

Package: wcsvatin (via r-universe)

July 20, 2026

Title Weather and Climate Inputs for 'SWAT'

Version 0.1.1

Description Provides workflows to prepare weather and climate time series from gridded and station data for 'SWAT' ('Soil and Water Assessment Tool'). Supports data extraction, aggregation, interpolation, quality control, unit conversion, and export of per-location model input files. For the underlying model, see Arnold et al. (1998) ``Large Area Hydrologic Modeling and Assessment Part I: Model Development" <doi:10.1111/j.1752-1688.1998.tb05961.x>.

License GPL (>= 3)

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

Depends R (>= 4.1.0)

Suggests future, knitr, rmarkdown, testthat (>= 3.0.0), withr

Config/testthat/edition 3

Imports dplyr, tidyr, raster, ncdf4, data.table, glue, stringr, sf, lubridate, hyfo, ggplot2, terra, future.apply, progressr, methods

URL <https://github.com/reginalexavier/wcsvatin>,
<https://reginalexavier.github.io/wcsvatin/>

BugReports <https://github.com/reginalexavier/wcsvatin/issues>

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev libicu-dev libnetcdf-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev

Repository <https://reginalexavier.r-universe.dev>

Date/Publication 2026-07-10 16:20:29 UTC

RemoteUrl <https://github.com/reginalexavier/wcsvatin>

RemoteRef HEAD

RemoteSha db201bbb977b43a33292d629684edd62e3fb56b2

Contents

wcsvatin-package	2
count_na	3
cube2table	3
daily_aggregation	5
datacube_aggregation	7
files_to_table	8
fill_gap	9
input_raster	10
layervalues2pixel	11
main_input_var	12
point_to_daily	12
raster_info	13
rh_calculator	14
save_daily_tbl	15
study_area_records	15
summary_plot	16
summary_table	17
table_to_files	18
tbl_from_references	18
ts_point_to_files	20
ts_to_area	20
ts_to_point	21
unit_converter	22
var_main_creator	22
var_names	23
windspeed_calculator	24
Index	25

wcsvatin-package

wcsvatin: Climate & Weather SWAT Input.

Description

wcsvatin (Climate & Weather SWAT input) is an open-source R package for preparing weather and climate data from different sources for input in the Soil & Water Assessment Tool (SWAT), funded by the Critical Ecosystem Partnership Fund (CEPF). Currently two blocks of processing routines are implemented, one for the pre-processing of NetCDF and tif raster files as made available from a increasing number of data-providing institutions around the globe and a second one for the upscaling of physical station data by interpolation methods. For processing all used datasets MUST have geographic coordinates using WGS 84 as datum.

Author(s)

Reginal Exavier <reginalexavier@rocketmail.com>, Fernando Shinji Kawakubo <fskgeo@gmail.com>, Peter Zeilhofer <zeilhoferpeter@gmail.com>

count_na	<i>Count the amount or percentage of NA in a table by column</i>
----------	--

Description

Count the amount or percentage of NA in a table by column

Usage

```
count_na(dataset, percent = FALSE)
```

Arguments

dataset	A dataframe containing rainfall data from different gauges in its columns.
percent	logical, controls whether to calculate the amount or percentage of NA.

Value

A dataframe.

cube2table	<i>Convert a Cube format data into a Table format</i>
------------	---

Description

The function extracts the values of a NetCDF/raster layer and converts it to a table format containing the values of the pixels and the layer name as two columns. The pixel is identified by the ID (row and col), and the layer name represents the date of the data collected. All layers are stacked in a single table, each layer is differentiated by the column layer_name containing the date of the data collected. The function, due to the large amount of data, counts with the structure of parallel processing based on the future package to speed up the process. By default, the computation is done in sequential mode `future::plan(future::sequential)`, for parallel processing, the user must change to the desired mode (ex: `future::plan(future::multisession, workers = 6)`).

