

# Package ‘CSHShydRology’

November 26, 2025

**Type** Package

**Title** Canadian Hydrological Analyses

**Version** 1.4.4

**Date** 2025-11-26

**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 Flood-net project are also included. CSHShydRology 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, lubridate, plotrix, timeDate, stringr, ggplot2,  
ggspatial, stats, raster, sf, dplyr, magrittr, httr, tidyhydat,  
whitebox, datasets, circular, MGBT, outliers, TeachingDemos

**Suggests** knitr, testthat, rmarkdown, readr

**VignetteBuilder** knitr

**LazyData** true  
**Encoding** UTF-8  
**RoxygenNote** 7.3.2  
**NeedsCompilation** no  
**Repository** CRAN  
**Date/Publication** 2025-11-26 14:50:02 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_circular_colors . . . . .	16
ch_circ_mean_reg . . . . .	17
ch_clear_wd . . . . .	18
ch_color_gradient . . . . .	19
ch_col_gradient . . . . .	20
ch_col_transparent . . . . .	21
ch_contours . . . . .	22
ch_create_wd . . . . .	23
ch_cut_block . . . . .	24
ch_date_subset . . . . .	24
ch_decades_plot . . . . .	25
ch_doys . . . . .	26
ch_fdcurve . . . . .	27
ch_ffa_screen_plot . . . . .	28
ch_flow_raster . . . . .	29
ch_flow_raster_qa . . . . .	30
ch_get_ECDE_metadata . . . . .	31
ch_get_peaks . . . . .	32
ch_get_url_data . . . . .	34
ch_get_wscstation . . . . .	35
ch_gg_hydrographs . . . . .	36
ch_high_Grubbs_test . . . . .	38
ch_model_hydrograph . . . . .	39
ch_polar_plot . . . . .	41
ch_polar_plot_peaks . . . . .	42
ch_polar_plot_prep . . . . .	44
ch_qa_hydrograph . . . . .	46

ch_read_AHCCD_daily . . . . .	47
ch_read_AHCCD_monthly . . . . .	48
ch_read_ECDE_flows . . . . .	49
ch_regime_plot . . . . .	50
ch_rfa_distseason . . . . .	51
ch_rfa_extractamax . . . . .	53
ch_rfa_julianplot . . . . .	54
ch_rfa_seasonstat . . . . .	55
ch_sh_get_amax . . . . .	56
ch_slice . . . . .	57
ch_sub_set_Years . . . . .	58
ch_tidyhydat_ECDE . . . . .	59
ch_tidyhydat_ECDE_meta . . . . .	60
ch_tr_sign . . . . .	62
ch_tr_signif . . . . .	63
ch_volcano_pourpoints . . . . .	63
ch_volcano_raster . . . . .	64
ch_wbt_catchment . . . . .	65
ch_wbt_catchment_onestep . . . . .	66
ch_wbt_channels . . . . .	68
ch_wbt_filenames . . . . .	69
ch_wbt_flow_accumulation . . . . .	71
ch_wbt_flow_direction . . . . .	72
ch_wbt_pourpoints . . . . .	73
ch_wbt_removesinks . . . . .	74
ch_wtr_yr . . . . .	76
flowAtlantic . . . . .	76
HYDAT_list . . . . .	77
Spatial_hydrology_functions . . . . .	78
StatisticalHydrology-functions . . . . .	79
Visualization-functions . . . . .	79
<b>Index</b>	<b>80</b>

---

CSHShydRology-package *Functions for Canadian hydrological analyses*

---

## 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/> which is an affiliated society of the Canadian Water Resources Association (CWRA) <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/adaptor 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 ECDataExplorer

**ch\_get\_ECDE\_metadata** Reads station meta data from ECDataExplorer

**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/>

---

CAN05AA008	CAN05AA008
------------	------------

---

**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 m<sup>3</sup>/s
  - SYM** Water Survey FLags A, B, D, E

**Source**

Water Survey of Canada

---

ch_axis_doy	<i>Generates the x axis beginning on specified day of year</i>
-------------	--

---

**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.

**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**

DF	A data frame of hydrometric data from <code>ch_read_ECDE_flows</code>
step	An integer indicating the degree of smoothing eg. 1, 5, 11.
range1	The first and last year of first period, as <code>c(first, last)</code>
range2	The first and last year of second period, as <code>c(first, last)</code>
ptest	The significance level default is 0.05.
variable	Name of variable. Default is 'discharge'
metadata	dataframe of station metadata, default is <code>HYDAT_list</code>

**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**

Whitfield, P.H., Cannon, A.J., 2000. Recent variations in climate and hydrology in Canada. Canadian Water Resources Journal 25: 19-65.

