenized Canadian Climate Data (AHCCD) of daily precipitation or temperatures. The values are arranged as month x day, which makes them difficult to read using standard R functions.

Usage
ch_read_AHCCD_daily(daily_file)

Arguments
daily_file Required. Name of the file to be read.

Value
If successful, returns the values in a data frame, consisting of the date, the value and the data code.

Author(s)
Kevin Shook

References

See Also
ch_read_AHCCD_monthly

Examples
```r
## Not run:
# Don't run this example as it requires a file, and use of the dummy
# file will cause an error message

stoon_daily_tmax <- ch_read_AHCCD_daily("dx40657120.txt")
## End(Not run)
```
**ch_read_AHCCD_monthly**  
Reads AHCCD monthly file

**Description**

This program reads an Adjusted and Homogenized Canadian Climate Data (AHCCD) data of precipitation or temperatures. The values are arranged as year x month, which makes them difficult to read using standard R functions.

**Usage**

```r
ch_read_AHCCD_monthly(monthly_file = NULL)
```

**Arguments**

- `monthly_file`  
  Required. Name of the file to be read.

**Value**

If successful, returns the values in a dataframe, consisting of the year, the month, the value and the data code.

**Author(s)**

Kevin Shook

**References**


**See Also**

`ch_read_AHCCD_daily`

**Examples**

```r
## Not run:
# Don't run these examples as use of the dummy files will cause error messages
Stoon_monthly_precip <- ch_read_AHCCD_monthly("mt4057120.txt")
NB_monthly_tmean <- ch_read_AHCCD_monthly("mm4045695.txt")
## End(Not run)
```
ch_read_ECDE_flows

Reads a file of WSC daily flows from ECDataExplorer (ECDE)

Description
Reads in a file WSC daily flows as returned from the Windows program ECDataExplorer, converts the Date, and omits the last 3 lines as these contain the data disclaimer and not data. The function can read values from a url.

Usage
ch_read_ECDE_flows(filename)

Arguments
filename Datafile retrieved from ECDataExplorer.

Value
Returns a dataframe with the last three rows removed:

<table>
<thead>
<tr>
<th>ID</th>
<th>stationID</th>
</tr>
</thead>
<tbody>
<tr>
<td>PARAM</td>
<td>Parameter 1 for Flow 2 for Level</td>
</tr>
<tr>
<td>Date</td>
<td>original character string converted to date format</td>
</tr>
<tr>
<td>Flow</td>
<td>Daily mean flow m³/sec</td>
</tr>
<tr>
<td>SYM</td>
<td>Quality flag</td>
</tr>
</tbody>
</table>

Author(s)
Paul Whitfield

Examples
## Not run:
# Not run as requires a file returned by the Windows program ECDataExplorer
# Using a dummy file name as an example
mfile <- "04JD005_Daily_Flow_ts.csv"
mdata <- ch_read_ECDE_flows(mfile)
## End(Not run)

# Not tested automatically as it is slow to read from a url
url1 <- "https://zenodo.org/record/7007830/files/08NL007_Daily_Flow_ts.csv"
values <- ch_read_ECDE_flows(url1)
**ch_regime_plot**

*Plots the regime of daily streamflows using quantiles*

**Description**

Produces a regime hydrograph similar to that in the reference. It shows the flow quantiles for each day of the year and the maximum and minimum. Parameters can be set to change colours and set the y-scale to allow plots of same scale to be produced.

**Usage**

```r
ch_regime_plot(
  DF,
  wyear = 1,
  colour = TRUE,
  mx = 1,
  metadata = NULL,
  quant = c(0.95, 0.9, 0.75, 0.5, 0.25, 0.1, 0.05)
)
```

**Arguments**

- **DF**: data frame of daily flow data
- **wyear**: set `wyear = 10` for October, `water.year = 1` for calendar year, can be any month
- **colour**: if `TRUE` plot is in colour, if `FALSE` plot is grayscale.
- **mx**: set the maximum y value; if `1` then maximum value of the flows is used to set
- **metadata**: a data frame of metadata, defaults to `HYDAT_list`. the y-axis value. The value of `mx` can be specified to produce a series of plots with the same scale.
- **quant**: quantiles; default is `quant = c(0.95, 0.9, 0.75, 0.5, 0.25, 0.1, 0.05)`. Can be changed but the length must be `7` and the `4th` value must be `0.5` (median)

**Value**

No value is returned; a standard `R` graphic is created.

**Author(s)**

Paul Whitfield

**References**

**ch_rfa_distseason**

**Examples**

```r
data(CAN05AA008)
ch_regime_plot(CAN05AA008, colour = TRUE, wyear = 1)
```

---

**ch_rfa_distseason**  
*Distance in seasonal space*

**Description**

Calculates a matrix of distances between points in the seasonal space that characterizes timing and regularity. It is equivalent to Euclidean distance applied to regularity (radius) and timing (angle) separately.