Usage

```
cube2table(
  input_path,
  var = NA,
  n_layers,
  study_area,
  future_scheduling = 1,
  missing_value = -99,
  final_dir = NULL,
  side_effect = "only",
  temp_dir = NULL,
  clean_after = FALSE
)
```

Arguments

<code>input_path</code>	Path to the NetCDF or raster file.
<code>var</code>	The variable to be extracted. The default is NA. For NetCDF files containing multiple variables, the user must provide the name of the variable to be extracted. If the file contains only one variable, the user can leave this argument as NA.
<code>n_layers</code>	Number of layers in the raster file to be extracted
<code>study_area</code>	The table from 'study_area_records'
<code>future_scheduling</code>	Controlling how the future will be scheduled and distributed between the workers. The default is 1, which means that the future will be scheduled by core. See the documentation of future package for more details future.apply::future_lapply() .
<code>missing_value</code>	The value to be used when the data is missing
<code>final_dir</code>	The directory to save the final table. If NULL, the final table will not be saved.
<code>side_effect</code>	The side effect of the function. The default is "only", which means that the function will only save the final table in disk (if final_dir is provided). The other options are "both" and "none". If "both", the function will save the final table in disk and return it within the R environment. If "none", the function will only return the final table within the R environment.
<code>temp_dir</code>	The directory to save the intermediate tables. If the directory already exists, the tables will be saved in the existing directory. If the directory does not exist, it will be created. If NULL, the tables will be saved in a temporary directory.
<code>clean_after</code>	Logical. If TRUE, the directory with the intermediate tables will be deleted after the process is finished. If FALSE, the directory will be kept. The default is FALSE. And when the temp_dir is NULL, what implies that the tables will be saved in a temporary directory, the temp_dir will be deleted after the process is finished.

Value

A table containing the:

- ID: The pixel ID (row and col);
- values: The values of the pixel;
- layer_name: The layer name (date of the data collected).

Examples

```
cube_file <- tempfile("wswatin-cube-", fileext = ".tif")
cube <- terra::rast(
  nrows = 1,
  ncols = 2,
  nlyrs = 2,
  vals = c(10, 20, 11, 21)
)
names(cube) <- c("X20200101", "X20200102")
terra::writeRaster(cube, cube_file, overwrite = TRUE)
cube2table(
  input_path = cube_file,
  var = NULL,
  n_layers = 2,
  study_area = data.frame(ID = 1:2),
  side_effect = "none"
)
unlink(cube_file)
```

daily_aggregation

Create a daily aggregation from an hourly dataset

Description

<https://confluence.ecmwf.int/display/CKB/ERA5+family+post-processed+daily+statistics+documentation>
 # nolint: line_length_linter

Usage

```
daily_aggregation(
  folder_in,
  folder_out,
  pattern = ".txt$",
  from = "2002-01-01 00",
  to = "2021-05-31 23",
  drop_first_record = TRUE,
  aggregation_function = mean,
  mode = c("agg_fun", "max_min", "value_at_hour")[1],
  value_hour = 0,
  na.rm = FALSE
)
```

Arguments

folder_in	Path of the input files
folder_out	Path where to save the transformed files
pattern	an optional regular expression . Only file names which match the regular expression will be returned.
from	The first date of the series, including the hour part.
to	The last date of the series, including the hour part.
drop_first_record	Logical. If TRUE, the first row of each input file is removed before the date sequence is assigned. This is useful when the input file contains a leading 00:00 record that belongs to the previous day and should not be part of the requested from/to range. After this optional removal, the number of rows in each input file must match the number of hours between from and to.
aggregation_function	The function to use on the hourly groups like mean, sum, mode, etc
mode	The mode of aggregation. The options are agg_fun, max_min or value_at_hour.
value_hour	Integer hour between 0 and 23 used when mode = "value_at_hour". The default is 0, which matches products whose daily accumulated value is timestamped at 00:00 at the end of the accumulation period. In this case, users should include the following day's 00:00 record in the requested period.
na.rm	a logical value indicating whether NA values should be removed before the computation proceeds.

