Package 'HYPEtools'

May 5, 2025

Version 1.6.5

Title Tools for Processing and Analyzing Files from the Hydrological Catchment Model HYPE

Description Work with model files (setup, input, output) from the hydrological catchment model HYPE: Streamlined file import and export, standard evaluation plot routines, diverse post-processing and aggregation routines for hydrological model analysis. The HYPEtools package is also archived at <doi:10.5281/zenodo.7627955> and can be cited in publications with Brendel et al. (2024) <doi:10.1016/j.envsoft.2024.106094>.

Depends R (>= 3.5.0)

Imports clipr, colorspace, data.table (>= 1.9.8), dplyr, ggplot2, ggpubr, ggrepel, grDevices, graphics, lubridate, methods, ncdf4, parallel, patchwork, pbapply, purrr, rlang, scales, stats, stringr, tidyr, tidyselect, tools, utils, zoo

License LGPL-3

URL https://hypeweb.smhi.se/, https://github.com/rcapell/HYPEtools

BugReports https://github.com/rcapell/HYPEtools/issues

RoxygenNote 7.3.1

Encoding UTF-8

Language en-US

Suggests rmarkdown, knitr, beepr, DT, htmlwidgets, leaflet, leaflet.extras, mapview, plotly, sf, shiny, shinyalert, shinyFiles, shinyWidgets, webshot

VignetteBuilder knitr

NeedsCompilation no

Author Rene Capell [aut, cre] (ORCID: <https://orcid.org/0000-0002-7784-1313>), Conrad Brendel [aut] (ORCID: <https://orcid.org/0000-0002-5199-0580>), Jafet Andersson [ctb], David Gustafsson [ctb], Jude Musuuza [ctb], Jude Lubega [ctb]

Maintainer Rene Capell <hypetools.rene@smhi.se>

Repository CRAN

Date/Publication 2025-05-05 18:20:02 UTC

Contents

`AllDownstreamSubids` *Find All Downstream SUBIDs*

Description

Function to find all SUBIDs of downstream sub-catchments along the main stem for a single sub-catchment.

Usage

```
AllDownstreamSubids(subid, gd, bd = NULL, write.arcgis = FALSE)
```

Arguments

`subid`	Integer, SUBID of a target sub-catchment (must exist in `gd`).
`gd`	Dataframe, an imported 'GeoData.txt' file. Mandatory argument. See 'Details'.
`bd`	Dataframe, an imported 'BranchData.txt' file. Optional argument. See 'Details'.
`write.arcgis`	Logical. If `TRUE`, a string containing an SQL expression suitable for ArcGIS's 'Select By Attributes' feature will be written to the clipboard.

Details

`AllDownstreamSubids` finds all downstream SUBIDs of a given SUBID along the main stem (including itself but not including potential irrigation links or groundwater flows) using GeoData columns 'SUBID' and 'MAINDOWN'. If a BranchData file is provided, the function will also include information on downstream bifurcations.

Value

`AllDownstreamSubids` returns a vector of downstream SUBIDs to the outlet if no BranchData is provided, otherwise a data frame with two columns `downstream` with downstream SUBIDs and `is.branch` with logical values indicating if a downstream SUBID contains a bifurcation ('branch' in HYPE terms). Downstream SUBIDs are ordered from source to final outlet SUBID.

See Also

`AllUpstreamSubids`, `OutletSubids`, `OutletIds`

Examples

```
te <- ReadGeoData(filename = system.file("demo_model", "GeoData.txt", package = "HYPEtools"))
AllDownstreamSubids(subid = 3344, gd = te)
```

`AllUpstreamSubids` *Find All Upstream SUBIDs*

Description

Function to find all SUBIDs of upstream sub-catchments for a single sub-catchment.

Usage

```
AllUpstreamSubids(
  subid,
  gd,
  bd = NULL,
  sort = FALSE,
  get.weights = FALSE,
  write.arcgis = FALSE
)
```

Arguments

`subid`	SUBID of a target sub-catchment (must exist in gd).
`gd`	A data frame, containing 'SUBID' and 'MAINDOWN' columns, e.g. an imported 'GeoData.txt' file. Mandatory argument. See 'Details'.
`bd`	A data frame, containing 'BRANCHID' and 'SOURCEID' columns, and 'MAINPART' with argument `get.weights`, e.g. an imported 'BranchData.txt' file. Optional argument.
`sort`	Logical. If `TRUE`, the resulting upstream SUBID vector will be sorted according to order in argument gd, i.e. in downstream order for a working GeoData table.
`get.weights`	Logical. If `TRUE`, flow weights are computed along the upstream SUBID sequence. See details.
`write.arcgis`	Logical. If `TRUE`, a string containing an SQL expression suitable for ArcGIS's 'Select By Attributes' feature will be written to the clipboard.

Details

`AllUpstreamSubids` finds all upstream SUBIDs of a given SUBID (including itself but not including potential irrigation links or groundwater flows) using GeoData columns 'SUBID' and 'MAINDOWN', i.e the full upstream catchment. If a BranchData file is provided, the function will also include upstream areas which are connected through an upstream bifurcation. The results can be directly used as 'partial model setup file' ('pmsf.txt') using the export function `WritePmsf`.

If argument `get.weights` is set to `TRUE`, weighting fractions are returned along with upstream SUBIDs. The fractions are based on column 'MAINPART' in argument `bd`. The function considers fractions from bifurcation branches which flow into the basin, and fractions where bifurcation branches remove discharge from the basin. Fractions are incrementally updated, i.e. nested bifurcation fractions are multiplied.

For details on bifurcation handling in HYPE, see the HYPE online documentation for BranchData.txt.

Value

If `get.weights` is `FALSE`, `AllUpstreamSubids` returns a vector of SUBIDs, otherwise a two-column data frame with SUBIDs in the first, and flow weight fractions in the second column.

See Also

`UpstreamGeoData`, `AllDownstreamSubids`

Examples

```
te <- ReadGeoData(filename = system.file("demo_model", "GeoData.txt", package = "HYPEtools"))
AllUpstreamSubids(subid = 63794, gd = te)
```

`AnnualRegime` *Calculate annual regimes*

Description

Calculate annual regimes based on long-term time series, typically imported HYPE basin output and time output result files.

Usage

```
AnnualRegime(
  x,
  stat = c("mean", "sum"),
  ts.in = NULL,
  ts.out = NULL,
  start.mon = 1,
  incl.leap = FALSE,
  na.rm = TRUE,
  format = c("list", "long")
)
```

Arguments

x	Data frame, with column-wise equally-spaced time series. Date-times in POSIXct format in first column. Typically an imported basin or time output file from HYPE. See details.
stat	Character string, either "mean" or "sum". Defines the type of aggregation to be computed for output time periods, see Details. Defaults to "mean".
ts.in	Character string, timestep of x, attribute timestep in x per default. Otherwise one of "month", "week", "day", or "nhour" (n = number of hours).
ts.out	Character string, timestep for results, defaults to ts.in. This timestep must be equal to or longer than ts.in.
start.mon	Integer between 1 and 12, starting month of the hydrological year, used to order the output.
incl.leap	Logical, leap days (Feb 29) are removed from results per default, set to TRUE to keep them. Applies to daily and shorter time steps only.
na.rm	Logical, indicating if NA values should be stripped before averages are calculated.
format	Character string. Output format, list (default) or long. See Value.

Details

AnnualRegime uses aggregate to calculate long-term average regimes for all data columns provided in x, including long-term arithmetic means, medians, minima and maxima, and 5%, 25%, 75%, and 95% percentiles. With HYPE result files, AnnualRegime is particularly applicable to basin and time output files imported using ReadBasinOutput and ReadTimeOutput. The function does not check if equally spaced time steps are provided in x or if the overall time period in x covers full years so that the calculated averages are based on the same number of values.

Values within each output time period can be aggregated either by arithmetic means or by sums within each period, e.g. typically means for temperatures and sums for precipitation. Long-term aggregated values are always computed as arithmetic means.

Value

If argument format is list, AnnualRegime returns a list with 8 elements and two additional attributes(). Each list element contains a named data frame with aggregated annual regime data: arithmetic means, medians, minima, maxima, and 5%, 25%, 75%, and 95% percentiles.

Each data frames contains, in column-wise order: reference dates in POSIXct format, date information as string, and aggregated variables found in x.

Reference dates are given as dates in either 1911, 1912, or 1913 (just because a leap day and outer weeks '00'/'53' occur during these years) and can be used for plots starting at the beginning of the hydrological year (with axis annotations set to months only). Daily and hourly time steps are given as is, weekly time steps are given as mid-week dates (Wednesday), monthly time steps as mid month dates (15th).

If argument format is long, AnnualRegime returns a four-column data frame with one value per row, and all variable information aligned with the values. Columns in the data frame: id with SUBIDs or HYPE variable IDs, month/week/day with aggregation time steps, name with short

names of regime data (means, medians, minima, maxima, percentiles), and `value` with the variable value.

Attribute `period` contains a two-element POSIXct vector containing start and end dates of the source data. Attribute `timestep` contains a timestep keyword corresponding to function argument `ts.out`.

Note

If weekly data are provided in `x`, `AnnualRegime` will inflate `x` to daily time steps before computing results. Values in `x` will be assigned to the preceeding week days, corresponding to HYPE file output, where weekly values are conventionally printed on the last day of the week. If `NA` values are present in the original weekly data, these will be filled with the next available value as a side effect of the inflation.

If weekly output time steps are computed in combination with a user-defined start month, the function will round up weeks to determine the first week of the hydrological year. Weeks are identified using Monday as first day of the week and the first Monday of the year as day 1 of week 1 (see conversion code `%W` in `strptime`). Boundary weeks `'00'` and `'53'` are merged to week `'00'` prior to average computations.

See Also

`PlotAnnualRegime`

Examples

```
# Source data, HYPE basin output with a number of result variables
te <- ReadBasinOutput(filename = system.file("demo_model", "results", "0003587.txt",
package = "HYPEtools"))
# Daily discharge regime, computed and observed, hydrologigical year from October
AnnualRegime(te[, c("DATE", "COUT", "ROUT")], ts.in = "day", start.mon = 10)
# Id., aggregated to weekly means
AnnualRegime(te[, c("DATE", "COUT", "ROUT")], ts.in = "day", ts.out = "week", start.mon = 10)
# Long format, e.g. for subsequent plotting with ggplot
AnnualRegime(te[, c("DATE", "COUT", "ROUT")], ts.in = "day", ts.out = "week", format = "long",
start.mon = 10)
# Precipitation regime, monthly sums
AnnualRegime(te[, c("DATE", "UPCPRC")], ts.in = "day", ts.out = "month", stat = "sum")
```

`BarplotUpstreamClasses` *Bar plots of upstream-averaged classes of HYPE sub-basins*

Description

Function to plot upstream-averaged landscape property classes of one or several sub-basins as bar plots, e.g. land use or soils. Builds on `barplot`.

Usage

```
BarplotUpstreamClasses(
  x,
  type = c("custom", "landuse", "soil", "crop"),
  desc = NULL,
  class.names = NULL,
  xlab = NULL,
  ylab = "Area fraction (%)",
  ylim = c(-0.05, max(x[, -1] * 150)),
  names.arg = rep("", ncol(x) - 1),
  cex.axis = 1,
  cex.names = 0.9,
  col = NULL,
  border = NA,
  legend.text = NULL,
  legend.pos = "left",
 pars = list(mar = c(1.5, 3, 0.5, 0.5) + 0.1, mgp = c(1.5, 0.3, 0), tcl = NA, xaxs =
    "i")
)
```

Arguments

x	Data frame, containing column-wise class group fractions with SUBIDs in first column. Typically a result from UpstreamGroupSLCClasses. Column names of class group fractions *must* end with _x, with x giving the class group ID, see details.
type	Character string keyword for class group labeling, used in combination with desc. Type of class groups, either "landuse", "soil", or "crop" (abbreviations allowed). If "custom" (default), labeling can be fine- tuned with argument class.names.
desc	List for use with type. Class description labels imported from a 'description.txt' file, for labeling of standard class groups. See ReadDescription for formatting details.
class.names	Character vector of class group names, with same length as number of class group fractions in x. Specification of bar labels, alternative to arguments type and desc, in particular for cases where a non-standard grouping was used for x.
xlab	Character string, x-axis label, with defaults for standard groups land use, soil, and crops.
ylab	Character string, y-axis label.
ylim	Numeric, two element vector with limits for the y-axis. Defaults to values which give ample space for bar labels.
names.arg	Character vector, see barplot. Defaults to no labeling below bars (text labels within plot through arguments above).
cex.axis	Numeric, character expansion factor for axis annotation and labels.
cex.names	Numeric, character expansion factor for class group labels.

col	Colors for bars. Defaults to type-specific pre-defined color.
border	Colors for bar borders. Defaults to no borders.
legend.text	Character, if provided, a legend will be plotted. Defaults to none if one sub-basin is plotted, and SUBIDs if several sub-basins are plotted. Set to NA to prevent legend plotting in any case.
legend.pos	Character keyword for legend positioning, most likely "left" or "right". For details, see legend.
pars	List of tagged values which are passed to par.

Details

BarplotUpstreamClasses is a wrapper for barplot, with vertical labels plotted over the class group bars. Most arguments have sensible defaults, but can be adapted for fine-tuning if necessary.

Column names of x are used to link class groups to class IDs in desc. HYPE has no formal requirements on how class IDs are numbered and when one of the standard groups land use, soil, or crop are provided in x, there might be missing class IDs. Class names in desc are matched against column name endings '_x' in x. If manual names are provided in class.names, the column name endings must be a consecutive sequence from 1 to number of elements in class.names.

Value

The function returns bar midpoints, see description in barplot.

See Also

UpstreamGroupSLCClasses barplot

Examples

```
# Import source data
te1 <- ReadGeoData(filename = system.file("demo_model", "GeoData.txt", package = "HYPEtools"))
te2 <- ReadGeoClass(filename = system.file("demo_model", "GeoClass.txt", package = "HYPEtools"))
te3 <- ReadDescription(filename = system.file("demo_model", "description.txt",
                       package = "HYPEtools"))
# Calculate plot data, upstream soil fractions
te4 <- UpstreamGroupSLCClasses(subid = 63794, gd = te1, gcl = te2, type = "soil")
# Function call
BarplotUpstreamClasses(x = te4, type = "s", desc = te4, ylim = c(0,100))
```

BoxplotSLCClasses *Box plots of SLC distributions*

Description

BoxplotSLCClasses plots SLC class distributions for all SUBIDs in a GeoData data frame as boxplots. Boxes can represent distributions of area fractions

Usage

```
BoxplotSLCClasses(
  gd,
  gcl,
  col.landuse = "rainbow",
  col.group = NULL,
  lab.legend = NULL,
  pos.legend = 1,
  abs.area = FALSE,
  log = "",
  ylim = NULL,
  range = 0,
  mar = c(3, 3, 1, 7) + 0.1,
  mgp = c(1.5, 0.2, 0),
  tcl = 0.1,
  xaxs = "i",
  xpd = TRUE
)
```

Arguments

gd	Data frame containing columns with SLC fractions, typically a 'GeoData.txt' file imported with ReadGeoData.
gcl	Data frame containing columns with SLCs and corresponding land use and soil class IDs, typically a 'GeoClass.txt' file imported with ReadGeoClass.
col.landuse	Specification of colors for box outlines, to represent land use classes. Either a keyword character string, or a vector of colors with one element for each land use class as given in argument gcl in ascending order. Possible keywords are 'rainbow' (default) or 'auto'. 'rainbow' triggers a generation of land use class colors using the rainbow palette. 'auto' triggers generation of a pretty color palette with similar colors for land use groups. This option requires specification of land use groups in argument col.group.
col.group	Integer vector of the same length as the number of land use classes given in gcl. Specifies a land use group ID for each land use class ID, in ascending order. Groups and group IDs to use (in parentheses): • Water, snow, and ice (1) • Urban (2) • Forests (3) • Agriculture and pastures (4) • Natural non-forested (5)
lab.legend	Character string giving optional land use and soil class names to label the legend. Land use classes first, then soil classes. Both following class IDs as given in gcl in ascending order.
pos.legend	Numeric, legend position in x direction. Given as position on the right hand outside of the plot area in x-axis units.

abs.area	Logical, if TRUE, boxes will be plotted for absolute areas instead of area fractions.
log	Character string, passed to boxplot. Empty string for linearly scaled axes, 'y' for log scaled y-axis (particularly in combination with abs.area = TRUE).
ylim	Numeric vector of length 2, y-axis minimum and maximum. Set automatically if not specified.
range	Argument to boxplot with changed default. See documentation in there.
mar, mgp, tcl, xaxs, xpd	Arguments passed to par. See documentation in there.

Details

BoxplotSLCClasses allows to analyze the occurrence of individual SLCs in a given model set-up. both in terms of area fractions (SLC values) and absolute areas. The function uses boxplot to plot distributions of SLCs of all SUBIDs in a GeoData data frame. Land use classes are color-coded, and soil classes marked by a point symbol below each box. Box whiskers extend to the data extremes.

Value

BoxplotSLCClasses returns a plot to the currently active plot device, and invisibly a data frame of SLC class fractions with 0 values replaced by NAs. If absolute areas are plotted, these are returned in the data frame.

Note

There is a maximum of 26 symbols available for marking soil classes. BoxplotSLCClasses can be quite crowded, depending on the number of SLCs in a model set-up. Tested and recommended plot device dimensions are 14 x 7 inches (width x height), e.g.:

```
> x11(width = 14, height = 7)
> png("mySLCdistri.png", width = 14, height = 7, units = "in", res = 600)
```

Examples

```
# Import source data
te1 <- ReadGeoData(filename = system.file("demo_model", "GeoData.txt", package = "HYPEtools"))
te2 <- ReadGeoClass(filename = system.file("demo_model", "GeoClass.txt", package = "HYPEtools"))
BoxplotSLCClasses(gd = te1, gcl = te2)
```

CleanSLCClasses *Clean Soil-Landuse classes (SLCs) from small fractions*

Description

CleanSLCClasses attempts to clean small SLC fractions within each SUBID (sub-catchment) from an imported GeoData file using user-provided area thresholds. Cleaning can be performed along class similarity rules or along SLC area alone.

Usage

```
CleanSLCClasses(
  gd,
  gcl,
  m1.file = NULL,
  m1.class = "s",
  m1.clean = rep(TRUE, 2),
  m1.precedence = rep(TRUE, 2),
  m2.frac = NULL,
  m2.abs = NULL,
  signif.digits = 3,
  verbose = TRUE,
  progbar = TRUE
)
```

Arguments

gd	Data frame containing columns with SUBIDs, SUBID areas in m^2, and SLC fractions, typically a 'GeoData.txt' file imported with ReadGeoData.
gcl	Data frame containing columns with SLCs and corresponding land use and soil class IDs, typically a 'GeoClass.txt' file imported with ReadGeoClass.
m1.file	Character string, path and file name of the soil or land use class transfer table, a tab-separated text file. Format see details. A value of NULL (default) prevents method 1 cleaning.
m1.class	Character string, either "soil" or "landuse", can be abbreviated. Gives the type of transfer class table for method 1 cleaning. See Details.
m1.clean	A logical vector of length 2 which indicates if cleaning should be performed for area fraction thresholds (position 1) and/or absolute area thresholds (position 2).
m1.precedence	A logical vector of length 2 which indicates if areas below cleaning threshold should be moved to similar areas according to precedence in the transfer table given in m1.file (TRUE) or to the largest area of available transfer classes (FALSE). Area fraction thresholds (position 1) and absolute area thresholds (position 2).
m2.frac	Numeric, area fraction threshold for method 2 cleaning, i.e. moving of small SLC areas to largest SLC in each SUBID without considering similarity between classes. Either a single value or a vector of the same length as the number of SLC classes in gd, giving area fraction thresholds for each SLC separately, with a value 0 for SLCs to omit from cleaning. A value of NULL (default) prevents method 2 area fraction cleaning.
m2.abs	Numeric, see m2.frac. Threshold(s) for absolute areas in m^2.
signif.digits	Integer, number of significant digits to round cleaned SLCs to. See also signif. Set to NULL to prevent rounding.
verbose	Logical, print some information during runtime.
progbar	Logical, display a progress bar while calculating SLC class fractions. Adds overhead to calculation time but useful when subid is NULL or contains many SUBIDs.

Details

`CleanSLCClasses` performs a clean-up of small SLC fractions in an imported GeoData file. Small SLCs are eliminated either by moving their area to similar classes according to rules which are passed to the function in a text file (*Method 1*), or by simply moving their area to the largest SLC in the SUBID (*Method 2*). Moving rules for the first method can be based on either soil classes or land use classes but these cannot be combined in one function call. Run the function two times to combine soil and land use based clean-up. Method 1 and 2, however, can be combined in one function call, in which case the rule-based classification will be executed first. Clean-up precedence in method 1: if clean-ups based on area fractions and absolute areas are combined (`m1.clean = rep(TRUE, 2)`), then area fractions will be cleaned first. In order to reverse precedence, call `CleanSLCClasses` two times with absolute area cleaning activated in first call and area fraction cleaning in second. In both methods, SLCs in each SUBID are cleaned iteratively in numerical order, starting with SLC_1. This implies a greater likelihood of eliminating SLCs with smaller indices.

Method 1

For method one, small SLC fractions are moved to either similar land use classes within the same soil class, or vice versa. Similarities are defined by the user in a tab-separated text file, which is read by `CleanSLCClasses` during runtime. Soil and land use classes correspond to the classes given in column two and three in the `GeoClass` file. The file must have the following format:

class.1	*thres.frac.1*	*thres.abs.1*	*transfer.1*	*...*	*transfer.n*
class.2	*thres.frac.2*	*thres.abs.2*	*transfer.1*	*...*	*transfer.o*
...	*...*	*...*	*...*	*...*	*...*
class.m	*thres.frac.m*	*thres.abs.m*	*transfer.1*	*...*	*transfer.p*

Column 1 contains the source land use or soil classes subjected to clean-up, columns 2 and 3 contain threshold values for area fractions and absolute areas. The remaining columns contain classes to which areas below threshold will be transferred, in order of precedence. Each class can have one or several transfer classes. `CleanSLCClasses` will derive SLC classes to clean from the given soil or land use class using the GeoClass table given in argument `gcl`. No header is allowed. At least one transfer class must exist, but classes can be omitted and will then be ignored by `CleanSLCClasses`. The order of transfer classes in the transfer file indicates transfer preference. `CleanSLCClasses` constructs a transfer list for each SLC class in the model set-up and per default uses the order to choose a preferred SLC to transfer to. However, if several SLCs exist for a given soil or land use class, one of them will be chosen without further sorting. If argument `m1.precedence` is set to `FALSE` for either area fractions or absolute areas, precedence will be ignored and the largest area available will be chosen to transfer small areas to. Area fraction thresholds are given as fractions of 1, absolute area thresholds as values in m^2. If an area below threshold is identified but there are no fitting SLCs available to transfer to, the area will remain unchanged.

Method 2

This method is more rigid than method one and can also be applied as a post-processor after clean-up using method 1 to force a removal of all SLCs below a given threshold from a GeoData file (method 1 cleaning can be be very selective, depending on how many transfer classes are provided in the transfer table). Cleaning thresholds for method 2 area fractions and absolute areas are given in arguments `m2.frac` and `m2.abs`. SLC areas below the given thresholds will be moved to the largest SLC in the given SUBID without considering any similarity between classes.

Value

CleanSLCClasses returns the GeoData data frame passed to the function in argument gd with cleaned SLC class columns.

See Also

RescaleSLCClasses for re-scaling of SLC area fraction sums.

Examples

```
# Import source data
te1 <- ReadGeoData(filename = system.file("demo_model", "GeoData.txt", package = "HYPEtools"))
te2 <- ReadGeoClass(filename = system.file("demo_model", "GeoClass.txt", package = "HYPEtools"))
# Clean-up using method 2, 0.5 % area fraction threshold and 100 m^2 absolute area threshold
te3 <- CleanSLCClasses(gd = te1, gcl = te2, m2.frac = 0.005, m2.abs = 100)
# Detailed comparison with function CompareFiles
te4 <- CompareFiles(te1, te3, type = "GeoData")
te4
```

CompareFiles *Compare HYPE model files to identify any differences.*

Description

Compare HYPE model files to identify any differences, typically used to check that no undesired changes were made when writing a new file.

Usage

```
CompareFiles(
  x,
  y,
  type = c("AquiferData", "BasinOutput", "BranchData", "CropData", "DamData", "ForcKey",
   "FloodData", "GeoClass", "GeoData", "Info", "LakeData", "MapOutput", "MgmtData",
   "Optpar", "Par", "PointSourceData", "Obs", "Simass", "Subass", "TimeOutput", "Xobs"),
  by = NULL,
  compare.order = TRUE,
  threshold = 1e-10,
  ...
)
```

Arguments

x	Path to a HYPE model file to read, or an existing list/data frame object for a HYPE model file. File contents are compared to those of y.
y	Path to a HYPE model file to read, or an existing list/data frame object for a HYPE model file. File contents are compared to those of x.

type	Character string identifying the type of HYPE model file. Used to determine appropriate read function. One of AquiferData, BasinOutput, BranchData, CropData, DamData, ForcKey, FloodData, GeoClass, GeoData, Info, LakeData, MapOutput, MgmtData, Optpar, Par, PointSourceData, Obs, Simass, Subass, TimeOutput, or Xobs.
by	Character vector, names of columns in x and y to use to join data. See dplyr::full_join.
compare.order	Logical, whether or not the order of the rows should be compared. If TRUE, then x and y will also be joined by row number. See dplyr::full_join.
threshold	Numeric, threshold difference for comparison of numeric values. Set to 0 to only accept identical values.
...	Other arguments passed on to functions to read the files to compare (e.g. ReadGeoData, ReadPar, etc.).

Details

CompareFiles compares two HYPE model files and identifies any differences in values. The function reads two model files, compares the values in columns with corresponding names, and returns a data frame consisting of rows/columns with any differences. Values that are the same in both files are set to NA. If numeric values in two columns aren't exactly the same, then the difference between the values will be taken and compare to theshold. If the difference is <= theshold, then the values will be considered the equal and set to NA. The function is primarily intended as a check to ensure that no unintended changes were made when writing model files using the various HYPEtools write functions. However, it can also be used to e.g. compare files between different model versions.

Value

Returns invisibly a data frame containing rows and columns in which differences exist between x and y. Values that are the same in both files are set to NA. If the returned data frame has only rows for the "FILE_ROW" column (and/or columns specified with by), then there were no differences between the files.

Examples

```
# Import demo model GeoData file, edit a SUBID
te1 <- ReadGeoData(filename = system.file("demo_model", "GeoData.txt", package = "HYPEtools"))
te1$SUBID[1] <- 1
# Compare with original file
te2 <- CompareFiles(system.file("demo_model", "GeoData.txt", package = "HYPEtools"), te1,
                    type = "GeoData")
te2
```

`ConvertDischarge` *Calculate Specific runoff from volumetric discharge and vice versa*

Description

`ConvertDischarge` converts volumetric discharge to specific discharge (unit area discharge) and vice versa.

Usage

```
ConvertDischarge(q, area, from = "m3s", to = "mmd")
```

Arguments

`q`	An object of type `numeric`, containing volumetric or specific discharge values, typically HYPE variables COUT or ROUT.
`area`	An object of type `numeric`, containing a catchment area in m^2.
`from`	Character string keyword, giving the current unit of `q`. Either a specific discharge, one of: `"mmy"` (mmy^{-1}) `"mmd"` (mmd^{-1}) `"mmh"` (mmh^{-1}), or a volumetric discharge, one of: `"m3s"` (m^3s^{-1}) `"ls"` (ls^{-1}).
`to`	Character string keyword, see `from`. Conversion will not work between units within volumetric or specific discharge groups.

Details

`ConvertDischarge` is a simple conversion function, most likely to be used in combination with `apply` or related functions.

Value

`ConvertDischarge` returns a numeric object of the same type as provided in argument `q`.

Examples

```
ConvertDischarge(6, 400000000)
ConvertDischarge(c(1.1, 1.2, 1.9, 2.8, 2, 1.5, 1.3, 1.2, 1.15, 1.1),
                 from = "mmd", to = "ls", area = 1.2e6)
```

`CreateOptpar` *Create an optpar list*

Description

`CreateOptpar` creates a list representing a HYPE optpar.txt file from an imported par.txt file and a selection of parameters.

Usage

```
CreateOptpar(
  x,
  pars,
  tasks = data.frame(character(), character()),
  comment = "",
  fun.ival = NULL
)
```

Arguments

`x`	a list with named vector elements, as an object returned from `ReadPar`.
`pars`	Character vector with HYPE parameter names to be included in optpar list. Parameters must exist in `x`. Not case-sensitive. For a complete list of HYPE parameters, see the par.txt online documentation.
`tasks`	Data frame with two columns providing optimization tasks and settings (key-value pairs) as described in the optpar.txt online documentation. Defaults to an empty task section.
`comment`	Character string, comment (first row in optpar.txt file).
`fun.ival`	Either `NULL` (default), or a function with a single argument. See Details.

