

Package ‘hydrographr’

June 19, 2024

Type Package

Title Scalable Hydrographic Data Processing in R

Date 2023

Version 1.1.0

Maintainer Marlene Schürz <marlene.schuerz@igb-berlin.de> and
Afroditi Grigoropoulou <afroditi.grigoropoulou@igb-berlin.de>

Description Scalable hydrographic geospatial data processing tools using open-source command-line utilities. The package provides functions to download the Hydrography90m data (<https://essd.copernicus.org/articles/14/4525/2022/>), processing, reading and extracting information, as well as assessing network distances and network connectivity. While the functions are, by default, tailored towards the Hydrography90m data, they can also be generalized towards other data and purposes, such as efficient cropping and merging of raster and vector data, point-raster extraction, raster reclassification, and data aggregation. The package depends on the open-source software GDAL/OGR, GRASS-GIS and the AWK programming language in the Linux environment, allowing a seamless language integration. Since the data is processed outside R, hydrographr allows creating scalable geo-processing workflows. Please see the installation guide of the additional software at <https://glowabio.github.io/hydrographr/articles/hydrographr.html>. Windows users need to first activate the Windows Subsystem for Linux (WSL) feature. Instructions on how to use hydrographr with other, finer resolution stream networks, can be found at https://glowabio.github.io/hydrographr/articles/example_other_stream_networks.html

License GPL-3

URL <https://glowabio.github.io/hydrographr/>

BugReports <https://github.com/glowabio/hydrographr/issues>

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Imports processx (>= 3.7.0),
 data.table (>= 1.14.2),
 tidyr (>= 1.2.1),
 dplyr (>= 1.0.10),
 stringi (>= 1.7.8),
 stringr (>= 1.4.1),
 rlang (>= 1.0.6),
 DBI (>= 1.1.3),
 RSQLite (>= 2.2.19),
 terra (>= 1.6-41),
 sf (>= 1.0-9),
 parallel (>= 4.2.2),
 doParallel (>= 1.0.17),
 foreach (>= 1.5.2),
 future (>= 1.29.0),
 doFuture (>= 0.12.2),
 future.apply (>= 1.10.0),
 memuse (>= 4.2-2),
 igraph (>= 1.3.5),
 magrittr (>= 2.0.3),
 methods (>= 4.3.0)

Collate 'check_tiles_filesize.R'
 'crop_to_extent.R'
 'download_test_data.R'
 'download_tiles.R'
 'download_tiles_base.R'
 'extract_from_gpkg.R'
 'extract_ids.R'
 'extract_zonal_stat.R'
 'get_all_upstream_distances.R'
 'get_catchment_graph.R'
 'get_centrality.R'
 'get_distance.R'
 'get_distance_parallel.R'
 'get_distance_graph.R'
 'get_pfafstetter_basins.R'
 'get_regional_unit_id.R'
 'get_segment_neighbours.R'
 'get_tile_id.R'
 'get_upstream_catchment.R'
 'get_upstream_variable.R'
 'utils.R'
 'merge_tile.R'
 'read_geopackage.R'
 'reclass_raster.R'
 'report_no_data.R'
 'set_no_data.R'
 'snap_to_network.R'

'snap_to_subc_segment.R'

Contents

crop_to_extent	3
download_test_data	5
download_tiles	6
extract_from_gpkg	9
extract_ids	11
extract_zonal_stat	13
get_all_upstream_distances	16
get_catchment_graph	17
get_centrality	20
get_distance	21
get_distance_graph	24
get_distance_parallel	27
get_pfafstetter_basins	29
get_regional_unit_id	32
get_segment_neighbours	33
get_tile_id	35
get_upstream_catchment	36
get_upstream_variable	39
merge_tiles	42
read_geopackage	44
reclass_raster	46
report_no_data	48
set_no_data	49
snap_to_network	50
snap_to_subc_segment	52
Index	56

crop_to_extent	<i>Crop raster to extent</i>
----------------	------------------------------

Description

This function crops an input raster layer (.tif) given a bounding box (xmin, ymin, xmax, ymax coordinates, or a spatial object from which to extract a bounding box) or to the boundary of a polygon vector layer (cutline source). The cropping is performed directly on disk, i.e. the input layer does not need to be loaded into R. The output is always written to disk, and can be optionally loaded into R as a SpatRaster (terra package) object (using read = TRUE).

Usage

```
crop_to_extent(
  raster_layer,
  vector_layer = NULL,
  bounding_box = NULL,
  out_dir,
  file_name,
  compression = "low",
  bigtiff = TRUE,
  read = TRUE,
  quiet = TRUE
)
```

Arguments

<code>raster_layer</code>	character. Full path to the input raster .tif layer.
<code>vector_layer</code>	character. Full path to a vector layer that is used as a cutline data source (similar to a mask operation).
<code>bounding_box</code>	numeric vector of the coordinates of the corners of a bounding box (xmin, ymin, xmax, ymax), <code>SpatRaster</code> , <code>SpatVector</code> , or other spatial object.
<code>out_dir</code>	character. The directory where the output will be stored.
<code>file_name</code>	character. Name of the cropped output raster .tif file.
<code>compression</code>	character. Compression of the written output file. Compression levels can be defined as "none", "low", or "high". Default is "low".
<code>bigtiff</code>	logical. Define whether the output file is expected to be a BIGTIFF (file size larger than 4 GB). If FALSE and size > 4GB no file will be written. Default is TRUE.
<code>read</code>	logical. If TRUE, the cropped raster .tif layer gets read into R. If FALSE, the layer is only stored on disk. Default is TRUE.
<code>quiet</code>	logical. If FALSE, the standard output will be printed. Default is TRUE.

Value

The function returns always a .tif raster file written to disk. Optionally, a `SpatRaster` (terra object) can be loaded into R with `read = TRUE`.

Author(s)

Yusdiel Torres-Cambas

Examples

```
# Download test data into the temporary R folder
# or define a different directory
my_directory <- tempdir()
download_test_data(my_directory)
```

```
# Define full path to the input raster .tif layer and vector layer
spi_raster <- paste0(my_directory, "/hydrography90m_test_data",
                     "/spi_1264942.tif")
basin_vector <- paste0(my_directory, "/hydrography90m_test_data",
                       "/basin_59.gpkg")

# Crop the Stream Power Index to the basin
spi_basin <- crop_to_extent(raster_layer = spi_raster,
                           vector_layer = basin_vector,
                           out_dir = my_directory,
                           file_name = "spi_basin_cropped.tif",
                           read = TRUE)
```

download_test_data	<i>Download test data</i>
--------------------	---------------------------

Description

Download the test data of the package, which includes all Hydrography90m and species point observation data for a small geographic extent, to test the functions.