Details

This function allows to aggregate hourly observations to daily time series. The function for aggregation can be informed in the aggregation_function parameter, this parameter takes a function as argument. The default function is [mean](#), so a daily average is returned.

The function will create a daily aggregation from an hourly dataset. The function for aggregation can be informed in the aggregation_function parameter, this parameter takes a function as argument. The default function is [mean](#), so a daily average is returned. Alternatively, the user can choose the mode parameter to inform the function to use choosing between the agg_fun, max_min, and value_at_hour. The agg_fun will use the function informed in the aggregation_function parameter. The max_min will return the maximum and minimum values of the day. The value_at_hour mode will return the value timestamped at value_hour. For daily accumulated products timestamped at 00:00, set mode = "value_at_hour", keep value_hour = 0, and define from/to so that the 00:00 record ending the accumulation period is included. Use drop_first_record = TRUE only when the file also contains an extra leading row that must be discarded before assigning this date range.

Value

Files with a daily resolution

datacube_aggregation *Create a daily aggregation from an hourly datacube*

Description

datacube_aggregation() aggregates a datacube by a given function. The function can either apply an aggregation function to each day of the datacube or select the layer timestamped at a given hour.

Usage

```
datacube_aggregation(
  input_path,
  output_filename = "",
  fun = mean,
  cores = 1,
  mode = c("agg_fun", "value_at_hour")[1],
  value_hour = 0,
  date_shift_days = 0,
  drop_first_layer = FALSE,
  ...
)
```

Arguments

input_path	Path to the datasetcube
output_filename	Path to the output file
fun	Function to be applied to the datasetcube (default is mean). The function must be a function that takes a vector as input and returns a single value. The main functions to be used are: Sum, Mean, Min, Max, First and Last. For last, use <code>dplyr::last</code> . To use customized function say, for example "min", you could use the format <code>fun = \(\x) min(\x)</code> . See terra::tapp() for more information.
cores	Number of cores to use for the aggregation. Default is 1. See terra::tapp() for more information. See terra::tapp() for more information.
mode	The mode of aggregation. The options are <code>agg_fun</code> or <code>value_at_hour</code> . The <code>agg_fun</code> mode applies <code>fun</code> to all layers in each day. The <code>value_at_hour</code> mode returns the layer timestamped at <code>value_hour</code> .
value_hour	Integer hour between 0 and 23 used when <code>mode = "value_at_hour"</code> . The default is 0, which matches products whose daily accumulated value is timestamped at 00:00 at the end of the accumulation period.
date_shift_days	Whole number of days added to the output layer dates when <code>mode = "value_at_hour"</code> . Use -1 for products whose 00:00 timestamp represents the previous day.

drop_first_layer

Logical. If TRUE and mode = "value_at_hour", the first selected layer is removed. This is useful when the first selected layer represents the day before the requested period.

...

Additional arguments to pass to `names_to_date()`

Value

A raster object with the aggregated data

See Also

`terra::tapp()`, `names_to_date()`, [ERA5 family post-processed daily statistics documentation](#) # nolint: line_length_linter

files_to_table	<i>Turns multiple time series files into a single table</i>
----------------	---

Description

Turns multiple time series files into a single table

Usage

```
files_to_table(  
  files_path,  
  files_pattern,  
  start_date = "1970-12-31",  
  end_date = "1980-12-31",  
  interval = "day",  
  na_value = NA,  
  neg_to_zero = FALSE  
)
```

Arguments

files_path	path where the files are.
files_pattern	pattern for the observation/station points name.
start_date	Inform the start date of the series in the format %Y-%m-%d.
end_date	Inform the end date of the series in the format %Y-%m-%d.
interval	Inform the interval between two observations. See the function <code>x</code>
na_value	Value encoded as not available, use NA to leave it the way it is.
neg_to_zero	logical. inform whether negative values should be corrected to zero.

Value

A dataframe.