**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 <code>ch_get_peaks</code>
threshold	The threshold used by <code>ch_get_peaks</code>
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

Booth, E.G., Mount, J.F., Viers, J.H. 2006. Hydrologic Variability of the Cosumnes River Floodplain. San Francisco Estuary & Watershed Science 4:21.

Whitfield, P.H., and J.W. Pomeroy. 2016. Changes to flood peaks of a mountain river: implications for analysis of the 2013 flood in the Upper Bow River, Canada. Hydrological Processes 30:4657-73. doi: 10.1002/hyp.10957.

## See Also

[ch\\_get\\_peaks](#)

**Examples**

```
threshold <- 0.1 * max(CAN05AA008$Flow) # arbitrary threshold
peaks <- ch_get_peaks(CAN05AA008, threshold)
events <- peaks$P0Tevents
ch_booth_plot(events, threshold, title = "05AA008", type='mag')
ch_booth_plot(events, threshold, title = "05AA008", type='vol')
```

---

ch_catchment_hyps	<i>Catchment hypsometry</i>
-------------------	-----------------------------

---

**Description**

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

**Usage**

```
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 <b>sf</b> object containing the catchment divide.
dem	A <b>raster</b> object of the Digital Elevation Model.
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.

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

```
# 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_url <- "https://zenodo.org/record/4781469/files/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_url <- "https://zenodo.org/record/4781469/files/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	<i>Check Catchments</i>
-------------------	-------------------------

---

## Description

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

## Usage

```

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.
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.

### 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 <code>ch_wbt_channels</code> ) (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")
snapped_pourpoints <- ch_wbt_pourpoints(pourpoints, flow_acc_file, pourpoint_file,
snapped_pourpoint_file, snap_dist = 10)
ch_checkchannels(test_raster, channels, snapped_pourpoints)
} else {
  message("Examples not run as Whitebox executable not found")
}

```

---

ch\_circular\_colors      *Color Palettes for Circular*

---

## Description

Produces a vector of  $n$  contiguous colours. The original version from the package `circular` produced  $n$  colours with 1 and  $n$  being the same colour. This version internally produces  $n + 1$  colours and removes the repeated end colour so there are  $n$  unique colours. Hue is calculated for points on the circle. Saturation and value default to 0.85. Default alpha is 1.0 - no transparency.

## Usage

```

ch_circular_colors(
  n,
  m = 0,
  M = 2 * pi,
  offset = 0,
  s = 0.85,
  v = 0.85,
  alpha = 1
)

```



**Arguments**

n	the number of colours ( $\leq 1$ ) to be in the palette
m	the smallest angle in radians
M	the largest angle in radians
offset	the zero in radians
s	saturation further arguments passed to the function hsv
v	value [0.85]
alpha	alpha [1.0] no transparency

**Value**

a vector of length n

**Author(s)**

Claudio Agostinelli (original) Paul Whitfield (update)

**Examples**

```
ch_circular_colors(n=10)
```

---

ch_circ_mean_reg	<i>Calculates the circular mean, median, and regularity</i>
------------------	---

---

**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

Pewsey, A., M. Neuhauser, and G. D. Ruxton. 2014. Circular Statistics in R, 192 pp., Oxford University Press. Whitfield, P. H. 2018. Clustering of seasonal events: A simulation study using circular methods. Communications in Statistics - Simulation and Computation 47(10): 3008-3030. Burn, D. H., and P. H. Whitfield. 2021\*. Changes in the timing of flood events resulting from climate change.

## See Also

[ch\\_sh\\_get\\_amax](#)

## Examples

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

---

ch_clear_wd	<i>Clear Working Directory</i>
-------------	--------------------------------

---

## Description

Empties and removes a working directory.