The test data will be automatically downloaded and unzipped with this function to a desired path, or can be alternatively downloaded at

https://drive.google.com/file/d/1kYNWxmtVm6X7MZLIS0ePGpvxB1pk1scD/view?usp=share_link.

Usage

```
download_test_data(download_dir = ".")
```

Arguments

download_dir	character. The directory where the files will be downloaded. Default location is the working directory.
--------------	---

Details

Downloads the test data of the Hydrography90m dataset

Author(s)

Afroditi Grigoropoulou

References

Amatulli, G., Garcia Marquez, J., Sethi, T., Kiesel, J., Grigoropoulou, A., Üblacker, M. M., Shen, L. Q., and Domisch, S.: Hydrography90m: a new high-resolution global hydrographic dataset, Earth Syst. Sci. Data, 14, 4525–4550, <https://doi.org/10.5194/essd-14-4525-2022>, 2022.

Amatulli G., Garcia Marquez J., Sethi T., Kiesel J., Grigoropoulou A., Üblacker M., Shen L. & Domisch S. (2022-08-09). Hydrography90m: A new high-resolution global hydrographic dataset. IGB Leibniz-Institute of Freshwater Ecology and Inland Fisheries. dataset. <https://doi.org/10.18728/igb-fred-762.1>

Examples

```
# Download the test data to the current working directory
download_test_data()

# Download the data to a specific (existing) directory
download_test_data("path/to/your/directory")
```

download_tiles	<i>Download files of the Hydrography90m dataset</i>
----------------	---

Description

The function downloads data of the Hydrography90m dataset, which is split into 20°x20° tiles. If a tile ID is specified, then the selected layers (variable) will be downloaded. In addition, the Hydrography90m is organized in non-interrupted drainage basins called regional units. If a regional unit ID (reg_unit_id) is specified, then only the raster mask of the drainage basin is downloaded (useful for later processing). Multiple regular tiles, e.g. belonging to regional units, can be downloaded in a single request. The tile or regional unit IDs can be obtained using the functions "get_tile_id" and "get_regional_unit_id", respectively. The files will be stored locally in a folder architecture, similar as in the data repository, available at <https://public.igb-berlin.de/index.php/s/agciopgzXjWswF4?path=%2F>.

Usage

```
download_tiles(
  variable,
  file_format = "tif",
  tile_id = NULL,
  reg_unit_id = NULL,
  global = FALSE,
  download_dir = "."
)
```

Arguments

variable	character vector of variable names. See Details for all the variable names.
file_format	character. Format of the requested file ("tif" or "gpkg"). See Details.
tile_id	character vector. The IDs of the requested tiles.
reg_unit_id	character vector. The IDs of the requested regional units.
global	logical. If TRUE, the global extent file is downloaded. Default is FALSE.
download_dir	character. The directory where the files will be downloaded. Default is the working directory.

Details

In the following table you can find all the variables included in the Hydrography90m dataset. The column "Variable" includes the variable names that should be used as an input in the parameter "variable" of the function. Likewise, the column "File format" contains the input that should be given to the "file_format" parameter. For more details and visualisations of the spatial layers, please refer to https://hydrography.org/hydrography90m/hydrography90m_layers/.

Variable type	Variable name	Variable	Unit	File format
Network	Drainage basin	basin		tif
Network	Drainage basin	basin		gpkg
Network	Sub-catchment	sub_catchment		tif
Network	Sub-catchment	sub_catchment		gpkg
Network	Stream segment	segment		tif
Network	Outlet	outlet		tif
Network	Outlet	outlet		gpkg
Network	Regional unit	regional_unit		tif
Network	Flow direction	direction		tif
Flow	Flow accumulation	accumulation	km ²	tif
Stream slope	Cell maximum curvature	slope_curv_max_dw_cel	1/m	tif
Stream slope	Cell minimum curvature	slope_curv_min_dw_cel	1/m	tif
Stream slope	Cell elevation difference	slope_elv_dw_cel	m	tif
Stream slope	Cell gradient	slope_grad_dw_cel		tif
Stream distance	Shortest distance to drainage divide	stream_dist_up_near	m	tif
Stream distance	Longest distance to drainage divide	stream_dist_up_farth	m	tif
Stream distance	Nearest down stream stream grid cell	stream_dist_dw_near	m	tif
Stream distance	Outlet grid cell in the network	outlet_dist_dw_basin	m	tif
Stream distance	Down stream stream node grid cell	outlet_dist_dw_scatch	m	tif
Stream distance	Euclidean distance	stream_dist_proximity	m	tif
Elevation difference	Shortest path	stream_diff_up_near	m	tif
Elevation difference	Longest path	stream_diff_up_farth	m	tif
Elevation difference	Nearest downstream stream pixel	stream_diff_dw_near	m	tif
Elevation difference	Outlet grid cell in the network	outlet_diff_dw_basin	m	tif
Elevation difference	Downstream stream node grid cell	outlet_diff_dw_scatch	m	tif
Segment properties	Segment downstream mean gradient	channel_grad_dw_seg		tif
Segment properties	Segment upstream mean gradient	channel_grad_up_seg		tif
Segment properties	Cell upstream gradient	channel_grad_up_cel		tif
Segment properties	Cell stream course curvature	channel_curv_cel		tif