Examples

```
series_dir <- tempfile("wcsvatin-series-")
dir.create(series_dir)
utils::write.csv(
  data.frame(value = c(1, 2)),
  file.path(series_dir, "station_a.csv"),
  row.names = FALSE
)
utils::write.csv(
  data.frame(value = c(3, 4)),
  file.path(series_dir, "station_b.csv"),
  row.names = FALSE
)
files_to_table(
  files_path = series_dir,
  files_pattern = "station",
  start_date = "2020-01-01",
  end_date = "2020-01-02"
)
unlink(series_dir, recursive = TRUE)
```

fill_gap

A wrapper function for filling gaps in the rainfall time series

Description

This function is a wrapper for the [fillGap](#) in the hyfo **hyfo** package. The main idea here for this wrapping function is to preserve the column names as they are in the dataset input.

Usage

```
fill_gap(dataset, corPeriod = "daily")
```

Arguments

dataset	A dataframe with first column the time, the rest columns are rainfall data of different gauges.
corPeriod	A string showing the period used in the correlation computing, e.g. daily, monthly, yearly.

Value

A dataframe.

See Also

For more detail about the algorithm used to fill the gaps, please see [fillGap](#).

input_raster	<i>Input Raster</i>
--------------	---------------------

Description

Method to load ncdf or tiff file and convert them into a SpatRaster object.

Usage

```
input_raster(x, ...)  
  
## S4 method for signature 'character'  
input_raster(x, ...)  
  
## S4 method for signature 'SpatRaster'  
input_raster(x, ...)  
  
## S4 method for signature 'RasterLayer'  
input_raster(x, ...)  
  
## S4 method for signature 'RasterBrick'  
input_raster(x, ...)  
  
## S4 method for signature 'RasterStack'  
input_raster(x, ...)
```

Arguments

x	path (character) to the file or a SpatRaster object.
...	additional arguments to terra:: rast .

Value

SpatRaster

See Also

terra::rast

layervalues2pixel	<i>Series of Pixel Values</i>
-------------------	-------------------------------

Description

Converts layer-wise values from `cube2table()` into SWAT-style input: a time series for each pixel in the study area.

Usage

```
layervalues2pixel(
  layer_values,
  main_tbl,
  col_name = "20220101",
  inline_output = TRUE,
  path_output = NULL,
  append = FALSE
)
```

Arguments

<code>layer_values</code>	List. Values extracted per raster layer (from <code>cube2table()</code>).
<code>main_tbl</code>	A table with pixel metadata (e.g., from <code>main_input_var()</code>), used to name each output table.
<code>col_name</code>	Column name for each SWAT input table. Typically the first date in the time series (e.g., "20220101").
<code>inline_output</code>	Logical. If TRUE, returns a list of data.tables.
<code>path_output</code>	Directory to write one file per pixel when <code>inline_output = FALSE</code> .
<code>append</code>	Logical. If TRUE, append to existing files; otherwise overwrite.

Value

A list of tables (when `inline_output = TRUE`) or a set of files in `path_output` (one for each pixel).

Examples

```
layer_values <- data.frame(
  ID = c(1, 2, 1, 2),
  values = c(10, 20, 11, 21),
  layer_name = c("day_1", "day_1", "day_2", "day_2")
)
main_tbl <- data.frame(NAME = c("tmin_1", "tmin_2"))
layervalues2pixel(
  layer_values = layer_values,
  main_tbl = main_tbl,
  col_name = "20200101"
)
```

main_input_var	<i>Main table constructor by Variable</i>
----------------	---

Description

Construct a main table needed for the input in SWAT

Usage

```
main_input_var(study_area, var_name = "temp")
```

Arguments

study_area	The object from 'study_area_records'
var_name	The name of the variable to be extracted

Value

A table

Examples

```
study_area <- data.frame(
  ID = 1:2,
  LAT = c(-15.0, -15.5),
  LON = c(-56.0, -56.5),
  ELEVATION = c(100, 120)
)
main_input_var(study_area, var_name = "tmin")
```

point_to_daily	<i>Transforms the raw value from point perspective to a daily perspective</i>
----------------	---

Description

Having the collected observation from a point perspective, this function transform the input to a vertical perspective, like a a daily perspective.