**Usage**

```r
ch_rfa_distseason(x, ...)
```

- **S3 method for class 'numeric'**
  ```r
  ch_rfa_distseason(x, a, w = 1/pi, ...)
  ```

- **S3 method for class 'data.frame'**
  ```r
  ch_rfa_distseason(x, w = 1/pi, ...)
  ```

- **S3 method for class 'formula'**
  ```r
  ch_rfa_distseason(form, x, w = 1/pi, ...)
  ```

**Arguments**

- **x**, **a**  
  Coordinates in the seasonal space. Can be a data.frame or vectors with radius `x` and angle `a`.

- **...**  
  Other parameters.

- **w**  
  Weight to favor angle over radius. By default it is `1/pi`, which bring angle in the interval `[0,1]`.

- **form**  
  Formula and dataset providing the coordinates of the seasonal space. Must be of the form `radius ~ angle`.

**Value**

Returns a matrix of distances between points in the seasonal space that characterizes timing and regularity.

**Author(s)**

Martin Durocher
References


See Also

ch_rfa_seasonstat

Examples

scoord <- data.frame(radius = runif(5),
                     angle = runif(5,0,2*pi))

ch_rfa_distseason(radius ~ angle , scoord)

ch_rfa_extractamax Extracts the annual maxima of a daily time series

Description

Extracts the annual maxima of a daily time series

Usage

ch_rfa_extractamax(x, ...)

## S3 method for class 'formula'
ch_rfa_extractamax(form, x, tol = 0, ...)

## Default S3 method:
ch_rfa_extractamax(x, tol = 0, nlab = "n", ylab = "yy", ...)

Arguments

x             Data. If no formula is passed, the first column must be the value and the second the date.
...
form          Formula of the form value ~ date that specifies the variable from which the annual maximums are extracted and a date variable.
tol           Filter the years having less than tol days.
nlab, ylab    Names for the added columns representing respectively the number of yearly observations and the year. If set to NULL the given column is not added.
Ch_rfa_julianplot

Value

Returns a data frame containing the annual (Monthly) maxima, the date and the number of observations during the year.

Author(s)

Martin Durocher

Examples

out <- ch_rfa_extractamax(flow ~ date, CAN01AD002, tol = 350)
head(out)

ch_rfa_julianplot

Circular plotting by day of year

Description

Create axis for plotting circular statistics in a unitary circle.

Usage

ch_rfa_julianplot(
  rose.col = "gray40",
  rose.lwd = 1.5,
  rose.cex = 1.5,
  rose.radius = seq(0.25, 1, 0.25),
  ...
)

Arguments

rose.col, rose.lwd, rose.cex
  Properties of the polar axes.
rose.radius
  Vector of the position of the circular axis.
...
  Other parameter passed to points.

Value

Returns a empty rose plot by day of year

Author(s)

Martin Durocher
See Also

ch_rfa_seasonstat.

Examples

data(flowAtlantic)

ss <- ch_rfa_seasonstat(date ~ id, flowAtlantic$ams)

ch_rfa_julianplot()
points(y ~ x, ss, pch = 16, col = cut(ss[, 'radius'], c(0,.5,.75,1)))

ch_rfa_seasonstat  Seasonal statistics for flood peaks

Description

Return the circular or seasonal statistics of flood peaks. The angle represents the mean timing of
the floods and the radius its regularity. For example, a radius of one represents perfect regularity.
Can perform the analyses on multiple sites.

Usage

ch_rfa_seasonstat(x, ...)

## S3 method for class 'data.frame'
ch_rfa_seasonstat(x, ...)

## S3 method for class 'formula'
ch_rfa_seasonstat(form, x, ...)

Arguments

x  Data. If data.frame with two columns, they must be respectively the date and a
site variable.

...  Other parameters.

form  Formula that specifies the date and site variable. Must be of the form date ~
site.

Value

Returns the circular or seasonal statistics of flood peaks.

Author(s)

Martin Durocher
ch_sh_get_amax

References


See Also

ch_rfa_distseason

Examples

dt <- ch_rfa_extractamax(flow~date, CAN01AD002)$date

ch_rfa_seasonstat(dt)

## Illustration of the analysis of multiple sites

F0 <- function(ii) data.frame(site = ii, dt = sample(dt, replace = TRUE))
x <- lapply(1:10, F0)
x <- do.call(rbind, x)
st <- ch_rfa_seasonstat(dt ~ site, x)

ch_rfa_julianplot()
points(y ~ x, st, col = 2, pch = 16)

 ch_sh_get_amax  Extracts annual maximum values from ECDE dataframe.

Description

Extracts annual maximum values, the date of occurrence, the day of year, and the completeness from ECDE dataframe. Uses functions from timeDate (as.timeDate, dayOfYear).