Details

`CreateOptpar` makes it a bit more convenient to compose a HYPE optimization file. The function creates a template with all parameters to be included in an optimization run.

Parameter boundaries for individual classes have to be adapted after creation of the template, the function takes the existing parameter value(s) in `x` as upper and lower boundaries.

Parameter step width intervals (third parameter rows in optpar.txt files) are calculated with an internal function which per default returns the nearest single 1/1000th of the parameter value, with conditional replacement of '0' intervals:

```
function(x) {
  res <- 10^floor(log10(x/1000))
  ifelse(res == 0, .1, res)
}
```

Alternative functions can be passed to `CreateOptpar` using argument `fun.ival`. Such functions must have a single argument `x`, which represents the parameter value taken from argument `x`. The function is applied to all parameters in the resulting optpar list.

Value

The function returns a list with elements as described in `ReadOptpar`.

See Also

`ReadOptpar` `WriteOptpar` `OptimisedClasses`

Examples

```
# Import a HYPE parameter file
te1 <- ReadPar(filename = system.file("demo_model", "par.txt", package = "HYPEtools"))
# Create optimization parameters for a Monte Carlo run with 1000 iterations
te2 <- data.frame(key = c("task", "num_mc", "task"), value = c("MC", 1000, "WS"))
# Create an optpar file structure for HYPE recession coefficients
te3 <- CreateOptpar(x = te1, pars = c("rrcs1", "rrcs2"), tasks = te2)
te3
```

CustomColors *Custom color ramp palettes*

Description

Pre-defined color ramp palettes which are used in other `HYPEtools` functions.

Usage

```
ColNitr(n)

ColPhos(n)

ColPrec(n)

ColTemp(n)

ColQ(n)

ColDiffTemp(n)

ColDiffGeneric(n)

ColBlues(n)
```

```
ColReds(n)

ColGreens(n)

ColYOB(n)

ColPurples(n)
```

Arguments

n	Integer, number of colors to generate.

Details

These functions build on calls to `colorRampPalette`.

Value

All functions return vectors of length `n` with interpolated RGB color values in hexadecimal notation (see `rgb`).

Examples

```
ColNitr(10)
ColGreens(6)
barplot(rep(1, 11), col = ColTemp(11))
```

DirectUpstreamSubids *Find Direct Upstream SUBIDs, with Flow Fractions*

Description

Function to find **direct** upstream SUBIDs including flow fractions for MAINDOWN/BRANCHDOWN splits for a single sub-catchment or all sub-catchments in a GeoData-like data frame.

Usage

```
DirectUpstreamSubids(subid = NULL, gd, bd = NULL)
```

Arguments

subid	Integer, SUBID of a target sub-catchment (must exist in gd), defaults to `NULL`. If non-`NULL`, `DirectUpstreamSubids` returns the direct upstream SUBIDs for this sub-catchment, otherwise for all sub-catchments in gd.
gd	Data frame, typically an imported 'GeoData.txt' file. Mandatory argument. See 'Details'.
bd	Data frame, typically an imported 'BranchData.txt' file. Optional argument, defaults to an empty placeholder. See 'Details'.

Details

DirectUpstreamSubids identifies **direct** upstream SUBIDs for a user-provided target SUBID or for all SUBIDs given in a data frame gd, typically an imported GeoData file.

A sub-catchment in HYPE can have several upstream sub-catchments. If there are more than one upstream sub-catchments, the downstream sub-catchment is a confluence. HYPE stores these connections in the GeoData file, in downstream direction, given as downstream SUBID in column 'MAINDOWN'. Bifurcations, i.e. splits in downstream direction, are also possible to model in HYPE. These additional downstream connections are provided in the BranchData file, together with flow fractions to each downstream SUBID.

Formally, gd can be any data frame which contains columns 'SUBID' and 'MAINDOWN' (not case-sensitive), and bd any data frame which contains three columns: 'BRANCHID', 'SOURCEID', and 'MAINPART', and optionally columns 'MAXQMAIN', 'MINQMAIN', 'MAXQBRANCH'. Typically, these are HYPE data files imported through ReadGeoData and ReadBranchData. See HYPE documentation for further details on connections Between SUBIDs in the model.

Value

DirectUpstreamSubids always returns a list. If argument subid is non-NULL, a list with two elements is returned: subid contains an integer giving the target SUBID and upstr.df contains a data frame with columns upstream (upstream SUBID), is.main (logical, TRUE if it is a MAINDOWN connection), fraction (fraction of flow going into the target SUBID), and llim and ulim giving upper and lower flow boundaries which optionally limit flow into the target SUBID.

If no specific SUBID was provided, DirectUpstreamSubids returns a list with upstream information for all SUBIDs in argument gd, each list element containing the list described above, i.e. with an integer element (SUBID) and a data frame element (upstream connections).

See Also

AllUpstreamSubids, which returns all upstream SUBIDs, i.e. the full upstream network up to the headwaters, for a given SUBID.

Examples

```
te <- ReadGeoData(filename = system.file("demo_model", "GeoData.txt", package = "HYPEtools"))
DirectUpstreamSubids(subid = 3594, gd = te)
```

---

distinctColorPalette *Generate optimally distinct color palettes*

Description

distinctColorPalette generates an attractive palette of random colors.

Usage

```
distinctColorPalette(count = 1, seed = NULL, darken = 0)
```

Arguments

count	Integer, number of colors (>= 1). May be ineffective for count > 40.
seed	Integer, seed number to produce repeatable palettes.
darken	Numeric specifying the amount of darkening applied to the color palette. See colorspace::darken. Negative values will lighten the palette.

Details

Adapted from the randomcoloR package https://cran.r-project.org/package=randomcoloR.

Value

distinctColorPalette returns a character vector of count optimally distinct colors in hexadecimal codes.

Examples

```
distinctColorPalette()
```

EquallySpacedObs *Create an equally spaced time series from irregular observations*

Description

EquallySpacedObs creates equally spaced time series with missing observations from a data frame with irregular observations.

Usage

```
EquallySpacedObs(x, sort.data = TRUE, timestep, ts.col = 1)
```

Arguments

x	A data.frame, with a date-time column in POSIXt or Date format, and one or several columns with observed variables.
sort.data	Logical, if TRUE, x will be sorted by date-time.
timestep	Character string keyword, giving the target time step length. Either "day" or "hour".
ts.col	Integer, column index of datetime column.

Details

EquallySpacedObs will preserve additional attributes present in x. If datetime column is of class Date, there may occur problems with daylight saving time shifts. To avoid problems, use class POSIXct and set time zone to "UTC".

Value

EquallySpacedObs returns a dataframe.

Examples

```
te <- data.frame(date = as.POSIXct(c("2000-01-01", "2000-02-01"), tz = "gmt"), obs = c(1, 2))
EquallySpacedObs(x = te, timestep = "day")
```

ExtractFreq — *Extract quantiles for use in a frequency distribution plot, e.g. a flow duration curve*

Description

This function calculates quantiles suitable for duration curves of environmental time series data.

Usage

```
ExtractFreq(
  data,
 probs = c(0, 1e-05, 1e-04, 0.001, seq(0.01, 0.99, by = 0.01), 0.999, 0.9999, 0.99999,
    1)
)
```

Arguments

data	either a numeric vector or an all-numeric dataframe (NAs allowed) which holds the variables for which quantiles are computed.
probs	numeric, vector of probabilities as in quantile with default suitable for flow duration curves.

Details

ExtractFreq is a convenience wrapper function, it uses quantile to calculate the quantiles of one or more time series with a density appropriate for duration curves. NAs are allowed in the input data. For the results to be meaningful, input should represent equally-spaced time series, e.g. HYPE basin output files.

Value

ExtractFreq returns a dataframe with probabilities in the first column, and quantiles of data in the following columns. Number of observations per variable in data are given in an attribute n.obs (see attributes).

See Also

PlotDurationCurve

Examples

```
ExtractFreq(rnorm(1000))
```

`ExtractStats` *Extract statistics from time series*

Description

Calculate aggregated statistics from long-term time series, typically imported HYPE time output files.

Usage

```
ExtractStats(
  x,
  start.mon = 1,
  aggperiod = c("year", "season1", "season2", "month"),
  timestep = attr(x, "timestep"),
  subid = attr(x, "subid"),
  FUN,
  ...
)
```

Arguments

`x`	Data frame, with column-wise equally-spaced time series. Date-times in `POSIXct` format in first column. Typically an imported time output file from HYPE.
`start.mon`	Integer between 1 and 12, starting month of the hydrological year.
`aggperiod`	Character string, timestep for aggregated results. One of `"year"` for annual statistics, `"season1"` (winter/summer half-years), `"season2"` (4 seasons), or `"month"`. See details.
`timestep`	Character string, timestep of data in `x`. Attribute `timestep` in `x` per default. Otherwise one of `"month"` (not allowed with `aggperiod = "month"`), `"week"`, `"day"`, or `"nhour"` (n = number of hours).
`subid`	Integer, a vector of HYPE subbasin IDs for data in `x`. Attribute `subid` in `x` per default.
`FUN`	A function to compute for each `aggperiod` in `x`.
`...`	Optional arguments to `FUN`.

Details

`ExtractStats` uses `aggregate` to calculate statistics for all data columns provided in `x`. Argument `start.mon` allows to define the start of the hydrological year. Hydrological seasons begin with winter (season1) or autumn (season2).

Value

ExtractStats returns a dataframe with starting dates for each aggregation period in the first column, and a descriptive aggregation period name in the second. Remaining columns contain aggregated results as ordered in x. Additional attributes subid with subbasin IDs, timestep with time step of the source data, and period with a two-element POSIXct vector containing start and end dates of the source data.

Note

If FUN returns several values per aggregation period, these are returned in nested columns in the resulting dataframe. See Value section for aggregate and example code below.

Examples

```
# Import example data
te1 <- ReadTimeOutput(filename = system.file("demo_model", "results",
"timeCOUT.txt", package = "HYPEtools"), dt.format = "%Y-%m")
# Extract maxima
ExtractStats(x = te1, start.mon = 1, FUN = max)
# Multiple result stats: Extract min, mean, and max in one go:
te2 <- ExtractStats(x = te1, start.mon = 1,
FUN = function(x) {c(min(x), mean(x), max(x))})
# extract mean from resulting nested dataframe:
data.frame(te2[, 1:2], sapply(te2[, -c(1:2)], function(x){x[, 2]}))
```

GOF *Goodness of Fit Functions*

Description

Numerical goodness-of-fit measures between sim and obs, with treatment of missing values.

Usage

```
gof(sim, obs, ...)

## Default S3 method:
gof(
  sim,
  obs,
  na.rm = TRUE,
  do.spearman = FALSE,
  s = c(1, 1, 1),
  method = c("2009", "2012"),
  start.month = 1,
  digits = 2,
  fun = NULL,
```

```
  ...,
  epsilon.type = c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
  epsilon.value = NA
)

valindex(sim, obs, ...)

## Default S3 method:
valindex(sim, obs, ...)

rPearson(sim, obs, ...)

## Default S3 method:
rPearson(
  sim,
  obs,
  fun = NULL,
  ...,
  epsilon.type = c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
  epsilon.value = NA
)

sKGE(sim, obs, ...)

## Default S3 method:
sKGE(
  sim,
  obs,
  s = c(1, 1, 1),
  na.rm = TRUE,
  method = c("2009", "2012"),
  start.month = 1,
  out.PerYear = FALSE,
  fun = NULL,
  ...,
  epsilon.type = c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
  epsilon.value = NA
)

KGE(sim, obs, ...)

## Default S3 method:
KGE(
  sim,
  obs,
  s = c(1, 1, 1),
  na.rm = TRUE,
  method = c("2009", "2012", "2021"),
```

```
  out.type = c("single", "full"),
  fun = NULL,
  ...,
  epsilon.type = c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
  epsilon.value = NA
)

NSE(sim, obs, ...)

## Default S3 method:
NSE(
  sim,
  obs,
  na.rm = TRUE,
  fun = NULL,
  ...,
  epsilon.type = c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
  epsilon.value = NA
)

pbias(sim, obs, ...)

## Default S3 method:
pbias(
  sim,
  obs,
  na.rm = TRUE,
  dec = 1,
  fun = NULL,
  ...,
  epsilon.type = c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
  epsilon.value = NA
)

mae(sim, obs, ...)

## Default S3 method:
mae(
  sim,
  obs,
  na.rm = TRUE,
  fun = NULL,
  ...,
  epsilon.type = c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
  epsilon.value = NA
)

VE(sim, obs, ...)
```

```
## Default S3 method:
VE(
  sim,
  obs,
  na.rm = TRUE,
  fun = NULL,
  ...,
  epsilon.type = c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
  epsilon.value = NA
)
```

Arguments

sim	numeric, vector of simulated values
obs	numeric, vector of observed values
...	further arguments passed to/from other methods.
na.rm	a logical value indicating whether 'NA' should be stripped before the computation proceeds. When an 'NA' value is found at the i-th position in obs OR sim, the i-th value of obs AND sim are removed before the computation.
do.spearman	logical, indicates if the Spearman correlation should be computed. The default is FALSE.
s	argument passed to the KGE function.
method	argument passed to the KGE function.
start.month	argument passed to the sKGE function.
digits	integer, numer of decimal places used for rounding the goodness of fit indexes.
fun	function to be applied to sim and obs in order to obtain transformed values thereof before applying any goodness-of-fit function
epsilon.type	argument used to define a numeric value to be added to both sim and obs before applying fun. It was designed to allow the use of logarithm and other similar functions that do not work with zero values. It must be one of the following possible values: • *none*: no value added to sim or obs. • *Pushpalatha2012*: one hundredth of the mean observed values is added to both sim and obs as described in Pushpalatha et al., 2012. • *otherFactor*: the numeric value defined in epsilon.value is used to multiply the mean observed values instead of the one hundredth (1/100) described in Pushpalatha et al., (2012). The resulting value is then added to both sim and obs. • *otherValue*: the numeric value defined in epsilon.value is directly added to both sim and obs.
epsilon.value	numeric, value to be added to both sim and obs when epsilon = "otherValue".
out.PerYear	logical, argument passed to the sKGE function.
out.type	argument passed to the KGE function.
dec	argument passed to the pbias function.

Details

The gof, mae, pbias, NSE, rPearson, sKGE, and KGE functions are provided to calculate goodness of fit statistics. The functions were adapted from the hydroGOF package https://github.com/hzambran/hydroGOF.

Value

gof Returns a matrix of goodness of fit statistics. mae, pbias, NSE, rPearson, sKGE, and KGE return a numeric of the goodness of fit statistic.

Examples

```
gof(sim = sample(1:100), obs = sample(1:100))
```

GroupSLCClasses *Calculate grouped sums for SLC classes in a GeoData file*

Description

GroupSLCClasses calculates grouped sums for SLC classes (area fractions or absolute areas) based on land use, soil, or crop groups in a GeoClass table, or any other user-provided grouping index.

Usage

```
GroupSLCClasses(
  gd,
  gcl = NULL,
  type = c("landuse", "soil", "crop"),
  group = NULL,
  abs.area = FALSE,
  verbose = TRUE
)
```

Arguments

gd	Data frame containing columns with SUBIDs, SLC fractions, and SUBID areas if abs.area = TRUE. Typically a 'GeoData.txt' file imported with ReadGeoData.
gcl	Data frame containing columns with SLCs and corresponding landuse and soil class IDs, typically a 'GeoClass.txt' file imported with ReadGeoClass. Must be provided if no group argument is given.
type	Character string keyword for use with gcl. Type of grouping index, either "landuse", "soil", or "crop", can be abbreviated.
group	Integer vector, of same length as number of SLC classes in gd. Alternative grouping index specification to gcl + type.
abs.area	Logical, if TRUE, then absolute areas will be calculated for each group, rather than area fractions.
verbose	Logical, if TRUE then information and progress bar will be printed.

Details

If absolute areas are calculated, area units will correspond to areas provided in gd.

Value

GroupSLClasses returns the data frame with SUBIDs, SUBID areas, and grouped SLC class columns.

Examples

```
# Import source data
te1 <- ReadGeoData(filename = system.file("demo_model", "GeoData.txt", package = "HYPEtools"))
te2 <- ReadGeoClass(filename = system.file("demo_model", "GeoClass.txt", package = "HYPEtools"))
# Calculate soil groups
GroupSLCClasses(gd = te1, gcl = te2, type = "s")
```

GwRetention *Calculate groundwater retention of nutrients*

Description

Function to calculate nutrient load retention fractions in groundwater parts of HYPE, i.e. after root zone retention. See Details for exact definition.

Usage

```
GwRetention(nfrz, nfs3, gts3, gd, par, unit.area = TRUE, nutrient = "tn")
```

Arguments

nfrz	Data frame with two-columns. Sub-basin IDs in first column, net loads from root zone in kg/year in second column. Typically an imported HYPE map output file, HYPE output variable SL06. See Details.
nfs3	Data frame with two-columns. Sub-basin IDs in first column, net loads from soil layer 3 in kg/year in second column. Typically an imported HYPE map output file, HYPE output variable SL18. See Details.
gts3	Data frame with two-columns. Sub-basin IDs in first column, gross loads to soil layer 3 in kg/year in second column. Typically an imported HYPE map output file, HYPE output variable SL17. See Details.
gd	Data frame, with columns containing sub-basin IDs and rural household emissions, e.g. an imported 'GeoData.txt' file. See details.
par	List, HYPE parameter list, typically an imported 'par.txt' file. Must contain parameter *locsoil* (not case-sensitive).
unit.area	Logical, set to FALSE to calculate incoming load (leaching rates) in kg/year instead of kg/(ha year).

`nutrient` Character keyword, one of the HYPE-modeled nutrient groups, for which to calculate groundwater retention. Not case-sensitive. *Currently, only* `tn` *(total nitrogen) is implemented.*

Details

`GwRetention` calculates a groundwater nutrient retention as fractions of outgoing and incoming loads using HYPE soil load variables. Incoming loads include drainage into layer 3 from the root zone (defined as soil layer 1 and 2), rural load fractions into soil (dependent on parameter *locsoil*), tile drainage, surface flow, and flow from layer 1 and 2. Outgoing loads include runoff from all soil layers, tile drain, and surface flow.

The retention fraction *R* is calculated as (see also the variable description in the HYPE online documentation):

$R = 1 - \frac{OUT}{IN} = 1 - \frac{nfrz - gts3 + nfs3 + locsoil * lr}{nfrz + locsoil * lr}$

$li = nfrz + locsoil * lr$ [kg/y]

$lr = LOC_V OL * LOC_T N * 0.365$ [kg/y]

, where *li* is incoming load to groundwater (leaching rates), *lr* is rural load (total from GeoData converted to kg/yr; *locsoil* in the formula converts it to rural load into soil layer 3), and *nfrz*, *gts3*, *nfs3* are soil loads as in function arguments described above. See Examples for HYPE variable names for `TN` loads.

Columns `SUBID`, `LOC_VOL`, and `LOC_TN` must be present in `gd`, for a description of column contents see the GeoData file description in the HYPE online documentation. Column names are not case-sensitive.

Value

`GwRetention` returns a three-column data frame, containing SUBIDs, retention in groundwater as a fraction of incoming loads (if multiplied by 100, it becomes \ .

Examples

```
# Create dummy data
te1 <- ReadGeoData(filename = system.file("demo_model",
"GeoData.txt", package = "HYPEtools"))
te1$loc_tn <- runif(n = nrow(te1), min = 0, max = 100)
te1$loc_vol <- runif(n = nrow(te1), min = 0, max = 2)
te2 <- ReadPar(filename = system.file("demo_model",
"par.txt", package = "HYPEtools"))
te2$locsoil <- .3
# HYPE soil load (sl) variables for TN, dummy loads
GwRetention(nfrz = data.frame(SUBID = te1$SUBID, SL06 = runif(n = nrow(te1), 10, 50)),
            gts3 = data.frame(SUBID = te1$SUBID, SL17 = runif(n = nrow(te1), 10, 50)),
            nfs3 = data.frame(SUBID = te1$SUBID, SL18 = runif(n = nrow(te1), 10, 50)),
            gd = te1, par = te2)
```

HeadwaterSubids *Find all headwater SUBIDs of a model domain*

Description

Function to find all headwater SUBIDs of a HYPE model domain.

Usage

```
HeadwaterSubids(gd)
```

Arguments

gd	A data frame, containing among others two columns subid and maindown. Column names are not case-sensitive and column positions in the data frame are irrelevant. Typically a 'GeoData.txt' file imported using ReadGeoData.

Details

HeadwaterSubids finds all headwater SUBIDs of a model domain as provided in a 'GeoData.txt' file, i.e. all subcatchments which do not have any upstream subcatchments.

Value

HeadwaterSubids returns a vector of outlet SUBIDs.

See Also

AllUpstreamSubids

Examples

```
te <- ReadGeoData(filename = system.file("demo_model", "GeoData.txt", package = "HYPEtools"))
HeadwaterSubids(gd = te)
```

HypeAttrAccess *Quickly query and set HYPE-specific attributes*

Description

These are simple convenience wrapper functions to quickly query and assign values of attributes which are added to HYPE data on import.

Usage

```
datetime(x)

datetime(x) <- value

hypeunit(x)

hypeunit(x) <- value

obsid(x)

obsid(x) <- value

outregid(x)

outregid(x) <- value

subid(x)

subid(x) <- value

timestep(x)

timestep(x) <- value

variable(x)

variable(x) <- value
```

Arguments

x	Object whose attribute is to be accessed
value	Value to be assigned

Details

These functions are just shortcuts for attr.

Value

The extractor functions return the value of the respective attribute or NULL if no matching attribute is found.

Examples

```
te <- ReadBasinOutput(filename = system.file("demo_model", "results",
"0003587.txt", package = "HYPEtools"))
hypeunit(te)
timestep(te)
```

```
subid(te)
```

HypeDataExport *Write HYPE data files*

Description

These are simple convenience wrapper functions to export various HYPE data files from R.

Usage

```
WriteAquiferData(x, filename, verbose = TRUE)

WriteOutregions(x, filename, verbose = TRUE)

WriteBranchData(x, filename, verbose = TRUE)

WriteCropData(x, filename, verbose = TRUE)

WriteDamData(x, filename, verbose = TRUE)

WriteFloodData(x, filename, verbose = TRUE)

WriteLakeData(x, filename, verbose = TRUE)

WriteMgmtData(x, filename, verbose = TRUE)

WritePointSourceData(x, filename, verbose = TRUE)

WriteForcKey(x, filename)

WriteGlacierData(x, filename, verbose = TRUE)
```

Arguments

x	The object to be written, a dataframe as returned from the HypeDataImport functions.
filename	A character string naming a path and file name to write to. Windows users: Note that Paths are separated by '/', not '\'.
verbose	Logical, display informative warning messages if columns contain NA values or if character strings are too long. See Details.

Details

Hype data file exports, simple `fwrite` wrappers with formatting options adjusted to match HYPE file specifications:

- LakeData.txt
- DamData.txt
- MgmtData.txt
- AquiferData.txt
- PointSourceData.txt
- GlacierData.txt
- CropData.txt
- BranchData.txt
- forckey.txt
- Outregions.txt

In most files, HYPE requires `NA`-free input in required columns, but empty values are allowed in additional comment columns which are not read by HYPE. Informative warnings will be thrown if `NA`s are found during export. Character string lengths in comment columns of HYPE data files are restricted to 100 characters, the functions will return with a warning if longer strings were exported.

Value

No return value, called for export to text files.

Examples

```
te <- ReadForcKey(filename = system.file("demo_model", "ForcKey.txt", package = "HYPEtools"))
WriteForcKey(x = te, filename = tempfile())
```

`HypeDataImport` *Read HYPE data files*

Description

These are simple convenience wrapper functions to import various HYPE data files as data frame into R.

Usage

```
ReadAquiferData(
  filename = "AquiferData.txt",
  verbose = TRUE,
  header = TRUE,
  na.strings = "-9999",
  sep = "\t",
  stringsAsFactors = FALSE,
  encoding = c("unknown", "latin1", "UTF-8"),
  ...
)

ReadOutregions(
  filename = "Outregions.txt",
  verbose = TRUE,
  header = TRUE,
  na.strings = "-9999",
  sep = "\t",
  stringsAsFactors = FALSE,
  encoding = c("unknown", "latin1", "UTF-8"),
  ...
)

ReadBranchData(
  filename = "BranchData.txt",
  verbose = TRUE,
  header = TRUE,
  na.strings = "-9999",
  sep = "\t",
  stringsAsFactors = FALSE,
  encoding = c("unknown", "latin1", "UTF-8"),
  ...
)

ReadCropData(
  filename = "CropData.txt",
  verbose = TRUE,
  header = TRUE,
  na.strings = "-9999",
  sep = "\t",
  stringsAsFactors = FALSE,
  encoding = c("unknown", "latin1", "UTF-8"),
  ...
)

ReadDamData(
  filename = "DamData.txt",
  verbose = TRUE,
```

```
  header = TRUE,
  na.strings = "-9999",
  sep = "\t",
  quote = "",
  stringsAsFactors = FALSE,
  encoding = c("unknown", "latin1", "UTF-8"),
  ...
)

ReadFloodData(
  filename = "FloodData.txt",
  verbose = TRUE,
  header = TRUE,
  na.strings = "-9999",
  sep = "\t",
  quote = "",
  stringsAsFactors = FALSE,
  encoding = c("unknown", "latin1", "UTF-8"),
  ...
)

ReadGlacierData(
  filename = "GlacierData.txt",
  verbose = TRUE,
  header = TRUE,
  na.strings = "-9999",
  sep = "\t",
  stringsAsFactors = FALSE,
  encoding = c("unknown", "latin1", "UTF-8"),
  ...
)

ReadLakeData(
  filename = "LakeData.txt",
  verbose = TRUE,
  header = TRUE,
  na.strings = "-9999",
  sep = "\t",
  quote = "",
  stringsAsFactors = FALSE,
  encoding = c("unknown", "latin1", "UTF-8"),
  ...
)

ReadMgmtData(
  filename = "MgmtData.txt",
  verbose = TRUE,
  header = TRUE,
```

```
  na.strings = "-9999",
  sep = "\t",
  stringsAsFactors = FALSE,
  encoding = c("unknown", "latin1", "UTF-8"),
  ...
)

ReadPointSourceData(
  filename = "PointSourceData.txt",
  verbose = TRUE,
  header = TRUE,
  na.strings = "-9999",
  sep = "\t",
  stringsAsFactors = FALSE,
  encoding = c("unknown", "latin1", "UTF-8"),
  data.table = FALSE,
  ...
)

ReadAllsim(filename = "allsim.txt", na.strings = "-9999")

ReadForcKey(
  filename = "ForcKey.txt",
  sep = "\t",
  encoding = c("unknown", "latin1", "UTF-8")
)

ReadUpdate(
  filename = "update.txt",
  header = TRUE,
  sep = "\t",
  stringsAsFactors = FALSE,
  encoding = c("unknown", "latin1", "UTF-8"),
  data.table = FALSE,
  ...
)
```

Arguments

filename	Path to and file name of HYPE data file file to import. Windows users: Note that Paths are separated by '/', not '\'.
verbose	Logical, display message if columns contain NA values.
header	read.table or fread argument, with appropriate default for HYPE data file import.
na.strings	See header.
sep	See header.
stringsAsFactors	See header.

encoding	read.table argument. Specify character encoding when importing files created under Windows (default encoding "latin1") in Linux (default encoding "UTF-8") and vice versa.
...	Other parameters passed to read.table.
quote	See header.
data.table	Logical, return data.table instead of data frame. fread argument.

Details

Hype data file imports, simple read.table or fread wrappers with formatting arguments set to match HYPE file specifications:

- LakeData.txt
- DamData.txt
- MgmtData.txt
- AquiferData.txt
- PointSourceData.txt
- GlacierData.txt
- CropData.txt
- BranchData.txt
- Outregions.txt
- allsim.txt
- update.txt

In most files, HYPE requires NA-free input in required columns, but empty values are allowed in additional comment columns. Informative warnings will be thrown if NAs are found during import.

Value

Imported files are returned as data frames.

Examples

```
te <- ReadForcKey(filename = system.file("demo_model", "ForcKey.txt", package = "HYPEtools"))
```

HypeGeoData *HypeGeoData data frames*

Description

Constructor function for data frames which hold HYPE GeoData tables with information on sub-basins.

Usage

```
HypeGeoData(x)
```

Arguments

x	Data frame with at least five mandatory columns, see details.

Details

S3 constructor function for data frames which hold HYPE GeoData tables. These are normal data frames with at least five mandatory columns, all `numeric`: *AREA*, *SUBID*, *MAINDOWN*, *RIVLEN*, and *SLC_n*, where *n* are consecutive SLC class numbers (up to 999).See also the HYPE file description for GeoData.txt files for reference.

Usually, this class will be assigned to GeoData tables on import with `ReadGeoData`. A `summary` method exists for `HypeGeoData` data frames.

Value

Returns a data frame with added `class` attribute `HypeGeoData`.