## Usage

```
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	<i>ch_color_gradient</i>
-------------------	--------------------------

---

**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

**Author(s)**

Paul Whitfield

**Examples**

```
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	<i>Creates a colour gradient</i>
-----------------	----------------------------------

---

**Description**

Creates a colour gradient for plotting.

**Usage**

```
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	<i>Add Transparency to plot colours</i>
--------------------	---

---

### 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

### References

See original code on post in Stack Overflow <https://stackoverflow.com/questions/12995683/any-way-to-make-plot-points-in-scatterplot-more-transparent-in-rmaking>

### Examples

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

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

# 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

```
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

## 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	<i>Create working directory</i>
--------------	---------------------------------

---

**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	<i>Extracts a specified time period from a longer record</i>
--------------	--

---

### 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

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

---

ch_date_subset	<i>Subsets dates by string</i>
----------------	--------------------------------

---

### Description

Subsets a data frame by an 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'



**Value**

df                      subsetted data frame

**Author(s)**

Robert Chlumsky

**Examples**

```
{
  dd <- seq.Date(as.Date("2010-10-01"), as.Date("2013-09-30"), by = 1)
  x <- rnorm(length(dd))
  y <- abs(rnorm(length(dd)))*2
  df <- data.frame("Date" = dd,x,y)
  prd <- "2011-10-01/2012-09-30"
  summary(ch_date_subset(df,prd))}
```

---

ch_decades_plot	<i>Plots output from ch_binned_MannWhitney for decades</i>
-----------------	--

---

**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](#)

**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	<i>Days of year and water year</i>
---------	------------------------------------

---

**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 <code>as.Date()</code>
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)
```

---

ch\_fdcurve*Plot Flow Duration Curve*

---

**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**

Gustard, A., A. Bullock, and J.M. Dixon. 1992. Low flow estimation in the United Kingdom. Institute of Hydrology, 292. Wallingford: Institute of Hydrology.

Vogel, R.M., and N.M. Fennessy. 1994. Flow-duration curves. I: New Interpretation and confidence intervals. Journal of Water Resources Planning and Management ASCE 120:485-504.

Vogel, R.M., and N.M. Fennessy. 1995. Flow duration curves II: A review of applications in water resources planning. Water Resources Bulletin 31:1030-9.

## Examples

```
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\_ffa\_screen\_plot      *FFA screening plot*

---

## Description

Generates a flood frequency plot with symbols indicating whether an observation is a high or low outlier, and colour symbols indicating month of the year in which the flood occurred.

## Usage

```
ch_ffa_screen_plot(
  df,
  stn = "unspecified",
  mtitle = "",
  n = 12,
  m = 0,
  M = 2,
  offset = 0,
  mcol = c("orange", "gray70", "red")
)
```

## Arguments

df	dataframe with date of event [maxdate] and annual maximum [amax]
stn	stationID
mtitle	title for plot
n	default is 12, number of colours for number of months
m	smallest angle in radians parameter for generating circular colours default = 0
M	largest angle in radians parameter for generating circular colours default = 2 (*pi)
offset	the zero in radians, default is 0.
mcol	array of three colours, default is c("orange", "gray70", "red") use to outline or emphasize low outliers, normal, and high outliers.

**Value**

a list containing

- Station if specified else "unspecified"
- number of amax events
- array of amax magnitudes
- array of outlier codes (1 = low, 2 = not, 3 = high outlier)

**References**

Cohn, T. A., J. F. England, C. E. Berenbrock, R. R. Mason, J. R. Stedinger and J. R. Lamontagne (2013). "A generalized Grubbs-Beck test statistic for detecting multiple potentially influential low outliers in flood series." *Water Resources Research* 49(8): 5047-5058 10.1002/wrcr.20392: 10.1002/wrcr.20392.