Usage

ch_sh_get_amax(df)

Arguments

df  A dataframe of daily streamflow data from ECDE

Value

Returns a dataframe with the following variables

year
annual maximum
ch_slice

```r
date of annual maximum
day of year of annual maximum
days number of days with observations

Author(s)
Paul Whitfield

See Also
ch_read_ECDE_flows ch_circ_mean_reg

Examples
```r
data(CAN05AA008)
amax <- ch_sh_get_amax(CAN05AA008)
str(amax)
```

---

Description

Converts a series of a variable such as day of year into numbered bins. Whenever the number of bins does not divide in 365 evenly a message showing the number of bins created and the number of days added to the last bin is provided.

Simply put, ch_slice is used to convert doy into a factor which is a number of bins per year. A year can be converted into any number of bins; slice does it based upon a number of days. So when you send it an array of doy it slices that into bins of the desired width. For example, if the step is 5. They 365/5 gives 73 bins and because of leap years there might be one extra day added every four years to the final bin.

To illustrate for a bin of 5 days: doy: 1 2 3 4 5 6 7 8 9 10 11 12 Bin: 1 1 1 1 1 2 2 2 2 2 2 3

Usage

```r
ch_slice(doy, step)
```

Arguments

doy A vector of the day of calendar year for the dataset
step Width of bin in days

Value

Returns a vector of bin numbers that is used as a factor for each day in the dataset and provides a message indicating the handling of partial bins
ch_sub_set_Years

Author(s)
Paul Whitfield, Kevin Shook

See Also
ch_binned_MannWhitney ch_flow_raster_trend

Examples

doy <- c(1:365)
# first 30 days are 1, 31-60 are 2 etc
dice <- ch_slice(doy, 30)
plot(doy, dice)

ch_sub_set_Years

Helper function for selecting points for an axis

Description
Sub-samples a vector every n places. Many times there are so many years the labels on the plot overlap. ch_sub_set_Years returns the position and label for the subset. The function can be used on any type of simple array.

Usage
ch_sub_set_Years(years, n)

Arguments
years a vector of years
n sample size

Value
a list containing:
position array of axis positions
label array of labels

Author(s)
Paul Whitfield
ch_tidyhydat_ECDE

Converts a tidyhydat daily flow data tibble to ECDE format

Description
Accessing daily flow data using tidyhydat is quick and efficient. However, it sometimes conflicts with other functions as tidyhydat changes variable names and some default entries. This function converts a tibble obtained from a tidyhydat tibble to a dataframe with standard Environment and Climate Change Canada Data Explorer (ECDE) names.

Usage
ch_tidyhydat_ECDE(data)

Arguments
- data Tibble of daily flows retrieved using tidyhydat function hy_daily_flows.

Value
A dataframe or a list of flows with formats consistent with datafiles read using ch_read_ECDE_flows:

<table>
<thead>
<tr>
<th>ID</th>
<th>stationID</th>
</tr>
</thead>
<tbody>
<tr>
<td>PARAM</td>
<td>Parameter 1 for Flow 2 for Level</td>
</tr>
<tr>
<td>Date</td>
<td>Original character string converted to date format</td>
</tr>
<tr>
<td>Flow</td>
<td>Daily mean flow m³/sec</td>
</tr>
<tr>
<td>SYM</td>
<td>Quality flag</td>
</tr>
</tbody>
</table>

Author(s)
Paul Whitfield

See Also
ch_tidyhydat_ECDE_meta
ch_tidyhydat_ECDE_meta

Examples

# This example uses the built-in test database, by setting the hydat_path parameter
# You will want to use it with your actual HYDAT database
library(tidyhydat)
# check for existence of test database
test_db <- hy_test_db()
if (file.exists(test_db)) {
  hydat_path = hy_set_default_db(test_db)
  mdata <- hy_daily_flows(station_number=c("05AA008"))
  m_data <- ch_tidyhydat_ECDE(mdata)

  mdata <- hy_daily_flows(station_number=c("05AA008", "08MF005", "05HD008"))
  mnew <- ch_tidyhydat_ECDE(mdata)
  str(mnew[[1]])
  str(mnew[[2]])
  str(mnew[[3]])
  # note the order is in increasing alphabetical order
  hy_set_default_db(NULL) # Reset HYDAT database
}

ch_tidyhydat_ECDE_meta

Creates an ECDE-like dataframe of metadata from tidyhydat

Description

Extracts tombstone (meta) data for stations from tidyhydat in a format similar to that used by the Environment Canada Data Explorer (ECDE). The default does not capture all the fields in ECDE, which includes the most recent status of many fields such as operating schedule. Returning these values slows the function, particularly when all WSC stations are selected.

Usage

ch_tidyhydat_ECDE_meta(stations, all_ECDE = FALSE)

Arguments

stations A vector of WSC station IDs, i.e. c("05BB001", "05BB003", "05BB004", "05BB005"). If stations = "all" then values are returned for all stations. Note that you should ensure that that the tidyhydat database is up to date, if you select stations = "all", so that the most recent set of stations is used.

all_ECDE Should all ECDE values be returned? If FALSE the default, then values of Flow, Level, Sed, OperSched, Region, Datum, and Operator are omitted or will differ from the ECDE values. If all_ECDE = TRUE, then the function will return values identical to ECDE. Note that setting all_ECDE = TRUE will result in very long execution times, as it is necessary to extract many daily values for each station to determine the values of Flow, Level, Sed, and OperSched to determine the final values.
Value

Returns a list with three items:

- `meta` - a dataframe of metadata from `tidyhydat` in ECDE form (not all ECDE fields are reproduced in this summary)
- `H_version` - version information, and
- `th_meta` - a dataframe with all `tidyhydat` fields including:
  - Station - StationID
  - StationName - Station Name
  - HYDStatus - Active or Discontinued
  - Prov - Province
  - Latitude
  - Longitude
  - DrainageArea - km²
  - Years - number of years with data
  - From - Start Year
  - To - End Year
  - Reg. - Regulated?
  - Flow - not captured (differs from ECDE), unless all_ECDE = TRUE
  - Level - not captured (differs from ECDE), unless all_ECDE = TRUE
  - Sed - not captured (differs from ECDE), unless all_ECDE = TRUE
  - OperSched - not captured (differs from ECDE), unless all_ECDE = TRUE
  - RealTime - if TRUE/Yes
  - RHBN - if TRUE/Yes is in the reference hydrologic basin network
  - Region - number of region instead of name (differs from ECDE), unless all_ECDE = TRUE
  - Datum - reference number (differs from ECDE), unless all_ECDE = TRUE
  - Operator - reference number (differs from ECDE), unless all_ECDE = TRUE

Author(s)

Paul Whitfield, Kevin Shook

See Also

`ch_get_ECDE_metadata` `ch_tidyhydat_ECDE`

Examples

# This example uses the built-in test database, by setting the hydat_path parameter
# You will want to use it with your actual HYDAT database
library(tidyhydat)
# check for existence of test database
test_db <- hy_test_db()
if (file.exists(test_db)) {
  stations <- c("05AA008", "08MF005", "05HD008")
  hy_set_default_db(test_db)
result <- ch_tidyhydat_ECDE_meta(stations)
metadata <- result[[1]]
version <- result[[2]]
hy_set_default_db(NULL)  # Reset HYDAT database
}
## Not run:
# This example is not run, as it will take several hours to execute and will
# return many warnings for stations having no data. Note that it is using the actual
# HYDAT database, which must have been installed previously
# This use of the function is intended for the package maintainers to
# update the HYDAT_list data frame
result <- ch_tidyhydat_ECDE_meta("all", TRUE)
HYDAT_list <- result$meta
## End(Not run)

---

**ch_tr_sign**

**ch_tr_sign**

---

**Description**

Converts MK (or other) slopes to integers 1-2-3 (negative, none, positive). These indices can be used to indicate trend direction.

**Usage**