Segment properties	Segment downstream elevation difference	channel_elv_dw_seg		tif
Segment properties	Segment upstream elevation difference	channel_elv_up_seg		tif
Segment properties	Cell upstream elevation difference	channel_elv_up_cel		tif
Segment properties	Cell downstream elevation difference	channel_elv_dw_cel		tif
Segment properties	Segment downstream distance	channel_dist_dw_seg		tif
Segment properties	Segment upstream distance	channel_dist_up_seg		tif
Segment properties	Cell upstream distance	channel_dist_up_cel		tif
Stream order	Strahler's stream order	order_strahler		tif
Stream order	Shreve's stream magnitude	order_shreve		tif
Stream order	Horton's stream order	order_horton		tif
Stream order	Hack's stream order	order_hack		tif
Stream order	Topological dimension of streams	order_topo		tif
Stream order	Strahler's stream order	order_vect_segment		gpkg
Stream order	Shreve's stream magnitude	order_vect_segment		gpkg
Stream order	Horton's stream order	order_vect_segment		gpkg
Stream order	Hack's stream order	order_vect_segment		gpkg
Stream order	Topological dimension of streams	order_vect_segment		gpkg
Stream reach	Length of the stream reach	order_vect_segment	m	gpkg
Stream reach	Straight length	order_vect_segment	m	gpkg
Stream reach	Sinusoid of the stream reach	order_vect_segment		gpkg
Stream reach	Accumulated length	order_vect_segment	m	gpkg
Stream reach	Flow accumulation	order_vect_segment	km ²	gpkg
Stream reach	Distance to outlet	order_vect_segment	m	gpkg
Stream reach	Source elevation	order_vect_segment	m	gpkg
Stream reach	Outlet elevation	order_vect_segment	m	gpkg
Stream reach	Elevation drop	order_vect_segment		gpkg
Stream reach	Outlet drop	order_vect_segment		gpkg
Stream reach	Gradient	order_vect_segment		gpkg
Flow index	Stream power index	spi		tif
Flow index	Sediment transportation index	sti		tif
Flow index	Compound topographic index	cti		tif

Note

If there is an error during the download of a file (more likely in case of files bigger than 3-4GB), you can try to manually download this file by pasting the link that is returned by the error message in your browser.

Author(s)

Afroditi Grigoropoulou

References

Amatulli G., Garcia Marquez J., Sethi T., Kiesel J., Grigoropoulou A., Üblacker M., Shen L. & Domisch S. (2022-08-09) Hydrography90m: A new high-resolution global hydrographic dataset. IGB Leibniz-Institute of Freshwater Ecology and Inland Fisheries. dataset. <https://doi.org/10.18728/igb-fred-762.1>

Examples

```
# Download data for two variables in three regular tiles
# to the current working directory
download_tiles(variable = c("sti", "stream_dist_up_farth"),
              file_format = "tif",
              tile_id = c("h00v02", "h16v02", "h16v04"))

# Download the global .tif layer for the variable "direction"
# into the temporary R folder or define a different directory
# Define directory
my_directory <- tempdir()
# Download layer
download_tiles(variable = "direction",
              file_format = "tif",
              global = TRUE,
              download_dir = my_directory)

# Download the raster mask of two regional units
# to the current working directory.
download_tiles(variable = "regional_unit",
              file_format = "tif",
              reg_unit_id = c("33", "34"))

# Download the raster mask of all regional units
# to the current working directory.
download_tiles(variable = "regional_unit",
              file_format = "tif",
              global = TRUE)
```

extract_from_gpkg	<i>Extract values from the stream order .gpkg files.</i>
-------------------	--

Description

The function reads the attribute table of the stream network GeoPackage file (.gpkg) stored on disk and extracts the data for one or more (or all) input sub-catchment (i.e. stream segment) IDs. The output is a data.table, and only the output is loaded into R.

Usage

```
extract_from_gpkg(
  data_dir,
  subc_id,
  subc_layer,
  var_layer,
  out_dir = NULL,
  file_name = NULL,
  n_cores = NULL,
  quiet = TRUE
)
```

Arguments

<code>data_dir</code>	character. Path to the directory containing all input data.
<code>subc_id</code>	a numeric vector of sub-catchment IDs or "all". If "all", the attribute table is extracted for all the stream segments of the input .gpkg layer. The stream segment IDs are the same as the sub-catchment IDs. A vector of the sub-catchment IDs can be acquired from the <code>extract_ids()</code> function, by sub-setting the resulting <code>data.frame</code> .
<code>subc_layer</code>	character. Full path to the sub-catchment ID .tif layer
<code>var_layer</code>	character vector of .gpkg files on disk, e.g. "order_vect_point_h18v04.gpkg".
<code>out_dir</code>	character. The directory where the output will be stored. If the <code>out_dir</code> is specified, the attribute tables will be stored as .csv files in this location, named after their input variable vector files (e.g. "/path/to/stats_order_vect_point_h18v04.csv"). If NULL, the output is only loaded in R and not stored on disk.
<code>file_name</code>	character. Name of the .csv file where the output table will be stored. <code>out_dir</code> should also be specified for this purpose.
<code>n_cores</code>	numeric. Number of cores used for parallelization, in case multiple .gpkg files are provided to <code>var_layer</code> . If NULL, available cores - 1 will be used.
<code>quiet</code>	logical. If FALSE, the standard output will be printed. Default is TRUE.

Details

The following attributes are stored in the stream network .gpkg files (as produced by the GRASS GIS function `r.stream.order`):

- `cat` - category
- `stream` - sub-catchment / stream segment ID (equal to `cat`)
- `next_stream` - downstream sub-catchment / stream segment ID
- `prev_streams`; two or more upstream sub-catchment / stream segment IDs
- `strahler` - Strahler's stream order
- `horton` - Horton's stream order
- `shreve` - Shreve's stream magnitude
- `hack` - Hack's main streams or Gravelius order
- `topo_dim` - Topological dimension streams order
- `scheidegger` - Scheidegger's Consisted Associated Integers
- `drwal` - Drwal's stream hierarchy
- `length` - length of the stream segment
- `stright` - length of the stream segment as a stright line
- `sinusoid` - fractal dimension: stream segment length / stright stream
- `segment length`;
- `cum_length` - length of the stream from the source
- `flow_accum` - flow accumulation within the sub-catchment of a stream segment

- out_dist - distance of current stream initialisation from outlet
- source_elev - elevation at stream segment initialisation
- outlet_elev - elevation at stream segment outlet
- elev_drop difference between source_elev and outlet_elev + drop outlet
- out_drop - drop at the outlet of the stream segment
- gradient - drop/length

Author(s)

Afroditi Grigoropoulou, Jaime Garcia Marquez, Marlene Schürz

References

<https://grass.osgeo.org/grass82/manuals/v.in.ogr.html> <https://grass.osgeo.org/grass82/manuals/addons/r.stream.order.html>

Examples

```
# Download test data into temporary R folder
# or define a different directory
my_directory <- tempdir()
download_test_data(my_directory)

# Define path to the directory containing all input data
test_data <- paste0(my_directory, "/hydrography90m_test_data")

# Define sub-catchment ID layer
subc_raster <- paste0(my_directory, "/hydrography90m_test_data",
                      "/subcatchment_1264942.tif")

# Extract the attribute table of the file order_vect_59.gpkg for all the
# sub-catchment IDs of the subcatchment_1264942.tif raster layer
attribute_table <- extract_from_gpkg(data_dir = test_data,
                                     subc_id = "all",
                                     subc_layer = subc_raster,
                                     var_layer = "order_vect_59.gpkg",
                                     n_cores = 1)

# Show the output table
attribute_table
```

extract_ids

Extract sub-catchment and/or basin IDs

Description

Extracts the ID value of the basin and/or sub-catchment raster layer at given point locations. Can also be used for point-based extraction of any .tif layer by specifying the layer in the "basin" parameter.