See Also

`ReadGeoData`

Examples

```
te <- data.table::fread(file = system.file("demo_model",
"GeoData.txt", package = "HYPEtools"), data.table = FALSE)
HypeGeoData(x = te)
summary(te)
```

HypeMultiVar *HypeMultiVar arrays*

Description

Constructor function for arrays which hold equidistant time series of multiple HYPE variables for a single sub-basin and multiple model runs, typically imported HYPE basin output results.

Usage

```
HypeMultiVar(
  x,
  datetime,
  hype.var,
  hype.unit,
  subid = NULL,
  outregid = NULL,
  hype.comment = ""
)
```

Arguments

x	numeric array with three dimensions, which holds HYPE results for one sub-basin as (in order) [datetime, variable, iteration].
datetime	POSIXct date-time vector of the same length as time dimension of x with equidistant time steps (starting day for time steps from weekly to annual), or character string for full model period averages, e.g. "2000-2010".
hype.var, hype.unit	Character vectors of keywords to specify HYPE variable IDs, corresponding to second dimension (columns) in x. See list of HYPE variables
subid	Integer, HYPE sub-basin ID. Either this or outregid needs to be supplied.
outregid	Integer, HYPE output region ID, alternative to subid.
hype.comment	Character, first-row optional comment string of basin output file. Empty string, if non-existing.

Details

S3 class constructor function for array objects which can hold (multiple) HYPE basin output results.

Value

Returns a 3-dimensional array with [time, variable, iteration] dimensions and additional attributes:

datetime A vector of date-times. Corresponds to 1st array dimension.

variable A character vector of HYPE output variable IDs.

hypeunit A character vector of HYPE output variable units.

subid A single SUBID.

outregid A single OUTREGID.

timestep A character keyword for the time step.

comment A comment string, currently used for class group outputs.

Examples

```
# import a basin output file
te1 <- ReadBasinOutput(filename = system.file("demo_model",
"results", "0003587.txt", package = "HYPEtools"))
# create a dummy array with two iterations from imported basin file
te2 <- array(data = c(unlist(te1[, -1]),  unlist(te1[, -1])),
             dim = c(nrow(te1), ncol(te1) - 1, 2),
             dimnames = list(rownames(te1), colnames(te1)[-1]))
# Construct HypeMultiVar array
HypeMultiVar(te2, datetime = te1$DATE, hype.var = variable(te1),
hype.unit = hypeunit(te1), subid = 3587)
```

`HypeSingleVar` *HypeSingleVar arrays*

Description

Constructor function for arrays which hold equidistant time series of a single HYPE variable for multiple sub-basins and multiple model runs, typically imported time and map output results.

Usage

```
HypeSingleVar(x, datetime, subid = NULL, outregid = NULL, hype.var)
```

Arguments

`x`	numeric `array` with three dimensions, which holds HYPE results for one variable as (in order) `[datetime, subid/outregid, iteration]`.
`datetime`	`POSIXct` date-time vector of the same length as `time` dimension of `x` with equidistant time steps (starting day for time steps from weekly to annual), or character string for full model period averages, e.g. `"2000-2010"`.
`subid`	Integer vector with HYPE sub-basin IDs, of the same length as `subid` dimension of `x`. Either this or `outregid` must be supplied.
`outregid`	Integer vector with HYPE output region IDs, alternative to `subid`.
`hype.var`	Character string, keyword to specify HYPE variable ID, see list of HYPE variable. Not case-sensitive.

Details

S3 class constructor function for array objects which can hold (multiple) HYPE time or map output results.

Value

Returns a 3-dimensional array with `[time, subid, iteration]` dimensions and additional `attributes`:

datetime A vector of date-times. Corresponds to 1st array dimension.

subid A vector of SUBIDs. Corresponds to 2nd array dimension (`NA`, if it does not apply to data contents).

outregid A vector of OUTREGIDs. Corresponds to 2nd array dimension (`NA`, if it does not apply to data contents).

variable HYPE output variable ID.

timestep A character keyword for the time step.

Examples

```
# Import a time output file
te1 <- ReadTimeOutput(filename = system.file("demo_model", "results",
"timeCOUT.txt", package = "HYPEtools"), dt.format = "%Y-%m")
# Create a dummy array with two iterations from imported time file
te2 <- array(data = c(unlist(te1[, -1]),  unlist(te1[, -1])),
             dim = c(nrow(te1), ncol(te1) - 1, 2),
             dimnames = list(rownames(te1), colnames(te1)[-1]))
# Construct HypeSingleVar array
HypeSingleVar(x = te2, datetime = te1$DATE,
subid = subid(te1), hype.var = variable(te1))
```

HypeSubidChecks *Check HYPE SUBID properties*

Description

Quickly query vectors of HYPE sub-basin IDs (SUBID) for various properties.

Usage

```
IsHeadwater(subid, gd)

IsOutlet(subid, gd)

IsRegulated(subid, gd, dd = NULL, ld = NULL)
```

Arguments

subid	Numeric, vector of SUBIDs to be queried
gd	HypeGeoData or base data frame with columns SUBID and MAINDOWN, typically an imported GeoData.txt file.
dd	Data frame, typically an imported DamData.txt file. Defaults to NULL. dd or ld has to be provided in IsRegulated.
ld	Data frame, typically an imported LakeData.txt file. Defaults to NULL. dd or ld has to be provided in IsRegulated.

Details

These are convenience functions to query subbasin properties. Some functions can be inefficient if applied to many or all subbasins of a HYPE model setup and more efficient functions may exist in HYPEtools, see links in *See also* section below or browse the package index.

Value

The functions return a logical vector of the same length as subid, with NA values for all SUBIDs which do not exist in gd.

See Also

AllUpstreamSubids(); AllDownstreamSubids(); OutletSubids(); OutletIds()

Examples

```
te <- ReadGeoData(filename = system.file("demo_model", "GeoData.txt", package = "HYPEtools"))
IsHeadwater(subid = 40556, gd = te)
IsHeadwater(subid = te$SUBID, gd = te)
```

HypeXobs *HypeXobs data frames*

Description

Constructor function for data frames which hold HYPE Xobs.txt file contents, i.e. time series of a multiple observation variables for multiple sub-basins and equidistant time steps in POSIXct format in the first column.

Usage

```
HypeXobs(x, comment, variable, subid, verbose = TRUE)
```

Arguments

x	data.frame with POSIXct formatted time steps in the first, and numeric variables in the remaining columns.
comment	Character string, metadata or other information, first line of a HYPE Xobs.txt file.
variable	Character vector of four-letter keywords to specify HYPE variable IDs, corresponding to second to last column in x.
subid	Integer vector with HYPE sub-basin IDs, corresponding to second to last column in x.
verbose	Logical, throw warning if attribute timestep cannot be computed. Not case-sensitive.

Details

S3 class constructor function for HypeXobs data frame objects which hold HYPE Xobs.txt file contents. Xobs.txt files contain three header rows, see the Xobs.txt description in the HYPE documentation. These headers are stored as additional attributes in objects.

Value

Returns a data frame of class HypeXobs with additional attributes:

comment A character vector.

variable A character vector of HYPE variable IDs.

subid A vector of SUBIDs.

timestep Time step keyword, "day", or "n hour" (n = number of hours). NULL, if x contains just one row.

Examples

```
# Use the Xobs file import function instead of the class constructor for standard work flows
te <- ReadXobs(file = system.file("demo_model", "Xobs.txt", package = "HYPEtools"))
summary(te)
# Class constructor
HypeXobs(x = as.data.frame(te), comment = comment(te), variable = variable(te), subid = subid(te))
```

InfoManipulation *Functions to Manipulate HYPE Info Files*

Description

Add/Remove lines to HYPE info.txt files

Usage

```
AddInfoLine(info, name, value, after = NULL)

RemoveInfoLine(info, name)
```

Arguments

info	Named list containing the info.txt file data, typically created using ReadInfo with the exact mode.
name	Name of info.txt code to add/remove.
value	Value of the info.txt code to add/remove.
after	String vector containing the name(s) of info.txt codes that the new info.txt code should be inserted below. If multiple values are specified and all codes are present in info, then the new code will be inserted below the match that is farthest down in the info.txt file.

Details

The AddInfoLine and RemoveInfoLine functions provide features to add/remove lines to an imported info.txt file. Info.txt codes can be found on the HYPE Wiki.

Value

AddInfoLine and RemoveInfoLine return a named list in the info.txt file structure.

Examples

```
info <- ReadInfo(filename = system.file("demo_model",
"info.txt", package = "HYPEtools"))
info <- AddInfoLine(info, name = "testline", value = "testvalue")
info <- RemoveInfoLine(info, name = "testline")
```

MapRegionalSources *Map regional irrigation source connection as spatial lines*

Description

By default, this function creates an sf object which contains regional irrigation connections between source and target HYPE sub-catchments. However, this function can also be used to create interactive Leaflet maps.

Usage

```
MapRegionalSources(
  data,
  map,
  map.subid.column = 1,
  group.column = NULL,
  group.colors = NULL,
  digits = 3,
  progbar = FALSE,
  map.type = "default",
  plot.scale = TRUE,
  plot.searchbar = FALSE,
  weight = 0.5,
  opacity = 1,
  fillColor = "#4d4d4d",
  fillOpacity = 0.25,
  line.weight = 5,
  line.opacity = 1,
  seed = NULL,
  darken = 0,
  font.size = 10,
  file = "",
  vwidth = 1424,
  vheight = 1000,
  html.name = ""
)
```

Arguments

data	Dataframe, containing a column SUBID and a column REGSRCID (not case-sensitive), which identify irrigation target and source sub-catchments, respectively. Typically a HYPE 'MgmtData.txt' file, imported with ReadMgmtData.
map	A sf, SpatialPointsDataFrame, or SpatialPolygonsDataFrame object providing sub-catchment locations as points or polygons. Typically an imported SUBID center-point shape file or geopackage. If provided polygon data, then the polygon centroids will be calculated and used as the point locations (See sf::st_centroid()). Spatial data import requires additional packages, e.g. sf.
map.subid.column	Integer, index of the column in map holding SUBIDs (sub-catchment IDs).
group.column	Integer, optional index of the column in data providing grouping of connections to allow toggling of groups in Leaflet maps. Default NULL will produce maps without grouping.
group.colors	Named list providing colors for connection groups in Leaflet maps. List names represent the names of the groups in the group.column of data, and list values represent the colors. Example: groups.colors = list("GROUP 1" = "black",

	"GROUP 2" = "red"). If a group is not included in group.colors, then random colors will be assigned to the connections in the group. Default NULL will produce maps using random colors for all groups.
digits	Integer, number of digits to which irrigation connection lengths are rounded to.
progbar	Logical, display a progress bar while calculating.
map.type	Map type keyword string. Choose either "default" for the default static plots or "leaflet" for interactive Leaflet maps.
plot.scale	Logical, include a scale bar on Leaflet maps.
plot.searchbar	Logical, if TRUE, then a search bar will be included on Leaflet maps. See leaflet.extras::addSearchFeatures().
weight	Numeric, weight of subbasin boundary lines in Leaflet maps. Used if map contains polygon data. See leaflet::addPolygons().
opacity	Numeric, opacity of subbasin boundary lines in Leaflet maps. Used if map contains polygon data. See leaflet::addPolygons().
fillColor	String, color of subbasin polygons in Leaflet maps. Used if map contains polygon data. See leaflet::addPolygons().
fillOpacity	Numeric, opacity of subbasin polygons in Leaflet maps. Used if map contains polygon data. See leaflet::addPolygons().
line.weight	Numeric, weight of connection lines in Leaflet maps. See leaflet::addPolylines().
line.opacity	Numeric, opacity of connection lines in Leaflet maps. See leaflet::addPolylines().
seed	Integer, seed number to to produce repeatable color palette.
darken	Numeric specifying the amount of darkening applied to the random color palette. Negative values will lighten the palette. See distinctColorPalette.
font.size	Numeric, font size (px) for subbasin labels in Leaflet maps.
file	Save a Leaflet map to an image file by specifying the path to the desired output file using this argument. File extension must be specified. See mapview::mapshot(). You may need to run webshot::install_phantomjs() the first time you save a map to an image file.
vwidth	Numeric, width of the exported Leaflet map image in pixels. See webshot::webshot().
vheight	Numeric, height of the exported Leaflet map image in pixels. See webshot::webshot().
html.name	Save a Leaflet map to an interactive HTML file by specifying the path to the desired output file using this argument. File extension must be specified. See htmlwidgets::saveWidget().

Details

MapRegionalSources can return static plots or interactive Leaflet maps depending on value provided for the argument map.type. By default, MapRegionalSources creates an sf object from HYPE SUBID centerpoints using a table of SUBID pairs. Regional irrigation sources in HYPE are transfers from outlet lakes or rivers in a source sub-catchment to the soil storage of irrigated SLC classes (Soil, Land use, Crop) in a target sub-catchment. If map.type is set to "leaflet", then MapRegionalSources returns an object of class leaflet.

Value

For default static maps, `MapRegionalSources` returns an `sf` object containing columns `SUBID` (irrigation target sub-catchment), `REGSRCID` (irrigation source sub-catchment), and `Length_[unit]` (distance between sub-catchments) where 'unit' is the actual length unit of the distances. The projection of the returned object is always identical to the projection of argument `map`. For interactive Leaflet maps, `PlotMapOutput` returns an object of class `leaflet`. If `map` contains polygon data, then the interactive map will include the polygons as a background layer.

Examples

```
# Import subbasin centroids and subbasin polygons (to use as background)
require(sf)
te1 <- st_read(dsn = system.file("demo_model", "gis",
"Nytorp_centroids.gpkg", package = "HYPEtools"))
te2 <- st_read(dsn = system.file("demo_model", "gis",
"Nytorp_map.gpkg", package = "HYPEtools"))
# Create dummy MgmtData file with irrigation links
te3 <- data.frame(SUBID = c(3594, 63794), REGSRCID = c(40556, 3486))

# Plot regional irrigation links between subbasins with subbasin outlines as background
MapRegionalSources(data = te3, map = te1, map.subid.column = 25)
plot(st_geometry(te2), add = TRUE, border = 2)
```

---

merge *Merge HypeGeoData object*

Description

Merge an imported HYPE GeoData table of class `link{HypeGeoData}` with another data frame.

Usage

```
## S3 method for class 'HypeGeoData'
merge(x, y, all.x = TRUE, sort = NA, ...)
```

Arguments

`x`	`HypeGeoData` data frame, HYPE GeoData table to be extended with new columns.
`y`	Data frame, with mandatory `SUBID` column.
`all.x`	Logical, keep all rows from `x`. Defaults to `TRUE`, as opposed to default method, thus extending the GeoData table with columns in `y`.
`sort`	Logical, result sorting by by columns. In addition to the default method's choices `TRUE`, `FALSE`, a third option `NA` (default) will use sorting of `x` for results. I.e. a sorted GeoData table will be runnable in HYPE even after merging.
`...`	Arguments passed to S3 method for data frames, see `merge` and Details.

Details

`merge.HypeGeoData` allows to merge new columns to an existing HYPE GeoData table, while preserving the `HypeGeoData` class attribute. Duplicate columns are marked with a `".y"`-suffix for the merged `y` data frame.

The following arguments of the default method are hard-coded:

- `by`, `by.x`, `by.y`, set to `"SUBID"`
- `suffixes`, set to `c("", ".y")`

The method warns if any of these arguments is supplied by the user. To override, use the GeoData table as argument `y` or call the data frame method explicitly (`merge.data.frame()`).

Value

A `HypeGeoData` data frame.

See Also

`merge`, the S3 generic function.

Examples

```
# import and create dummy data
te1 <- ReadGeoData(filename = system.file("demo_model",
"GeoData.txt", package = "HYPEtools"))
te2 <- data.frame(SUBID = sample(x = te1$SUBID, size = 10),
loc_vol = runif(n = 10, 10, 50))
merge(x = te1, y = te2)
```

MergeObs *Merge two HYPE observation data frames*

Description

Function to merge two HYPE observation data frames, with handling of overlapping time periods and time periods gaps as well as merging of common columns.

Usage

```
MergeObs(x, y)
```

Arguments

`x`, `y`	Data frames containing observation timeseries data. Typically imported using `ReadObs`.

Details

MergeObs handles time steps of different lengths (e.g. daily, hourly), but requires identical time step lengths from both inputs data frames.

In case of common columns (identical date and SUBID combinations in x and y), values from columns in x will take precedence, and values from y will only be added if x values are missing.

Value

MergeObs returns a data frame with merged Obs data.

Examples

```
# Import dummy data, add new observations to second Obs table, and merge
te1 <- ReadObs(filename = system.file("demo_model", "Tobs.txt", package = "HYPEtools"))
te2 <- ReadObs(filename = system.file("demo_model", "Tobs.txt", package = "HYPEtools"))
te2$X0000[1:365] <- runif(n = 365, -20, 25)
MergeObs(x = te1, y = te2)
```

MergeXobs *Merge two Xobs data frames*

Description

Function to merge two Xobs data frames, with handling of overlapping time periods and time periods gaps as well as merging of common columns.

Usage

```
MergeXobs(x, y, comment = "")
```

Arguments

x, y	Data frames of class HypeXobs, including additional attributes comment, variable, subid, and timestep, typically imported using ReadXobs. For details on attribute format, see the class description. Class attribute not formally necessary.
comment	Character string, will be added to the result as attribute comment. If empty, comment attributes from x and y will be merged to new comment string.

Details

MergeXobs handles time steps of different lengths (e.g. daily, hourly), but requires identical time step lengths from both inputs data frames. The functions expects data frames of class HypeXobs or data frames with comparable structure and will throw a warning if the class attribute is missing.

In case of common columns (identical observation variable and SUBID combinations in x and y), values from columns in x will take precedence, and values from y will only be added if x values are missing

Value

MergeXobs returns a data frame with attributes for Xobs data.

Examples

```
# Import dummy data, add new observations to second Xobs table
te1 <- ReadXobs(filename = system.file("demo_model", "Xobs.txt", package = "HYPEtools"))
te2 <- ReadXobs(filename = system.file("demo_model", "Xobs.txt", package = "HYPEtools"))
te2$WSTR_40541[1:10] <- runif(n = 10, 50, 100)
MergeXobs(x = te1, y = te2)
```

NSE.HypeSingleVar *Nash-Sutcliffe Efficiency*

Description

Nash-Sutcliffe Efficiency calculation for imported HYPE outputs with single variables for several catchments, i.e. time and map files, optionally multiple model run iterations combined.

Usage

```
## S3 method for class 'HypeSingleVar'
NSE(sim, obs, na.rm = TRUE, progbar = TRUE, ...)
```

Arguments

sim	HypeSingleVar array with simulated variable (one or several iterations).
obs	HypeSingleVar array with observed variable, (one iteration). If several iterations are present in the array, only the first will be used.
na.rm	Logical. If TRUE, incomplete sim-obs pairs will be removed prior to NSE computation.
progbar	Logical, if TRUE progress bars will be printed for main computational steps.
...	ignored

Value

NSE.HypeSingleVar returns a 2-dimensional array of NSE performances for all SUBIDs and model iterations provided in argument sim, with values in the same order as the second and third dimension in sim, i.e. [subid, iteration].

Examples

```
# Create dummy data, discharge observations with added white noise as model simulations
te1 <- ReadObs(filename = system.file("demo_model", "Qobs.txt", package = "HYPEtools"))
te1 <- HypeSingleVar(x = array(data = unlist(te1[, -1]) +
                                      runif(n = nrow(te1), min = -.5, max = .5),
                               dim = c(nrow(te1), ncol(te1) - 1, 1),
                               dimnames = list(rownames(te1), colnames(te1)[-1])),
                     datetime = te1$DATE, subid = obsid(te1), hype.var = "cout")
te2 <- ReadObs(filename = system.file("demo_model", "Qobs.txt", package = "HYPEtools"))
te2 <- HypeSingleVar(x = array(data = unlist(te2[, -1]),
                               dim = c(nrow(te2), ncol(te2) - 1, 1),
                               dimnames = list(rownames(te2), colnames(te2)[-1])),
                     datetime = te2$DATE, subid = obsid(te2), hype.var = "rout")
# Nash-Sutcliffe Efficiency
NSE(sim = te1, obs = te2, progbar = FALSE)
```

OptimisedClasses *Get optimized classes from an imported optpar.txt file*

Description

`OptimisedClasses` checks which classes (land use or soil) of parameters in an imported optpar list are actually optimized, i.e. have a min/max range larger than zero.

Usage

```
OptimisedClasses(x)
```

Arguments

`x`	list with named elements, as an object returned from `ReadOptpar`.

Details

`OptimisedClasses` allows to quickly check which classes of parameters in an optpar.txt file are actually optimized during a HYPE optimization run. The function compares min and max values in the `pars` element of an imported HYPE optpar.txt file to identify those.

Value

`OptimisedClasses` returns a named list with one vector element for each parameter found in `x`. List element names are HYPE parameter names. Each vector contains the optimized class numbers for the respective parameter.

Examples

```
te <- ReadOptpar(filename = system.file("demo_model", "optpar.txt", package = "HYPEtools"))
OptimisedClasses(te)
```

`OutletIds` *Find Outlet IDs*

Description

Function to find the identifier(s) used to signify model domain outlets, i.e. the "downstream" ID of outlet catchments, in a GeoData file. This is typically just one number, often e.g. '0' or '-9999', but can be one or several IDs if the GeoData file originates from a HYPE sub-model set-up, e.g. created with the 'SelectAro' program. Use `OutletSubids` to find the actual SUBID values of the outlet catchments.

Usage

```
OutletIds(gd)
```

Arguments

`gd`	Data frame with two columns `subid` and `maindown` (not case-sensitive). Typically a 'GeoData.txt' file imported using `ReadGeoData`.

Details

`OutletIds` finds the unique outlet IDs of a GeoData file. The outlet ID of a typical model is a single placeholder number, often e.g. '0' or '-9999', but there can be several outlet IDs, e.g. one or several SUBIDs if the GeoData file originates from a HYPE sub-model set-up, created with the 'SelectAro' tool.

Value

`OutletIds` returns a vector of outlet IDs.

See Also

`AllDownstreamSubids`, `OutletSubids`

Examples

```
te <- ReadGeoData(filename = system.file("demo_model", "GeoData.txt", package = "HYPEtools"))
OutletIds(gd = te)
```

OutletNearObs *Find outlet-near observations in HYPE observation data files.*

Description

Find observation stations close to specified outlet subbasins of a HYPE model set-up. Proximity threshold as upstream area fraction of target outlet subbasin(s). Currently, only upstream observations are identified.

Usage

```
OutletNearObs(
  gd,
  file.qobs = NULL,
  file.xobs = NULL,
  variable = NULL,
  outlets = NULL,
  frac.drain = 0.8,
  nearest.only = TRUE,
  verbose = TRUE
)
```

Arguments

gd	Data frame with two columns subid and maindown (not case-sensitive). Typically a 'GeoData.txt' file imported using ReadGeoData.
file.qobs, file.xobs	Character string, file location of HYPE observation data file. *Only one of these needs to be supplied*, with file.qobs taking precedence if both are provided. Either an Xobs.txt or a Qobs.txt file.
variable	Character string, HYPE variable to use. Needed only with argument file.xobs. If NULL (default), a vector of available variables in file.xobs is returned.
outlets	Integer vector, HYPE SUBIDs of subbasins to be considered outlets. If NULL (default), all outlet subbasins in gd are used.
frac.drain	Numeric, minimum fraction of drainage area at corresponding outlet to be covered by observation site.
nearest.only	Logical, if TRUE (default), only the nearest observation site SUBID is returned. If FALSE, all observation site SUBIDs available within frac.drain are returned.
verbose	Logical, print status messages and progress bars during runtime.

Details

OutletNearObs finds observation sites for observation variables in HYPE 'Qobs.txt' and HYPE 'Xobs.txt' files located upstream an outlet sub-basin. For file.xobs files, which can hold several observation variables, a single variable has to be selected (the function conveniently prints available variables in file.xobs, if no variable is provided). Any number of SUBIDs present in gd

can be defined as outlet subbasins with argument `outlets`. The function handles nested outlets, i.e. cases where user-provided subbasins in `outlets` are upstream basins of one another. Outlet proximity is defined by drainage area size compared to the respective outlet. The function returns either the nearest or all sites matching or exceeding fraction `frac.drain`, depending on argument `nearest.only`.

Value

`OutletNearObs` returns a data frame with 4 columns, containing row-wise all observation sites which match the search criteria:

subid.outlet SUBID of outlet subbasin

subid.obs SUBID of observation site

area.fraction Relative drainage area fraction of observation site, compared to corresponding outlet subbasin

area.outlet Drainage area of outlet subbasin, in km^2

area.obs Drainage area of observation site, in km^2

If `file.xobs` is provided without `variable`, the function prints available HYPE observation variables in `file.xobs` and silently returns the same information as character vector.

Examples

```
# Import source data
te <- ReadGeoData(filename = system.file("demo_model", "GeoData.txt", package = "HYPEtools"))
# Find observation near domain outlet
OutletNearObs(file.qobs = system.file("demo_model", "Qobs.txt", package = "HYPEtools"), gd = te,
verbose = FALSE)
# get vector of variables in an Xobs file
OutletNearObs(file.xobs = system.file("demo_model", "Xobs.txt", package = "HYPEtools"), gd = te,
verbose = FALSE)
```

OutletSubids *Find all Outlet SUBIDs of a model domain*

Description

Function to find all outlet SUBIDs of a HYPE model domain.

Usage

```
OutletSubids(gd)
```

Arguments

gd	A data frame, with two columns `subid` and `maindown`, (not case-sensitive). Typically a 'GeoData.txt' file imported using `ReadGeoData`.

Details

OutletSubids finds all outlet SUBIDs of a model domain as provided in a 'GeoData.txt' file, i.e. all SUBIDs from which stream water leaves the model domain.

Value

OutletSubids returns a vector of outlet SUBIDs.

See Also

AllDownstreamSubids, OutletIds

Examples

```
te <- ReadGeoData(filename = system.file("demo_model", "GeoData.txt", package = "HYPEtools"))
OutletSubids(gd = te)
```

PartyParrot *Create a Party Parrot.*

Description

Creates a Party Parrot.

Usage

```
PartyParrot(sound = 8)
```

Arguments

sound	Character string or number specifying which sound to play when showing the Party Parrot. See the beep function in the beepr package.

Details

PartyParrot generates a Party Parrot. Uses for Party Parrots include, for example, celebrating the successful execution of a script.

Value

Returns a Party Parrot to the Console.

Examples

```
PartyParrot()
```

`pbias.HypeSingleVar` *Percent bias*

Description

Percent bias (PBIAS) calculation for imported HYPE outputs with single variables for several catchments, i.e. time and map files, optionally multiple model runs combined.

Usage

```
## S3 method for class 'HypeSingleVar'
pbias(sim, obs, na.rm = TRUE, progbar = TRUE, ...)
```

Arguments

`sim`	`HypeSingleVar` array with simulated variable (one or several iterations).
`obs`	`HypeSingleVar` array with observed variable, (one iteration). If several iterations are present in the array, only the first will be used.
`na.rm`	Logical. If `TRUE`, incomplete sim-obs pairs will be removed prior to PBIAS computation.
`progbar`	Logical. If `TRUE`, progress bars will be printed for main computational steps.
`...`	ignored

Value

`pbias.HypeSingleVar` returns a 2-dimensional array of NSE performances for all SUBIDs and model iterations provided in argument `sim`, with values in the same order as the second and third dimension in `sim`, i.e. `[subid, iteration]`.

Examples

```
# Create dummy data, discharge observations with added white noise as model simulations
te1 <- ReadObs(filename = system.file("demo_model", "Qobs.txt", package = "HYPEtools"))
te1 <- HypeSingleVar(x = array(data = unlist(te1[, -1]) +
                                 runif(n = nrow(te1), min = -.5, max = .5),
                                 dim = c(nrow(te1), ncol(te1) - 1, 1),
                                 dimnames = list(rownames(te1), colnames(te1)[-1])),
                     datetime = te1$DATE, subid = obsid(te1), hype.var = "cout")
te2 <- ReadObs(filename = system.file("demo_model", "Qobs.txt", package = "HYPEtools"))
te2 <- HypeSingleVar(x = array(data = unlist(te2[, -1]),
                                 dim = c(nrow(te2), ncol(te2) - 1, 1),
                                 dimnames = list(rownames(te2), colnames(te2)[-1])),
                     datetime = te2$DATE, subid = obsid(te2), hype.var = "rout")
# Percentage bias
pbias(sim = te1, obs = te2, progbar = FALSE)
```

PlotAnnualRegime *Plot annual regimes*

Description

Convenience wrapper function for a combined line `plot` with `polygon` variation ranges.