```r
ch_tr_sign(x, offset = 2)
```

**Arguments**

- `x`: an array of slopes
- `offset`: the amount of shift to make values positive integers, default is 2.

**Value**

Returns an array of indices (1, 2, 3)

**Author(s)**

Paul Whitfield

**Examples**

```r
mkin <- c(-0.23, 0.34, 0.0, 0.033, -0.55)
mkout <- ch_tr_sign(mkin)
# 1 3 2 3 1
```
ch_tr_signif  \hspace{1cm} ch_tr_signif()

Description
Convert pvalues to integers 1 for NS and 2 for significant using a pvalue that can be set (default is 0.05)

Usage
ch_tr_signif(x, pvalue = 0.05)

Arguments
x \hspace{1cm} an array of pvalues from statistical test
pvalue \hspace{1cm} critical value, default is 0.05

Value
Returns an array of indices 1 and 2, where 1 is NS and 2 is significant

Author(s)
Paul Whitfield

Examples
\begin{verbatim}
  sin <- c( -0.052, 0.34, 0.012, -.033, -0.55)
  sout <- ch_tr_signif(sin)
  # 1 1 2 2 1
\end{verbatim}

ch_volcano_pourpoints  \hspace{1cm} Creates a sample file of pour points

Description
Creates a file of pour points for the volcano DEM. The pour points define the outlets of sub-basins. These pour points are used by examples within other functions.

Usage
ch_volcano_pourpoints(pp_shp)

Arguments
pp_shp  \hspace{1cm} Name for shapefile to hold pour points
Value

Returns an sf object containing 2 pour points for the volcano DEM. The pour points are also written to the specified file.

Author(s)

Dan Moore and Kevin Shook

See Also

ch_volcano_raster ch_wbt_pourpoints

Examples

```r
pourpoint_file <- tempfile("volcano_pourpoints", fileext = c(\".shp\")
pourpoints <- ch_volcano_pourpoints(pourpoint_file)
plot(pourpoints)
```

---

### ch_volcano_raster

**Create Test Raster**

Description

Creates a raster object of land surface elevations, as used to test/demonstrate many functions requiring a digital elevation model (DEM).

Usage

```r
ch_volcano_raster()
```

Details

No arguments are required as the DEM is created from the base volcano matrix of elevations.

Value

Returns a raster object of land surface elevations.

Author(s)

Dan Moore and Kevin Shook

Examples