Usage

```
point_to_daily(
  my_folder,
  var_pattern = "p-",
  main_pattern = "pcp",
  start_date = "20170301",
  end_date = "20170331",
```

```

    interval = "day",
    na_value = -99,
    neg_to_zero = TRUE,
    prefix = "day_"
  )

```

Arguments

my_folder	character. The path to the raw files.
var_pattern	character. A pattern for the observation/station points name.
main_pattern	character. A pattern for the main file containing all the points with ID, NAME, LAT, LON and ELEVATION.
start_date	character. Inform the start date of the serie in the format yyyyymmdd.
end_date	character. Inform the end date of the serie in the format yyyyymmdd.
interval	character. Inform the interval between two observations. See the function x
na_value	numeric. Value encoded as not available, use NA for NA.
neg_to_zero	logical. Inform whether negative values should be corrected to zero after applying na_value.
prefix	character. A prefix for naming the table in the format of "prefix+date". for more detail.

Value

A list of table

Examples

```

folder <- system.file("extdata/pcp_stations", package = "wswatin")
test01 <- point_to_daily(my_folder = folder)

```

raster_info	<i>Summarize raster file metadata</i>
-------------	---------------------------------------

Description

Summarize raster file metadata

Usage

```
raster_info(path)
```

Arguments

path	Path to one or more raster files.
------	-----------------------------------

Value

A data.table with one row per raster variable and the columns: file, variable, long_name, unit, n_layers, n_rows, n_cols, x_min, x_max, y_min, y_max, crs, has_time, time_start, time_end, time_step and time_resolution. The file and CRS columns are short labels intended for console inspection.

rh_calculator	<i>Calculate the Relative Humidity from dewpoint and ambient temperature</i>
---------------	--

Description

This function performs the calculation of a relative humidity with dewpoint and ambient temperature as input applying the formula: $RH = 100 \cdot 10^{m \cdot [(Td / (Td + Tn)) - (T_{ambient} / (T_{ambient} + Tn))]}$. Where m and Tn are constants (Vaisala, 2013).

Usage

```
rh_calculator(
  folder_dpt,
  folder_tas,
  folder_out,
  file_name_output = "rh",
  m_value = 7.591386,
  Tn_value = 240.7263,
  pattern = ".txt$"
)
```

Arguments

folder_dpt	Path of the input 2m dewpoint temperature files as Td.
folder_tas	Path of the input Near-Surface Air Temperature files as T _{ambient} .
folder_out	Path where to save the transformed files
file_name_output	Character string for the Relative humidity files on output.
m_value	The value for the constant m (Vaisala, 2013).
Tn_value	The value for T _n (Triple point temperature 273.16 K), constant (Vaisala, 2013).
pattern	an optional regular expression. Only file names which match the regular expression will be returned.

Value

Files with the same temporal resolution as the input

References

VAISALA (2013) HUMIDITY CONVERSION FORMULAS, Calculation formulas for humidity.

save_daily_tbl	<i>Save the csv files after transformation to daily form</i>
----------------	--

Description

Save the csv files after transformation to daily form

Usage

```
save_daily_tbl(tbl_list, path)
```

Arguments

tbl_list	list. A list, the output of the x function
path	The path where the files must be saved

Value

No return value (NULL), called for side effects. Writes one CSV file for every named element of tbl_list to path; output files are named <element-name>.csv and retain the element table structure.