**Examples**

```
# Not tested automatically as can be very slow to execute
data(CAN05AA008)
amax <- ch_sh_get_amax(CAN05AA008)
ch_ffa_screen_plot(amax)
```

---

ch_flow_raster	<i>Raster plot of daily streamflows</i>
----------------	---

---

**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**

```
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\\_qa](#)

**Examples**

```
ch_flow_raster(CAN05AA008)
```

---

ch_flow_raster_qa	<i>Raster plot of daily streamflows with WSC quality flags</i>
-------------------	--

---

**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 <code>ch_read_ECDE_flows</code>
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](#)**Examples**

```
data(HYDAT_list)
data(CAN05AA008)
qaplot <- ch_flow_raster_qa(CAN05AA008)
```

---

ch\_get\_ECDE\_metadata    *Reads Environment Canada Date Explorer (ECDE) meta data file*

---

**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	km <sup>2</sup>
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

**Author(s)**

Paul Whitfield <paul.h.whitfield@gmail.com>

**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)
```

---

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)

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**

Burn, D.H., Whitfield, P.H., Sharif, M., 2016. Identification of changes in floods and flood regimes in Canada using a peaks over threshold approach. *Hydrological Processes*, 39: 3303-3314. DOI:10.1002/hyp.10861

Whitfield, P.H., and J.W. Pomeroy. 2016. Changes to flood peaks of a mountain river: implications for analysis of the 2013 flood in the Upper Bow River, Canada. *Hydrological Processes* 30:4657-73. doi: 10.1002/hyp.10957.

**See Also**

[ch\\_booth\\_plot](#)

**Examples**

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

---

ch_get_url_data	<i>Gets remote data sets</i>
-----------------	------------------------------

---

### 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

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

### 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_url <- "https://zenodo.org/record/4781469/files/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_url <- "https://zenodo.org/record/4781469/files/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)") +
  coord_sf(datum = 32611) +
  theme_bw()

```

---

ch_get_wscstation	<i>Reads station information from a data file produced by ECDE</i>
-------------------	--

---

## 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 km <sup>2</sup>
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("05BB001", metadata = HYDAT_list)
title <- s_info[21]
print(title)
```

---

ch\_gg\_hydrographs

*Hydrographs for WSC stations using ggplot2*

---

### Description

Acquires and plots values for WSC streamflows, using **ggplot2**. The existing functions `ch_qa_hydrograph` and `ch_model_hydrograph` use basic R plotting, require other functions to assemble the values, and only plot values for a single station. This function is able to plot hydrographs for more than one station, which may be useful, particularly when comparing responses for several streams in the same region.

**Usage**

```
ch_gg_hydrographs(
  WSC_stations,
  daily = TRUE,
  instantaneous = FALSE,
  facets = TRUE,
  common_dates = FALSE,
  start_date = NULL,
  end_date = NULL,
  hydat_path = NULL,
  inst_colour = "black",
  daily_colour = "black",
  ...
)
```

**Arguments**

WSC_stations	Required. A vector of WSC station numbers.
daily	Optional. If TRUE, mean daily streamflows are plotted as stair-steps.
instantaneous	Optional. If TRUE, annual instantaneous peak flows are plotted as points.
facets	Optional. If TRUE, the plot is faceted by station number.
common_dates	Optional. If TRUE, a common date range is used for all time series.
start_date	Optional. If specified (format = yyyy-mm-dd), only values on or following the date are plotted.
end_date	Optional. If specified (format = yyyy-mm-dd), only values on or before the date are plotted.
hydat_path	Optional. Path to the HYDAT database. Usually omitted unless you want to use a specific database file.
inst_colour	Optional. Colour to be used for annual instantaneous peaks, if either faceted or only a single station is plotted. Default is "black".
daily_colour	Optional. Colour to be used for daily flows, if either faceted or only a single station is plotted. Default is "black".
...	Other parameters for the <b>ggplot</b> facets, if specified.

**Value**

Returns a ggplot2 object of the hydrographs.

**Author(s)**

Kevin Shook

**See Also**

[ch\\_qa\\_hydrograph](#) [ch\\_model\\_hydrograph](#)

**Examples**

```
## Not run:
# Not run as this requires the installation of the HYDAT database

# plot a single station
stations <- c("05HH003")
p <- ch_gg_hydrographs(stations, daily = TRUE, instantaneous = TRUE)

# plot a group of stations in a region, as all appear to be responding to the same event
stations <- c("05CC001", "05CC011", "05CD006", "05CD007", "05CE002", "05CE006",
"05CE010", "05CE012", "05CE018", "05CE020", "05CG004", "05CG006")

p <- ch_gg_hydrographs(stations, daily = TRUE, instantaneous = FALSE,
common_dates = FALSE, start_date = "2011-06-01", end_date = "2011-06-30",
facets = TRUE, scales = "free_y", ncol= 3)

## End(Not run)
```

---

ch\_high\_Grubbs\_test      *Grubbs test for high events*

---

**Description**

Checks for outliers among the highest fifteen events, assumes the data are normally distributed and uses the Grubbs test to determine if the highest value is an outlier. If it is an outlier the next largest is tested until the value is not an outlier, or 15 values have been tested.