```r
test_raster <- ch_volcano_raster()
```
ch_wbt_catchment  Delineate catchment boundaries

Description
Delineate catchment boundaries

Usage

ch_wbt_catchment(
  fn_pp_snap,
  fn_flowdir,
  fn_catchment_ras,
  fn_catchment_vec,
  return_vector = TRUE
)

Arguments

  fn_pp_snap  Name of file containing snapped pour points
  fn_flowdir  Name of file containing flow accumulations.
  fn_catchment_ras  Raster file to contain delineated catchment.
  fn_catchment_vec  Vector file to contain delineated catchment.
  return_vector  If TRUE (the default) a vector of the catchment will be returned.

Value
If return_vector == TRUE a vector of the catchment is returned. Otherwise nothing is returned.

Author(s)
Dan Moore and Kevin Shook

See Also
ch_wbt_catchment_onestep

Examples

# Only proceed if Whitebox executable is installed
library(whitebox)
if (check_whitebox_binary()){
  library(raster)
test_raster <- ch_volcano_raster()
dem_raster_file <- tempfile(fileext = "tif")
no_sink_raster_file <- tempfile("no_sinks", fileext = "tif")
# write test raster to file
writeRaster(test_raster, dem_raster_file, format = "GTiff")

# remove sinks
removed_sinks <- ch_wbt_removesinks(dem_raster_file, no_sink_raster_file, method = "fill")

# get flow accumulations
flow_acc_file <- tempfile("flow_acc", fileext = ".tif")
flow_acc <- ch_wbt_flow_accumulation(no_sink_raster_file, flow_acc_file)

# get pour points
pourpoint_file <- tempfile("volcano_pourpoints", fileext = ".shp")
pourpoints <- ch_volcano_pourpoints(pourpoint_file)
snapped_pourpoint_file <- tempfile("snapped_pourpoints", fileext = ".shp")
snapped_pourpoints <- ch_wbt_pourpoints(pourpoints, flow_acc_file, pourpoint_file,
                                       snapped_pourpoint_file, snap_dist = 10)

# get flow directions
flow_dir_file <- tempfile("flow_dir", fileext = ".tif")
flow_dir <- ch_wbt_flow_direction(no_sink_raster_file, flow_dir_file)
fn_catchment_ras <- tempfile("catchment", fileext = ".tif")
fn_catchment_vec <- tempfile("catchment", fileext = ".shp")
catchments <- ch_wbt_catchment(snapped_pourpoint_file, flow_dir_file,
                               fn_catchment_ras, fn_catchment_vec)
}

} else {
  message("Examples not run as Whitebox executable not found")
}

ch_wbt_catchment_onestep

Delineates a catchment in a single step

Description

Calls all of the ch_wbt and other functions required to do the sub-tasks required to delineate a catchment. The names of files to be created are taken from the list created by the function ch_wbt_filenames.

Usage

ch_wbt_catchment_onestep(
  wd,
  in_dem,
  pp_sf,
  sink_method = "breach_leastcost",
  dist = NULL,
  check_catchment = TRUE,
  threshold = NULL,
snip_dist = NULL,
cb_colour = "red",
pp_colour = "red",
channel_colour = "blue",
contour_colour = "grey",
plot_na = TRUE,
plot_scale = TRUE,
na_location = "tr",
scale_location = "bl",
...
)

Arguments

wd            Name of working directory.
in_dem        File name for original DEM.
pp_sf         Vector containing pour points.
sink_method   Method for sink removal as used by ch_wbt_removesinks.
dist          Maximum search distance for breach paths in cells. Required if sink_method = "breach_leastcost".
check_catchment If TRUE (the default) ch_checkcatchment will be called after the catchment is created.
threshold     Threshold for channel initiation.
snap_dist     Maximum pour point snap distance in map units.
cb_colour     Colour for catchment outline. Default is "red".
pp_colour     Colour for catchment pour points. Default is "red".
channel_colour Colour for channel. Default is "blue".
contour_colour Colour for contours Default is "grey".
plot_na       If TRUE (the default) a north arrow is added to the plot.
plot_scale    If TRUE (the default) a scale bar is added to the plot.
na_location   Location for the north arrow. Default is ‘tr’, i.e. top-right.
scale_location Location for the scale bar. Default is ‘bl’, i.e. bottom-left.
...           Extra parameters for ch_wbt_removesinks.

Value

Returns an sp object of the delineated catchment.

Author(s)

Dan Moore and Kevin Shook

See Also