Usage

```
extract_ids(  
  data,  
  lon,  
  lat,  
  id = NULL,  
  basin_layer = NULL,  
  subc_layer = NULL,  
  quiet = TRUE  
)
```

Arguments

data	a data.frame or data.table that contains the columns regarding the longitude / latitude coordinates in WGS84.
lon	character. The name of the column with the longitude coordinates.
lat	character. The name of the column with the latitude coordinates.
id	character. The name of a column containing unique IDs for each row of "data" (e.g., occurrence or site IDs).
basin_layer	character. Full path to the .tif layer with the basin ID.
subc_layer	character. Full path to the .tif layer with the sub-catchment ID.
quiet	logical. If FALSE, the standard output will be printed. Default is TRUE.

Details

For the extraction of a value at a given point location from the basin and/or sub-catchment raster layer of the Hydrography90m dataset, the GDAL function 'gdallocationinfo' is used. The point locations have to be defined by coordinates in the WGS84 reference system. The function can also be used to extract any value from a given raster layer in a WGS84 projection, such as environmental information that is stored in the input raster file.

Note

Duplicated rows will be removed.

Author(s)

Afroditi Grigoropoulou, Marlene Schürz

References

<https://gdal.org/programs/gdallocationinfo.html>

Examples

```
# Download test data into the temporary R folder
# or define a different directory
my_directory <- tempdir()
download_test_data(my_directory)

# Load occurrence data
species_occurrence <- read.table(paste0(my_directory,
                                          "/hydrography90m_test_data",
                                          "/spdata_1264942.txt"),
                                 header = TRUE)

# Define full path to the basin and sub-catchments raster layer
basin_raster <- paste0(my_directory,
                       "/hydrography90m_test_data/basin_1264942.tif")
subc_raster <- paste0(my_directory,
                      "/hydrography90m_test_data/basin_1264942.tif")

# Extract basin and sub-catchment IDs from the Hydrography90m layers
hydrography90m_ids <- extract_ids(data = species_occurrence,
                                  lon = "longitude",
                                  lat = "latitude",
                                  id = "occurrence_id",
                                  subc_layer = subc_raster,
                                  basin_layer = basin_raster)

# Show the output table
hydrography90m_ids
```

extract_zonal_stat	<i>Calculate zonal statistics</i>
--------------------	-----------------------------------

Description

Calculate zonal statistics based on one or more environmental variable raster .tif layers. This function aggregates data to 12 summary statistics (mean, min, max, range, ...) for selected or all sub-catchments of the input file. The sub-catchment raster (.tif) input file is read directly from disk. The output is a data.table which is loaded into R. This function can also be used for any zonal statistic calculation by specifying the raster layer zones in the subc_layer parameter and optionally, also the target zone IDs in the subc_id parameter.

Usage

```
extract_zonal_stat(
  data_dir,
  subc_id,
  subc_layer,
  var_layer,
  out_dir = NULL,
```

```

    file_name = NULL,
    n_cores = NULL,
    quiet = TRUE
  )

```

Arguments

<code>data_dir</code>	character. Path to the directory containing all input data.
<code>subc_id</code>	Vector of sub-catchment IDs or "all". If "all", zonal statistics are calculated for all sub-catchments of the given sub-catchment raster layer. A vector of the sub-catchment IDs can be acquired from the <code>extract_ids()</code> function, and by sub-setting the resulting <code>data.frame</code> .
<code>subc_layer</code>	character. Full path to the sub-catchment ID .tif layer.
<code>var_layer</code>	character vector of variable raster layers on disk, e.g. "slope_grad_dw_cel_h00v00.tif". Note that the variable name appears in the output table columns (e.g. <code>slope_grad_dw_cel_mean</code>). To speed up the processing, the selected variable raster layers can be cropped to the extent of the sub-catchment layer, e.g. with <code>crop_to_extent()</code> .
<code>out_dir</code>	character. The directory where the output will be stored. If the <code>out_dir</code> and <code>file_name</code> are specified, the output table will be stored as a .csv file in this location. If they are NULL, the output is only loaded in R and not stored on disk.
<code>file_name</code>	character. Name of the .csv file where the output table will be stored. <code>out_dir</code> should also be specified for this purpose.
<code>n_cores</code>	numeric. Number of cores used for parallelization, in case multiple .tif files are provided to <code>var_layer</code> . Default is 1.
<code>quiet</code>	logical. If FALSE, the standard output will be printed. Default is TRUE.

Value

Returns a table with

- sub-catchment ID (`subc_id`)
- number of cells with a value (`data_cells`)
- number of cells with a NoData value (`nodata_cells`)
- minimum value (`min`)
- maximum value (`max`)
- value range (`range`)
- arithmetic mean (`mean`)
- arithmetic mean of the absolute values (`mean_abs`)
- standard deviation (`sd`)
- variance (`var`)
- coefficient of variation (`cv`)
- sum (`sum`)
- sum of the absolute values (`sum_abs`).

get_all_upstream_distances

Get all upstream distances for all subc_id

Description

Calculates the upstream distance from each subc_id to all upstream subc_id. The output can be directly used in spatial prioritization analyses for e.g. Marxan, Gurobi etc. to specify the longitudinal connectivity. Note that the stream segment and sub-catchment IDs are identical, and for consistency, we use the term "subc_id".

Note that the distance can be extremely long and for the subsequent spatial prioritization analyses you might want to consider setting a cap at a certain distance.

Usage

```
get_all_upstream_distances(network_table = network_table, n_cores = 1)
```

Arguments

network_table	A data.table that includes the columns c(stream, next_stream, out_dist, the latter specifying the distance to the outlet (which is included in the Hydrography90m vector attribute table).
n_cores	numeric. Number of cores used for parallelisation in the case of multiple stream segments / s. Default is 1. Currently, the parallelisation process requires copying the data to each core. In case the graph is very large, and many segments are used as an input, setting n_cores to a higher value can speed up the computation. This comes however at the cost of possible RAM limitations and even slower processing since the large data will be copied to each core. Hence consider testing with n_cores = 1 first. Optional.

Value

A data.table that reports the distance (in meters) from each subc_id to all upstream subc_ids.