**Usage**

```
ch_high_Grubbs_test(amax)
```

**Arguments**

amax                      A vector of the annual maxima

**Value**

result A vector of the same length as amax with a 1 for those values that are high outliers else zero.

**Author(s)**

Paul Whitfield

**Examples**

```
set.seed(1234)
x = rnorm(10)
x[11] <- 15
x[12] <- 35
ch_high_Grubbs_test(x)
```

---

ch\_model\_hydrograph     *Hydrograph plot for model outputs and gauged flows*


---

### Description

Creates a hydrograph plot for simulated and observed flows, including precipitation if provided. The secondary y axis is used to plot the precipitation time series.

### Usage

```
ch_model_hydrograph(
  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
)
```

### 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 <sup>3</sup> /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 stages are typically reported as geodetic elevations, where the minimum values are much greater than zero.

### 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

### See Also

[ch\\_qa\\_hydrograph](#) [ch\\_gg\\_hydrographs](#)

### 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_model_hydrograph(flows = df, winter_shading = FALSE)

# with different labels and winter shading
ch_model_hydrograph(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_model_hydrograph(flows = df, precip = precip, range_mult_flow = 1.7,
range_mult_precip = 2, leg_box = TRUE)
```

---

ch_polar_plot	<i>Polar plot of daily streamflows</i>
---------------	--

---

## 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 <a href="#">ch_binned_MannWhitney</a>
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")

## Value

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

## Author(s)

Paul Whitfield

## References

Whitfield, P.H. and A.J. Cannon. 2000. Polar plotting of seasonal hydrologic and climatic data. *Northwest Science* 74: 76-80.

Whitfield, P.H., Cannon, A.J., 2000. Recent variations in climate and hydrology in Canada. *Canadian Water Resources Journal* 25: 19-65.

**See Also**

[ch\\_binned\\_MannWhitney](#) [ch\\_polar\\_plot\\_prep](#)

**Examples**

```
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**

```
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 <code>ch_circ_mean_reg</code> (optional)
regularity	a value or array of regularity from <code>ch_circ_mean_reg</code> (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, otherwise 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")
in_detail	an array of indices indicating symbol [1] shape, [2] colour, [3] background, and [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**

Pewsey, A., M. Neuhauser, and G. D. Ruxton. 2014. Circular Statistics in R, 192 pp., Oxford University Press.

Whitfield, P. H. 2018. Clustering of seasonal events: A simulation study using circular methods. Communications in Statistics - Simulation and Computation 47(10): 3008-3030.

Burn, D. H., and P. H. Whitfield. 2023. Changes in the timing of flood events resulting from climate change. Journal of Hydrology.

## Examples

```
# 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	<i>Creates a data structure to be passed to ch_polar_plot</i>
--------------------	---

---

## 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.

## 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
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
series	A data frame containing six columns

The series data frame contains

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 for each period
code	significance codes for each bin

Author(s)

Paul Whitfield

References

Whitfield, P.H. and A.J. Cannon. 2000. Polar plotting of seasonal hydrologic and climatic data. Northwest Science 74: 76-80.

Whitfield, P.H., Cannon, A.J., 2000. Recent variations in climate and hydrology in Canada. Canadian Water Resources Journal 25: 19-65.

See Also

[ch\\_binned\\_MannWhitney](#) [ch\\_polar\\_plot](#)

---

ch_qa_hydrograph	<i>Plots a hydrograph with the data quality symbols and returns a report on qa symbols and missing data.</i>
------------------	--

---

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 is 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.