ch_wbt_filenames
Examples

# Only proceed if Whitebox executable is installed
library(whitebox)
if (check_whitebox_binary()){
  library(raster)
test_raster <- ch_volcano_raster()
dem_raster_file <- tempfile(fileext = c(".tif"))
  # write test raster to file
  writeRaster(test_raster, dem_raster_file, format = "GTiff")
  wd <- tempdir()
pourpoint_file <- tempfile("volcano_pourpoints", fileext = ".shp")
pourpoints <- ch_volcano_pourpoints(pourpoint_file)
catchment <- ch_wbt_catchment_onestep(wd = wd, in_dem = dem_raster_file,
pp_sf = pourpoints, sink_method = "fill", threshold = 1, snap_dist = 10)
} else {
  message("Examples not run as Whitebox executable not found")
}

ch_wbt_channels

Generate stream network

Description
Generate stream network

Usage

ch_wbt_channels(
  fn_flowacc,
  fn_flowdir,
  fn_channel_ras,
  fn_channel_vec,
  threshold = NULL,
  ...
)

Arguments

fn_flowacc File name for flow accumulation grid.
fn_flowdir File name for flow direction grid.
fn_channel_ras File name for raster version of channel network.
fn_channel_vec File name for vector version of channel networks.
threshold Threshold for channel initiation.
...

Other parameters for whitebox function wbt_extract_streams

Value

Returns a sf vector object of the stream channels.
ch_wbt_filenames

Creates names for Whitebox function input and output files

Description

Creates a list of the files used for inputs and outputs by the Whitebox functions. This function needs to be called before calling any of the other Whitebox (i.e. those prefixed by cd_wbt) functions. If the file names are not specified, default names will be used. All raster files are TIFF (.tif), all vector files are shapefiles (.shp).

Usage

ch_wbt_filenames(
    wd = NULL,
)
ch_wbt_filenames

fn_dem = "dem.tif",
fn_dem_fsc = "dem_fsc.tif",
fn_dem_ns = "dem_ns.tif",
fn_flowacc = "flow_acc.tif",
fn_flowdir = "flow_dir.tif",
fn_channel_ras = "channel.tif",
fn_channel_vec = "channel.shp",
fn_catchment_ras = "catchment.tif",
fn_catchment_vec = "catchment.shp",
fn_pp = "pp.shp",
fn_pp_snap = "pp_snap.shp"
)

Arguments

wd Required. Name of working directory.
fn_dem File name of input DEM. Default is ‘dem.tif’.
fn_dem_fsc File name for dem after filling single-cell pits. Default is ‘dem_fsc.tif’.
fn_dem_ns File name for dem removing sinks. Default is ‘dem_ns.tif’.
fn_flowacc File name for DEM flow accumulation grid. Default is ‘flow_acc.tif’.
fn_flowdir File name for DEM flow direction grid. Default is ‘flow_dir.tif’.
fn_channel_ras File name for raster version of channel network. Default is ‘channel.tif’.
fn_channel_vec File name for vector version of channel networks. Default is ‘channel.shp’.
fn_catchment_ras File name for raster version of catchment. Default is ‘catchment.tif’.
fn_catchment_vec File name for vector version of catchment. Default is ‘catchment.shp’.
fn_pp File name for pour points (input). Vector file. Default is ‘pp.shp’.
fn_pp_snap File name for pour points after snapping to channel network. Vector file. Default is ‘pp.snap.shp’.

Value

Returns a list of the input and output file names

Author(s)

Dan Moore

Examples

wbt_file_names <- ch_wbt_filenames(getwd())
ch_wbt_flow_accumulation

Creates flow accumulation grid file

Description

Creates flow accumulation grid file

Usage

ch_wbt_flow_accumulation(fn_dem_ns, fn_flowacc, return_raster = TRUE)

Arguments

fn_dem_ns File name of dem with sinks removed.
fn_flowacc File name for flow accumulation grid to be created.
return_raster If TRUE (the default), the flow accumulation grid will be returned as a raster object, in addition to being written to ‘fn_flowacc’. If FALSE, the output file will still be created but a NULL value is returned.

Value

If return_raster = TRUE, the flow accumulation grid will be returned as a raster object, otherwise NULL is returned.

Author(s)

Dan Moore

Examples

# Only proceed if Whitebox executable is installed
library(whitebox)
if (check_whitebox_binary()){ library(raster)
test_raster <- ch_volcano_raster()
dem_raster_file <- tempfile(fileext = c(".tif"))
no_sink_raster_file <- tempfile("no_sinks", fileext = c(".tif"))