Note

Currently the attributes are not provided for the (the selected subc_id segment). If the attributes are also needed for the outlet subc_id, then the next downstream sub_id can be selected (enlarge the study area)

Author(s)

Sami Domisch

References

Csardi G, Nepusz T: The igraph software package for complex network research, InterJournal, Complex Systems 1695. 2006. <https://igraph.org>

See Also

[read_geopackage\(\)](#) and [get_catchment_graph\(\)](#) to create a network graph. Alternatively, see [get_segment_neighbours\(\)](#) to obtain the upstream variables for a specified neighbourhood, or [get_upstream_variable\(\)](#) to aggregate a set of variables across the upstream catchment.

Examples

```
# Download test data into the temporary R folder
# or define a different directory
my_directory <- tempdir()
download_test_data(my_directory)

# Load stream network as a graph
my_graph <- read_geopackage(gpkg = paste0(my_directory,
                                           "/hydrography90m_test_data",
                                           "/order_vect_59.gpkg"),
                           import_as = "graph")

# Pick a random subc_id
subc_id = "513867228"
# Get the upstream catchment as a data.table
network_table <- hydrographr::get_catchment_graph(g = my_graph ,
                                                  subc_id = subc_id,
                                                  mode = "in",
                                                  use_outlet = FALSE,
                                                  as_graph = FALSE,
                                                  n_cores = 1)

## Condense the table supplied to the function to save RAM
keep_these <- c("stream", "next_stream", "out_dist")
network_table <- network_table[, ..keep_these]

## Change to integers
network_table$stream <- as.integer(network_table$stream)
network_table$next_stream <- as.integer(network_table$next_stream)

## Calculate the network distance (in meter) from each subc_id to
## all upstream subc_id using four CPUs for the parallelization
result <- get_all_upstream_distances(network_table = network_table,
                                     n_cores = 4)
```

get_catchment_graph	<i>Get catchment from stream network graph</i>
---------------------	--

Description

Subset the stream network graph by extracting the upstream, downstream or entire catchment, for one or multiple stream segments. The function will return either one or more data.tables or graph

objects for each input stream segment. Note that the stream segment and sub-catchment IDs are identical, and for consistency, we use the term "subc_id".

By switching the mode to either "in", "out" or "all", only the upstream, downstream or all connected segments will be returned, respectively. The function `read_geopackage()` can be used to create the input network graph.

Usage

```
get_catchment_graph(
  g,
  subc_id = NULL,
  use_outlet = FALSE,
  mode = NULL,
  as_graph = FALSE,
  n_cores = 1,
  max_size = 1500
)
```

Arguments

<code>g</code>	igraph object. A directed graph.
<code>subc_id</code>	numeric vector of a single or multiple IDs, e.g (c(ID1, ID2, ID3, ...)). The sub-catchment (equivalent to stream segment) IDs for which to delineate the upstream drainage area. If empty, then outlets will be used as sub-catchment IDs (with <code>use_outlet = TRUE</code>). Note that you can browse the entire network online at https://geo.igb-berlin.de/maps/351/view and to left hand side, select the "Stream segment ID" layer and click on the map to get the ID. Optional.
<code>use_outlet</code>	logical. If TRUE, the outlets of the given network graph will be used as additional input subc_ids. Outlets will be identified internally as those stream segments that do not have any downstream connected segment. Default is FALSE.
<code>mode</code>	character. One of "in", "out" or "all". "in" returns the upstream catchment, "out" returns the downstream catchment (all catchments that are reachable from the given input segment), and "all" returns both.
<code>as_graph</code>	logical. If TRUE, the output will be a new graph or a list of new graphs with the original attributes. If FALSE, the output will be a new data.table or a list of data.tables. List objects are named after the subc_ids. Default is FALSE.
<code>n_cores</code>	numeric. Number of cores used for parallelisation in the case of multiple stream segments / s. Default is 1. Currently, the parallelisation process requires copying the data to each core. In case the graph is very large, and many segments are used as an input, setting <code>n_cores</code> to a higher value can speed up the computation. This comes however at the cost of possible RAM limitations and even slower processing since the large data will be copied to each core. Hence consider testing with <code>n_cores = 1</code> first. Optional.
<code>max_size</code>	numeric. Specifies the maximum size of the data passed to the parallel back-end in MB. Default is 1500 (1.5 GB). Consider a higher value for large study areas (more than one 20°x20° tile). Optional.

get_centrality

Get centrality indexes from stream network graph

Description

Calculate centrality indexes from a directed stream network graph. By switching the mode to either "in", "out" or "all", only the upstream, downstream or all connected segments will be considered, respectively. The function `read_geopackage()` can be used to create the input network graph.

Usage

```
get_centrality(g, index = "all", mode = NULL)
```

Arguments

<code>g</code>	igraph object. A directed graph.
<code>index</code>	character. One of "all", "closeness", "farness", "betweenness", "degree", "eccentricity". See @Details
<code>mode</code>	character. One of "in", "out" or "all". Defines whether the shortest paths to (upstream) or from (downstream) the given segments/sub-catchments should be calculated. If "out", then only downstream segments will be considered. If "in", then only upstream segments will be considered. If "all", then the flow direction will be ignored and all streams will be considered.

Details

The degree of a node is the number of its adjacent edges. Closeness centrality measures how many steps are required to access every other node from a given node. Farness centrality is the sum of the length of the shortest paths between the node and all other nodes. It is the reciprocal of closeness (Altermatt, 2013). The eccentricity of a node is its shortest path distance from the farthest other node in the graph (West, 1996). The node betweenness is (roughly) defined by the number of geodesics (shortest paths) going through a node.

Value

A data.table that reports all subc_id and their centrality values.

Author(s)

Afroditi Grigoropoulou

References

Csardi G, Nepusz T: The igraph software package for complex network research, InterJournal, Complex Systems 1695. 2006. <https://igraph.org>

See Also

[read_geopackage\(\)](#) to create a network graph.