Usage

```
PlotAnnualRegime(
  x,
  line = c("mean", "median"),
  band = c("none", "p05p95", "p25p75", "minmax"),
  add.legend = FALSE,
  l.legend = NULL,
 l.position = c("topright", "bottomright", "right", "topleft", "left", "bottomleft"),
  log = FALSE,
  ylim = NULL,
  ylab = expression(paste("Q (m"^3, " s"^{
     -1
 }, ")")),
  xlab = paste(format(attr(x, "period"), format = "%Y"), collapse = " to "),
  col = "blue",
  alpha = 30,
  lty = 1,
  lwd = 1,
  mar = c(3, 3, 1, 1) + 0.1,
  verbose = TRUE
)
```

Arguments

x	List, typically a result from `AnnualRegime`, containing data frames with aggregated long-term average regime data and two attributes `period` and `timestep`. See Details and Value sections there.
line	Character string, keyword for type of average line to plot. Either `"mean"` or `"median"`.
band	Character vector, keyword for variation bands. If `"none"` (default), plot average line(s) only. `"minmax"`, `"p25p75"`, or `p5p95` to include bands of variation. Combinations of bands are allowed, but providing `"none"` will always prevent plotting of any band. See details.
add.legend	Logical. If `TRUE`, a legend will be added to the plot.
l.legend	Character vector. If non-NULL, legend labels are read from here instead of from column names in `x$mean`.
l.position	Legend position, keyword string. One of `"left"`, `"topleft"`, `"topright"`, `"right"`, `"bottomright"`, `"bottomleft"`.

log	Logical, if TRUE, y-axis will be log-scaled.
ylim	Numeric vector of length two, giving y-axis limits. Defaults to min-max range of all plotted data.
ylab	Character or plotmath expression string. Y-axis label, with a default for discharge regimes.
xlab	Character string or plotmath expression string, x-axis label. Default prints the time period on which the regime is based, read from x$period.
col	Line color specification, see par for details. Defaults to blue. Either a single value or a vector of the same length as quantile series in freq.
alpha	Numeric, alpha transparency value for variation bands. Value between 0 (transparent) and 255 (opaque), see also rgb
lty	Line type specification, see par for details. Either a single value or a vector of the same length as quantile series in freq.
lwd	Line width specification, see par for details. Either a single value or a vector of the same length as quantile series in freq.
mar	Numeric vector of length 4, margin specification as in par with modified default. Details see there.
verbose	Logical, print warnings if NA values are found in x. Defaults to TRUE.

Details

PlotAnnualRegime plots contents from lists as returned by AnnualRegime (for format details, see there). If NA values are present in the plot data, the function will throw a warning if verbose = TRUE and proceed with plotting all available data.

Argument band allows to plot variation bands to be plotted in addition to average lines. These can be (combinations of) ranges between minima and maxima, 5th and 95th percentiles, and 25th and 75th percentiles, i.e. all moments available in AnnualRegime results.

Grid lines plotted in the background are mid-month lines.

Value

PlotAnnualRegime returns a plot to the currently active plot device.

See Also

AnnualRegime, PlotSimObsRegime

Examples

```
## Not run:
# Source data, HYPE basin output with a number of result variables
te1 <- ReadBasinOutput(filename = system.file("demo_model",
                                              "results", "0003587.txt",
                                              package = "HYPEtools"))
# Daily discharge regime, computed and observed,
# hydrological year from October, aggregated to weekly means
te2 <- AnnualRegime(te1[, c("DATE", "COUT", "ROUT")],
```

```
                    ts.in = "day",
                    ts.out = "week", start.mon = 10)

# Screen devices should not be used in examples
PlotAnnualRegime(x = te2)
PlotAnnualRegime(x = te2, line = "median", band = "p05p95",
  add.legend = TRUE, col = c("red", "blue"))

## End(Not run)
```

`PlotBasinOutput` *Plot a suite of time series plots from a HYPE basin output file*

Description

Plot a standard suite of time series plots from a basin output file, typically used for model performance inspection and/or during manual calibration

Usage

```
PlotBasinOutput(
  x,
  filename,
  driver = c("default", "pdf", "png", "screen"),
  timestep = attr(x, "timestep"),
  hype.vars = "all",
  vol.err = TRUE,
  log.q = FALSE,
  start.mon = 1,
  from = 1,
  to = nrow(x),
  date.format = "",
  name = "",
  area = NULL,
  subid = attr(x, "subid"),
  gd = NULL,
  bd = NULL,
  ylab.t1 = "Conc."
)
```

Arguments

x	Data frame, with column-wise equally-spaced time series of HYPE variables. Date-times in `POSIXct` format in first column. Typically an imported basin output file from HYPE using `ReadBasinOutput`. See details for HYPE output variables required for plotting.

filename	String, file name for plotting to file device, see argument driver. *No file extension!* Ignored with plotting to screen device. *Device dimensions are currently hard-coded, see Details.*
driver	String, device driver name, one of default, pdf, png, or screen. Defaults to default, which plots using default plotting device getOption("device").
timestep	Character string, timestep of x, one of "month", "week", "day", or "nhour" (n = number of hours). If not provided, an attribute timestep is required in x.
hype.vars	Either a keyword string or a character vector of HYPE output variables. User-specified selection of HYPE variables to plot. Default ("all") is to plot all variables which the function knows and which are available in x. See details for a list of known variables. Other possible keywords are "hydro" and "wq" (water quality), for which a pre-selected range of (available) result variables is plotted. Alternatively, a character vector holding HYPE output variables to be plotted. Variables unknown to the function will be ignored with a warning.
vol.err	Logical, if TRUE and both observed and simulated discharge are available in x, the accumulated volume error will be plotted.
log.q	Logical, y-axis scaling for flow duration curve and discharge time series, set to TRUE for log-scaling.
start.mon	Integer between 1 and 12, starting month of the hydrological year. For runoff regime plot, see also AnnualRegime.
from, to	Integer or date string of format \ interpreted as row indices of x.
date.format	String format for x-axis dates/times. See strptime.
name	Character string, name to be printed on the plot.
area	Numeric, upstream area of sub-basin in m^2. Required for calculation of accumulated volume error. Optional argument, either this or arguments subid, gd, and bd are required.
subid	Integer, HYPE SUBID of a target sub-catchment (must exist in gd). Mandatory in combination with gd and optionally bd if argument area is not defined. If not provided, an attribute subid is required in x. Used to calculate upstream area internally with function SumUpstreamArea. For repeated calls to PlotBasinOutput providing area in combination with a one-off separate call to SumUpstreamArea saves computation time, especially in basins with many upstream sub-basins.
gd	A data frame, containing 'SUBID' and 'MAINDOWN' columns, e.g. an imported 'GeoData.txt' file. Mandatory with argument subid, details see there.
bd	A data frame, containing 'BRANCHID' and 'SOURCEID' columns, e.g. an imported 'BranchData.txt' file. Optional with argument subid, details see there.
ylab.t1	String or plotmath expression, y axis label for T1 tracer time series panel (tracer concentration units are not prescribed in HYPE).

Details

PlotBasinOutput plots a suite of time series along with a flow duration curve, a flow regime plot, and a selection of goodness-of-fit measures from an imported HYPE basin output file. The function

selects from a range of "known" variables, and plots those which are available in the user-supplied basin output. It is mostly meant as a support tool during calibration, manual or automatic, providing a quick and comprehensive overview of model dynamics in a subbasin of interest.

HYPE outputs which are known to `PlotBasinOutput` include:

- precipitation
- air temperature
- discharge
- lake water level
- water temperature
- evapotranspiration
- snow water equivalent
- sub-surface storage components
- nitrogen concentrations
- phosphorus concentrations
- organic carbon concentrations
- suspended sediment concentrations
- total sediment concentrations
- tracer concentration

Below a complete list of HYPE variables known to the function in HYPE info.txt format, ready to copy-paste into an info.txt file. For a detailed description of the variables, see the HYPE online documentation.

`basinoutput variable upcprf upcpsf temp upepot upevap cout rout soim sm13 upsmfp snow upcprc cct2 ret2 ccin rein ccon reon cctn retn ccsp resp ccpp repp cctp retp wcom wstr ccss ress ccts rets cct1 ret1 ccoc reoc`

Device dimensions are hard-coded to a width of 15 inches and height depending on the number of plotted time series. When plotting to a screen device, a maximum height of 10 inches is enforced in order to prevent automatic resizing with slow redrawing. `PlotBasinOutput` throws a warning if the plot height exceeds 10 inches, which can lead to overlapping plot elements. On screens with less than 10 inch screen, redrawing is inhibited, which can lead to an empty plot. The recommended solution for both effects is to plot to pdf or png file devices instead.

Value

Returns a multi-panel plot in a new graphics device.

See Also

`PlotBasinSummary`, `PlotAnnualRegime`, `PlotDurationCurve`, `ReadBasinOutput`

Examples

```
# Source data, HYPE basin output with a number of result variables
te1 <- ReadBasinOutput(filename = system.file("demo_model",
"results","0003587.txt", package = "HYPEtools"))
te2 <- ReadGeoData(filename = system.file("demo_model",
"GeoData.txt", package = "HYPEtools"))

## Not run:
# Plot selected water variables on screen device
PlotBasinOutput(x = te1, gd = te2, driver = "screen",hype.vars = c("cout", "rout",
"snow", "upcprf", "upcpsf"))

## End(Not run)
```

`PlotBasinSummary` *Plot a summary of model results for a single sub-basin*

Description

Plot a standard suite of plots summarizing properties of a sub-basin including upstream area and model performance for discharge and concentrations of nutrients, organic carbon, sediment, and tracers.

Usage

```
PlotBasinSummary(
  x,
  filename,
  driver = c("default", "pdf", "png", "screen"),
  panels = 1,
  gd = NULL,
  bd = NULL,
  gcl = NULL,
  psd = NULL,
  subid = NULL,
  desc = NULL,
  timestep = attr(x, "timestep"),
  hype.vars = "all",
  from = 1,
  to = nrow(x),
  log = FALSE,
  xscale = "gauss",
  start.mon = 10,
  name = "",
  ylab.t1 = "Conc."
)
```

Arguments

x	Data frame, with column-wise daily time series of HYPE variables. Date-times in POSIXct format in first column. Typically an imported basin output file from HYPE using ReadBasinOutput. See details for HYPE output variables required for plotting.
filename	String, file name for plotting to file device, see argument driver. *No file extension!* Ignored with plotting to screen device. *Device dimensions are currently hard-coded, see Details.*
driver	String, device driver name, one of default, pdf, png, or screen. Defaults to default, which plots using default plotting device getOption("device").
panels	Integer, either 1, 2, or 3, indicating which panels to plot. See Details.
gd	A data frame, containing 'SUBID', 'MAINDOWN', and 'AREA' columns, e.g. an imported 'GeoData.txt' file. Only needed with bar chart panels, see Details.
bd	A data frame, containing 'BRANCHID' and 'SOURCEID' columns, e.g. an imported 'BranchData.txt' file. Optional argument. Only needed with bar chart panels, see Details.
gcl	Data frame containing columns with SLCs and corresponding land use and soil class IDs, typically a 'GeoClass.txt' file imported with ReadGeoClass. Only needed with bar chart panels, see Details.
psd	A data frame with HYPE point source specifications, typically a 'PointSourceData.txt' file imported with ReadPointSourceData. Only needed with bar chart panels, see Details.
subid	Integer, SUBID of sub-basin for which results are plotted. If NULL (default), a subid attribute is required in x. Only needed with bar chart panels, see Details.
desc	List for use with type. Class description labels imported from a 'description.txt' file, for bar labeling. See ReadDescription for formatting details. Only needed with bar chart panels, see Details.
timestep	Character string, timestep of x, one of "month", "week", "day", or "nhour" (n = number of hours). If not provided, an attribute timestep is required in x.
hype.vars	Either a keyword string or a character vector of HYPE output variables. User-specified selection of HYPE variables to plot. Default ("all") is to plot all variables which the function knows and which are available in x. See details for a list of known variables. Other possible keywords are "hydro" and "wq" (water quality), for which a pre-selected range of (available) result variables is plotted. Alternatively, a character vector holding HYPE output variable IDs to be plotted. Variables unknown to the function will be ignored with a warning.
from, to	Integer or date string of format \ interpreted as row indices of x.
log	Logical, log scaling discharge and concentrations.
xscale	Character string, keyword for x-axis scaling. Either "lin" for linear scaling or "gauss" for gaussian scaling. See description in PlotDurationCurve.
start.mon	Integer between 1 and 12, starting month of the hydrological year. For regime plots, see also AnnualRegime.
name	Character or expression string. Site name to plot besides bar chart panels. Only relevant with panels 1 or 3.

`ylab.t1` String or `plotmath` expression, y axis label for T1 tracer time series panel (tracer concentration units are not prescribed in HYPE).

Details

`PlotBasinSummary` plots a multi-panel plot with a number of plots to evaluate model properties and performances for a chosen sub-basin. Performance plots include discharge, HYPE-modeled nutrient species for nitrogen (total, inorganic, organic) phosphorus (total, particulate, soluble), organic carbon, and suspended and total sediment concentrations.

Plotted panels show:

- *Summarized catchment characteristics as bar charts*: Upstream-averaged land use, soil, and crop group fractions; modeled nutrient loads in sub-basin outlet, and summed upstream gross loads from point sources and rural households (if necessary variables available, omitted otherwise).
- *Goodness-of-fit measures for discharge and concentrations*: KGE (Kling-Gupta Efficiency), NSE (Nash-Sutcliffe Efficiency), PBIAS (Percentage Bias, aka relative error), MAE (Mean Absolute Error), r (Pearson product-moment correlation coefficient), VE (Volumetric Efficiency).
- *Simulation-observation relationships for discharge and concentrations*: Simulated and observed concentration-discharge relationships, relationship between observed and simulated nutrient, organic carbon, sediment, and tracer concentrations.
- *Duration curves for flow and concentrations*: Pairwise simulated and observed curves.
- *Annual regimes for flow and concentrations*: Pairwise simulated and observed regime plots at monthly aggregation, with number of observations for concentration regimes.
- *Corresponding plots for IN/TN and SP/TP ratios*.

Per default, the function plots from available model variables in an imported HYPE basin output file, and missing variables will be automatically omitted. Variable selection can be additionally fine-tuned using argument `hype.vars`.

Argument `panels` allows to choose if bar chart panels should be plotted. This can be time-consuming for sites with many upstream sub-basins and might not necessary e.g. during calibration. If `1` (default), all panels are plotted. If set to `2`, bar charts will be excluded. If `3`, only bar charts will be plotted. Arguments `gd`, `bd`, `gcl`, `psd`, `subid`, and `desc` are only needed for bar chart plotting.

Below a complete list of HYPE variables known to the function in HYPE info.txt format, ready to copy-paste into an info.txt file. For a detailed description of the variables, see the HYPE online documentation.

`basinoutput variable cout rout ccin rein ccon reon cctn retn ccsp resp ccpp repp cctp retp ctnl ctpl ccss ress ccts rets cct1 ret1 ccoc reoc`

#' *Device dimensions* are hard-coded to a width of 13 inches and height depending on the number of plotted time series. When plotting to a screen device, a maximum height of 10 inches is enforced in order to prevent automatic resizing with slow redrawing. `PlotBasinOutput` throws a warning if the plot height exceeds 10 inches, which can lead to overlapping plot elements. On screens with less than 10 inch screen height, redrawing is inhibited, which can lead to an empty plot. The recommended solution for both effects is to plot to pdf or png file devices instead.

Value

Returns a multi-panel plot in a new graphics device.

See Also

PlotBasinOutput, BarplotUpstreamClasses, PlotSimObsRegime, PlotAnnualRegime, PlotDurationCurve, ReadBasinOutput

Examples

```
# Source data, HYPE basin output with a number of result variables
te1 <- ReadBasinOutput(filename = system.file("demo_model", "results", "0003587.txt",
                       package = "HYPEtools"))
te2 <- ReadGeoData(filename = system.file("demo_model", "GeoData.txt", package = "HYPEtools"))

# Screen devices should not be used in examples
## Not run:
# Plot basin summary for discharge on screen device
PlotBasinSummary(x = te1, gd = te2, driver = "screen", panels = 2)

## End(Not run)
```

PlotDurationCurve *Plot duration curves*

Description

Convenience wrapper function for a (multiple) line plot, with pretty defaults for axis annotation and a Gaussian scaling option for the x-axis.

Usage

```
PlotDurationCurve(
  freq,
  xscale = "lin",
  yscale = "log",
  add.legend = FALSE,
  l.legend = NULL,
  ylim = NULL,
  xlab = "Flow exceedance percentile",
  ylab = "m3s",
  col = "blue",
  lty = 1,
  lwd = 1,
  mar = c(3, 3, 1, 1) + 0.1
)
```

Arguments

freq	Data frame with at least two columns, containing probabilities in the first and series of data quantiles in the remaining columns. Typically an object as returned by ExtractFreq or a subset thereof.
xscale	Character string, keyword for x-axis scaling. Either "lin" for linear scaling or "gauss" for gaussian scaling as in a normal probability plot, which allows for for better comparison of low flow and high flow frequencies.
yscale	Character string, keyword for y-axis scaling. Either "lin" for linear scaling or "log" for common logarithm scaling.
add.legend	Logical. If TRUE, a legend will be added to the plot, including the number of observations on which the quantiles are based for each curve if freq is a result from ExtractFreq.
l.legend	Character vector. If non-NULL, legend labels are read from here instead of from column names in freq.
ylim	Numeric vector of length two, giving y-axis limits. NULL for default values.
xlab	Character string, x-axis label.
ylab	Character or plotmath expression string. Y-axis label, either as keyword "m3s" or "mmd" for pre-defined pretty discharge labels, or any other string which will be plotted unchanged.
col	Line color specification, see par for details. Defaults to blue. Either a single value or a vector of the same length as quantile series in freq.
lty	Line type specification, see par for details. Either a single value or a vector of the same length as quantile series in freq.
lwd	Line width specification, see par for details. Either a single value or a vector of the same length as quantile series in freq.
mar	Numeric vector of length 4, margin specification as in par with modified default. Details see there.

Details

PlotDurationCurve plots a duration curve with pretty formatting defaults. The function sets par parameters tcl and mgp internally and will override previously set values for the returned plot. It typically uses results from ExtractFreq as input data and via that function it can be used to visualize and compare time series properties.

Value

PlotDurationCurve returns a plot to the currently active plot device.

See Also

ExtractFreq

Examples

```
# Import source data
te1 <- ReadBasinOutput(filename = system.file("demo_model", "results", "0003587.txt",
                       package = "HYPEtools"))
te2 <- ExtractFreq(te1[, c("COUT", "ROUT")])
# Plot flow duration curves for simulated and observed discharge
PlotDurationCurve(freq = te2, add.legend = TRUE, col = c("red", "blue"))
```

PlotMapOutput *Plot function for HYPE map results.*

Description

Draw HYPE map results, with pretty scale discretizations and color ramp defaults for select HYPE variables.

Usage

```
PlotMapOutput(
  x,
  map = NULL,
  map.subid.column = 1,
  var.name = "",
  map.type = "default",
  shiny.data = FALSE,
  plot.legend = TRUE,
  legend.pos = "right",
  legend.title = NULL,
  legend.signif = 2,
  col = "auto",
  col.ramp.fun,
  col.breaks = NULL,
  col.labels = NULL,
  col.rev = FALSE,
  plot.scale = TRUE,
  scale.pos = "br",
  plot.arrow = TRUE,
  arrow.pos = "tr",
  weight = 0.15,
  opacity = 0.75,
  fillOpacity = 0.5,
  outline.color = "black",
  na.color = "#808080",
  plot.searchbar = FALSE,
  plot.label = FALSE,
  plot.label.size = 2.5,
```

```
  plot.label.geometry = c("centroid", "surface"),
  file = "",
  width = NA,
  height = NA,
  units = c("in", "cm", "mm", "px"),
  dpi = 300,
  vwidth = 1424,
  vheight = 1000,
  html.name = "",
  map.adj = 0,
  legend.outer = FALSE,
  legend.inset = c(0, 0),
  par.cex = 1,
  par.mar = rep(0, 4) + 0.1,
  add = FALSE,
  sites = NULL,
  sites.subid.column = NULL
)
```

Arguments

x	HYPE model results, typically 'map output' results. Data frame object with two columns, first column containing SUBIDs and second column containing model results to plot. See details.
map, sites	A SpatialPolygonsDataFrame or sf object. Typically an imported sub-basin vector polygon file. Import of vector polygons requires additional packages, e.g. sf::st_read. For interactive Leaflet maps a small/simplified polygon file should be used as larger files can take an excessive amount of time to render.
map.subid.column, sites.subid.column	Integer, column index in the map 'data' slot holding SUBIDs (sub-catchment IDs).
var.name	Character string. HYPE variable name to be plotted. Mandatory for automatic color ramp selection of pre-defined HYPE variables (col = "auto"). Not case-sensitive. See details.
map.type	Map type keyword string. Choose either "default" for the default static plots or "leaflet" for interactive Leaflet maps. Use "legacy" for deprecated static plots.
shiny.data	Logical, if map.type is "leaflet", then should the output be a list containing the basemap, formatted data, legend colors, and legend labels? Typically set to FALSE unless using PlotMapOutput to create Shiny apps or custom Leaflet maps.
plot.legend	Logical, plot a legend along with the map.
legend.pos	Keyword string for legend position. For static plots, one of: "none", "left", "right", "bottom", "top", or a two-element numeric vector. For interactive Leaflet maps, one of: "topleft", "topright", "bottomright", "bottomleft". For legacy static plots, one of: "left", "topleft", "topright", "right", "bottomright", "bottomleft".

`legend.title` Character string or mathematical expression. An optional title for the legend. If none is provided here, `var.name` is used as legend title string. For select HYPE variables, pretty legend titles are in-built.

`legend.signif` Integer, number of significant digits to display in legend labels.

`col` Colors to use on the map. One of the following:

- `"auto"` to allow for automatic selection from tailored color ramp palettes and break points based on argument `var.name`, see details
- A color ramp palette function, e.g. as returned from a call to `colorRampPalette`. A number of tailored functions are available in `HYPEtools`, see `CustomColors`
- A vector of colors. This can be a character vector of R's built-in color names or hexadecimal strings as returned by `rgb`, or an integer vector of current `palette` indices.

`col.ramp.fun` DEPRECATED, for backwards compatibility only.

`col.breaks` A numeric vector, specifying break points for discretization of model result values into classes. Used if a color palette is specified with `col` argument. Class boundaries will be interpreted as right-closed, i.e upper boundaries included in class. Lowest class boundary included in lowest class as well. Meaningful results require the lowest and uppermost breaks to bracket all model result values, otherwise there will be unclassified white spots on the map plot. Not mandatory, can optionally be combined with one of the pre-defined palettes, including `"auto"` selection. Per default, a generic classification will be applied (see details).

`col.labels` A character vector, specifying custom labels to be used for each legend item. Works with `map.type` set to `default` or `leaflet`.

`col.rev` Logical, If `TRUE`, then color palette will be reversed.

`plot.scale` Logical, plot a scale bar on map. NOTE: Scale bar may be inaccurate for geographic coordinate systems (Consider switching to projected coordinate system).

`scale.pos` Keyword string for scalebar position for static maps. One of `bl`, `br`, `tr`, or `tl`.

`plot.arrow` Logical, plot a North arrow in static maps.

`arrow.pos` Keyword string for north arrow position for static maps. One of `bl`, `br`, `tr`, or `tl`.

`weight` Numeric, weight of subbasin boundary lines. See ggplot2::geom_sf for static maps and leaflet::addPolygons for Leaflet maps.

`opacity` Numeric, opacity of subbasin boundary lines in Leaflet maps. See leaflet::addPolygons.

`fillOpacity` Numeric, opacity of subbasin polygons in Leaflet maps. See leaflet::addPolygons.

`outline.color` Character string of color to use to for subbasin polygon outlines. Use `NA` to hide the outlines.

`na.color` Character string of color to use to symbolize subbasin polygons in maps which correspond to `NA` values.

`plot.searchbar` Logical, if `TRUE`, then a search bar will be included within Leaflet maps. See leaflet.extras::addSearchFeatures.

`plot.label` Logical, if `TRUE`, then labels will be displayed on default static maps and in Leaflet maps when the cursor hovers over subbasins. See ggplot2::geom_sf_text for default maps and leaflet::addPolygons for Leaflet maps.

`plot.label.size` Numeric, size of text for labels on default static plots. See ggplot2::geom_sf_text.

`plot.label.geometry` Keyword string to select where plot labels should be displayed on the default static plots. Either `centroid` to use sf::st_centroid or `surface` to use sf::st_point_on_surface.

`file` Save map to an image file by specifying the path to the desired output file using this argument. File extension must be specified. See ggplot2::ggsave for static maps and mapview::mapshot for Leaflet maps. You may need to run `webshot::install_phantomjs()` the first time you save a Leaflet map to an image file. See webshot::install_phantomjs.

`width` Numeric, width of output plot for static maps in units of `units`. See ggplot2::ggsave.

`height` Numeric, height of output plot for static maps in units of `units`. See ggplot2::ggsave.

`units` Keyword string for units to save static map. One of `"in"`, `"cm"`, `"mm"`, `"px"`. See ggplot2::ggsave.

`dpi` Integer, resolution to save static map. See ggplot2::ggsave.

`vwidth` Numeric, width of the exported Leaflet map image in pixels. See mapview::mapshot.

`vheight` Numeric, height of the exported Leaflet map image in pixels. See mapview::mapshot.

`html.name` Save Leaflet map to an interactive HTML file by specifying the path to the desired output file using this argument. File extension must be specified. See htmlwidgets::saveWidget.

`map.adj` Numeric, map adjustment in direction where it is smaller than the plot window. A value of `0` means left-justified or bottom-justified, `0.5` (the default) means centered, and `1` means right-justified or top-justified. Only used for default maps.

`legend.outer` Logical. If `TRUE`, outer break point values will be plotted in legend.

`legend.inset` Numeric, inset distance(s) from the margins as a fraction of the plot region for legend, scale and north arrow. See `legend` and details below.

`par.cex` Numeric, character expansion factor. See description of `cex` in `par`. Only used for default maps.

`par.mar` Plot margins as in `par` argument `mar`. Defaults to a nearly margin-less plot. In standard use cases of this function, plot margins do not need to be changed. Only used for default maps.

`add` Logical, default `FALSE`. If `TRUE`, add to existing plot. In that case `map.adj` has no effect. Only used for default maps.

Details

`PlotMapOutput` plots HYPE results from 'map[variable name].txt' files, typically imported using `ReadMapOutput`. `x` arguments **must** contain the variable of interest in the second column.

For map results with multiple columns, i.e. several time periods, pass index selections to x, e.g. mymapresult[, c(1, 3)].

PlotMapOutput can return static plots or interactive Leaflet maps depending on value provided for the argument map.type. For backwards compatibility, legacy static plots can still be generated by setting map.type to legacy. For legacy plots, legend.pos and map.adj should be chosen so that legend and map do not overlap, and the legend position can be fine-tuned using argument legend.inset. This is particularly useful for legend titles with more than one line. In order to move map and legend closer to each other, change the plot device width. For details on inset specification for the default maps, see inset in legend.

Mapped variables are visualized using color-coded data intervals. HYPEtools provides a number of color ramps functions for HYPE variables, see CustomColors. These are either single-color ramps with less saturated colors for smaller values and more saturated values for higher values, suitable for e.g. concentration or volume ranges, or multi-color ramps suitable for calculated differences, e.g. between two model runs.

Break points between color classes of in-built or user-provided color ramp palettes can optionally be provided in argument col.breaks. This is particularly useful when specific pretty class boundaries are needed, e.g. for publication figures. Per default, break points for internal single color ramps and user-provided ramps are calculated based on 10\ x. Default break points for internal color ramp ColDiffGeneric are based on an equal distance classification of log-scaled x ranges, centered around zero. For internal color ramp ColDiffTemp, they are breaks in an interval from -7.5 to 7.5 K.

For select common HYPE variables, given in argument var.name, an automatic color ramp selection including pretty breaks and legend titles is built into PlotMapOutput. These are 'CCTN', 'CCTP', 'COUT', and 'TEMP'. Automatic selection is activated by choosing keyword "auto" in col. All other HYPE variables will be plotted using a generic color ramp palette and generic break points with "auto" color selection.

Value

For default static maps, PlotMapOutput returns an object of class ggplot. This plot can also be assigned to a variable in the environment. For interactive Leaflet maps, PlotMapOutput returns an object of class leaflet. For legacy static plots, PlotMapOutput returns a plot to the currently active plot device, and invisibly an object of class SpatialPolygonsDataFrame as provided in argument map, with plotted values and color codes added as columns in the data slot.

See Also

ReadMapOutput for HYPE result import; PlotMapPoints for plotting HYPE results at points, e.g. sub-basin outlets.

Examples

```
# Import plot data and subbasin polygons
require(sf)
te1 <- ReadMapOutput(filename = system.file("demo_model",
"results", "mapCRUN.txt", package = "HYPEtools"), dt.format = NULL)
te2 <- st_read(dsn = system.file("demo_model",
"gis", "Nytorp_map.gpkg", package = "HYPEtools"))
```

```
# plot runoff map
PlotMapOutput(x = te1, map = te2, map.subid.column = 25,
var.name = "CRUN", col = ColQ)
```

PlotMapPoints *Plot function for mapped point information*

Description

Plot mapped point information, e.g. model performances at observation sites.