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 <code>ch_read_ECDE_flows</code> .
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 data
n	the number of data points
sym_count	summary of the SYM counts
missing	number of missing data

**Author(s)**

Paul Whitfield

**See Also**

[ch\\_gg\\_hydrographs](#) [ch\\_model\\_hydrograph](#)

**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**

Daily AHCCD data are available from [https://crd-data-donnees-rdc.ec.gc.ca/CDAS/products/EC\\_data/AHCCD\\_daily/](https://crd-data-donnees-rdc.ec.gc.ca/CDAS/products/EC_data/AHCCD_daily/). Any use of the data must cite *Mekis, E and L.A. Vincent, 2011: An overview of the second generation adjusted daily precipitation dataset for trend analysis in Canada. Atmosphere-Ocean, 49 (2), 163-177.*

**See Also**

[ch\\_read\\_AHCCD\\_monthly](#)

**Examples**

```
## 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**

```
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**

Any use of the data must cite Mekis, E and L.A. Vincent, 2011: An overview of the second generation adjusted daily temperature and precipitation dataset for trend analysis in Canada. *Atmosphere-Ocean*, 49 (2), 163-177.

**See Also**

[ch\\_read\\_AHCCD\\_daily](#)

**Examples**

```
## 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	<i>Reads a file of WSC daily flows from ECDataExplorer (ECDE)</i>
--------------------	---

---

**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:

ID	stationID
PARAM	Parameter 1 for Flow 2 for Level
Date	original character string converted to date format
Flow	Daily mean flow m <sup>3</sup> /sec
SYM	Quality flag

**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**

```
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**

MacCulloch, G. and P. H. Whitfield (2012). Towards a Stream Classification System for the Canadian Prairie Provinces. Canadian Water Resources Journal 37: 311-332.

**Examples**

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

---

ch_rfa_distseason	<i>Distance in seasonal space</i>
-------------------	-----------------------------------

---

**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**

```
ch_rfa_distseason(x, ...)

## S3 method for class 'numeric'
ch_rfa_distseason(x, a, w = 1/pi, ...)

## S3 method for class 'data.frame'
```

```
ch_rfa_distseason(x, w = 1/pi, ...)

## S3 method for class 'formula'
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

Durocher, M., Burn, D. H., & Ashkar, F. (2019). Comparison of estimation methods for a nonstationary index-flood model in flood frequency analysis using peaks over threshold. <https://doi.org/10.31223/osf.io/rnepc>

### 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	<i>Extracts the annual maxima of a daily time series</i>
--------------------	--

---

## 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.
...	Other parameters.
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.

## 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	<i>Circular plotting by day of year</i>
-------------------	---

---

## 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 <a href="#">points</a> .

## 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	<i>Seasonal statistics for flood peaks</i>
-------------------	--

---

## 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

## References

Burn, D.H. (1997). Catchment similarity for regional flood frequency analysis using seasonality measures. Journal of Hydrology 202, 212-230. [https://doi.org/10.1016/S0022-1694\(97\)00068-1](https://doi.org/10.1016/S0022-1694(97)00068-1)

## 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	<i>Extracts annual maximum values from ECDE dataframe.</i>
----------------	--

---

**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

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**

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

---

ch_slice	<i>Converts doy or dwy into a factor that is used to bin data</i>
----------	---

---

**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 3 3

**Usage**

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

**Arguments**

<code>doy</code>	A vector of the day of calendar year for the dataset
<code>step</code>	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

**Author(s)**

Paul Whitfield, Kevin Shook

**See Also**

[ch\\_binned\\_MannWhitney](#)

**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

**Examples**

```
myyears <- c(1900:2045)
myyears <- ch_sub_set_Years(myyears, 20)
myyears

a <- LETTERS
my_alpha <- ch_sub_set_Years(a, 5)
my_alpha
```

---

ch_tidyhydat_ECDE	<i>Converts a tidyhydat daily flow data tibble to ECDE format</i>
-------------------	---

---

### 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 <b>tidyhydat</b> function hy_daily_flows.
------	---

### Value

A dataframe or a list of flows with formats consistent with datafiles read using ch\_read\_ECDE\_flows:

ID	stationID
PARAM	Parameter 1 for Flow 2 for Level
Date	Original character string converted to date format
Flow	Daily mean flow m <sup>3</sup> /sec
SYM	Quality flag

### Author(s)

Paul Whitfield

### See Also

[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 the <b>tidyhydat</b> 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 - km<sup>2</sup>
- 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	<i>ch_tr_sign</i>
------------	-------------------

---

### Description

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

### Usage