Examples

```
# Download test data into the temporary R folder
# or define a different directory
my_directory <- tempdir()
download_test_data(my_directory)

# Load stream network as a graph
my_graph <- read_geopackage(gpkg = paste0(my_directory,
                                           "/hydrography90m_test_data",
                                           "/order_vect_59.gpkg"),
                           import_as = "graph")

# Get the all the centrality indexes
centrality <- get_centrality(g = my_graph, index = "all", mode = "in")

# Load stream network as a vector
stream_vect <- read_geopackage(gpkg = paste0(my_directory,
                                           "/hydrography90m_test_data",
                                           "/order_vect_59.gpkg"),
                              import_as = "SpatVect")

# Merge the centrality table with the vector
stream_vect <- terra::merge(stream_vect, centrality,
                           by.x = c('stream'), by.y="subc_id")

# Write out the stream network vector including the centrality indices
writeVector(stream_vect, paste0(my_directory,
                                "/hydrography90m_test_data",
                                "/order_vect_59_centra.gpkg"))
```

get_distance	<i>Calculate euclidean or along the network distance between points located in one basin</i>
--------------	--

Description

Calculate euclidean or along-the-network distance (in meters) between points located in one basin. To calculate the distance along the network, point coordinates need to be snapped to the stream network using the function [snap_to_network\(\)](#) or [snap_to_subc_segment\(\)](#).

Usage

```
get_distance(
  data,
  lon,
```

```

    lat,
    id,
    stream_layer = NULL,
    distance = "both",
    n_cores = 1,
    quiet = TRUE
)

```

Arguments

data	a data.frame or data.table that contains the columns regarding the longitude / latitude coordinates in WGS84.
lon	character. The name of the column with the longitude coordinates.
lat	character. The name of the column with the latitude coordinates.
id	character. The name of a column containing unique IDs for each row of "data" (e.g., occurrence or site IDs). The unique IDs need to be numeric and less than 10 characters long.
stream_layer	character. Full path of the stream network .gpkg file. Needs to be defined to calculate the distance along the network. (In the case of Hydrography90m, the relevant files have the format "order_vect_segment_h??v??gpkg")
distance	character. One of "euclidean", "network", or "both". If "euclidean", the euclidean distances between all pairs of points are calculated. If "network", the shortest path along the network between all pairs of points is calculated. (see "Details" for more information). If method is set to "both", both distance measures are calculated. Distances are given in meters. Default is "both".
n_cores	numeric. Number of cores used for parallelisation. Default is 1.
quiet	logical. If FALSE, the standard output will be printed. Default is TRUE.

Details

To calculate the euclidean distance between all pairs of points the function uses the `v.distance` command of GRASS GIS, which has been set up to produce a square matrix of distances. The calculation of distances along the stream network has been implemented with the command `v.net.allpairs` of GRASS GIS. The along-the-network distance calculation is done for all pairs of points located within the same basin. If the points are located in different basins, the function `get_distance_parallel()` should be used.

Value

If `distance='euclidean'`, a distance matrix, in meters, of the euclidean distances between all the pairs of points (object of class `data.frame`) is returned. If `distance='network'`, a `data.frame` with three columns: `from_id`, `to_id`, `dist` is returned. The 'dist' column includes the distance, in meters, of the shortest path along the network from the point "from_id" to the point "to_id". If `distance='both'`, a list containing both objects is returned.

Author(s)

Afroditi Grigoropoulou, Marlene Schürz, Jaime Garcia Marquez

References

<https://grass.osgeo.org/grass82/manuals/v.net.allpairs.html> <https://grass.osgeo.org/grass82/manuals/v.distance.html>

See Also

- `snap_to_network()` to snap the data points to the next stream segment within a given radius and/or a given flow accumulation threshold value.
- `snap_to_subc_segment()` to snap the data points to the next stream segment of the sub-catchment the data point is located.
- `get_distance_parallel()` to calculate the distance along the network in more than two basins in parallel.

Examples

```
# Download test data into the temporary R folder
# or define a different directory
my_directory <- tempdir()
download_test_data(my_directory)

# Load occurrence data
species_occurrence <- read.table(paste0(my_directory,
                                         "/hydrography90m_test_data/spdata_1264942.txt"),
                                header = TRUE)

basin_rast <- paste0(my_directory,
                    "/hydrography90m_test_data/basin_1264942.tif")

# Define full path to the sub-catchment raster layer
subc_rast <- paste0(my_directory,
                    "/hydrography90m_test_data/subcatchment_1264942.tif")

# Define full path to the vector file of the stream network
stream_vect <- paste0(my_directory,
                      "/hydrography90m_test_data/order_vect_59.gpkg")

# Automatically extract the basin and sub-catchment IDs and
# snap the data points to the stream segment
snapped_coordinates <- snap_to_subc_segment(data = species_occurrence,
                                           lon = "longitude",
                                           lat = "latitude",
                                           id = "occurrence_id",
                                           basin_layer = basin_rast,
                                           subc_layer = subc_rast,
                                           stream_layer = stream_vect,
                                           n_cores = 2)

# Show head of output table
head(snapped_coordinates)

# Get the euclidean distance and the distance along the network between all
# pairs of points
```

```

distance_table <- get_distance(data = snapped_coordinates,
                              lon = "lon_snap",
                              lat = "lat_snap",
                              id = "occurrence_id",
                              stream_layer = stream_vect,
                              distance = "network")

# Show table
distance_table

```

get_distance_graph	<i>Get the distance in meters between stream segments along a network graph</i>
--------------------	---

Description

Given a set of input sub-catchment IDs, the function will calculate the network distance in meters between all input pairs. Alternatively, only the number of stream segments (sub-catchment) along the paths are reported. Note that the stream segment and sub-catchment IDs are identical, and for consistency, we use the term "subc_id".

See the example code under "Note" that explains how to use standard igraph functions to obtain the actual stream segment IDs along the path. The function `read_geopackage()` can be used to create the input network graph, and the function `get_catchment_graph()` can be used to subset a network graph. Note that graph-based function includes also the entire length of the "from" and "to" stream segment, opposed to the `get_distance()` which, depending on the snapping method, includes only the part of the "from" and "to" segment where the points are located, resulting in minor differences.

Usage

```

get_distance_graph(
  g,
  subc_id = NULL,
  variable = "length",
  distance_m = TRUE,
  max_size = 1500
)

```

Arguments

g	igraph object. A directed graph.
subc_id	numeric. Vector of a single or multiple IDs, e.g (c(ID1, ID2, ID3, ...)). The sub-catchment (equivalent to stream segment) IDs for calculating the distances. Note that you can browse the entire network online at https://geo.igb-berlin.de/maps/351/view and to the left hand side, select the "Stream segment ID" layer and click on the map to get the ID.
variable	character. Specify the attribute / column name in the graph object that should be cumulated along the network path. Default is "length" to be used with the Hydrography90m dataset. Not needed when using "distance_m = FALSE".

distance_m	logical. If TRUE, and in case of the Hydrography90m dataset, the length (in meters) of each network segment along the path will be cumulated and the total length between all pairs will be reported in data.table. If FALSE, only the number of segments that are traversed through will be reported in the output matrix. If the subc_ids of the actual path is needed, please see the example code under "Note". Default is TRUE.
max_size	numeric. Specifies the maximum size of the data passed to the parallel back-end in MB. Default is 1500 (1.5 GB). Consider a higher value for large study areas (more than one 20°x20° tile). Optional.