Usage

```
PlotMapPoints(
  x,
  sites = NULL,
  sites.subid.column = 1,
  sites.groups = NULL,
  bg = NULL,
  bg.label.column = 1,
  var.name = "",
  map.type = "default",
  shiny.data = FALSE,
  plot.legend = TRUE,
  legend.pos = "right",
  legend.title = NULL,
  legend.signif = 2,
  col = NULL,
  col.breaks = NULL,
  col.labels = NULL,
  col.rev = FALSE,
  plot.scale = TRUE,
  scale.pos = "br",
  plot.arrow = TRUE,
  arrow.pos = "tr",
  radius = 5,
  weight = 0.15,
  opacity = 0.75,
  fillOpacity = 0.5,
  na.color = "#808080",
  jitter = 0.01,
  bg.weight = 0.15,
  bg.opacity = 0.75,
  bg.fillColor = "#e5e5e5",
```

```
  bg.fillOpacity = 0.75,
  plot.label = FALSE,
  plot.label.size = 2.5,
  plot.label.geometry = c("centroid", "surface"),
  noHide = FALSE,
  textOnly = FALSE,
  font.size = 10,
  plot.bg.label = NULL,
  file = "",
  width = NA,
  height = NA,
  units = c("in", "cm", "mm", "px"),
  dpi = 300,
  vwidth = 1424,
  vheight = 1000,
  html.name = "",
  map.adj = 0,
  legend.outer = FALSE,
  legend.inset = c(0, 0),
  pt.cex = 1,
  par.cex = 1,
  par.mar = rep(0, 4) + 0.1,
  pch = 21,
  lwd = 0.8,
  add = FALSE,
  map = NULL,
  map.subid.column = NULL
)
```

Arguments

x	Information to plot, typically model performances from imported HYPE 'subassX.txt' files. Data frame object with two columns, first column containing SUBIDs and second column containing model results to plot. See details.
sites, map	A SpatialPointsDataFrame or sf object. Typically an imported outlet point vector point file. Import of vector points requires additional packages, e.g. sf::st_read().
sites.subid.column, map.subid.column	Integer, column index in the sites 'data' slot holding SUBIDs (sub-catchment IDs).
sites.groups	Named list providing groups of SUBIDs to allow toggling of point groups in Leaflet maps. Default NULL will produce maps without point groups. List names represent the names of the groups to plot, and list values represent the SUBIDs within the group. Example: sites.groups = list("GROUP 1" = c(1, 2, 3), "GROUP 2" = c(4, 5, 6)).
bg	A SpatialPolygonsDataFrame or sf object to plot in the background. Typically an imported sub-basin vector polygon file. For default maps with several background layers, use add = TRUE and plot background layer(s) first.

`bg.label.column`
: Integer, column index in the `bg` 'data' `slot` holding labels (e.g. SUBIDs) to use for plotting.

`var.name`
: Character string. HYPE variable name to be plotted. Mandatory for automatic color ramp selection of pre-defined HYPE variables (`col = "auto"`). Not case-sensitive.

`map.type`
: Map type keyword string. Choose either `"default"` for the default static plots or `"leaflet"` for interactive Leaflet maps. Use `"legacy"` for deprecated static plots.

`shiny.data`
: Logical, if `map.type` is `"leaflet"`, then should the output be a list containing the basemap, formatted data, legend colors, and legend labels? Typically set to `FALSE` unless using `PlotMapOutput` to create Shiny apps or custom Leaflet maps.

`plot.legend`
: Logical, plot a legend along with the map.

`legend.pos`
: Keyword string for legend position. For static plots, one of: `"none"`, `"left"`, `"right"`, `"bottom"`, `"top"`, or a two-element numeric vector. For interactive Leaflet maps, one of: `"topleft"`, `"topright"`, `"bottomright"`, `"bottomleft"`. For legacy static plots, one of: `"left"`, `"topleft"`, `"topright"`, `"right"`, `"bottomright"`, `"bottomleft"`.

`legend.title`
: Character string or mathematical expression. An optional title for the legend. If none is provided here, the name of the second column in `x` is used as legend title string.

`legend.signif`
: Integer, number of significant digits to display in legend labels.

`col`
: Colors to use on the map. One of the following:
  - `NULL`, to use a default purple-red-yellow-blue color ramp, best used with `col.breaks = NULL`.
  - A color ramp palette function, e.g. as returned from a call to `colorRampPalette`
  - A vector of colors. This can be a character vector of R's built-in color names or hexadecimal strings as returned by `rgb`, or an integer vector of current `palette` indices.

`col.breaks`
: A numeric vector, specifying break points for discretization of model result values into classes. Class boundaries will be interpreted as right-closed, i.e upper boundaries included in class. Lowest class boundary included in lowest class as well. Meaningful results require the lowest and uppermost breaks to bracket all model result values, otherwise there will be unclassified white spots on the map plot. If `NULL` (the default), `col.breaks` covers a range from 0 to 1 with 9 intervals, and an additional interval for negative values. This is suitable for e.g. NSE performances.

`col.labels`
: A character vector, specifying custom labels to be used for each legend item. Works with `map.type` set to `default` or `leaflet`.

`col.rev`
: Logical, If `TRUE`, then color palette will be reversed.

`plot.scale`
: Logical, plot a scale bar on map. NOTE: Scale bar may be inaccurate for geographic coordinate systems (Consider switching to projected coordinate system).

scale.pos	Keyword string for scalebar position for static maps. One of bl, br, tr, or tl.
plot.arrow	Logical, plot a North arrow in static maps.
arrow.pos	Keyword string for north arrow position for static maps. One of bl, br, tr, or tl.
radius	Numeric, radius of markers maps. See ggplot2::geom_sf for static maps and leaflet::addCircleMarkers for Leaflet maps.
weight	Numeric, weight of marker outlines in Leaflet maps. See leaflet::addCircleMarkers.
opacity	Numeric, opacity of marker outlines in Leaflet maps. See leaflet::addCircleMarkers.
fillOpacity	Numeric, opacity of markers in Leaflet maps. See leaflet::addCircleMarkers.
na.color	Character string of color to use to symbolize markers in maps which correspond to NA values.
jitter	Numeric, amount to jitter points with duplicate geometries. See sf::st_jitter.
bg.weight	Numeric, weight of bg subbasin outlines in Leaflet maps. See leaflet::addPolygons.
bg.opacity	Numeric, opacity of bg subbasin outlines in Leaflet maps. See ggplot2::geom_sf for static maps and leaflet::addPolygons for Leaflet maps.
bg.fillColor	Character string of color to use to symbolize bg subbasin polygons in maps. See ggplot2::geom_sf for static maps and leaflet::addPolygons for Leaflet maps.
bg.fillOpacity	Numeric in range 0-1, opacity of bg subbasin polygons in maps. See ggplot2::geom_sf for static maps and leaflet::addPolygons for Leaflet maps.
plot.label	Logical, if TRUE, then labels will be displayed on default static maps and in Leaflet maps when the cursor hovers over markers. See ggplot2::geom_sf_text for default maps and leaflet::addCircleMarkers for Leaflet maps.
plot.label.size	Numeric, size of text for labels on default static plots. See ggplot2::geom_sf_text.
plot.label.geometry	Keyword string to select where plot labels should be displayed on the default static plots. Either centroid to use sf::st_centroid or surface to use sf::st_point_on_surface.
noHide	Logical, set to TRUE to always display marker labels in Leaflet maps. See leaflet::labelOptions.
textOnly	Logical, set to TRUE to hide marker label background in Leaflet maps. See leaflet::labelOptions.
font.size	Numeric, font size (px) for marker labels in Leaflet maps.
plot.bg.label	String, if hover, then labels will be displayed in Leaflet maps for bg when the cursor hovers over polygons. If static, then static labels for bg will be displayed in Leaflet maps. If any string is specified, then background labels will be added to default static maps.
file	Save map to an image file by specifying the path to the desired output file using this argument. File extension must be specified. See ggplot2::ggsave for static maps and mapview::mapshot for Leaflet maps. You may need to run webshot::install_phantomjs() the first time you save a Leaflet map to an image file. See webshot::install_phantomjs.
width	Numeric, width of output plot for static maps in units of units. See ggplot2::ggsave.

height	Numeric, height of output plot for static maps in units of units. See ggplot2::ggsave.
units	Keyword string for units to save static map. One of "in", "cm", "mm", "px". See ggplot2::ggsave.
dpi	Integer, resolution to save static map. See ggplot2::ggsave.
vwidth	Numeric, width of the exported Leaflet map image in pixels. See webshot::webshot.
vheight	Numeric, height of the exported Leaflet map image in pixels. See webshot::webshot.
html.name	Save Leaflet map to an interactive HTML file by specifying the path to the desired output file using this argument. File extension must be specified. See htmlwidgets::saveWidget.
map.adj	Numeric, map adjustment in direction where it is smaller than the plot window. A value of 0 means left-justified or bottom-justified, 0.5 (the default) means centered, and 1 means right-justified or top-justified. Only used for legacy static maps.
legend.outer	Logical. If TRUE, outer break point values will be plotted in legend. Only used for legacy static maps.
legend.inset	Numeric, inset distance(s) from the margins as a fraction of the plot region for legend, scale and north arrow. See legend and details below. Only used for legacy static maps.
pt.cex	Numeric, plot point size expansion factor, works on top of par.cex.
par.cex	Numeric, character expansion factor. See description of cex in par. Only used for legacy maps.
par.mar	Plot margins as in par argument mar. Defaults to a nearly margin-less plot. In standard use cases of this function, plot margins do not need to be changed. Only used for legacy maps.
pch, lwd	Integer, plotting symbol and line width. See points. Only used for legacy maps.
add	Logical, default FALSE. If TRUE, add to existing plot. In that case map.adj has no effect. Only used for legacy maps.

Details

PlotMapPoints can be used to print point information on a mapped surface. The primary target are model performance measures as written to HYPE 'subassX.txt' files, but color scale and break point arguments are flexible enough to also be used with e.g. HYPE output variables or other data.

PlotMapOutput can return static plots or interactive Leaflet maps depending on value provided for the argument map.type. For backwards compatibility, legacy static plots can still be generated by setting map.type to legacy. For legacy plots, legend.pos and map.adj should be chosen so that legend and map do not overlap, and the legend position can be fine-tuned using argument legend.inset. This is particularly useful for legend titles with more than one line. For details on inset specification for the default maps, see inset in legend.

Value

For default static maps, PlotMapPoints returns an object of class ggplot. This plot can also be assigned to a variable in the environment. For interactive Leaflet maps, PlotMapOutput returns an

object of class `leaflet`. For legacy static plots, `PlotMapOutput` returns a plot to the currently active plot device and invisibly an object of class `SpatialPointsDataFrame` as provided in argument `sites`, with plotted values and color codes added as columns in the data slot.

See Also

`ReadSubass` for HYPE result import; `ReadMapOutput` for a similar plot function

Examples

```
# Import plot data and subbasin points
require(sf)
te1 <- ReadSubass(filename = system.file("demo_model",
"results", "subass1.txt", package = "HYPEtools"))
te2 <- st_read(dsn = system.file("demo_model",
"gis", "Nytorp_station.gpkg", package = "HYPEtools"))
te2$SUBID <- 3587 # add station SUBID to point
te3 <- st_read(dsn = system.file("demo_model",
"gis", "Nytorp_map.gpkg", package = "HYPEtools"))
# plot NSE performance for discharge
PlotMapPoints(x = te1[, 1:2], sites = te2, sites.subid.column = 4, bg = te3)
```

PlotPerformanceByAttribute

Plot model performance by SUBID attributes

Description

Create scatterplots of model performance by SUBID attributes.

Usage

```
PlotPerformanceByAttribute(
  subass,
  subass.column = 2,
  groups = NULL,
  attributes,
  join.type = c("join", "cbind"),
  group.join.type = c("join", "cbind"),
  groups.color.pal = NULL,
  drop = TRUE,
  alpha = 0.4,
  trendline = TRUE,
  trendline.method = "lm",
  trendline.formula = NULL,
  trendline.alpha = 0.5,
```

```
  trendline.darken = 15,
  density.plot = FALSE,
  density.plot.type = c("density", "boxplot"),
  scale.x.log = FALSE,
  scale.y.log = FALSE,
  xsigma = 1,
  ysigma = 1,
  xlimits = c(NA, NA),
  ylimits = c(NA, NA),
  xbreaks = waiver(),
  ybreaks = waiver(),
  xlabels = waiver(),
  ylabels = waiver(),
  xlab = NULL,
  ylab = NULL,
  ncol = NULL,
  nrow = NULL,
  align = "hv",
  common.legend = TRUE,
  legend.position = "bottom",
  group.legend.title = "Group",
  common.y.axis = FALSE,
  summary.table = FALSE,
  table.margin = 0.4,
  filename = NULL,
  width = NA,
  height = NA,
  units = c("in", "cm", "mm", "px"),
  dpi = 300
)

PlotJohan(
  subass,
  subass.column = 2,
  groups = NULL,
  attributes,
  join.type = c("join", "cbind"),
  group.join.type = c("join", "cbind"),
  groups.color.pal = NULL,
  drop = TRUE,
  alpha = 0.4,
  trendline = TRUE,
  trendline.method = "lm",
  trendline.formula = NULL,
  trendline.alpha = 0.5,
  trendline.darken = 15,
  density.plot = FALSE,
  density.plot.type = c("density", "boxplot"),
```

```
  scale.x.log = FALSE,
  scale.y.log = FALSE,
  xsigma = 1,
  ysigma = 1,
  xlimits = c(NA, NA),
  ylimits = c(NA, NA),
  xbreaks = waiver(),
  ybreaks = waiver(),
  xlabels = waiver(),
  ylabels = waiver(),
  xlab = NULL,
  ylab = NULL,
  ncol = NULL,
  nrow = NULL,
  align = "hv",
  common.legend = TRUE,
  legend.position = "bottom",
  group.legend.title = "Group",
  common.y.axis = FALSE,
  summary.table = FALSE,
  table.margin = 0.4,
  filename = NULL,
  width = NA,
  height = NA,
  units = c("in", "cm", "mm", "px"),
  dpi = 300
)
```

Arguments

subass	Information to plot, typically model performances from imported HYPE 'subassX.txt' files. Data frame object with first column containing SUBIDs and additional columns containing model results to plot. See details.
subass.column	Column index of information in subass to plot on the y-axis of the output plots.
groups	Optional data frame object to specify groups of SUBIDs to plot separately. First column should contain SUBIDs and second column should contain group IDs.
attributes	Data frame object containing the subbasin attribute information to plot on the x-axis of the output plots. Typically a data frame created by SubidAttributeSummary
join.type	Specify how to join subass to attributes. Default "join" will perform a dplyr::left_join in which the order of the SUBIDs does not need to match. Additional option "cbind" will perform a cbind in which the order of the SUBIDs needs to match; this can be helpful if you want to create plots where subass performance data is calculated according to a grouping variable (e.g. month).
group.join.type	Specify how to join subass to groups. Default "join" will perform a dplyr::left_join in which the order of the SUBIDs does not need to match. Additional option "cbind" will perform a cbind in which the order of the SUBIDs needs to match;

this can be helpful if you want to create plots where subass performance data is calculated according to a grouping variable (e.g. month).

groups.color.pal
: Vector containing colors to use when plotting groups. Only used if groups is not NULL.

drop
: Logical, should unused factor levels be omitted from the legend. See ggplot2::scale_color_manual and ggplot2::scale_fill_manual.

alpha
: Numeric value to set transparency of dots in output plots. Should be in the range 0-1.

trendline
: Logical, if TRUE, then trendlines will be added to the output plots. Set to FALSE to hide trendlines. See ggplot2::geom_smooth.

trendline.method
: Specify method used to create trendlines. See ggplot2::geom_smooth.

trendline.formula
: Specify formula used to create trendlines. See ggplot2::geom_smooth.

trendline.alpha
: Numeric value to set transparency of trendlines in output plots. Should be in the range 0-1.

trendline.darken
: Numeric value to make the trendlines darker color shades of their corresponding scatterplot points. Should be in the range 1-100.

density.plot
: Logical, if TRUE, then density plots will be added to the output plots. Set to FALSE to hide density plots.

density.plot.type
: String, type of plot geometry to use for density plots: "density" for ggplot2::geom_density or "boxplot" for ggplot2::geom_boxplot. Outliers are hidden from the boxplots.

scale.x.log
: Vector describing if output plots should use a log scale on the x-axis. A pseudo-log scale will be used if any zero or negative values are present. If length of vector == 1, then the value will be used for all output plots. Vector values should be either TRUE or FALSE. See ggplot2::scale_x_log10.

scale.y.log
: Vector describing if output plots should use a log scale on the y-axis. A pseudo-log scale will be used if any zero or negative values are present. If length of vector == 1, then the value will be used for all output plots. Vector values should be either TRUE or FALSE. See ggplot2::scale_y_log10.

xsigma
: Numeric, scaling factor for the linear part of psuedo-long transformation of x axis. Used if scale.x.log is TRUE and zero or negative values are present. See scales::pseudo_log_trans.

ysigma
: Numeric, scaling factor for the linear part of psuedo-long transformation of y axis. Used if scale.y.log is TRUE and zero or negative values are present. See scales::pseudo_log_trans.

xlimits
: Vector containing minimum and maximum values for the x-axis of the output plots. See ggplot2::scale_x_continuous.

ylimits
: Vector containing minimum and maximum values for the y-axis of the output plots. See ggplot2::scale_y_continuous.

xbreaks	Vector containing the break values used for the x-axis of the output plots. See ggplot2::scale_x_continuous.
ybreaks	Vector containing the break values used for the y-axis of the output plots. See ggplot2::scale_y_continuous.
xlabels	Vector containing the labels for each break value used for the x-axis of the output plots. See ggplot2::scale_x_continuous.
ylabels	Vector containing the labels for each break value used for the y-axis of the output plots. See ggplot2::scale_y_continuous.
xlab	String containing the text to use for the x-axis title of the output plots. See ggplot2::xlab.
ylab	String containing the text to use for the y-axis title of the output plots. See ggplot2::ylab.
ncol	Integer, number of columns to use in the output arranged plot. See ggpubr::ggarrange.
nrow	Integer, number of rows to use in the output arranged plot. See ggpubr::ggarrange.
align	Specify how output plots should be arranged. See ggpubr::ggarrange.
common.legend	Specify if arranged plot should use a common legend. See ggpubr::ggarrange.
legend.position	Specify position of common legend for arranged plot. See ggpubr::ggarrange. Use "none" to hide legend.
group.legend.title	String, title for plot legend when generating plots with groups.
common.y.axis	Logical, if TRUE, then only one y-axis label and marginal density plot will be provided. If FALSE, then separate y-axis labels and marginal density plots will be included for each subplot.
summary.table	Logical, if TRUE, then a table providing summary statistics will be included at the bottom of the output plot.
table.margin	Numeric, controls spacing between plots and summary table.
filename	String, filename used to save plot. File extension must be specified. See ggplot2::ggsave.
width	Numeric, specify width of output plot. See ggplot2::ggsave.
height	Numeric, specify height of output plot. See ggplot2::ggsave.
units	Specify units of width and height. See ggplot2::ggsave.
dpi	Specify resolution of output plot. See ggplot2::ggsave.

Details

PlotPerformanceByAttribute can be used to analyze model performance according to subbasin attributes. The function requires two primary inputs; Model performance information is contained in the subass input, and subbasin attribute information is contained in the attributes input. The subass.column argument controls which column of the subass data frame will be used as the y-coordinate of points. Plots will be generated for each column in the attributes data frame (except for the column named "SUBID") using the column values as the x-coordinate of the points.

A subbasin attribute summary table can be generated using SubidAttributeSummary, and additional columns can be joined to the data frame to add additional output plots.

Value

PlotPerformanceByAttribute returns a plot to the currently active plot device.

See Also

ReadSubass for HYPE result import; SubidAttributeSummary for subbasin attribute summary

Examples

```
subass <- ReadSubass(filename = system.file("demo_model", "results",
  "subass1.txt",
  package = "HYPEtools"
), check.names = TRUE)
gd <- ReadGeoData(filename = system.file("demo_model",
  "GeoData.txt",
  package = "HYPEtools"
))
gc <- ReadGeoClass(filename = system.file("demo_model",
  "GeoClass.txt",
  package = "HYPEtools"
))

attributes <- SubidAttributeSummary(subids <- subass$SUBID,
  gd = gd, gc = gc,
 mapoutputs = c(system.file("demo_model", "results", "mapCOUT.txt", package = "HYPEtools")),
  upstream.gd.cols = c("SLOPE_MEAN")
)

PlotPerformanceByAttribute(
  subass = subass,
  attributes = attributes[, c("SUBID", "landuse_1", "landuse_2", "landuse_3")],
  xlimits = c(0, 1)
)
```

PlotSimObsRegime *Plot annual regimes of simulated and observed variables*

Description

A combined plot for annual regimes with box plot elements for observed variables and ribbon elements for simulated variables. Particularly designed for comparisons of sparse observations with high-density model results, e.g. for in-stream nutrients.

Usage

```
PlotSimObsRegime(
  x,
  sim,
  obs,
  ts.in = NULL,
  ts.out = "month",
  start.mon = 1,
  add.legend = TRUE,
  pos.legend = "topright",
  inset = 0,
  l.legend = NULL,
  log = FALSE,
  ylim = NULL,
  xlab = NULL,
  ylab = NULL,
  mar = c(3, 3, 1, 1) + 0.1
)
```

Arguments

x	Data frame, with column-wise equally-spaced time series of HYPE variables. Date-times in POSIXct format in first column. Typically an imported basin output file from HYPE using ReadBasinOutput. See details for HYPE output variables required for plotting.
sim, obs	Character string keywords, observed and simulated HYPE variable IDs to plot. Not case-sensitive, but must exist in x. Set to NULL to omit corresponding elements in plot.
ts.in	Character string, timestep of x, searches for an attribute timestep in x per default. Otherwise one of "month", "week", "day", or "nhour" (n = number of hours).
ts.out	Character string, aggregation timestep for simulation results, defaults to ts.in. This timestep must be equal to or longer than ts.in.
start.mon	Integer between 1 and 12, starting month of the hydrological year, used to order the output.
add.legend	Logical. If TRUE, a legend will be added to the plot.
pos.legend	Character string keyword for legend positioning. See Details in link{legend}.
inset	Integer, legend inset as fraction of plot region, one or two values for x and y. See link{legend}.
l.legend	Character vector of length 2 containing variable labels for legend, first for sim, then for obs. If non-NULL, variable labels are read from here instead of sim and obs.
log	Logical, if TRUE, y-axis will be log-scaled.
ylim	Numeric vector of length two, giving y-axis limits. Defaults to min-max range of all plotted data.

xlab	Character string or plotmath expression string, x-axis label. Defaults to a string giving the time period on which the regime is based.
ylab	Character or plotmath expression string. Y-axis label. Defaults to a HYPE variable unit string taken from x attributes 'hypeunit'.
mar	Numeric vector of length 4, margin specification passed to par.

Details

PlotSimObsRegime combines ribbons and box plot elements. Box plot elements are composed as defaults from boxplot, i.e. boxes with 25\ extreme values as points. Observation counts per month over the observation period are printed above the x-axis.

Aggregation time length of the simulated variable can be chosen in argument ts.out, resulting in more or less smoothed ribbons. For the observed variable, the aggregation is fixed to months, in order to aggregate enough values for each box plot element.

Value

PlotSimObsRegime returns a plot to the currently active plot device, and invisibly a list object containing three elements with the plotted data and variable IDs. Element obs contains a list as returned by AnnualRegime. Element obs contains a list with two elements, a vector refdate with x positions of box plots elements, and a list reg.obs with observations for the monthly box plot elements. Element variable contains a named vector with HYPE variable IDs for observations and simulations. sim and obs returned empty if corresponding function argument was NULL.

See Also

PlotAnnualRegime for a more generic annual regime plot, AnnualRegime to compute annual regimes only.

Examples

```
# Plot observed and simulated discharge
te <- ReadBasinOutput(filename = system.file("demo_model",
"results", "0003587.txt", package = "HYPEtools"))
PlotSimObsRegime(x = te, sim = "cout", obs = "rout", start.mon = 10)
```

PlotSubbasinRouting *Plot HYPE model subbasin routing.*

Description

Plot routing of subbasins for a HYPE model on an interactive map.

Usage

```
PlotSubbasinRouting(
  map,
  map.subid.column = 1,
  gd = NULL,
  bd = NULL,
  plot.scale = TRUE,
  plot.searchbar = FALSE,
  weight = 0.5,
  opacity = 1,
  fillColor = "#4d4d4d",
  fillOpacity = 0.25,
  line.color = NULL,
  line.weight = 5,
  line.opacity = 1,
  seed = NULL,
  darken = 0,
  font.size = 10,
  file = "",
  vwidth = 1424,
  vheight = 1000,
  html.name = ""
)
```

Arguments

map	Path to file containing subbasin polygon GIS data (e.g. shapefile or geopackage) or a SpatialPolygonsDataFrame or sf object. For large maps, a small/simplified polygon file should be used as larger files can take an excessive amount of time to render.
map.subid.column	Integer, column index in the map 'data' slot holding SUBIDs (sub-catchment IDs). Only required if providing GeoData information with gd.
gd	Path to model GeoData.txt or a GeoData object from ReadGeoData. Only required if map does not contain SUBID and/or MAINDOWN fields.
bd	Path to model BranchData.txt or a BranchData object from ReadBranchData. Only required if model has a BranchData.txt file.
plot.scale	Logical, include a scale bar on the map.
plot.searchbar	Logical, if TRUE, then a search bar will be included. See leaflet.extras::addSearchFeatures().
weight	Numeric, weight of subbasin boundary lines. See leaflet::addPolygons().
opacity	Numeric, opacity of subbasin boundary lines. See leaflet::addPolygons().
fillColor	String, color of subbasin polygons. See leaflet::addPolygons().
fillOpacity	Numeric, opacity of subbasin polygons. See leaflet::addPolygons().
line.color	String, color codes to use for each routing line. If NULL, then random colors will be used. If a single value is provided, then that color will be used for all lines.

line.weight	Numeric, weight of routing lines. See leaflet::addPolylines().If a single value is provided, then that weight will be used for all lines.
line.opacity	Numeric, opacity of routing lines. See leaflet::addPolylines().
seed	Integer, seed number to to produce repeatable color palette.
darken	Numeric specifying the amount of darkening applied to the random color palette. Negative values will lighten the palette. See distinctColorPalette.
font.size	Numeric, font size (px) for map subbasin labels.
file	Save map to an image file by specifying the path to the desired output file using this argument. File extension must be specified. See mapview::mapshot(). You may need to run webshot::install_phantomjs() the first time you save a map to an image file.
vwidth	Numeric, width of the exported map image in pixels. See webshot::webshot().
vheight	Numeric, height of the exported map image in pixels. See webshot::webshot().
html.name	Save map to an interactive HTML file by specifying the path to the desired output file using this argument. File extension must be specified. See htmlwidgets::saveWidget().

Details

PlotSubbasinRouting generates an interactive Leaflet map with lines indicating the routing of flow between subbasins. GeoData information only needs to be provided if the map GIS data does not include SUBID and/or MAINDOWN fields. BranchData information only needs to be provided if model has a BranchData.txt file. Subbasin routing lines are randomly assigned a color using distinctColorPalette.

Value

Returns an interactive Leaflet map.

Examples

```
## Not run:
PlotSubbasinRouting(
  map = system.file("demo_model",
                    "gis", "Nytorp_map.gpkg",
                    package = "HYPEtools"
  ),
  gd = system.file("demo_model", "GeoData.txt", package = "HYPEtools"),
  map.subid.column = 25
)

## End(Not run)
```

r *Pearson product-moment correlation coefficient r*

Description

Pearson product-moment correlation coefficient calculation, a specific case of function cor.

Usage

```
r(sim, obs, ...)

## S3 method for class 'HypeSingleVar'
r(sim, obs, progbar = TRUE, ...)
```

Arguments

sim	HypeSingleVar array with simulated variable (one or several iterations).
obs	HypeSingleVar array with observed variable, (one iteration). If several iterations are present in the array, only the first will be used.
...	Ignored.
progbar	Logical, if TRUE progress bars will be printed for main computational steps.

Details

This function wraps a call to `cor(x = obs, y = sim, use = "na.or.complete", method = "pearson")`.

Method `r.HypeSingleVar` calculates Pearson's r for imported HYPE outputs with single variables for several catchments, i.e. time and map files, optionally multiple model runs combined, typically results from calibration runs.

Value

`r.HypeSingleVar` returns a 2-dimensional array of Pearson correlation coefficients for all SUBIDs and model iterations provided in argument `sim`, with values in the same order as the second and third dimension in `sim`, i.e. `[subid, iteration]`.

See Also

cor, on which the function is based. ReadWsOutput for importing HYPE calibration results.