Examples

```
daily_dir <- tempfile("wcsvatin-daily-")
dir.create(daily_dir)
daily_tables <- list(
  day_20200101 = data.frame(ID = 1:2, pcp = c(1.2, 0))
)
save_daily_tbl(daily_tables, daily_dir)
list.files(daily_dir)
unlink(daily_dir, recursive = TRUE)
```

study_area_records	<i>Study Area Records</i>
--------------------	---------------------------

Description

This function extracts the records position grid for the study area combining with the DEM for every point

Usage

```
study_area_records(raster_model, roi, dem)
```

Arguments

raster_model	One layer representing where the data will be extracted
roi	A shapefile delimiting the study area
dem	An elevation raster for the study area

Value

A table

summary_plot	<i>Plot a summary of the data</i>
--------------	-----------------------------------

Description

This function creates a graph of daily values observed from several points on a monthly basis, using a boxplot. You can choose the amount of points to be randomly computed in the total point set.

Usage

```
summary_plot(
  var_folder,
  sample = 5,
  percent = FALSE,
  from = "2002-01-01",
  to = "2021-05-31",
  x_lab = "Months of observation",
  y_lab = "Variable name and unit",
  pattern = ".txt$"
)
```

Arguments

var_folder	Path of the input files
sample	Numeric value, informing the number of files to be used. Until the total amount is informed, the choice of points to be computed is random.
percent	When TRUE, the values passed on sample is used as a percentage.
from	The first date of the series.
to	The last date of the series.
x_lab	Character. Title for the x, see labs .
y_lab	Character. Title for the y, see labs .
pattern	an optional regular expression. Only file names which match the regular expression will be returned.

Value

A summary plot

summary_table

*Table summary of the data***Description**

This function creates a table summary containing the min, the max, the mean, the sd (*standard deviation*) and n (*number of value*) of daily values observed from several points on a monthly basis. When the parameter `by_month` is FALSE the summary return is general, i.e., not on a monthly basis, so the table contains only one row with the same columns (*min, max, mean, sd, and n*). You can choose the amount of points to be randomly computed in the total point set.

Usage

```
summary_table(
  var_folder,
  sample = 5,
  percent = FALSE,
  by_month = TRUE,
  from = "2002-01-01",
  to = "2021-05-31",
  pattern = ".txt$"
)
```

Arguments

<code>var_folder</code>	Path of the input files
<code>sample</code>	Numeric value, informing the number of files to be used. Until the total amount is informed, the choice of points to be computed is random.
<code>percent</code>	When TRUE, the values passed on <code>sample</code> is use as as a percentage.
<code>by_month</code>	Either the summary should be done per month or in general. When true, the parameters <code>from</code> and <code>to</code> are ignored.
<code>from</code>	The first date of the series when <code>by_month</code> is TRUE. Remembering that when <code>by_month</code> is FALSE, this parameter is ignored.
<code>to</code>	The last date of the series when <code>by_month</code> is TRUE. Remembering that when <code>by_month</code> is FALSE, this parameter is ignored.
<code>pattern</code>	an optional regular expression. Only file names which match the regular expression will be returned.

Value

A summary table.

table_to_files	<i>Export tables to txt or csv files</i>
----------------	--

Description

Export tables to txt or csv files

Usage

```
table_to_files(table, folder_path, first_date, file_extension = "txt")
```

Arguments

table	A table <code>data.frame</code> containing all the observations
folder_path	Character string of the folder where the file must be saved.
first_date	Character string of the first date for the time series. This value is used to renaming the columns on every single file while saving. The actual name is used as the file names. The suggested format is <code>%y%m%d</code>
file_extension	Character. <code>txt</code> or <code>csv</code> .

Value

No return value (NULL), called for side effects. Writes one single-column `txt` or `csv` file per column in `table` to `folder_path`. Each file is named after its input column and uses `first_date` as its column name.

tbl_from_references	<i>Extract gridded values at station or reference points</i>
---------------------	--

Description

Extract values from a raster layer, stack or brick at reference point locations and return a wide table with one column per point. This prepares gridded or simulated data for point-based validation against station observations.