# write test raster to file
writeRaster(test_raster, dem_raster_file, format = "GTiff")

# remove sinks
removed_sinks <- ch_wbt_removesinks(dem_raster_file, no_sink_raster_file, method = "fill")

# get flow accumulations
flow_acc_file <- tempfile("flow_acc", fileext = c(".tif"))
flow_acc <- ch_wbt_flow_accumulation(no_sink_raster_file, flow_acc_file)
ch_wbt_flow_direction

Description

Creates flow direction grid file

Usage

ch_wbt_flow_direction(fn_dem_ns, fn_flowdir, return_raster = TRUE)

Arguments

fn_dem_ns File name of dem with sinks removed.
fn_flowdir File name for flow direction grid to be created.
return_raster Should a raster object be returned?

Value

If return_raster = TRUE (the default), the flow direction grid will be returned as a raster object, in addition to being written to ‘fn_flowdir’. If return_raster = FALSE, the output file will still be created but a NULL value is returned.

Author(s)

Dan Moore

Examples

# Only proceed if Whitebox executable is installed
library(whitebox)
if (check_whitebox_binary()){
  library(raster)
  test_raster <- ch_volcano_raster()
  dem_raster_file <- tempfile(fileext = c(".tif"))
  no_sink_raster_file <- tempfile("no_sinks", fileext = c(".tif"))
  
  # write test raster to file
  writeRaster(test_raster, dem_raster_file, format = "GTiff")
  
  # remove sinks
  removed_sinks <- ch_wbt_removesinks(dem_raster_file, no_sink_raster_file, method = "fill")
  
  # get flow directions
  plot(flow_acc)
} else {
  message("Examples not run as Whitebox executable not found")
}
ch_wbt_pourpoints

Snap pour points to channels

Description

Pour points describe the outlets of sub-basins within a DEM. To use the pour points to delineate catchments, they must align with the drainage network. This function snaps (forces the locations) of pour points to the channels.

Usage

ch_wbt_pourpoints(
    pp_sf = NULL,
    fn_flowacc,
    fn_pp,
    fn_pp_snap,
    check_crs = TRUE,
    snap_dist = NULL,
    ...
)

Arguments

pp_sf sf object containing pour points. These must be supplied by the user. See the code in ch_volcano_pourpoints for an example of creating the object.
fn_flowacc Name of file containing flow accumulations.
fn_pp File name to create un-snapped pour points.
fn_pp_snap File name for snapped pour points.
check_crs If TRUE the projections of the pour points and flow accumulation files will be checked to ensure they are identical.
snap_dist Maximum snap distance in map units.
...

Value

Returns a sf object of the specified pour points snapped to the channel network.

Author(s)

Dan Moore
ch_wbt_removesinks

Removes sinks from a DEM

Description

Sinks are removed from a DEM using one of several methods. The raster file types supported are listed in Spatial_hydrology_functions.

Usage

ch_wbt_removesinks(
  in_dem,
  out_dem,
  method = "breach_leastcost",
  dist = NULL,
  fn_dem_fsc = NULL,
ch_wbt_removesinks

Arguments

in_dem  File path for original dem. Required.
out_dem File path for dem after removing sinks.
method Method for removing sinks. Default method is 'breach_leastcost'. Other methods include 'breach', 'fill', 'fill_pd' (Planchon and Darboux), and 'fill_wl' (Wang and Liu).
dist  Maximum search distance for breach paths in cells. Required if method = "breach_leastcost".
fn_dem_fsc File path for dem after removing single-cell pits.

Value

Returns a raster object containing the processed dem.

Author(s)

Dan Moore

Examples

# Only proceed if Whitebox executable is installed
library(whitebox)
if (check_whitebox_binary()){
  library(raster)
  test_raster <- ch_volcano_raster()
  dem_raster_file <- tempfile(fileext = c(".tif"))
  no_sink_raster_file <- tempfile("no_sinks", fileext = c(".tif"))

  # write test raster to file
  writeRaster(test_raster, dem_raster_file, format = "GTiff")

  # remove sinks
  removed_sinks <- ch_wbt_removesinks(dem_raster_file, no_sink_raster_file, method = "fill")
} else {
  message("Examples not run as Whitebox executable not found")
}
### ch_wtr_yr

**Designation of the water year**

**Description**

Display water year

**Usage**