Examples

```
# Create dummy data, discharge observations with added white noise as model simulations
te1 <- ReadObs(filename = system.file("demo_model", "Qobs.txt", package = "HYPEtools"))
te1 <- HypeSingleVar(x = array(data = unlist(te1[, -1]) + runif(n = nrow(te1),
                                                                min = -.5, max = .5),
                               dim = c(nrow(te1), ncol(te1) - 1, 1),
                               dimnames = list(rownames(te1), colnames(te1)[-1])),
```

```
                      datetime = te1$DATE, subid = obsid(te1), hype.var = "cout")
te2 <- ReadObs(filename = system.file("demo_model", "Qobs.txt", package = "HYPEtools"))
te2 <- HypeSingleVar(x = array(data = unlist(te2[, -1]),
                               dim = c(nrow(te2), ncol(te2) - 1, 1),
                               dimnames = list(rownames(te2), colnames(te2)[-1])),
                      datetime = te2$DATE, subid = obsid(te2), hype.var = "rout")
# Pearson correlation
r(sim = te1, obs = te2, progbar = FALSE)
```

`ReadBasinOutput` *Read a Basin Output File*

Description

This is a convenience wrapper function to import a basin output file as data frame or matrix into R.

Usage

```
ReadBasinOutput(
  filename,
  dt.format = "%Y-%m-%d",
  type = c("df", "dt", "hmv"),
  id = NULL,
  warn.nan = FALSE
)
```

Arguments

`filename`	Path to and file name of the basin output file to import. Windows users: Note that Paths are separated by '/', not '\'.
`dt.format`	Date-time `format` string as in `strptime`. Incomplete format strings for monthly and annual values allowed, e.g. '\ be imported as `character`, applicable e.g. for files containing just one row of summary values over the model period.
`type`	Character, keyword for data type to return. `"df"` to return a standard data frame, `"dt"` to return a `data.table` object, or `"hmv"` to return a `HypeMultiVar` array.
`id`	Integer, SUBID or OUTREGID of the imported sub-basin or outregion results. If `NULL` (default), the function attempts to read this from the imported file's name, which only works for standard HYPE basin output file names or any where the first 7 digits give the SUBID or OUTREGID with leading zeros. See details.
`warn.nan`	Logical, check if imported results contain any `NaN` values. If `TRUE` and `NaN`s are found, a warning is thrown and affected SUBIDs saved in an attribute `subid.nan`. Adds noticeable overhead to import time for large files.

Details

ReadBasinOutput is a convenience wrapper function of fread from package data.table::data.table, with conversion of date-time strings to POSIX time representations. Monthly and annual time steps are returned as first day of the time step period.

HYPE basin output files can contain results for a single sub-basin or for a user-defined output region. ReadBasinOutput checks HYPE variable names (column headers in imported file) for an "RG"-prefix. If it is found, the ID read from either file name or argument id is saved to attribute outregid, otherwise to attribute subid.

Value

ReadBasinOutput returns a data.frame, data.table::data.table, or a HypeMultiVar array. Data frames and data tables contain additional attributes: hypeunit, a vector of HYPE variable units, subid and outregid, the HYPE SUBID/OUTREGID to which the time series belong (both attributes always created and assigned NA if not applicable to data contents), timestep with a time step keyword attribute, and comment with contents of an optional first-row comment (NA otherwise). An additional attribute subid.nan might be returned, see argument warn.nan.

Note

For the conversion of date/time strings, time zone "UTC" is assumed. This is done to avoid potential daylight saving time side effects when working with the imported data (and possibly converting to string representations during the process).

HYPE results are printed to files using a user-specified accuracy. This accuracy is specified in 'info.txt' as a number of decimals to print. If large numbers are printed, this can result in a total number of digits which is too large to print. Results will then contain values of '**********'. ReadBasinOutput will convert those cases to 'NA' entries.

Current versions of HYPE allow for defining significant numbers of digits instead of fixed ones, which should prevent this issue from arising.

Examples

```
te <- ReadBasinOutput(filename = system.file("demo_model",
"results", "0003587.txt", package = "HYPEtools"))
```

ReadClassData *Read a 'ClassData.txt' File*

Description

This is a convenience wrapper function to import a ClassData file as data frame into R. ClassData files contain definitions of SLC (**S**oil and **L**and use **C**rop) classes in five to 15 predefined columns, see ClassData.txt documentation.

Usage

```
ReadClassData(
  filename = "ClassData.txt",
  encoding = c("unknown", "UTF-8", "Latin-1"),
  verbose = TRUE
)
```

Arguments

filename	Path to and file name of the ClassData file to import. Windows users: Note that Paths are separated by '/', not '\'.
encoding	Character string, encoding of non-ascii characters in imported text file. Particularly relevant when importing files created under Windows (default encoding "Latin-1") in Linux (default encoding "UTF-8") and vice versa. See also argument description in fread.
verbose	Print information on number of data columns in imported file.

Details

ReadClassData is a convenience wrapper function of fread, with treatment of leading comment rows. Column names are created on import, optional comment rows are imported as strings in attribute 'comment'. Optional inline comments (additional non-numeric columns) are automatically identified and imported along with data columns.

Value

ReadClassData returns a data frame with added attribute 'comment'.

See Also

ReadGeoClass

Examples

```
te <- ReadClassData(filename = system.file("demo_model", "ClassData.txt", package = "HYPEtools"))
te
```

ReadDescription *Read a 'description.txt' file*

Description

Read a 'description.txt' file as list object into R. A 'description.txt' file contains land use, soil, and crop class names of a HYPE set-up, as well as model set-up name and version.

Usage

```
ReadDescription(
  filename,
  gcl = NULL,
  ps = NULL,
  encoding = c("unknown", "UTF-8", "latin1")
)
```

Arguments

filename	Path to and file name of the 'description.txt' file to import.
gcl	dataframe, GeoClass.txt file imported with ReadGeoClass to compare class IDs with. A warning will be thrown if not all class IDs in gcl exist in the description file.
ps	dataframe, PointSourceData.txt file imported with ReadPointSourceData to compare point source type IDs with. A warning will be thrown if not all type IDs in ps exist in the description file.
encoding	Character string, encoding of non-ascii characters in imported text file. Particularly relevant when importing files created under Windows (default encoding "Latin-1") in Linux (default encoding "UTF-8") and vice versa. See also argument description in scan.

Details

ReadDescription imports a 'description.txt' into R. This file is not used by HYPE, but is convenient for e.g. plotting legend labels or examining imported GeoClass files. E.g., PlotBasinSummary requires a list as returned from ReadDescription for labeling.

A 'description.txt' file consists of 28 lines, alternating names and semicolon-separated content. Lines with names are not read by the import function, they just make it easier to compose and read the actual text file.

File contents read by ReadDescription:

- HYPE set-up name (line 2)
- HYPE set-up version (line 4)
- Land use class IDs (line 6)
- Land use class names (line 8)
- Land use class short names (line 10)
- Soil class IDs (line 12)
- Soil class names (line 14)
- Soil class short names (line 16)
- Crop class IDs (line 18)
- Crop class names (line 20)
- Crop class short names (line 22)
- Point Source IDs (line 24)

- Point Source type names (line 26)
- Point Source type short names (line 28)

Note that Crop class IDs start from `0`, which means no crop, whereas land use and soil IDs start from 1 (or higher).

Formatting example for description.txt files:

```
# Name
MyHYPE
# Version
0.1
# Land use class IDs
1;2
# Land use class names
Agriculture;Coniferous forest
# Short land use class names
Agric.;Conif. f.
# Soil class IDs
1;2
# Soil class names
Coarse soils;Medium to fine soils
# Short soil class names
Coarse;Medium
# Crop class IDs
0;1;2
# Crop class names
None;Row crops;Autumn-sown cereal
# Short crop class names
None;Row;Aut.-sown
# Point source type IDs
-1;1;2
# Point source type names
Abstraction;Primary;Secondaryl
# Short point source type names
ABS;NP1;NP2
```

Value

`ReadDescription` returns a named list with named character elements, corresponding to the imported lines:

`Name`, `Version`, `lu.id`, `Landuse`, `lu` (short names), `so.id`, `Soil`, `so` (short names), `cr.id`, `Crop`, `cr` (short names), `ps.id`, `PointSource`, `ps` (short names)

Examples

```
te <- ReadDescription(filename = system.file("demo_model",
"description.txt", package = "HYPEtools"))
te
```

ReadGeoClass *Read a 'GeoClass.txt' File*

Description

This is a convenience wrapper function to import a GeoClass file as data frame into R. GeoClass files contain definitions of SLC (**S**oil and **L**and use **C**rop) classes in twelve to 14 predefined columns, see GeoClass.txt documentation.

Usage

```
ReadGeoClass(
  filename = "GeoClass.txt",
  encoding = c("unknown", "UTF-8", "Latin-1"),
  verbose = TRUE
)
```

Arguments

filename	Path to and file name of the GeoClass file to import. Windows users: Note that Paths are separated by '/', not '\'.
encoding	Character string, encoding of non-ascii characters in imported text file. Particularly relevant when importing files created under Windows (default encoding "Latin-1") in Linux (default encoding "UTF-8") and vice versa. See also argument description in fread.
verbose	Print information on number of data columns in imported file.

Details

ReadGeoClass is a convenience wrapper function of fread, with treatment of leading comment rows. Column names are created on import, optional comment rows are imported as strings in attribute 'comment'. Optional inline comments (additional non-numeric columns) are automatically identified and imported along with data columns.

Value

ReadGeoClass returns a data frame with added attribute 'comment'.

See Also

ReadClassData

Examples

```
te <- ReadGeoClass(filename = system.file("demo_model", "GeoClass.txt", package = "HYPEtools"))
te
```

ReadGeoData *Read a 'GeoData.txt' file*

Description

Import a GeoData file into R.

Usage

```
ReadGeoData(
  filename = "GeoData.txt",
  sep = "\t",
  encoding = c("unknown", "UTF-8", "Latin-1"),
  remove.na.cols = TRUE
)
```

Arguments

filename	Path to and file name of the GeoData file to import. Windows users: Note that Paths are separated by '/', not '\'.
sep	character string. Field separator character as described in read.table.
encoding	Character string, encoding of non-ascii characters in imported text file. Particularly relevant when importing files created under Windows (default encoding "Latin-1") in Linux (default encoding "UTF-8") and vice versa. See also argument description in fread.
remove.na.cols	Logical, remove columns which have all NA values.

Details

ReadGeoData uses fread from the data.table::data.table package with type numeric type for columns AREA and RIVLEN (if they exist), and upper-case column names.

Value

If the imported file is a HYPE-conform GeoData file, ReadGeoData returns an object of S3 class HypeGeoData (see the class description there), providing its own summary method. If mandatory GeoData columns are missing, a standard dataframe is returned along with informative warning messages.

Examples

```
te <- ReadGeoData(filename = system.file("demo_model", "GeoData.txt", package = "HYPEtools"))
summary(te)
```

`ReadInfo` *Read an 'info.txt' file*

Description

Import a HYPE model settings information file as list into R.

Usage

```
ReadInfo(
  filename = "info.txt",
  encoding = c("unknown", "UTF-8", "latin1"),
  mode = c("simple", "exact"),
  comment.duplicates = TRUE
)
```

Arguments

`filename`	Path to and file name of the info.txt file to import.
`encoding`	Character string, encoding of non-ascii characters in imported text file. Particularly relevant when importing files created under Windows (default encoding "Latin-1") in Linux (default encoding "UTF-8") and vice versa. See also argument description in `scan`.
`mode`	Use `simple` to read info.txt file as a nested list to that provides easy access to key information. Alternatively, use `exact` to read info.txt file as a list matching the exact info.txt file structure (including all comment lines).
`comment.duplicates`	Logical, if `TRUE`, then duplicate codes will be commented out when reading the input file. If `FALSE`, then the input file will not not be checked for duplicate codes.

Details

Using `ReadInfo` with the `simple` mode discards all comments of the imported file (comment rows and in-line comments). The function's purpose is to quickly provide access to settings and details of a model run, not to mirror the exact info.txt file structure into an R data object. If you would like to mirror the exact file structure, then use the `exact` mode.

Value

`ReadInfo` returns a named list. List names are settings codes (see info.txt documentation). Settings with two codes are placed in nested lists, e.g. `myinfo$basinoutput$variable`. Multi-line subbasin definitions for basin outputs and class outputs are merged to single vectors on import.

See Also

`WriteInfo` `AddInfoLine` `RemoveInfoLine`

Examples

```
te <- ReadInfo(filename = system.file("demo_model",
"info.txt", package = "HYPEtools"))
te
```

`ReadMapOutput` *Read a Map Output File*

Description

This is a convenience wrapper function to import a map output file ('map<*HYPE_output_variable*>.txt') into R.

Usage

```
ReadMapOutput(
  filename,
  dt.format = NULL,
  hype.var = NULL,
  type = c("df", "dt", "hsv"),
  warn.nan = FALSE,
  col.prefix = "X"
)
```

Arguments

`filename`	Path to and file name of the map output file to import. Windows users: Note that Paths are separated by '/', not '\'.
`dt.format`	Date-time `format` string as in `strptime`, for conversion of date-time information in column headers to POSIX dates, which are returned as attribute. Incomplete format strings for monthly and annual values allowed, e.g. `"\%Y"`. *Defaults to* `NULL`*, which prevents date-time conversion*, applicable e.g. for files containing just one column of summary values over the model period.
`hype.var`	Character string, a four-letter keyword to specify HYPE variable ID of file contents. See list of HYPE variables. If `NULL` (default), the variable ID is extracted from the provided file name, which only works for standard HYPE map output file names.
`type`	Character, keyword for data type to return. `"df"` to return a standard data frame, `"dt"` to return a `data.table` object, or `"hsv"` to return a `HypeSingleVar` array.
`warn.nan`	Logical, check if imported results contain any `NaN` values. If `TRUE` and `NaN`s are found, a warning is thrown and affected SUBIDs saved in an attribute `subid.nan`. Adds noticeable overhead to import time for large files.
`col.prefix`	String, prefix added to mapoutput column names. Default is `X`. Set to `NULL` to ignore.

Details

ReadMapOutput is a convenience wrapper function of fread from package data.table::data.table, with conversion of date-time strings to POSIX time representations. Monthly and annual time steps are returned as first day of the time step period.

Value

ReadMapOutput returns a data.frame, data.table::data.table, or a HypeSingleVar array. Data frames and data tables contain additional attributes: variable, giving the HYPE variable ID, date, a vector of date-times (corresponding to columns from column 2), timestep with a time step attribute, and comment with the first line of the imported file as text string. An additional attribute subid.nan might be returned, see argument warn.nan.

Note

HYPE results are printed to files using a user-specified accuracy. This accuracy is specified in 'info.txt' as a number of decimals to print. If large numbers are printed, this can result in a total number of digits which is too large to print. Results will then contain values of '**********'. ReadMapOutput will convert those cases to 'NA' entries.

Current versions of HYPE allow for defining significant instead of fixed number of digits, which should prevent this issue from arising.

Examples

```
te <- ReadMapOutput(filename = system.file("demo_model",
"results", "mapEVAP.txt", package = "HYPEtools"), dt.format = NULL)
te
```

ReadObs *Read HYPE observation data files*

Description

Import single-variable HYPE observation files into R.

Usage

```
ReadObs(
  filename,
  variable = "",
  dt.format = NULL,
  nrows = -1,
  type = c("df", "dt"),
  select = NULL,
  obsid = NULL
)
```

```
ReadPTQobs(
  filename,
  variable = "",
  dt.format = NULL,
  nrows = -1,
  type = c("df", "dt"),
  select = NULL,
  obsid = NULL
)
```

Arguments

`filename`	Path to and file name of the file to import. Windows users: Note that Paths are separated by '/', not '\'.
`variable`	Character string, HYPE variable ID of file contents. If `""` (default), the ID is extracted from `filename`, which only works with HYPE input data file names or file names including those names (e.g. 'Pobs_old.txt', 'testSFobs.txt'). Some of the observation data files have no corresponding HYPE variable ID. In these cases, a dummy ID is used, see table in Details. If automatic extraction fails, attribute `variable` is set to `"other"`. Alternatively, any other variable name can be provided.
`dt.format`	Optional date-time `format` string as in `strptime`. If `NULL`, then HYPEtools will try to identify the format automatically.
`nrows`	Number of rows to import. A value of `-1` indicates all rows, a positive integer gives the number of rows to import.
`type`	Character, keyword for data type to return. `"df"` to return a standard data frame or `"dt"` to return a `data.table` object.
`select`	Integer vector, column numbers to import. Note: first column with dates must be imported and will be added if missing.
`obsid`	Integer vector, HYPE OBSIDs to import. Alternative to argument `select`, takes precedence if both are provided.

Details

`ReadObs` is a convenience wrapper function of `fread` from package data.table::data.table, with conversion of date-time strings to POSIX time representations. Observation IDs (SUBIDs or IDs connected to SUBIDs with a ForcKey.txt file) are returned as integer attribute `obsid` (directly accessible through `obsid`).

Observation file types with automatic (dummy) `variable` attribute assignment:

File	HYPE variable ID (*: dummy ID)
Pobs.txt	prec
Tobs.txt	temp
Qobs.txt	rout
TMINobs.txt	tmin*

TMAXobs.txt	tmax*
VWobs.txt	vwnd*
UWobs.txt	uwnd*
SFobs.txt	snff*
SWobs.txt	swrd*
RHobs.txt	rhum*
Uobs.txt	wind*

Value

`ReadObs` returns a data frame or data table with additional attributes: `obsid` with observation IDs, `timestep` with a time step string, either `"day"` or `"nhour"` (only daily or n-hourly time steps supported), and `variable` with a HYPE variable ID string.

Note

For the conversion of date/time strings, time zone "UTC" is assumed. This is done to avoid potential daylight saving time side effects when working with the imported data (and e.g. converting to string representations during the process).

See Also

`WriteObs` `ReadXobs` for multi-variable HYPE observation files

Examples

```
te <- ReadObs(filename = system.file("demo_model", "Tobs.txt", package = "HYPEtools"))
head(te)
```

ReadOptpar *Read an 'optpar.txt' file*

Description

This function imports an 'optpar.txt' into a list.

Usage

```
ReadOptpar(filename = "optpar.txt", encoding = c("unknown", "UTF-8", "latin1"))
```

Arguments

`filename`	Path to and file name of the 'optpar.txt' file to import.
`encoding`	Character string, encoding of non-ascii characters in imported text file. Particularly relevant when importing files created under Windows (default encoding "Latin-1") in Linux (default encoding "UTF-8") and vice versa. See also argument description in `scan`.

Details

ReadOptpar imports HYPE 'optpar.txt' files. Optpar files contain instructions for parameter calibration/optimization and parameter value ranges, for details on the file format, see the optpar.txt online documentation.

Value

ReadOptpar returns a list object with three elements:

- comment, the file's first-row comment string.
- tasks, a two-column dataframe with row-wise key-value pairs for tasks and settings.
- pars, a list of dataframes, each containing values for one parameter. Three columns each, holding parameter range minima, maxima, and intervals. The number of rows in each dataframe corresponds to the number of soil or land use classes for class-specific parameters. Parameter names as list element names.

See Also

ReadPar

Examples

```
te <- ReadOptpar(filename = system.file("demo_model", "optpar.txt", package = "HYPEtools"))
te
```

ReadPar *Read a 'par.txt' file*

Description

Import a HYPE parameter file as list into R.

Usage

```
ReadPar(filename = "par.txt", encoding = c("unknown", "UTF-8", "latin1"))
```

Arguments

filename	Path to and file name of the parameter file to import. Windows users: Note that Paths are separated by '/', not '\'.
encoding	Character string, encoding of non-ascii characters in imported text file. Particularly relevant when importing files created under Windows (default encoding "Latin-1") in Linux (default encoding "UTF-8") and vice versa. See also argument description in scan.

Details

ReadPar checks for inline comments in 'par.txt' files, these are moved to separate "lines" (list elements).

Value

ReadPar returns a list of named vectors. Parameters are returned as numeric vectors with HYPE parameter names as list element names. Comments are returned in separate list elements as single character strings, former inline comments are moved to elements preceding the original comment position (i.e. to a line above in the par.txt file structure). Comment elements are named `!!`.

Examples

```
te <- ReadPar(filename = system.file("demo_model", "par.txt", package = "HYPEtools"))
te
```

ReadPmsf *Read a 'pmsf.txt' file*

Description

This is a small convenience function to import a 'partial model setup file' as integer vector into R.

Usage

```
ReadPmsf(filename = "pmsf.txt")
```

Arguments

filename	Path to and file name of the pmsf file to import. Windows users: Note that Paths are separated by '/', not '\'.

Details

ReadPmsf imports 'pmsf.txt' files, which contain SUBIDs and are used to run only parts of a HYPE setup's domain without having to extract a separate model setup. For details on the file format, see the pmsf.txt online documentation. Pmsf.txt files imported with ReadPmsf are stripped from the first value containing the total number of subcatchments in the file. No additional attribute is added to hold this number since it can be easily obtained using length.

Value

ReadPmsf returns an integer vector.

Examples

```
te <- ReadGeoData(filename = system.file("demo_model", "GeoData.txt", package = "HYPEtools"))
te
```

`ReadSimass` *Read a 'simass.txt' file*

Description

Import a HYPE simass.txt simulation assessment file as data frame into R. Simulation assessment files contain domain-wide aggregated performance criteria results, as defined in 'info.txt'.

Usage

```
ReadSimass(filename = "simass.txt")
```

Arguments

`filename`	Path to and file name of the 'simass.txt' file to import.

Details

`ReadSimass` imports a simulation assessment file into R. HYPE simass.txt files contain domain-wide performance measures for observed-simulated variable pairs as defined in HYPE info.txt files.

The function interprets character-coded time steps (e.g. `"DD"` for daily time steps), as used in some HYPE versions. **Sub-daily time steps are currently not treated** and will probably result in a warning during time step evaluation within the function. Please contact the developers if you need support for sub-daily time steps!

Value

`ReadSubass` returns a data frame with columns for HYPE variable names (observed, simulated), aggregation periods, and performance measure values of evaluated variable pairs. Aggregation periods are coded as in info.txt files, i.e. 1 = daily, 2 = weekly, 3 = monthly, 4 = annual. Metadata is added to the data frame as additional `attributes`:

- `names.long`, `character` vector with long names, corresponding to abbreviations uses as actual column names
- `n.simulation`, `integer`, simulation number (e.g. with Monte Carlo simulations)
- `crit.total`, `numeric`, total criteria value
- `crit.conditional`, `numeric`, conditional criteria value
- `threshold`, `integer`, data limit threshold

See Also

`ReadSubass`

Examples

```
te <- ReadSimass(filename = system.file("demo_model",
"results", "simass.txt", package = "HYPEtools"))
te
```

`ReadSubass` *Read a 'subassX.txt' file*

Description

This is a convenience wrapper function to import an subassX.txt sub-basin assessment file as data frame into R. Sub-basins assessment files contain performance criteria results, as defined in 'info.txt', for individual sub-basins with observations.

Usage

```
ReadSubass(
  filename = "subass1.txt",
  nhour = NULL,
  check.names = FALSE,
  na.strings = c("****************", "-9999")
)
```

Arguments

`filename`	Path to and file name of the 'subassX.txt' file to import.
`nhour`	Integer, time step of sub-daily model results in hours. See details.
`check.names`	Logical. If `TRUE`, then the names of the variables are check to make sure they are syntactically valid.
`na.strings`	Vector of strings that should be read as NA.

Details

`ReadSubass` imports a sub-basin assessment file into R. Information on model variables evaluated in the file is imported as additional `attributes` `variables`, the evaluation time step in an attribute `timestep`.

Sub-daily time steps are reported with time step code '0' in HYPE result files. In order to preserve the time step information in the imported R object, users must provide the actual model evaluation time step in hours in argument `nhour` in the sub-daily case.

Value

`ReadSubass` returns a data frame with two additional attributes: `variables` contains a 2-element character vector with IDs of evaluated observed and simulated HYPE variables, `timestep` contains a character keyword detailing the evaluation time step.

Examples

```
te <- ReadSubass(filename = system.file("demo_model",
"results", "subass1.txt", package = "HYPEtools"))
te
```

`ReadTimeOutput` *Read a Time Output File*

Description

Import a time output file 'time<*HYPE_output_variable*>.txt' or a converted time output file in netCDF format into R.

Usage

```
ReadTimeOutput(
  filename,
  dt.format = "%Y-%m-%d",
  hype.var = NULL,
  out.reg = NULL,
  type = c("df", "dt", "hsv"),
  select = NULL,
  id = NULL,
  nrows = -1L,
  skip = 0L,
  warn.nan = FALSE,
  verbose = TRUE
)
```

Arguments

`filename`	Path to and file name of the time output file to import. Acceptable file choices are `*.txt` files following HYPE time output file format or `.nc` files following the HYPE netCDF formatting standard. See also details for netCDF import.
`dt.format`	Date-time `format` string as in `strptime`. Incomplete format strings for monthly and annual values allowed, e.g. `"\%Y"`. If set to `NULL`, no date-time conversion will be attempted and the column will be imported as `character`, applicable e.g. for files containing just one row of summary values over the model period.
`hype.var`	Character, HYPE variable ID in `x`. See list of HYPE variables. If `NULL` (default), the variable ID is extracted from the provided file name, which only works for standard HYPE time output file names (incl. regional and class outputs).
`out.reg`	Logical, specify if file contents are sub-basin or output region results (i.e. SUBIDs or OUTREGIDs as columns). `TRUE` for output regions, `FALSE` for sub-basins. *Use only in combination with user-provided* `hype.var` *argument.*

type	Character, keyword for data type to return. "df" to return a standard data frame, "dt" to return a data.table object, or "hsv" to return a HypeSingleVar array.
select	Integer vector, column numbers to import. Note: first column with dates must be imported and will be added if missing.
id	Integer vector, HYPE SUBIDs/OUTREGIDs to import. Alternative to argument select, takes precedence if both are provided.
nrows	Integer, number of rows to import, see documentation in fread.
skip	Integer, number of *data* rows to skip on import. Time output header lines are always skipped.
warn.nan	Logical, check if imported results contain any NaN values. If TRUE and NaNs are found, a warning is thrown and affected IDs saved in an attribute id.nan. Adds noticeable overhead to import time for large files.
verbose	Logical, print information during import.

Details

ReadTimeOutput imports from text or netCDF files. *netCDF import is experimental and not feature-complete (e.g. attributes are not yet fully digested).* Text file import uses fread from package data.table::data.table, netCDF import extracts data and attributes using functions from package ncdf4. Date-time representations in data files are converted to POSIX time representations. Monthly and annual time steps are returned as first day of the time step period.

Import from netCDF files requires an id dimension in the netCDF data. Gridded data with remapped HYPE results in spatial x/y dimensions as defined in the HYPE netCDF formatting standard are currently not supported.

Value

ReadTimeOutput returns a data.frame, data.table::data.table, or a HypeSingleVar array. Data frames and data tables contain additional attributes: variable, giving the HYPE variable ID, subid and outregid, the HYPE SUBIDs/OUTREGIDs (corresponding to columns from column two onward) to which the time series belong (both attributes always created and assigned NA if not applicable to data contents), timestep with a time step attribute, and comment with first row comment of imported text file as character string or global attributes of imported netCDF file as character string of collated key-value pairs. An additional attribute id.nan might be returned, see argument warn.nan.

Note

For the conversion of date/time strings, time zone "UTC" is assumed. This is done to avoid potential daylight saving time side effects when working with the imported data (and possibly converting to string representations during the process).

HYPE results are printed to files using a user-specified accuracy. This accuracy is specified in 'info.txt' as a number of decimals to print. If large numbers are printed, this can result in a total number of digits which is too large to print. Results will then contain values of '**********'. ReadTimeOutput will convert those cases to 'NA' entries. Current versions of HYPE allow for defining significant instead of fixed number of digits, which should prevent this issue from arising.

Examples

```
te <- ReadTimeOutput(filename = system.file("demo_model",
"results", "timeCOUT.txt", package = "HYPEtools"), dt.format = "%Y-%m")
te
```

`ReadWsOutput` *Read optimization simulation results*

Description

Read and combine HYPE optimization simulation output files, generated with 'task WS' during HYPE optimization runs. Outputs can consist of basin, time, or map output files.

Usage

```
ReadWsOutput(
  path,
  type = c("time", "map", "basin"),
  hype.var = NULL,
  id = NULL,
  dt.format = NULL,
  select = NULL,
  from = NULL,
  to = NULL,
  progbar = TRUE,
  warn.nan = FALSE
)
```

Arguments

`path`	Character string, path to the directory holding simulation output files to import. Windows users: Note that Paths are separated by '/', not '\'.
`type`	Character string, keyword for HYPE output file type to import. One of `"time"`, `"map"`, or `"basin"`. Can be abbreviated. The first two require specification of argument `hype.var`, the latter of argument `subid`. Format of return value depends on output type, see details.
`hype.var`	Character string, keyword to specify HYPE output variable to import. Must include "RG"-prefix in case of output region files. Not case-sensitive. Required in combination with `type` `"time"` or `"map"`.
`id`	Integer, giving a single SUBID or OUTREGID for which to import basin output files. Required in combination with `type` `"basin"`.
`dt.format`	Date-time `format` string as in `strptime`, for conversion of date-time information in imported result files to POSIX dates, which are returned as attribute. Incomplete format strings for monthly and annual values allowed, e.g. '\ summary values over the model period.