Usage

```
tbl_from_references(raster_file, ref_points, prefix_colname = NULL, ...)
```

Arguments

raster_file	Raster object accepted by input_raster .
ref_points	Reference point locations. This input can be a data.frame-like table with NAME, LAT and LON columns, a .txt or .csv file path, an sf object, a SpatVector object or a vector file path accepted by input_vector .
prefix_colname	If not null, a character string used to prefix the station column names.
...	further arguments passed to extract . These arguments concern only extracting the data in the rasters.

Details

This function does not calculate validation metrics. Instead, it prepares the extracted raster values in a table shape that can be combined with observed station data and then passed to validation tools such as `hydroGOF::gof()`, `hydroGOF::ggof()` or any other function that compares observed and simulated series.

The returned table has one row per raster layer and one column per reference point. Column names are taken from the NAME field in `ref_points`, optionally prefixed with `prefix_colname`. When combining this output with observed data, make sure raster layers and observation rows represent the same dates or time steps in the same order.

Additional arguments passed through `...` are forwarded to [extract](#), allowing options such as `method`, `buffer` and `fun`.

Value

A data.frame with one row per raster layer and one column per reference point.

Examples

```
raster_layer <- terra::rast(
  nrows = 2,
  ncols = 2,
  vals = 1:4,
  crs = "EPSG:4326",
  extent = c(-1, 1, -1, 1)
)
raster_stack <- c(raster_layer, raster_layer + 10)

stations <- data.frame(
  NAME = c("station_a", "station_b"),
  LAT = c(-0.5, 0.5),
  LON = c(-0.5, 0.5)
)

simulated <- tbl_from_references(
  raster_file = raster_stack,
  ref_points = stations,
  prefix_colname = "sim"
)
```

simulated

ts_point_to_files	Save trend-surface point time series to files
-------------------	---

Description

Save the list returned by ts_to_point() as one file per target point. Each output file has a single column named with the first date of the time series in YYYYMMDD format.

Usage

```
ts_point_to_files(  
  points_list,  
  output_folder,  
  file_prefix = "pcp",  
  start_date = NULL  
)
```

Arguments

- points_list A list of tables returned by ts_to_point().
- output_folder Path where output files will be saved.
- file_prefix Prefix used in output file names. It is separated from the point ID by an under-score.
- start_date Optional column name to use in all output files. If NULL, the first date found in each point table is used.

Value

NULL. Called for side effects.

ts_to_area	Trend Surface Interpolation into Raster
------------	---

Description

This function make an interpolation with the trend surface method where the user have to inform the polynome degree. The interpolation is made over the tagged points for all the serie on the input.

Usage

```
ts_to_area(my_folder, bassin_limit_path, poly_degree = 2, resolution = 0.01)
```

Arguments

my_folder	Folder containing the ts input files.
bassin_limit_path	A shapefile containing the bassin limit where the trend surface function have to be predicted.
poly_degree	The degree to be used in the polynomial function for the trend surface.
resolution	The resolution for the output raster in degree.

Value

A rasterbrick

Examples

```
ts_to_area(
  my_folder = system.file("extdata/ts_input", package = "wswatin"),
  bassin_limit_path = system.file("extdata/sl_bassin/sl_bassin_limit.shp",
    package = "wswatin"
  ),
  poly_degree = 2,
  resolution = 0.5
)
```

ts_to_point

Trend Surface Interpolation into Targeted Points

Description

This function make an interpolation with the trend surface method where the user have to inform the polynome degree. The interpolation is made over the tagged points for all the serie on the input.

Usage

```
ts_to_point(my_folder, targeted_points_path, poly_degree = 2)
```

Arguments

my_folder	Folder containing the ts input files.
targeted_points_path	A shapefile containing the targeted points where the trend surface function have to predict values.
poly_degree	The degree to be used in the polynomial function for the trend surface.

Value

A list of tibbles by day

unit_converter	<i>Unit Converter</i>
----------------	-----------------------

Description

This function performs the same calculation on each observation of a time series, the time resolution is the same on input as on output. This feature allows you to convert one unit to another. Just inform the conversion function in the FUN parameter. The standard function performs the conversion from Kelvin temperatures to degrees Celsius.