Value

A data.table that reports either the distance or the number of segments between sub_ids.

Note

For getting the actual IDs of the path between two sub-catchments, you can use the igraph function "all_shortest_paths":

Specify the subc_ids:

```
from_subc_id = 513866854
```

```
to_subc_id = 513867238
```

```
subc_path <- all_shortest_paths(graph = my_graph, from = as.character(from_subc_id), to = as.character(to_subc_id),
mode = "all")
```

Extract only the subc_ids from the output:

```
subc_path <- as.numeric(as_ids(subc_path$res[1]))
```

We can then attach environmental data to the sub-catchments along the path, using the functions read_geopackage or extract_zonal_statistics

Author(s)

Sami Domisch

References

Csardi G, Nepusz T: The igraph software package for complex network research, InterJournal, Complex Systems 1695. 2006. <https://igraph.org>

See Also

- [read_geopackage\(\)](#) to create a network graph.
- [get_catchment_graph\(\)](#) to subset a network graph.


```

                                variable = "length",
                                distance_m = FALSE)
number_segments

```

get_distance_parallel *Calculate euclidean or along the network distance between points*

Description

Calculate euclidean or along-the-network distance (in meters) between points. To calculate the distance along the network, point coordinates need to be snapped to the stream network using the function [snap_to_network\(\)](#) or [snap_to_subc_segment\(\)](#).

Usage

```

get_distance_parallel(
  data,
  lon,
  lat,
  id,
  basin_id = NULL,
  basin_layer = NULL,
  stream_layer = NULL,
  distance = "both",
  n_cores = 1,
  quiet = TRUE
)

```

Arguments

data	a data.frame or data.table that contains the columns regarding the longitude / latitude coordinates in WGS84.
lon	character. The name of the column with the longitude coordinates.
lat	character. The name of the column with the latitude coordinates.
id	character. The name of a column containing unique IDs for each row of "data" (e.g., occurrence or site IDs). The unique IDs need to be numeric and less than 10 characters long.
basin_id	character. The name of the column with the basin IDs. If NULL and distance is set to 'network' or 'both', the basin IDs will be extracted automatically. Default is NULL.
basin_layer	character. Full path to the basin ID .tif layer. Needs to be defined to calculate the distance along the network.
stream_layer	character. Full path of the stream network .gpkg file. Needs to be defined to calculate the distance along the network.

distance	character. One of "euclidean", "network", or "both". If "euclidean", the euclidean distances between all pairs of points are calculated. If "network", the shortest path along the network between all pairs of points is calculated. (see "Details" for more information). If method is set to "both", both distance measures are calculated. Distances are given in meters. Default is "both".
n_cores	numeric. Number of cores used for parallelisation. Default is 1.
quiet	logical. If FALSE, the standard output will be printed. Default is TRUE.

Details

To calculate the euclidian distance between all pairs of points the function uses the `v.distance` command of GRASS GIS, which has been set up to produce a square matrix of distances. The calculation of distances along the stream network has been implemented with the command `v.net.allpairs` of GRASS GIS. The along-the-network distance calculation is done for all pairs of points located within the same basin. If the points are located in different basins, the function can be run in parallel (i.e., each core for the distance calculations of all points within one basin). The distance between points located in different basins is zero because they are not connected through the network.

Value

If `distance='euclidean'`, a distance matrix, in meters, of the euclidean distances between all the pairs of points (object of class `data.frame`) is returned. If `distance='network'`, a `data.frame` with three columns: `from_id`, `to_id`, `dist` is returned. The 'dist' column includes the distance, in meters, of the shortest path along the network from the point "from_id" to the point "to_id". If `distance='both'`, a list containing both objects is returned.

Author(s)

Afroditi Grigoropoulou, Marlene Schürz, Jaime Garcia Marquez

References

<https://grass.osgeo.org/grass82/manuals/v.net.allpairs.html> <https://grass.osgeo.org/grass82/manuals/v.distance.html>

See Also

- `get_distance()` to calculate the distance along the network in points located in only one basin.
- `snap_to_network()` to snap the data points to the next stream segment within a given radius and/or a given flow accumulation threshold value.
- `snap_to_subc_segment()` to snap the data points to the next stream segment of the sub-catchment the data point is located.
- `extract_ids()` to extract basin and sub-catchment IDs.

Examples

```

# Download test data into the temporary R folder
# or define a different directory
my_directory <- tempdir()
download_test_data(my_directory)

# Load occurrence data
species_occurrence <- read.table(paste0(my_directory,
                                          "/hydrography90m_test_data/spdata_1264942.txt"),
                                header = TRUE)

basin_rast <- paste0(my_directory,
                    "/hydrography90m_test_data/basin_1264942.tif")

# Define full path to the sub-catchment raster layer
subc_rast <- paste0(my_directory,
                   "/hydrography90m_test_data/subcatchment_1264942.tif")

# Define full path to the vector file of the stream network
stream_vect <- paste0(my_directory,
                     "/hydrography90m_test_data/order_vect_59.gpkg")

# Automatically extract the basin and sub-catchment IDs and
# snap the data points to the stream segment
snapped_coordinates <- snap_to_subc_segment(data = species_occurrence,
                                           lon = "longitude",
                                           lat = "latitude",
                                           id = "occurrence_id",
                                           basin_layer = basin_rast,
                                           subc_layer = subc_rast,
                                           stream_layer = stream_vect,
                                           n_cores = 2)

# Show head of output table
head(snapped_coordinates)

# Get the euclidean distance and the distance along the network between all
# pairs of points
distance_table <- get_distance_parallel(data = snapped_coordinates,
                                       lon = "lon_snap",
                                       lat = "lat_snap",
                                       id = "occurrence_id",
                                       basin_id = "basin_id",
                                       basin_layer = basin_rast,
                                       stream_layer = stream_vect,
                                       distance = "network")

# Show table
distance_table

```

get_pfafstetter_basins

Get Pfafstetter sub-basins

Description

Subset a basin or catchment into up to nine smaller sub-basins following the Pfafstetter basin delineation scheme. The function takes a network graph as the input and splits it into smaller sub-basins following a hierarchical topological coding scheme (see Verdin & Verdin (1999) for details), using the flow accumulation as the basis. The user has to define the sub-catchment (stream segment) ID that serves as the outlet of the basin. Note that this can be any stream segment that has an upstream catchment. The input graph can be created with [read_geopackage\(\)](#) and [get_catchment_graph\(\)](#).