```r
ch_wtr_yr(dates, start_month = 10)
```

**Arguments**

- `dates`: A vector of dates with actual year
- `start_month`: Month in which the year starts (defaults to October)

**Value**

Year starting in `start_month`

**Source**

http://stackoverflow.com/questions/27626533/r-create-function-to-add-water-year-column

**Examples**

```r
date <- seq(as.Date("1910/1/1"), as.Date("1912/1/1"), "days")
wtr_yr_date <- ch_wtr_yr(dates=date, start_month=10)
df <- data.frame(wtr_yr_date, date)
```

### flowAtlantic

**Annual maxima from sites in the Atlantic region of Canada**

**Description**

Contains the annual maxima of 45 hydrometric stations found in the region '01' of Water Survey of Canada. In additional to the annual maxima, the output list includes catchment descriptors (longitude, latitude, basin area, mean annual precipitation) and the geographical distance between each station.

**Usage**

```r
flowAtlantic
```

**Format**

An object of class `list` of length 2.
HYDAT_list

Author(s)
Martin Durocher

Source
https://wateroffice.ec.gc.ca/

HYDAT_list

List of Water Survey of Canada hydrometric stations.

Description
A dataframe of station information, as extracted from HYDAT using ECDataExplorer.

Usage
HYDAT_list

Format
A dataframe with a row for each station and 20 columns.

Details
Variables:
- **Station**  StationID
- **StationName**  Station Name
- **HYDStatus**  Active or Discontinued
- **Prov**  Province
- **Latitude**
- **Longitude**
- **DrainageArea**  km²
- **Years**  Number of years with data
- **From**  Start Year
- **To**  End Year
- **Reg.**  Regulated
- **Flow**  If TRUE/Yes
- **Level**  If TRUE/Yes
- **Sed**  If TRUE/Yes
- **OperSched**  Continuous or Seasonal
- **RealTime**  If TRUE/Yes
- **RHBN**  If TRUE/Yes the station is in the reference hydrologic basin network
- **Region**  ECCC Region
- **Datum**  Reference datum
- **Operator**  Operator
Spatial_hydrology_functions

Source

Water Survey of Canada

Spatial_hydrology_functions

Description

These functions perform spatial analyses important in hydrology. All of the functions with the prefix ch_wbt require the installation of the package Whitebox. The functions include:

- **ch_wbt_removesinks**: Removes sinks from a DEM by deepening drainage network
- **ch_wbt_fillsinks**: Removes sinks from a DEM by filling them
- **ch_wbt_catchment**: Generates catchment boundaries for a conditioned DEM based on specified points of interest
- **ch_wbt_channels**: Generates a drainage network from DEM
- **ch_wbt_flow_accumulation**: Accumulates flows downstream in a catchment
- **ch_wbt_flow_direction**: Calculated flow directions for each cell in DEM
- **ch_wbt_pourpoints**: Snaps pour points to channel
- **ch_wbt_catchment_onestep**: Performs all catchment delineations in a single function
- **ch_contours**: Creates contour lines from DEM
- **ch_checkcatchment**: Provides a simple map to check the outputs from ch_saga_catchment
- **ch_checkchannels**: Provides a simple map to check the outputs from ch_saga_channels
- **ch_volcano_raster**: Returns a raster object of land surface elevations

The Whitebox functions support the following file types for raster data:

- **type**  extension
  - GeoTIFF  *.tif, *.tiff
  - Big GeoTIFF  *.tif, *.tiff
  - Esri ASCII  *.txt, *.asc
  - Esri BIL  *.flt, *.hdr
  - GRASS ASCII  *.txt, *.asc
  - Idrisi  *.rdc, *.rst
  - SAGA Binary  *.sdat, *.sgrd
  - Surfer ASCII  *.grd
  - Surfer Binary  *.grd
  - Whitebox  *.tas, *.dep
StatisticalHydrology-functions

Statistical analysis functions

Description

These functions perform statistical analyses

- **ch_binned_MannWhitney**  Compares two time periods of data using Mann-Whitney test
- **ch_fdcurve**  Finds flow exceedence probabilities
- **ch_get_peaks**  Finds peak flows over a specified threshold

Visualization-functions

Visualization functions

Description

These functions are primarily intended for graphing, although some analyses may also be done.

- **ch_booth_plot**  Plot of peaks over a threshold
- **ch_flow_raster**  Raster plot of streamflows
- **ch_flow_raster_qa**  Raster plot of streamflows with WSC quality flags
- **ch_flow_raster_trend**  Raster plot and simple trends of observed streamflows
- **ch_hydrograph_plot**  Plots hydrographs and/or precipitation
- **ch_polar_plot**  Polar plot of daily streamflows
- **ch_regime_plot**  Plots the regime of daily streamflows
Index

* colour
  ch_col_transparent, 20
* datasets
  CAN01AD002, 5
  CAN05AA008, 6
  flowAtlantic, 73
  HYDAT_list, 74
* data
  ch_date_subset, 23
* date
  ch_date_subset, 23
* package
  CSHShydRology-package, 3
* plot
  ch_booth_plot, 9
  ch_flow_raster_trend, 29
  ch_polar_plot, 38
* subset
  ch_date_subset, 23
* transparency
  ch_col_transparent, 20

Basic_data_manipulation_functions, 5

CAN01AD002, 5
CAN05AA008, 6
ch_axis_doy, 6
ch_binned_MannWhitney, 7, 38, 39, 41, 43, 55
ch_booth_plot, 9, 33
ch_catchment_hyps, 10
ch_checkcatchment, 12, 15
ch_checkchannels, 13, 14
ch_circ_mean_reg, 16, 54
ch_clear_wd, 17, 22
ch_col_gradient, 19
ch_col_transparent, 20
ch_color_gradient, 18
ch_contours, 21
ch_create_wd, 18, 22
ch_cut_block, 23
ch_date_subset, 23
ch_decades_plot, 8, 24, 24
ch_doy, 25
ch_fdcurve, 26
ch_flow_raster, 27, 28, 30
ch_flow_raster_qa, 27, 28
ch_flow_raster_trend, 27, 28, 29, 55
ch_get_ECDE_metadata, 31, 58
ch_get_peaks, 9, 32
ch_get_url_data, 33
ch_get_wscstation, 35
ch_hydrograph_plot, 36
ch_polar_plot, 8, 38, 43
ch_polar_plot_peaks, 39
ch_polar_plot_prep, 8, 38, 39, 41
ch_qa_hydrograph, 43
ch_read_AHCCD_daily, 45, 46
ch_read_AHCCD_monthly, 45, 46
ch_read_ECDE_flows, 27, 28, 47, 54
ch_regime_plot, 7, 48
ch_rfa_distseason, 49, 53
ch_rfa_extractamax, 50
ch_rfa_julianplot, 51
ch_rfa_seasonstat, 50, 52, 52
ch_sh_get_amax, 17, 53
ch_slice, 54
ch_sub_set_Years, 55
ch_tidyhydat_ECDE, 56, 58
ch_tidyhydat_ECDE_meta, 56, 57
ch_tr_sign, 59
ch_tr_signif, 60
ch_volcano_pourpoints, 60, 70, 71
ch_volcano_raster, 61, 61
ch_wbt_catchment, 62
ch_wbt_catchment_onestep, 62, 63
ch_wbt_channels, 65
ch_wbt_filenames, 64, 66
ch_wbt_flow_accumulation, 68
ch_wbt_flow_direction, 69
ch_wbt_pourpoints, 61, 70
ch_wbt_removesinks, 71
ch_wtr_yr, 73
CSHShydRology (CSHShydRology-package), 3
CSHShydRology-package, 3
flowAtlantic, 73
HYDAT_list, 74
points, 51
Spatial_hydrology_functions, 71, 75
StatisticalHydrology-functions, 76
Visualization-functions, 76