```
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

```
mkkin <- c( -0.23, 0.34, 0.0, .033, -0.55)
mkkout <- ch_tr_sign(mkkin)
# 1 3 2 3 1
```

---

ch_tr_signif	<i>ch_tr_signif()</i>
--------------	-----------------------

---

**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	an array of pvalues from statistical test
pvalue	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**

```
sin <- c( -0.052, 0.34, 0.012, -.033, -0.55)
sout <- ch_tr_signif(sin)
# 1 1 2 2 1
```

---

ch\_volcano\_pourpoints *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	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**

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

---

ch_volcano_raster	<i>Create Test Raster</i>
-------------------	---------------------------

---

**Description**

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

**Usage**

```
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**

```
test_raster <- ch_volcano_raster()
```



---

ch_wbt_catchment	<i>Delineate catchment boundaries</i>
------------------	---------------------------------------

---

**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,

```

```

    snap_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 <code>ch_wbt_removesinks</code> .
dist	Maximum search distance for breach paths in cells. Required if <code>sink_method = "breach_leastcost"</code> .
check_catchment	If TRUE (the default) <code>ch_checkcatchment</code> 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 <code>ch_wbt_removesinks</code> .

### 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	<i>Generate stream network</i>
-----------------	--------------------------------

---

## 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 <b>whitebox</b> function <code>wbt_extract_streams</code>

## Value

Returns a **sf** vector object of the stream channels.

**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)

  # 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)
  plot(channels)
} else {
  message("Examples not run as Whitebox executable not found")
}
```

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,
```

```

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.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

<code>fn_dem_ns</code>	File name of dem with sinks removed.
<code>fn_flowacc</code>	File name for flow accumulation grid to be created.
<code>return_raster</code>	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)
```

```

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

```

---

ch\_wbt\_flow\_direction *Creates flow direction grid file*

---

### 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

```



```

    flow_dir_file <- tempfile("flow_dir", fileext = c(".tif"))
    flow_dir <- ch_wbt_flow_direction(no_sink_raster_file, flow_dir_file)
    plot(flow_dir)
  } else {
    message("Examples not run as Whitebox executable not found")
  }

```

---

ch_wbt_pourpoints	<i>Snap pour points to channels</i>
-------------------	-------------------------------------

---

## 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	<b>sf</b> object containing pour points. These must be supplied by the user. See the code in <a href="#">ch_volcano_pourpoints</a> 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.
...	Additional parameters for <b>whitebox</b> function <code>wbt_snap_pour_points</code> .

## Value

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

## Author(s)

Dan Moore

**See Also**[ch\\_volcano\\_pourpoints](#)**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 pour points
  pourpoint_file <- tempfile("volcano_pourpoints", fileext = c(".shp"))
  pourpoints <- ch_volcano_pourpoints(pourpoint_file)
  snapped_pourpoint_file <- tempfile("snapped_pourpoints", fileext = c(".shp"))
  snapped_pourpoints <- ch_wbt_pourpoints(pourpoints, flow_acc_file, pourpoint_file,
    snapped_pourpoint_file, snap_dist = 10)
} else {
  message("Examples not run as Whitebox executable not found")
}
```

---

ch_wbt_removesinks	<i>Removes sinks from a DEM</i>
--------------------	---------------------------------

---

**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,
```

```
    ...
  )
```

### 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.
...	Additional arguments to be passed to functions to remove sinks.

### 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	<i>Designation of the water year</i>
-----------	--------------------------------------

---

**Description**

Display water year

**Usage**

```
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**

```
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	<i>Annual maxima from sites in the Atlantic region of Canada</i>
--------------	--

---

**Description**

Contains the annual maxima of 45 hydrometric stations found in the region '01' of Water Survey of Canada. In addition 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**

```
flowAtlantic
```

**Format**

An object of class list of length 2.