`select`	Integer vector, column numbers to import, for use with `type = "time"`. Note: first column with dates must be imported.
`from`	Integer. For partial imports, number of simulation iteration to start from.
`to`	Integer. For partial imports, number of simulation iteration to end with.
`progbar`	Logical, display a progress bar while importing HYPE output files. Adds overhead to calculation time but useful when many files are imported.
`warn.nan`	Logical, check if imported results contain any `NaN` values. If `TRUE` and `NaN`s are found, a warning is thrown and affected SUBIDs and iterations are saved in an attribute `subid.nan`. Adds noticeable overhead to import time for large simulation file sets.

Details

HYPE optimization routines optionally allow for generation of simulation output files for each iteration in the optimization routine. For further details see documentation on 'task WS' in the optpar.txt online documentation.

`ReadWsOutput` imports and combines all simulation iterations in an `array`, which can then be easily used in further analysis, most likely in combination with performance and parameter values from an imported corresponding 'allsim.txt' file.

The result folder containing HYPE WS results, argument `path`, can contain other files as well, `ReadWsOutput` searches for file name pattern to filter targeted result files. However, if files of the same type exist from different model runs, e.g. from another calibration run or from a standard model run, the pattern search cannot distinguish these from the targeted files and `ReadWsOutput` will fail.

For large numbers of result files, simulations can be partially imported using arguments `from` and `to`, in order to avoid memory exceedance problems.

Value

`ReadWsOutput` returns a 3-dimensional array with additional attributes. The array content depends on the HYPE output file type specified in argument `type`. Time and map output file imports return an array of class `HypeSingleVar` with `[time, subid, iteration]` dimensions, basin output file imports return an array of class `HypeMultiVar` with `[time, variable, iteration]` dimensions. An additional attribute `subid.nan` might be returned, see argument `warn.nan`, containing a list with SUBID vector elements. Vectors contain iterations where `NaN` values occur for the given SUBID.

Returned arrays contain additional `attributes`:

date A vector of date-times, `POSIX` if argument `dt.format` is non-`NULL`. Corresponds to 1st array dimension.

subid A (vector of) SUBID(s). Corresponds to 2nd array dimension for time and map output files. `NA` if not applicable.

outregid A (vector of) OUTREGID(s). Corresponds to 2nd array dimension for time and map output files. `NA` if not applicable.

variable A vector of HYPE output variables. Corresponds to 2nd array dimension for basin output files.

nan (optional) A named list with SUBID or HYPE variable vector elements. Vectors contain iterations where `NaN` values occur for the given SUBID/HYPE variable.

Examples

```
te <- ReadWsOutput(path = system.file("demo_model",
"results", package = "HYPEtools"), type = "map",
hype.var = "cout", dt.format = "%Y-%m")
te
```

ReadXobs *Read an 'Xobs.txt' file*

Description

This is a convenience wrapper function to import an Xobs file into R.

Usage

```
ReadXobs(
  filename = "Xobs.txt",
  dt.format = NULL,
  variable = NULL,
  nrows = -1L,
  verbose = if (nrows %in% 0:2) FALSE else TRUE
)
```

Arguments

filename	Path to and file name of the Xobs file to import. Windows users: Note that Paths are separated by '/', not '\'.
dt.format	Date-time format string as in strptime.
variable	Character vector, HYPE variable ID(s) to select for import. Not case-sensitive. If NULL (default), all variables are imported. See Xobs.txt documentation for a list of variable IDs.
nrows	Integer, number of rows to import. A value of -1 indicates all rows, a positive integer gives the number of rows to import.
verbose	Logical, throw warning if class HypeXobs's attribute timestep cannot be computed.

Details

ReadXobs is a convenience wrapper function of fread from package data.table::data.table, with conversion of date-time strings to POSIX time representations. Variable names, SUBIDs, comment, and timestep are returned as attributes (see attr on how to access these).

Duplicated variable-SUBID combinations are not allowed in HYPE Xobs files, and the function will throw a warning if any are found.

Value

If datetime import to POSIXct worked, ReadXobs returns a HypeXobs object, a data frame with four additional attributes variable, subid, comment, and timestep: variable and subid each contain a vector with column-wise HYPE IDs (first column with date/time information omitted). comment contains the content of the Xobs file comment row as single string. timestep contains a keyword string. Column names of the returned data frame are composed of variable names and SUBIDs, separated by an underscore, i.e. [variable]_[subid]. If datetime conversion failed on import, the returned object is a data frame (i.e. no class HypeXobs).

Note

For the conversion of date/time strings, time zone "UTC" is assumed. This is done to avoid potential daylight saving time side effects when working with the imported data (and e.g. converting to string representations during the process).

Examples

```
te <- ReadXobs(filename = system.file("demo_model", "Xobs.txt", package = "HYPEtools"))
te
```

RescaleSLCClasses *Re-scale SLC classes in a GeoData data frame*

Description

RescaleSLCClasses re-scales several or all SLC classes for each SUBID in a GeoData data frame to a new target sum for all classes.

Usage

```
RescaleSLCClasses(gd, slc.exclude = NULL, target = 1, plot.box = TRUE)
```

Arguments

gd	A data frame containing columns 'SLC_n' ($n \geq 1$), typically an imported 'GeoData.txt' file.
slc.exclude	Integer, SLC class numbers. Area fractions of classes listed here are kept fixed during re-scaling. If NULL (default), all classes are re-scaled.
target	Numeric, target sum for SLC class fractions in each subbasin after re-scaling. Either a single number or a vector with one value for each row in gd.
plot.box	Logical, if TRUE, a box plot of SLC area sums is returned.

Details

RescaleSLCClasses allows to rescale SLC classes, e.g. as part of a post-processing work flow during HYPE model setup. Individual SLC classes can be excluded to protect. This can be useful e.g. for lake areas which maybe must correspond to areas a LakeData file. The function will throw a warning if excluded SLC class fractions are greater than sums provided in target, but not if they are smaller.

Value

RescaleSLCClasses returns the data frame provided in gd, with re-scaled SLC class fractions.

See Also

SumSLCClasses for inspection of SLC class fraction sums in each subbasin CleanSLCClasses for pruning of small SLC fractions.

Examples

```
# Import source data
te <- ReadGeoData(filename = system.file("demo_model", "GeoData.txt", package = "HYPEtools"))
# Re-scale SLC classes, protect the first two
RescaleSLCClasses(gd = te, slc.exclude = 1:2)
```

RunHYPE *Run HYPE model*

Description

RunHYPE is a wrapper to run a HYPE model using a specified HYPE executable.

Usage

```
RunHYPE(hype.path = NULL, info.dir = NULL, sequence = 0, p = FALSE, ...)
```

Arguments

hype.path	Path to HYPE .exe executable (Windows) or HYPE file on Linux
info.dir	Optional path to a separate directory containing the HYPE model setup (info.txt, GeoData.txt, etc.). If not specified, then the directory of hype.path will be used.
sequence	Optional integer between 0 and 999 specifying which forcing files to use.
p	Optional logical. If TRUE, then the sequence number will also be applied to the parameter file. Ignored if FALSE (default).
...	Additional arguments passed on to system.

Details

RunHYPE is a system wrapper to run a HYPE model via the system console. HYPE executables can be downloaded from the official code repository. The path to the HYPE executable must be specified with hype.path. If your HYPE model is saved to a different directory than the HYPE executable (hype.path), then you can specify the model directory with info.dir. The sequence argument can be used to specify which forcing files should be used for the simulation, and p can be used to specify if the sequence number should also be applied to the parameter file. Read more about how to run HYPE on the HYPE wiki.

Value

Output messageges from the HYPE executable are printed to the console.

Examples

```
## Not run:
RunHYPE()

## End(Not run)
```

ScaleAquiferData *Scale 'AquiferData.txt' files to different model time steps*

Description

ScaleAquiferData scales the RETRATE time step-dependent recession coefficient in an imported HYPE 'AquiferData.txt' file to a new target time step. See HYPE wiki tutorial on sub-daily time steps.

Usage

```
ScaleAquiferData(
  x = NULL,
  timestep.ratio = 1/24,
  digits = 3,
  verbose = TRUE,
  print.par = FALSE
)
```

Arguments

x	Data frame containing HYPE AquiferData contents. Typically imported with ReadAquiferData().
timestep.ratio	Numeric, time step scaling factor. Defaults to (1/24) to scale from daily to hourly time steps. To scale from hourly to daily time steps use 24.

digits	Integer, number of significant digits in scaled parameter values to export. See signif().
verbose	Logical, if TRUE, then information will be printed.
print.par	Logical, print known time-scale dependent recession coefficients instead of scaling an AquiferData data frame.

Details

ScaleAquiferData applies a user-specified scaling factor timestep.ratio to the RETRATE time step-dependent recession coefficient in a HYPE AquiferData data frame. All RETRATE values that are not equal to zero are assumed to be aquifer rows and will be converted to the new time step. Recession coefficients are matched against an inbuilt set of column names. To see these names, call ScaleAquiferData(print.par = TRUE). *Please notify us if you find any missing coefficients.*

Timestep-dependent recession coefficients are scaled using the relationship described in: Nalbantis, Ioannis (1995). "Use of multiple-time-step information in rainfall-runoff modelling", Journal of Hydrology 165, 1-4, pp. 135–159.

```
new_coefficient_value = 1 - (1 - old_coefficient_value)^time_step_ratio
```

Use the ScalePar and ScaleFloodData functions to scale the time-dependent parameters and recession coefficients in par.txt and FloodData.txt files, respectively. Note that ScalePar does not scale the values for the "gratk", "ilratk", "olratk", or "wetrate" rating curve recession coefficients in par.txt because they are not limited to the range 0-1. Likewise, HYPEtools does not provide any scaling function for the "RATE" columns in DamData.txt and LakeData.txt because these values are not limited to the range 0-1. We recommend looking at the results from the lakes/wetlands and recalibrating these parameters and their related power coefficients as needed.

Value

A data.frame() object as supplied in x, with re-scaled recession coefficients, or nothing if print.par = TRUE.

See Also

ScaleFloodData ScalePar

Examples

```
# Import daily HYPE AquiferData file
ad <- ReadAquiferData(filename = system.file("demo_model",
"AquiferData_Example.txt", package = "HYPEtools"))
# Scale to hourly time steps
ScaleAquiferData(x = ad)
# Print all time scale-dependent coefficients known to the function
ScaleAquiferData(print.par = TRUE)
```

ScaleFloodData *Scale 'FloodData.txt' files to different model time steps*

Description

ScaleFloodData scales the time step-dependent recession coefficients in an imported HYPE 'FloodData.txt' file to a new target time step. See HYPE wiki tutorial on sub-daily time steps.

Usage

```
ScaleFloodData(
  x = NULL,
  timestep.ratio = 1/24,
  digits = 3,
  verbose = TRUE,
  print.par = FALSE
)
```

Arguments

x	Data frame containing HYPE FloodData contents. Typically imported with ReadFloodData().
timestep.ratio	Numeric, time step scaling factor. Defaults to (1/24) to scale from daily to hourly time steps. To scale from hourly to daily time steps use 24.
digits	Integer, number of significant digits in scaled parameter values to export. See signif().
verbose	Logical, if TRUE, then information will be printed.
print.par	Logical, print known time-scale dependent recession coefficients instead of scaling a FloodData data frame.

Details

ScaleFloodData applies a user-specified scaling factor timestep.ratio to the time step-dependent recession coefficients in a HYPE FloodData data frame. ' Recession coefficients are matched against an inbuilt set of column names. To see these names, call ScaleFloodData(print.par = TRUE). *Please notify us if you find any missing coefficients.*

Timestep-dependent recession coefficients are scaled using the relationship described in: Nalbantis, Ioannis (1995). "Use of multiple-time-step information in rainfall-runoff modelling", Journal of Hydrology 165, 1-4, pp. 135–159.

new_coefficient_value = 1 - (1 - old_coefficient_value)^time_step_ratio

Use the ScalePar and ScaleAquiferData functions to scale the time-dependent parameters and recession coefficients in par.txt and AquiferData.txt files, respectively. Note that ScalePar does not scale the values for the "gratk", "ilratk", "olratk", or "wetrate" rating curve recession coefficients in par.txt because they are not limited to the range 0-1. Likewise, HYPEtools does not provide any scaling function for the "RATE" columns in DamData.txt and LakeData.txt because these values

are not limited to the range 0-1. We recommend looking at the results from the lakes/wetlands and recalibrating these parameters and their related power coefficients as needed.

Value

A `data.frame()` object as supplied in `x`, with re-scaled recession coefficients, or nothing if `print.par = TRUE`.

See Also

`ScaleAquiferData` `ScalePar`

Examples

```
# Import daily HYPE FloodData file
fd <- ReadFloodData(filename = system.file("demo_model",
"FloodData_Example.txt", package = "HYPEtools"))
# Scale to hourly time steps
ScaleFloodData(x = fd)
# Print all time scale-dependent coefficients known to the function
ScaleFloodData(print.par = TRUE)
```

ScalePar *Scale 'par.txt' files to different model time step*

Description

`ScalePar` scales time step-dependent parameters and recession coefficients in an imported HYPE 'par.txt' parameter file to a new target time step. See HYPE wiki tutorial on sub-daily time steps.

Usage

```
ScalePar(
  x = NULL,
  timestep.ratio = 1/24,
  digits = 3,
  verbose = TRUE,
  print.par = FALSE
)
```

Arguments

`x`	List containing HYPE parameters. Typically imported with `ReadPar()`.
`timestep.ratio`	Numeric, time step scaling factor. Defaults to (1/24) to scale from daily to hourly time steps. To scale from hourly to daily time steps use 24.
`digits`	Integer, number of significant digits in scaled parameter values to export. See `signif()`.

verbose	Logical, if TRUE, then information will be printed.
print.par	Logical, print known time-scale dependent parameters and recession coefficients instead of scaling a parameter list.

Details

ScalePar applies a user-specified scaling factor, timestep.ratio, to all time scale-dependent parameters and recession coefficients in a HYPE parameter list. Parameters are matched against an inbuilt set of parameter names. To see these parameters, call ScalePar(print.par = TRUE). *Please notify us if you find any missing parameters.*

If parameters are not timestep-dependent recession coefficients, then scaling is performed using the ratio between the two time step lengths (e.g. 1/24 when scaling from daily to hourly time steps). If parameters are timestep-dependent recession coefficients, then scaling is performed using the relationship described in: Nalbantis, Ioannis (1995). "Use of multiple-time-step information in rainfall-runoff modelling", Journal of Hydrology 165, 1-4, pp. 135–159.

new_parameter_value = 1 - (1 - old_parameter_value)^time_step_ratio

ScalePar does not scale the values for the "gratk", "ilratk", "olratk", or "wetrate" rating curve recession coefficients in par.txt because they are not limited to the range 0-1. Likewise, HYPEtools does not provide any scaling function for the "RATE" columns in DamData.txt and LakeData.txt because these values are not limited to the range 0-1. We recommend looking at the results from the lakes/wetlands and recalibrating these parameters and their related power coefficients as needed.

Use the ScaleAquiferData and ScaleFloodData functions to scale the time-dependent recession coefficients in AquiferData.txt and FloodData.txt files, respectively.

Value

A list() object as supplied in x, with re-scaled parameters and recession coefficients, or nothing if print.par = TRUE.

See Also

ScaleAquiferData ScaleFloodData

Examples

```
# Import daily HYPE parameter file
par <- ReadPar(filename = system.file("demo_model", "par.txt", package = "HYPEtools"))
# Scale to hourly time steps
ScalePar(x = par)
# Print all time scale-dependent parameters known to the function
ScalePar(print.par = TRUE)
```

SimToPar *HYPE Calibration Outputs to par.txt*

Description

Update par.txt with values from an allsim.txt or bestsims.txt file

Usage

```
AllSimToPar(simfile, row, par)

BestSimsToPar(simfile, row, par)
```

Arguments

simfile	Imported allsim.txt or bestsims.txt file imported as data frame.
row	Integer, row number indicating row containing the parameter values that should be replaced/added to par.
par	Imported par.txt file that should be updated using parameter values from simfile. Typically imported using ReadPar.

Details

AllSimToPar and BestSimsToPar can be used to update an existing par.txt file with the parameter values from a HYPE allsim.txt or bestsims.txt file. If a parameter in the allsim or bestsims file already exists in par, then the parameter values will be overwritten in par. If the parameter does not exist, then the parameter will be added to the bottom of the output.

Value

AllSimToPar and BestSimsToPar return a list of named vectors in the format used by ReadPar.

See Also

ReadPar for HYPE par.txt import; WritePar to export HYPE par.txt files

Examples

```
simfile <- read.table(file = system.file("demo_model", "results",
  "bestsims.txt",
  package = "HYPEtools"
), header = TRUE, sep = ",")
par <- ReadPar(filename = system.file("demo_model", "par.txt", package = "HYPEtools"))
BestSimsToPar(simfile, 1, par)
```

SortGeoData *Sort a GeoData dataframe in downstream order*

Description

Function to sort an imported GeoData.txt file in downstream order, so that all upstream sub-basins are listed in rows above downstream sub-basins.

Usage

```
SortGeoData(gd, bd = NULL, progbar = TRUE)
```

Arguments

gd	A data frame containing a column with SUBIDs and a column (MAINDOWN) containing the corresponding downstream SUBID, e.g. an imported 'GeoData.txt' file.
bd	A data frame with bifurcation connections, e.g. an imported 'BranchData.txt' file. Optional argument.
progbar	Logical, display a progress bar while calculating SUBID sorting.

Details

GeoData.txt files need to be sorted in downstream order for HYPE to run without errors. SortGeoData considers bifurcation connections, but not irrigation or groundwater flow links.

Value

SortGeoData returns a GeoData dataframe.

See Also

AllUpstreamSubids OutletSubids

Examples

```
te <- ReadGeoData(filename = system.file("demo_model", "GeoData.txt", package = "HYPEtools"))
SortGeoData(gd = te)
```

SubidAttributeSummary *Summarize subbasin attributes*

Description

Prepare data frame containing summary of subbasin attributes.

Usage

```
SubidAttributeSummary(
  subids = NULL,
  gd,
  bd = NULL,
  gc = NULL,
  desc = NULL,
  group = NULL,
  group.upstream = TRUE,
  signif.digits = NULL,
  progbar = FALSE,
  summarize.landuse = TRUE,
  summarize.soil = TRUE,
  summarize.crop = TRUE,
  summarize.upstreamarea = TRUE,
  unweighted.gd.cols = NULL,
  upstream.gd.cols = NULL,
  olake.slc = NULL,
  bd.weight = FALSE,
  mapoutputs = NULL
)
```

Arguments

subids	Vector containing SUBIDs of subbasins to summarize.
gd	Imported HYPE GeoData.txt file. See ReadGeoData.
bd	Imported HYPE BranchData.txt file. See ReadBranchData.
gc	Imported HYPE GeoClass.txt file. See ReadGeoClass.
desc	Optional, Imported HYPE Description file. If provided, then dataframe columns will be renamed using the short names in the description file. See ReadDescription.
group	Optional, Integer vector of same length as number of SLC classes in gd. Alternative grouping index specification to gcl + type for UpstreamGroupSLCClasses.
group.upstream	Logical, if TRUE, then SLC fractions will be summarized for upstream areas using UpstreamGroupSLCClasses. If FALSE, then SLC fractions will be summarized for subbasin area only using GroupSLCClasses.
signif.digits	Optional, Integer specifying number of significant digits to round outputs to. Used by UpstreamGroupSLCClasses and UpstreamGeoData.

progbar Logical, display a progress bar while calculating summary information. Used by UpstreamGroupSLCClasses and UpstreamGeoData.

summarize.landuse
Logical, specify whether or not subbasin upstream landuse fractions should be calculated.

summarize.soil Logical, specify whether or not subbasin upstream soil fractions should be calculated.

summarize.crop Logical, specify whether or not subbasin upstream crop fractions should be calculated.

summarize.upstreamarea
Logical, specify whether or not subbasin upstream area should be calculated.

unweighted.gd.cols
Vector, names of gd columns which should be joined to the output data frame without any additional processing.

upstream.gd.cols
Vector, specify column names of gd which should be summarized using UpstreamGeoData.

olake.slc Integer, SLC class number representing outlet lake fractions. Used by UpstreamGeoData.

bd.weight Logical, if set to TRUE, flow weights will be applied for areas upstream of stream bifurcations. See UpstreamGeoData.

mapoutputs Vector, paths to mapoutput files that should be read by ReadMapOutput and joined to the output data frame.

Details

SubidAttributeSummary can be used to create a data frame object containing subbasin attribute summary information. This data frame can then be used as the attributes input for PlotPerformanceByAttribute. The function can summarize subbasin upstream landuse, soil, and crop fractions using UpstreamGroupSLCClasses. In addition, the function can summarize upstream GeoData information using UpstreamGeoData. Finally, the function can join mapoutput and GeoData columns directly to the output data frame (i.e without further processing).

Value

SubidAttributeSummary returns a data frame object containing subbasin attribute summary information.

See Also

UpstreamGroupSLCClasses, GroupSLCClasses, UpstreamGeoData, ReadMapOutput for subbasin attribute summary functions; PlotPerformanceByAttribute for related plotting function.

Examples

```
subass <- ReadSubass(filename = system.file("demo_model", "results",
  "subass1.txt",
  package = "HYPEtools"
), check.names = TRUE)
gd <- ReadGeoData(filename = system.file("demo_model",
```

```
  "GeoData.txt",
  package = "HYPEtools"
))
gc <- ReadGeoClass(filename = system.file("demo_model",
  "GeoClass.txt",
  package = "HYPEtools"
))

SubidAttributeSummary(subids <- subass$SUBID,
  gd = gd, gc = gc,
 mapoutputs = c(system.file("demo_model", "results", "mapCOUT.txt", package = "HYPEtools")),
  upstream.gd.cols = c("SLOPE_MEAN")
)
```

`SumSLCClasses` *Calculate sums of SLC classes in a GeoData file*

Description

`SumSLCClasses` sums all SLC classes for each SUBID in a GeoData data frame and optionally plots the results.

Usage

```
SumSLCClasses(gd, plot.box = TRUE, silent = FALSE, ...)
```

Arguments

`gd`	Data frame containing columns with SLC fractions, typically a 'GeoData.txt' file imported with `ReadGeoData`.
`plot.box`	Logical, if `TRUE`, a box plot of SLC area sums is returned.
`silent`	Logical, if set to `TRUE`, the default printing of a result summary is suppressed.
`...`	Other arguments to be passed to `boxplot`.

Details

`SumSLCClasses` is a wrapper for `colSums` with a boxplot output option, and allows to quickly control if SLCs of all SUBIDs in a GeoData data frame sum up to 1.

Value

`SumSLCClasses` returns a vector of SLC sums, invisibly if `plot.box` is `TRUE`.

Examples

```
te <- ReadGeoData(filename = system.file("demo_model", "GeoData.txt", package = "HYPEtools"))
SumSLCClasses(gd = te, plot.box = TRUE)
SumSLCClasses(gd = te, plot.box = FALSE)
```

`SumUpstreamArea` *Calculate upstream area sums*

Description

Function to calculate upstream areas of a vector of SUBIDs or all SUBIDs in a GeoData table.

Usage

```
SumUpstreamArea(subid = NULL, gd, bd = NULL, cl = 2, progbar = FALSE)
```

Arguments

`subid`	Integer vector of SUBIDs to calculate upstream areas for (must exist in `gd`). If `NULL`, upstream areas for all SUBIDs will be calculated.
`gd`	A data frame, containing 'SUBID', 'MAINDOWN', and 'AREA' columns, e.g. an imported 'GeoData.txt' file.
`bd`	A data frame, containing 'BRANCHID' and 'SOURCEID' columns, e.g. an imported 'BranchData.txt' file. Optional argument.
`cl`	Integer, number of processes to use for parallel computation. Set to `1` for serial computation. See `parallel::detectCores()`.
`progbar`	Logical, display a progress bar while calculating upstream areas. Adds overhead to calculation time but useful if you want HYPEtools to decide how long your coffee break should take.

Details

`SumUpstreamArea` sums upstream areas of all connected upstream SUBIDs, including branch connections in case of stream bifurcations but not including potential irrigation links or groundwater flows.

Value

`SumUpstreamArea` returns a data frame with two columns containing SUBIDs and total upstream areas (in area units as provided in `gd`). Upstream areas include areas of outlet SUBIDs.

See Also

`AllUpstreamSubids`

Examples

```
te <- ReadGeoData(filename = system.file("demo_model", "GeoData.txt", package = "HYPEtools"))
SumUpstreamArea(subid = c(3361, 63794), gd = te, progbar = FALSE)
```

`UpstreamGeoData` *Calculate upstream sums and averages of selected GeoData contents*

Description

Function to calculate upstream sums and averages for selected variables of imported GeoData.txt files.

Usage

```
UpstreamGeoData(
  subid = NULL,
  gd,
  bd = NULL,
  olake.slc = NULL,
  bd.weight = FALSE,
  signif.digits = 5,
  progbar = TRUE
)
```

Arguments

`subid`	Integer vector of SUBIDs for which to calculate upstream properties (must exist in `gd`). If `NULL` (default), upstream areas for all SUBIDs will be calculated.
`gd`	A data frame containing a column with SUBIDs and a column with areas, e.g. an imported 'GeoData.txt' file.
`bd`	A data frame with bifurcation connections, e.g. an imported 'BranchData.txt' file. Optional argument.
`olake.slc`	Integer,SLC class number which represents outlet lake fractions. Mandatory for weighted averaging of outlet lake depths.
`bd.weight`	Logical, if set to `TRUE`, flow weights will be applied for areas upstream of stream bifurcations. See `AllUpstreamSubids` for further details on flow fraction computation.
`signif.digits`	Integer, number of significant digits to round upstream variables to. See also `signif`. Set to `NULL` to prevent rounding.
`progbar`	Logical, display a progress bar while calculating SLC class fractions. Adds overhead to calculation time but useful when `subid` is `NULL` or contains many SUBIDs.

Details

UpstreamGeoData calculates upstream averages or sums of selected variables in a GeoData data frame, including branch connections in case of stream bifurcations but not including potential irrigation links or groundwater flows. Averages are weighted by sub-catchment area, with the exception of outlet lake depths and rural household emission concentrations provided in GeoData variables 'lake_depth', 'loc_tn', and 'loc_tp'. Outlet lake depths are weighted by outlet lake area and the GeoData column with SLC class fractions for outlet lakes must be provided in function argument col.olake.slc. Rural household emissions are weighted by emission volume as provided in column 'loc_vol'. Elevation and slope standard deviations are averaged if the corresponding mean values exist (sample means are required to calculate overall means of standard deviations).

Currently, the following variables are considered:

Area-weighted average elev_mean, slope_mean, buffer, close_w, latitude, longitude, all SLC classes, lake depths, elev_std, slope_std

Volume-weighted average loc_tn, loc_tp

Sum area, rivlen, loc_vol

Value

UpstreamGeoData returns a data frame with the same number of columns as argument gd and number of rows corresponding to number of SUBIDs in argument subid, with updated upstream columns marked with a leading 'UP_' in the column names.

See Also

UpstreamSLCClasses SumUpstreamArea AllUpstreamSubids

Examples

```
te <- ReadGeoData(filename = system.file("demo_model", "GeoData.txt", package = "HYPEtools"))
# Upstream stats for domain outlet
UpstreamGeoData(subid = OutletSubids(te), gd = te, olake.slc = 1, progbar = FALSE)
```

UpstreamGroupSLCClasses

Calculate area-weighted upstream averages of grouped SLC class fractions.

Description

Function to calculate averages of grouped SLC class fractions calculated from imported GeoData.txt and GeoClass.txt or any other user-defined grouping.

Usage

```
UpstreamGroupSLCClasses(
  subid = NULL,
  gd,
  bd = NULL,
  gcl = NULL,
  type = c("landuse", "soil", "crop"),
  group = NULL,
  signif.digits = 3,
  progbar = TRUE
)
```

Arguments

subid	Integer vector of SUBIDs for which to calculate upstream properties (must exist in gd). If NULL (default), upstream areas for all SUBIDs will be calculated.
gd	A data frame containing a column with SUBIDs and a column with areas, e.g. an imported 'GeoData.txt' file imported with ReadGeoData.
bd	A data frame, containing 'BRANCHID' and 'SOURCEID' columns, e.g. an imported 'BranchData.txt' file. Optional argument.
gcl	Data frame containing columns with SLCs and corresponding land use and soil class IDs, typically a 'GeoClass.txt' file imported with ReadGeoClass. Must be provided if no group argument is given.
type	Keyword character string for use with gcl. Type of grouping index, choice of "landuse", "soil", and/or "crop", can be abbreviated.
group	Integer vector, of same length as number of SLC classes in gd. Alternative grouping index specification to gcl + type.
signif.digits	Integer, number of significant digits to round upstream SLCs to. See also signif. Set to NULL to prevent rounding.
progbar	Logical, display a progress bar while calculating SLC class fractions. Adds overhead to calculation time but useful when subid is NULL or contains many SUBIDs.