Usage

```
unit_converter(
  folder_in,
  folder_out,
  pattern = ".txt$",
  FUN = function(x) (x - 273.15)
)
```

Arguments

folder_in	Path of the input files
folder_out	Path where to save the transformed files
pattern	an optional regular expression. Only file names which match the regular expression will be returned.
FUN	The function to use for transforming the unit of the variable on input.

Value

Files with the same temporal resolution as the input

var_main_creator	<i>Main table creator for SWAT Input from Trend SURface Interpolation</i>
------------------	---

Description

This function is to create the main table for the input table for SWAT.

Usage

```
var_main_creator(targeted_points_path, var_name = "pcp", col_elev = "Elev")
```

Arguments

targeted_points_path	Shapefile path
var_name	The variable name
col_elev	The column contain the elevation values

Value

A table

Examples

```
var_main_creator(targeted_points_path = system.file("extdata/sl_centroides",
"Centroide_watershed_grau.shp",
package = "wswatin"
))
```

var_names	<i>Shows raster variables present in a NetCDF</i>
-----------	---

Description

Shows raster variables present in a NetCDF

Usage

```
var_names(path)
```

Arguments

path	Path where the netcdf file is located
------	---------------------------------------

Value

Character vector of raster variable names

windspeed_calculator	<i>Calculate the wind speed from Eastward and Northward Near-Surface Wind</i>
----------------------	---

Description

This function performs the calculation of the wind speed from Eastward and Northward Near-Surface Wind as input applying the formula: $ws = \sqrt{u^2 + v^2}$.

Usage

```
windspeed_calculator(
  folder_uas,
  folder_vas,
  folder_out,
  col_name = "20020101",
  file_name_output = "ws",
  pattern = ".txt$"
)
```

Arguments

folder_uas	Path of the input Eastward Near-Surface Wind files (<i>as u component</i>).
folder_vas	Path of the input Northward Near-Surface Wind files (<i>as v component</i>).
folder_out	Path where to save the transformed files
col_name	The column name for the tables on the output. Usually, the first date of the time series.
file_name_output	Character string for the Wind speed files on output.
pattern	an optional regular expression. Only file names which match the regular expression will be returned.

Value

Files with the same temporal resolution as the input.

Index

character-raster (input_raster), [10](#)
count_na, [3](#)
cube2table, [3](#)

daily_aggregation, [5](#)
datacube_aggregation, [7](#)

extract, [19](#)

files_to_table, [8](#)
fill_gap, [9](#)
fillGap, [9](#)
future.apply::future_lapply(), [4](#)

input_raster, [10](#), [19](#)
input_raster,character-method
 (input_raster), [10](#)
input_raster,RasterBrick-method
 (input_raster), [10](#)
input_raster,RasterLayer-method
 (input_raster), [10](#)
input_raster,RasterStack-method
 (input_raster), [10](#)
input_raster,SpatRaster-method
 (input_raster), [10](#)
input_vector, [19](#)

labs, [16](#)
layervalues2pixel, [11](#)

main_input_var, [12](#)
mean, [6](#)

names_to_date(), [8](#)

point_to_daily, [12](#)

rast, [10](#)
raster_info, [13](#)
RasterBrick (input_raster), [10](#)
RasterLayer (input_raster), [10](#)

RasterStack (input_raster), [10](#)
rh_calculator, [14](#)

save_daily_tbl, [15](#)
SpatRaster (input_raster), [10](#)
study_area_records, [15](#)
summary_plot, [16](#)
summary_table, [17](#)

table_to_files, [18](#)
tbl_from_references, [18](#)
terra::tapp(), [7](#), [8](#)
ts_point_to_files, [20](#)
ts_to_area, [20](#)
ts_to_point, [21](#)

unit_converter, [22](#)

var_main_creator, [22](#)
var_names, [23](#)

wcsvatin (wcsvatin-package), [2](#)
wcsvatin-package, [2](#)
windspeed_calculator, [24](#)