**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** km<sup>2</sup>**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

**Source**

Water Survey of Canada

---

Spatial\_hydrology\_functions

*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** \*.sdatt, \*.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\_ffa\_screen\_plot** FFA screening plot

**ch\_flow\_raster** Raster plot of streamflows

**ch\_flow\_raster\_qa** Raster plot of streamflows with WSC quality flags

**ch\_gg\_hydrographs** ggplot2 hydrographs of WSC flows

**ch\_model\_hydrograph** Plots hydrographs and/or precipitation

**ch\_polar\_plot** Polar plot of daily streamflows

**ch\_qa\_hydrograph** Plots a hydrograph with the data quality symbols

**ch\_regime\_plot** Plots the regime of daily streamflows

# Index

- \* **colour**
    - ch\_col\_transparent, 21
  - \* **datasets**
    - CAN01AD002, 5
    - CAN05AA008, 6
    - flowAtlantic, 76
    - HYDAT\_list, 77
  - \* **data**
    - ch\_date\_subset, 24
  - \* **date**
    - ch\_date\_subset, 24
  - \* **package**
    - CSHShydRology-package, 3
  - \* **plot**
    - ch\_booth\_plot, 9
    - ch\_polar\_plot, 41
  - \* **subset**
    - ch\_date\_subset, 24
  - \* **transparency**
    - ch\_col\_transparent, 21
- Basic\_data\_manipulation\_functions, 5
- CAN01AD002, 5
- CAN05AA008, 6
- ch\_axis\_doy, 6
- ch\_binned\_MannWhitney, 7, 41, 42, 44, 46, 57
- ch\_booth\_plot, 9, 33
- ch\_catchment\_hyps, 10
- ch\_checkcatchment, 12, 15
- ch\_checkchannels, 13, 14
- ch\_circ\_mean\_reg, 17, 57
- ch\_circular\_colors, 16
- ch\_clear\_wd, 18, 23
- ch\_col\_gradient, 20
- ch\_col\_transparent, 21
- ch\_color\_gradient, 19
- ch\_contours, 22
- ch\_create\_wd, 19, 23
- ch\_cut\_block, 24
- ch\_date\_subset, 24
- ch\_decades\_plot, 8, 25, 25
- ch\_doys, 26
- ch\_fdcurve, 27
- ch\_ffa\_screen\_plot, 28
- ch\_flow\_raster, 29, 31
- ch\_flow\_raster\_qa, 30, 30
- ch\_get\_ECDE\_metadata, 31, 61
- ch\_get\_peaks, 9, 32
- ch\_get\_url\_data, 34
- ch\_get\_wscstation, 35
- ch\_gg\_hydrographs, 36, 40, 47
- ch\_high\_Grubbs\_test, 38
- ch\_model\_hydrograph, 37, 39, 47
- ch\_polar\_plot, 8, 41, 46
- ch\_polar\_plot\_peaks, 42
- ch\_polar\_plot\_prep, 8, 41, 42, 44
- ch\_qa\_hydrograph, 37, 40, 46
- ch\_read\_AHCCD\_daily, 47, 49
- ch\_read\_AHCCD\_monthly, 48, 48
- ch\_read\_ECDE\_flows, 30, 31, 49, 57
- ch\_regime\_plot, 7, 50
- ch\_rfa\_distseason, 51, 55
- ch\_rfa\_extractamax, 53
- ch\_rfa\_julianplot, 54
- ch\_rfa\_seasonstat, 52, 54, 55
- ch\_sh\_get\_amax, 18, 56
- ch\_slice, 57
- ch\_sub\_set\_Years, 58
- ch\_tidyhydat\_ECDE, 59, 61
- ch\_tidyhydat\_ECDE\_meta, 59, 60
- ch\_tr\_sign, 62
- ch\_tr\_signif, 63
- ch\_volcano\_pourpoints, 63, 73, 74
- ch\_volcano\_raster, 64, 64
- ch\_wbt\_catchment, 65
- ch\_wbt\_catchment\_onestep, 65, 66
- ch\_wbt\_channels, 68
- ch\_wbt\_filenames, 67, 69



ch\_wbt\_flow\_accumulation, [71](#)  
ch\_wbt\_flow\_direction, [72](#)  
ch\_wbt\_pourpoints, [64](#), [73](#)  
ch\_wbt\_removesinks, [74](#)  
ch\_wtr\_yr, [76](#)  
CSHShydRology (CSHShydRology-package), [3](#)  
CSHShydRology-package, [3](#)  
  
flowAtlantic, [76](#)  
  
HYDAT\_list, [77](#)  
  
points, [54](#)  
  
Spatial\_hydrology\_functions, [74](#), [78](#)  
StatisticalHydrology-functions, [79](#)  
  
Visualization-functions, [79](#)