Details

UpstreamGroupSLCClasses calculates area-weighted upstream averages of CropID fractions from SLC class fractions in a GeoData table and corresponding grouping columns in a GeoClass table or a user-provided vector. Upstream calculations include branch connections in case of stream bifurcations but not potential irrigation links or groundwater flows. Averages are weighted by sub-catchment area.

The function builds on GroupSLCClasses, which provides grouped sums of SLC classes for several or all sub-basins in a GeoData dataframe.

Value

UpstreamGroupSLCClasses returns a data frame with SUBIDs in the first column, and upstream group fractions in the following columns.

Note

UpstreamGroupSLCClasses expects SLC class columns in argument gd to be ordered in ascending order.

See Also

GroupSLCClasses UpstreamSLCClasses UpstreamGeoData SumUpstreamArea AllUpstreamSubids

Examples

```
# Import source data
te1 <- ReadGeoData(filename = system.file("demo_model", "GeoData.txt", package = "HYPEtools"))
te2 <- ReadGeoClass(filename = system.file("demo_model", "GeoClass.txt", package = "HYPEtools"))
# Upstream land use fractions for single SUBID
UpstreamGroupSLCClasses(subid = 63794, gd = te1, gcl = te2, type = "landuse", progbar = FALSE)
# Upstream soil fraction for all SUBIDs in GeoData
UpstreamGroupSLCClasses(gd = te1, gcl = te2, type = "soil")
```

UpstreamPointSources *Summarize point source emissions of all upstream areas*

Description

Function to calculate point source emissions over all upstream areas of a vector of SUBIDs or all SUBIDs in a GeoData table.

Usage

```
UpstreamPointSources(
  subid = NULL,
  gd,
  psd,
  bd = NULL,
  signif.digits = 4,
  progbar = TRUE
)
```

Arguments

subid	Integer vector of SUBIDs to calculate upstream point sources for (must exist in gd). If NULL, upstream point sources for all SUBIDs in 'gd' will be calculated.
gd	A data frame containing columns 'SUBID' with SUBIDs and 'MAINDOWN' with downstream SUBIDs, e.g. an imported 'GeoData.txt' file.
psd	A data frame with HYPE point source specifications, typically a 'PointSourceData.txt' file imported with ReadPointSourceData.

bd	A data frame, containing 'BRANCHID' and 'SOURCEID' columns, e.g. an imported 'BranchData.txt' file. Optional argument.
signif.digits	Integer, number of significant digits to round upstream SLCs to. See also signif. Set to NULL to prevent rounding.
progbar	Logical, display a progress bar while calculating SLC class fractions. Adds overhead to calculation time but useful when subid is NULL or contains many SUBIDs.

Details

UpstreamPointSources calculates summarized upstream point source emissions. For each sub-basin with at least one upstream point source (including the sub-basin itself), summed emission volumes and volume weighted emission concentrations are calculated. HYPE point source types ('ps_type') are returned in separate rows. UpstreamPointSources requires point source types to be one of -1, 0, 1, 2, 3, corresponding to water abstractions, no differentiation/tracer, and type 1 to 3 (e.g. wastewater treatment plants, industries, and urban stormwater). For water abstraction point sources, only summed upstream volumes are returned, i.e., concentrations are simply set to zero in results.

Value

UpstreamPointSources returns a data frame with columns containing SUBIDs, point source types, volumes, and concentrations found in psd: total nitrogen, total phosphorus, total suspended sediment, tracer, and temperature.

Examples

```
te1 <- ReadPointSourceData(filename = system.file("demo_model",
"PointSourceData.txt", package = "HYPEtools"))
te2 <- ReadGeoData(filename = system.file("demo_model",
"GeoData.txt", package = "HYPEtools"))
UpstreamPointSources(subid = OutletSubids(te2), gd = te2,
psd = te1, progbar = FALSE)
```

UpstreamSLCClasses *Calculate SLC class fractions of all upstream areas*

Description

Function to calculate SLC class fractions over all upstream areas of a vector of SUBIDs or all SUBIDs in a GeoData table.

Usage

```
UpstreamSLCClasses(
  subid = NULL,
  gd,
  bd = NULL,
  signif.digits = 3,
  progbar = TRUE
)
```

Arguments

subid	Integer vector of SUBIDs to calculate upstream SUBID fractions for (must exist in gd). If NULL, upstream areas for all SUBIDs will be calculated.
gd	A data frame containing columns 'SUBID' with SUBIDs, 'MAINDOWN' with downstream SUBIDs, and 'AREA' with sub-basin areas, e.g. an imported 'GeoData.txt' file.
bd	A data frame with bifurcation connections, e.g. an imported 'BranchData.txt' file. Optional argument.
signif.digits	Integer, number of significant digits to round upstream SLCs to. See also signif. Set to NULL to prevent rounding.
progbar	Logical, display a progress bar while calculating SLC class fractions. Adds overhead to calculation time but useful when subid is NULL or contains many SUBIDs.

Details

UpstreamSLCClasses sums upstream areas of all connected upstream SUBIDs, including branch connections in case of stream bifurcations but not including potential irrigation links or groundwater flows.

Value

UpstreamSLCClasses returns a data frame with columns containing SUBIDs, total upstream areas (in area unit as provided in gd), and SLC class fractions for upstream areas.

Note

This function is now superseded by UpstreamGeoData, which returns more upstream variables.

See Also

SumUpstreamArea, UpstreamGeoData, UpstreamGroupSLCClasses

Examples

```
# Import source data
te1 <- ReadGeoData(filename = system.file("demo_model", "GeoData.txt", package = "HYPEtools"))
# Upstream SLCs for single SUBID
```

```
UpstreamSLCClasses(subid = 3361, gd = te1, progbar = FALSE)
```

`VariableLookup` *Lookup Functions For HYPE Variables*

Description

Lookup information (e.g. Name, Units) for a specific HYPE variable ID, or find HYPE variable information for a search term.

Usage

```
VariableInfo(
  variable,
 info = c("ID", "Name", "Unit", "Description", "Aggregation", "Reference", "Component")
)

VariableSearch(
  search,
 info = c("ID", "Name", "Unit", "Description", "Aggregation", "Reference", "Component"),
  ignore_case = TRUE
)
```

Arguments

`variable`	String, HYPE Variable ID (e.g. "COUT").
`info`	A vector of strings describing HYPE variable attribute information to return/search: "ID", "Name", "Unit", "Description", "Aggregation", and/or "Component".
`search`	String, search HYPE variable info for string matches in `info` attributes.
`ignore_case`	Logical, should case differences be ignored in the match?

Details

The `VariableInfo` and `VariableSearch` functions provide features to lookup information on HYPE variables from the HYPE Wiki. `VariableInfo` can be used to return information (e.g. Name, Units) for a known HYPE Variable ID. `VariableSearch` can be used to search for e.g. an unknown HYPE variable ID based on a `search` term. The `info` argument can be used to select which information to return or search.

Value

`VariableInfo` Returns a named list of the selected `info` for the specified `variable` ID. `VariableInfo` returns a tibble of the search results.

Examples

```
VariableInfo(variable = "COUT", info = c("Name","Unit"))
VariableSearch(search = "ccSS", info = c("ID", "Name", "Description"))
```

`VisualizeMapOutput` *Shiny App for visualizing HYPE MapOutputs.*

Description

Interactive maps and plots for visualizing MapOutput files.

Usage

```
VisualizeMapOutput(
  results.dir = NULL,
  file.pattern = "^map.*\\.(txt|csv)$",
  map = NULL,
  map.subid.column = 1,
  output.dir = NULL
)

VisualiseMapOutput(
  results.dir = NULL,
  file.pattern = "^map.*\\.(txt|csv)$",
  map = NULL,
  map.subid.column = 1,
  output.dir = NULL
)
```

Arguments

`results.dir`	Optional string, path to a directory containing MapOutput files that should be loaded on app initialization.
`file.pattern`	Optional string, filename pattern to select files in `results.dir` that should be loaded on app initialization. See `list.files`.
`map`	Optional string, path to GIS file for subbasin polygons that should be loaded on app initialization. Typically a GeoPackage (.gpkg) or Shapefile (.shp).
`map.subid.column`	Optional integer, column index in the `map` 'data' `slot` holding SUBIDs (sub-catchment IDs) that should be used on app initialization.
`output.dir`	Optional string, path to a default output directory to save captured map images.

Details

VisualizeMapOutput is a Shiny app that provides interactive maps, plots, and tables for visualizing HYPE MapOutput files. The interactive Leaflet map is generated using PlotMapOutput. The app can be launched with or without the input arguments. All necessary input buttons and menus are provided within the app interface. For convenience, however, the input arguments can be provided in order to quickly launch the app with desired settings.

Value

VisualizeMapOutput returns a Shiny application object.

See Also

ReadMapOutput; PlotMapOutput

Examples

```
## Not run:
if (interactive()) {
  VisualizeMapOutput(
    results.dir = system.file("demo_model", "results", package = "HYPEtools"),
    map = system.file("demo_model", "gis", "Nytorp_map.gpkg", package = "HYPEtools"),
    map.subid.column = 25
  )
}

## End(Not run)
```

VisualizeMapPoints *Shiny App for visualizing Mapped Point Information.*

Description

Interactive maps and plots for visualizing mapped point information, e.g. HYPE MapOutput files or model performances at observation sites.

Usage

```
VisualizeMapPoints(
  results.dir = NULL,
  file.pattern = "^(map|subass).*\\.(txt|csv)$",
  sites = NULL,
  sites.subid.column = 1,
  bg = NULL,
  output.dir = NULL
)
```

```
VisualiseMapPoints(
  results.dir = NULL,
  file.pattern = "^(map|subass).*\\.(txt|csv)$",
  sites = NULL,
  sites.subid.column = 1,
  bg = NULL,
  output.dir = NULL
)
```

Arguments

results.dir	Optional string, path to a directory containing e.g. MapOutput or Subass files that should be loaded on app initialization.
file.pattern	Optional string, filename pattern to select files in results.dir that should be loaded on app initialization. See list.files.
sites	Optional string, path to GIS file for outlet points that should be loaded on app initialization. Typically a GeoPackage (.gpkg) or Shapefile (.shp).
sites.subid.column	Optional integer, column index in the map 'data' slot holding SUBIDs (sub-catchment IDs) that should be used on app initialization.
bg	Optional string, path to GIS file with polygon geometry to plot in the background. Typically an imported sub-basin vector polygon file.
output.dir	Optional string, path to a default output directory to save captured map images.

Details

VisualizeMapPoints is a Shiny app that provides interactive maps, plots, and tables for visualizing mapped point information. The interactive Leaflet map is generated using PlotMapPoints. The app can be launched with or without the input arguments. All necessary input buttons and menus are provided within the app interface. For convenience, however, the input arguments can be provided in order to quickly launch the app with desired settings.

Value

VisualizeMapPoints returns a Shiny application object.

See Also

ReadMapOutput; PlotMapPoints

Examples

```
## Not run:
if (interactive()) {
  VisualizeMapPoints(
    results.dir = system.file("demo_model", "results", package = "HYPEtools"),
   sites = system.file("demo_model", "gis", "Nytorp_centroids.gpkg", package = "HYPEtools"),
    sites.subid.column = 25,
    bg = system.file("demo_model", "gis", "Nytorp_map.gpkg", package = "HYPEtools")
```

```
  )
}

## End(Not run)
```

`WriteBasinOutput` *Write a basin output '[SUBID].txt' file*

Description

Function to export a basin output file from R.

Usage

```
WriteBasinOutput(x, filename, dt.format = "%Y-%m-%d")
```

Arguments

`x`	The object to be written, a dataframe with `hypeunit` attribute, as an object returned from `ReadBasinOutput`.
`filename`	A character string naming a file to write to. Windows users: Note that Paths are separated by '/', not '\'.
`dt.format`	Date-time `format` string as in `strptime`. Incomplete format strings for monthly and annual values allowed, e.g. '\%Y'.

Details

`WriteBasinOutput` exports a dataframe with headers and formatting options adjusted to match HYPE's basin output files.

Value

No return value, called for file export.

Examples

```
te <- ReadBasinOutput(filename = system.file("demo_model", "results", "0003587.txt",
                      package = "HYPEtools"))
WriteBasinOutput(x = te, filename = tempfile())
```

WriteGeoClass *Write a 'GeoClass.txt' file*

Description

This is a convenience wrapper function to export a 'GeoClass.txt' file from R.

Usage

```
WriteGeoClass(x, filename, use.comment = FALSE)
```

Arguments

x	The object to be written, a dataframe, as an object returned from ReadGeoClass.
filename	A character string naming a file to write to. Windows users: Note that Paths are separated by '/', not '\'.
use.comment	Logical, set to TRUE to export comment lines saved in attribute 'comment'. Per default, column names are exported as header. See details.

Details

WriteGeoClass exports a GeoClass dataframe. HYPE accepts comment rows with a leading '!' in the beginning rows of a GeoClass file. Comment rows typically contain some class descriptions in a non-structured way. With argument use.comment = TRUE, the export function looks for those in attribute 'comment', where ReadGeoClass stores such comments. Description files (see ReadDescription) offer a more structured way of storing that information.

Value

No return value, called for export to text files.

Examples

```
te <- ReadGeoClass(filename = system.file("demo_model", "GeoClass.txt", package = "HYPEtools"))
WriteGeoClass(x = te, filename = tempfile())
```

WriteGeoData *Write a 'GeoData.txt' file*

Description

This is a convenience wrapper function to export a 'GeoData.txt' file from R.

Usage

```
WriteGeoData(x, filename, digits = 6)
```

Arguments

x	The object to be written, a dataframe, as an object returned from ReadGeoData. NAs in any column will result in a warning (no NAs allowed in GeoData data columns).
filename	A character string naming a file to write to. Windows users: Note that Paths are separated by '/', not '\'.
digits	Integer, number of significant digits **in SLC class columns** to export. See signif.

Details

WriteGeoData exports a GeoData dataframe using fwrite. SUBID and MAINDOWN columns are forced to non-scientific notation by conversion to text strings prior to exporting. For all other numeric columns, use fwrite argument scipen. HYPE does neither allow empty values in any GeoData column nor any string elements with more than 50 characters. The function will return with warnings if NAs or long strings were exported.

Value

No return value, called for export to text files.

Examples

```
te <- ReadGeoData(filename = system.file("demo_model", "GeoData.txt", package = "HYPEtools"))
summary(te)
WriteGeoData(x = te, filename = tempfile())
```

`WriteHarmonizedData` *Write a Harmonized Data File*

Description

This is a convenience wrapper function to export a data frame to the required Harmonized Data File format. See the HYPEObsMetadataTools documentation.

Usage

```
WriteHarmonizedData(
  df,
  filename = "",
  replace.accents = FALSE,
  strip.punctuation = FALSE,
  ignore.cols = NULL,
  nThread = NULL
)
```

Arguments

`df`	Data frame containing the harmonized data.
`filename`	Path to and file name (including ".csv" file extension) of the Harmonized Data CSV file to export. Windows users: Note that Paths are separated by '/', not '\'.
`replace.accents`	Logical, if `TRUE`, then accented characters (e.g. ä, ö, å) will be replaced with non-accented characters in all strings. If `FALSE`, then strings will be left unmodified.
`strip.punctuation`	Logical, if `TRUE`, then punctuation characters (e.g. "-", ".", ".") will be removed from all strings. If `FALSE`, then strings will be left unmodified.
`ignore.cols`	Vector of columns in `df` that should be ignored when `replace.accents` or `strip.punctuation` are set to `TRUE`.
`nThread`	Integer, set number of threads to be used when writing file. If `NULL` (default), then the output of data.table::getDTthreads will be used.

Details

`WriteHarmonizedData` is a convenience wrapper function of `fread` to export harmonized data in the HYPEObsMetadataTools Harmonized Data Format. The function checks that all required columns are present, includes options to format strings, and exports data to output CSV files with the correct encoding and formatting.

Value

`WriteHarmonizedData` exports a CSV file if `filename` is specified. Otherwise, the function outputs a data frame to the console.

Examples

```
df <- data.frame(
  "STATION_ID" = "A1",
  "DATE_START" = "2002-06-18 12:00",
  "DATE_END" = "2002-06-18 12:00",
  "PARAMETER" = "NH4_N",
  "VALUE" = 0.050,
  "UNIT" = "mg/L",
  "QUALITY_CODE" = "AA"
)
WriteHarmonizedData(df)
```

`WriteHarmonizedSpatialDescription`

Write a Harmonized Spatial Description File

Description

This is a convenience wrapper function to export a data frame to the required Harmonized Spatial Description File format. See the HYPEObsMetadataTools documentation.

Usage

```
WriteHarmonizedSpatialDescription(
  df,
  filename = "",
  replace.accents = FALSE,
  strip.punctuation = FALSE,
  ignore.cols = NULL,
  nThread = NULL
)
```

Arguments

`df`	Data frame containing the harmonized spatial description data.
`filename`	Path to and file name (including ".csv" file extension) of the Harmonized Spatial Description CSV file to export. Windows users: Note that Paths are separated by '/', not '\'.
`replace.accents`	Logical, if `TRUE`, then accented characters (e.g. ä, ö, å) will be replaced with non-accented characters in all strings. If `FALSE`, then strings will be left unmodified.
`strip.punctuation`	Logical, if `TRUE`, then punctuation characters (e.g. "-", ".", ".") will be removed from all strings. If `FALSE`, then strings will be left unmodified.
`ignore.cols`	Vector of columns in `df` that should be ignored when `replace.accents` or `strip.punctuation` are set to `TRUE`.

nThread — Integer, set number of threads to be used when writing file. If NULL (default), then the output of data.table::getDTthreads will be used

Details

WriteHarmonizedSpatialDescription is a convenience wrapper function of fread to export harmonized spatial description data in the HYPEObsMetadataTools Harmonized Spatial Description Format. The function checks that all required columns are present, includes options to format strings, and exports data to output CSV files with the correct encoding and formatting.

Value

WriteSpatialDescrption exports a CSV file if filename is specified. Otherwise, the function outputs a data frame to the console.

Examples

```
df <- data.frame(
  "STATION_ID" = "A1",
  "SRC_NAME" = "Example",
  "DOWNLOAD_DATE" = "2022-10-19",
  "SRC_STATNAME" = "Station",
  "SRC_WBNAME" = "River",
  "SRC_UAREA" = NA,
  "SRC_XCOORD" = 28.11831,
  "SRC_YCOORD" = -25.83053,
  "SRC_EPSG" = 4326,
  "ADJ_XCOORD" = 28.11831,
  "ADJ_YCOORD" = -25.83053,
  "ADJ_EPSG" = 4326
)

WriteHarmonizedSpatialDescription(df)
```

WriteInfo *Write a 'info.txt' File*

Description

WriteInfo writes its required argument x to a file.

Usage

```
WriteInfo(x, filename)
```

Arguments

x	The object to be written, a list with named vector elements, as an object returned from ReadInfo using the exact mode.
filename	A character string naming a file to write to. Windows users: Note that Paths are separated by '/', not '\'.

Details

WriteInfo writes an 'info.txt' file, typically originating from an imported and modified 'info.txt'.

Value

No return value, called for export to text files.

See Also

ReadInfo with a description of the expected content of x. AddInfoLine RemoveInfoLine

Examples

```
te <- ReadInfo(filename = system.file("demo_model",
"info.txt", package = "HYPEtools"), mode = "exact")
WriteInfo(x = te, filename = tempfile())
```

WriteMapOutput *Write a 'mapXXXX.txt' file*

Description

Function to export a map output file from R.

Usage

```
WriteMapOutput(x, filename, dt.format = "%Y-%m-%d")
```

Arguments

x	The object to be written, a dataframe with comment, date, and timestep attributes, as an object returned from ReadMapOutput.
filename	A character string naming a file to write to. Windows users: Note that Paths are separated by '/', not '\'.
dt.format	Date-time format string as in strptime. Date format for export of column headers. Incomplete format strings for monthly and annual values allowed, e.g. '\ Use NULL for single-column dataframes, i.e. long-term average map files.

Details

`WriteMapOutput` exports a dataframe with headers and formatting options adjusted to match HYPE's map output files. The function attempts to format date-time information to strings and will return a warning if the attempt fails.

Value

No return value, called for export to text files.

Examples

```
te <- ReadMapOutput(filename = system.file("demo_model", "results", "mapEVAP.txt",
                    package = "HYPEtools"), dt.format = NULL)
WriteMapOutput(x = te, filename = tempfile())
```

WriteObs *Write 'Pobs.txt', 'Tobs.txt', 'Qobs.txt', and other observation data files*

Description

Export forcing data and discharge observation files from R.

Usage

```
WriteObs(
  x,
  filename,
  dt.format = "%Y-%m-%d",
  round = NULL,
  signif = NULL,
  obsid = NULL,
  append = FALSE,
  comment = NULL
)

WritePTQobs(
  x,
  filename,
  dt.format = "%Y-%m-%d",
  round = NULL,
  signif = NULL,
  obsid = NULL,
  append = FALSE,
  comment = NULL
)
```

Arguments

x	The object to be written, a dataframe containing observation date-times in first and observations in SUBIDs or OBSIDs in remaining columns. If argument obsid is not provided, x must have an additional attribute obsid containing observation IDs/SUBIDs in column order.
filename	Path to and file name of the file to import. Windows users: Note that Paths are separated by '/', not '\'.
dt.format	Date-time format string as in strptime.
round, signif	Integer, number of decimal places and number of significant digits to export, respectively. See round and signif. Applied in sequence (round first and signif second). If NULL (default), the data to export is not touched.
obsid	Integer vector containing observation IDs/SUBIDs in same order as columns in x. To be exported as header in the obs file. Must contain the same number of IDs as observation series in x. If NULL, an attribute obsid in x is mandatory. An existing obsid argument takes precedence over a obsid attribute.
append	Logical, if TRUE, then table will be joined to the data in existing file and the output will be sorted by DATE (Rows will be added for any missing dates).
comment	A character string to be exported as first row comment in the Obs file. Comments are only exported if append is FALSE.

Details

WriteObs is a convenience wrapper function of fwrite to export a HYPE-compliant observation file. Headers are generated from attribute obsid on export (see attr on how to create and access it).

Observation IDs are SUBIDs or IDs connected to SUBIDs with a ForcKey.txt file.

If the first column in x contains dates of class POSIXt, then they will be formatted according to dt.format before writing the output file.

If round is specified, then WriteObs() will use round to round the observation values to a specified number of decimal places. Alternatively, signif can be used to round the observation values to a specified number of significant digits using signif. Finally, if both round and signif are specified, then the observation values will be first rounded to the number of decimal places specified with round and then rounded to the number of significant digits specified with signif.

Value

No return value, called for export to text files.

See Also

ReadObs WriteXobs

Examples

```
te <- ReadObs(filename = system.file("demo_model", "Tobs.txt", package = "HYPEtools"))
WriteObs(x = te, filename = tempfile())
```

WriteOptpar *Write an 'optpar.txt' File*

Description

WriteOptpar prints a HYPE parameter optimization list to a file.

Usage

```
WriteOptpar(x, filename, digits = 10, nsmall = 1)
```

Arguments

x	The object to be written, a list with named elements, as an object returned from ReadOptpar.
filename	A character string naming a file to write to. Windows users: Note that Paths are separated by '/', not '\'.
digits	Integer, number of significant digits to export. See format.
nsmall	Integer, number of significant decimals to export. See format.

Value

No return value, called for export to text files.

See Also

ReadOptpar with a description of the expected content of x.

Examples

```
te <- ReadOptpar(filename = system.file("demo_model", "optpar.txt", package = "HYPEtools"))
WriteOptpar(x = te, filename = tempfile())
```

WritePar *Write a 'par.txt' File*

Description

WritePar prints its required argument x to a file.

Usage

```
WritePar(x, filename, digits = 10, nsmall = 1)
```

Arguments

x	The object to be written, a list with named vector elements, as an object returned from ReadPar.
filename	A character string naming a file to write to. Windows users: Note that Paths are separated by '/', not '\'.
digits	Integer, number of significant digits to export. See format.
nsmall	Integer, number of significant decimals to export. See format.

Details

WritePar writes a 'par.txt' file, typically originating from an imported and modified 'par.txt'.

Value

No return value, called for export to text files.

See Also

ReadPar with a description of the expected content of x.

Examples

```
te <- ReadPar(filename = system.file("demo_model", "par.txt", package = "HYPEtools"))
# Note that par files lose all comment rows on import
WritePar(x = te, filename = tempfile())
```

WritePmsf *Write a 'pmsf.txt' file*

Description

This is a small convenience function to export a 'partial model setup file' from R.

Usage

```
WritePmsf(x, filename)
```

Arguments

x	The object to be written, an integer vector containing SUBIDs.
filename	A character string naming a file to write to. Windows users: Note that Paths are separated by '/', not '\'.

Details

Pmsf files are represented as integer vectors in R. The total number of subcatchments in the file are added as first value on export. pmsf.txt files need to be ordered as downstream sequence.

Value

No return value, called for export to text files.

See Also

`AllUpstreamSubids`, which extracts upstream SUBIDs from a GeoData dataframe.

Examples

```
te <- ReadGeoData(filename = system.file("demo_model", "GeoData.txt", package = "HYPEtools"))
WritePmsf(x = te$SUBID[te$SUBID %in% AllUpstreamSubids(3564, te)], filename = tempfile())
```

WriteTimeOutput *Write a 'timeXXXX.txt' file*

Description

Function to export a time output file from R.

Usage

```
WriteTimeOutput(x, filename, dt.format = "%Y-%m-%d")
```

Arguments

`x`	The object to be written, a dataframe with `comment` and `subid` attributes, as an object returned from `ReadTimeOutput`.
`filename`	A character string naming a file to write to. Windows users: Note that Paths are separated by '/', not '\'.
`dt.format`	Date-time `format` string as in `strptime`. Incomplete format strings for monthly and annual values allowed, e.g. '\%Y'.

Details

`WriteTimeOutput` exports a data frame with headers and formatting options adjusted to match HYPE's time output files.

Value

No return value, called for export to text files.

Examples

```
te <- ReadTimeOutput(filename = system.file("demo_model", "results", "timeCOUT.txt",
                     package = "HYPEtools"), dt.format = "%Y-%m")
WriteTimeOutput(x = te, filename = tempfile(), dt.format = "%Y-%m")
```

`WriteXobs` *Write an 'Xobs.txt' File*

Description

`WriteXobs` writes or appends an observation data set to an Xobs file.

Usage

```
WriteXobs(
  x,
  filename,
  append = FALSE,
  comment = NULL,
  variable = NULL,
  subid = NULL,
  last.date = NULL,
  timestep = "d"
)
```

Arguments

`x`	A data frame, e.g. an object originally imported with `ReadXobs`. Date-time information in the first column and measured values in the remaining columns. Column names are ignored on export, but if attributes `comment`, `variable`, and `subid` are available, these can be exported as Xobs headers (see also arguments of the same names below).
`filename`	A character string naming a file to write to. Windows users: Note that Paths are separated by '/', not '\'.
`append`	Logical. If `TRUE`, `x` will be appended to file `filename`. File must exist and have an identical column structure as `x`. If `FALSE`, existing file in `filename` will be overwritten!
`comment`	A character string to be exported as first row comment in the Xobs file. If provided, it takes precedence over a `comment` attribute of `x`. Comments are only exported if `append` is `FALSE`.
`variable`	A character vector to be exported as second row in the Xobs file. Must contain the same number of variables as `x`. If omitted or `NULL`, an attribute `variable` in `x` is mandatory. Will take precedence over a `variable` attribute of `x`. If `append` is `TRUE` the values are used to test for consistency between export object and the existing file.

subid	Third row in Xobs, containing SUBIDs (integer). Behavior otherwise as argument variable.
last.date	Optional date-time of last observation in existing Xobs file as text string. Only relevant with append = TRUE. Formatting depending on time step, e.g. '2000-01-01' (day) or '2000-01-01 00:00' (hour). Will be automatically read from file per default, but can be provided to reduce execution time when appending to large files.
timestep	Character string, either "day" or "hour", giving the time step between observations. Can be abbreviated.

Details

WriteXobs writes a 'Xobs.txt' file, typically originating from an imported and modified 'Xobs.txt'. HYPE Xobs files contain a three-row header, with a comment line first, next a line of variables, and then a line of SUBIDs. Objects imported with ReadXobs include attributes holding this information, and WriteXobs will use this information. Otherwise, these attributes can be added to objects prior to calling WriteXobs, or passed as function arguments.

If argument append is TRUE, the function requires daily or hourly time steps as input. The date-time column must be of class POSIXct, see as.POSIXct. Objects returned from ReadXobs per default have the correct class for the date-time column. When appending to existing file, the function adds new rows with '-9999' values in all data columns to fill any time gaps between existing and new data. If time periods overlap, the export will stop with an error message. Argument last.date can be provided to speed up appending exports, but per default, WriteXobs extracts the last observation in the existing file automatically.

Value

No return value, called for export to text files.

Note

Both variable and subid do not include elements for the first column in the Xobs file/object, in accordance with ReadXobs. These elements will be added by the function.

Examples

```
te <- ReadXobs(filename = system.file("demo_model", "Xobs.txt", package = "HYPEtools"))
WriteXobs(x = te, filename = tempfile())
```

Index