Usage

```
get_pfafstetter_basins(
  g,
  subc_raster,
  out_dir,
  file_name,
  data_table = FALSE,
  n_cores = NULL
)
```

Arguments

<code>g</code>	igraph object. A directed graph of a basin with one outlet. The outlet can be any stream / sub-catchment for which the upstream basin should be split into smaller sub-basins. The input graph can be created with read_geopackage() and get_catchment_graph() .
<code>subc_raster</code>	character. Full path to the sub-catchment raster file of the basin. Does not need to be cropped / masked to the basin, but the IDs of the sub-catchments need to match with those in the input graph.
<code>out_dir</code>	character. The path of the output directory where the Pfafstetter raster layer will be written. Only needed when <code>data.table=FALSE</code> .
<code>file_name</code>	character. The filename and extension of the Pfafstetter raster layer (e.g. 'pfafstetter_raster.tif'). Only needed when <code>data.table=FALSE</code> .
<code>data_table</code>	Logical. If TRUE, then the result will be loaded into R as a 2-column data.table (sub-catchment ID and Pfafstetter code). If FALSE, the result is loaded as a raster (terra object) in R and written to disk. Default is FALSE.
<code>n_cores</code>	numeric. Number of cores used for parallelisation. Default is NULL (= <code>detectCores(logical=FALSE)-1</code>). Optional.

Value

Either a data.table, or a raster (terra object) loaded into R. In case the result is a raster, then a .tif file is written to disk.

Note

You can use the online map at <https://geo.igb-berlin.de/maps/351/view> to identify an ID of a stream segment (use the "Stream segment ID" layer to the left)

Author(s)

Sami Domisch

References

Verdin, K.L. & Verdin, J.P. (1999). A topological system for delineation and codification of the Earth's river basins. *Journal of Hydrology*, 218(1-2), 1-12. doi:10.1016/s0022-1694(99)00011-6

See Also

[read_geopackage\(\)](#) and [get_catchment_graph.\(\)](#) to create the input graph.

Examples

```
# Download test data into the temporary R folder
# or define a different directory
my_directory <- tempdir()
download_test_data(my_directory)

# Import the stream network as a graph
# Load stream network as a graph
my_graph <- read_geopackage(gpkg = paste0(my_directory,
                                           "/hydrography90m_test_data",
                                           "/order_vect_59.gpkg"),
                           import_as = "graph")

# Subset the graph such that it contains only one basin. You can use
# a random ID, i.e. it does not need to be the real outlet of the basin.
g_subset <- get_catchment_graph(g = my_graph,
                               subc_id = "513867227",
                               outlet = FALSE,
                               mode = "in",
                               as_graph = TRUE)

# Specify the sub-catchment raster file
subc_raster <- paste0(my_directory, "/hydrography90m_test_data",
                     "/subcatchment_1264942.tif")

# Specify the output directory
out_dir <- my_directory

# Calculate the Pfafstetter sub-basins and write the raster layer to disk (
# and import into R)
pfafstetter <- get_pfafstetter_basins(g = g_subset ,
                                     subc_raster = subc_raster,
                                     out_dir = out_dir,
                                     file_name = "pfafstetter_raster.tif",
                                     data_table = FALSE,
                                     n_cores = 4)
```

get_regional_unit_id *Get Hydrography90m regional unit IDs*

Description

Given the coordinates of input points (in WGS84), the function identifies the IDs of the regional units of the Hydrography90m in which the points are located. Input is a point data frame. The regional units refer to non-interrupted basins (as opposed to the 20°x20° tiles). These IDs can then be used to download the Hydrography90m regional unit raster mask(s) using [download_tiles\(\)](#).

Usage

```
get_regional_unit_id(data, lon, lat, quiet = TRUE)
```

Arguments

data	a data.frame or data.table that contains the columns regarding the longitude / latitude coordinates in WGS84.
lon	character. The name of the column with the longitude coordinates.
lat	character. The name of the column with the latitude coordinates.
quiet	logical. If FALSE, the standard output will be printed. Default is TRUE.

Author(s)

Afroditi Grigoropoulou

References

<https://gdal.org/programs/gdallocationinfo.html>

Examples

```
# Download test data into the temporary R folder
# or define a different directory
my_directory <- tempdir()
download_test_data(my_directory)

# Read the species data
species_occurrence <- read.table(paste0(my_directory,
                                         "/hydrography90m_test_data",
                                         "/spdata_1264942.txt"),
                                header = TRUE)

# Get the regional unit ID
get_regional_unit_id(species_occurrence, lon = "longitude",
                     lat = "latitude")
```

get_segment_neighbours

Get stream segment neighbours

Description

For each input stream segment, the function reports those upstream, downstream, or up-and down-stream segments that are connected to the input segment(s) within a specified neighbour order, with the option to summarize attributes across these segments. Note that the stream segment and sub-catchment IDs are identical, and for consistency, we use the term "subc_id". The input graph can be created with [read_geopackage\(\)](#) and [get_catchment_graph\(\)](#).

This function can be used to obtain the neighbour stream segments / sub-catchments for spatially explicit models or for spatial prioritization analyses (e.g. Marxan/Gurobi). By selecting a wider radius, the spatial dependency of the neighbouring segments / sub-catchments can be increased.

Usage

```
get_segment_neighbours(
  g,
  subc_id = NULL,
  var_layer = NULL,
  stat = NULL,
  attach_only = FALSE,
  order = 5,
  mode = "in",
  n_cores = 1,
  max_size = 1500
)
```

Arguments

<code>g</code>	igraph object. A directed graph.
<code>subc_id</code>	numeric vector of the input sub-catchment IDs (=stream segment IDs) for which to search the connected segments.
<code>var_layer</code>	character vector. One or more attributes (variable layers) of the input graph that should be reported for each output segment_id ("to_stream"). Optional. Default is NULL.
<code>stat</code>	one of the functions mean, median, min, max, sd (without quotes). Aggregates (or summarizes) the variables for the neighbourhood of each input segment (e.g., the average land cover in the next five upstream segments or sub-catchments). Default is NULL.
<code>attach_only</code>	logical. If TRUE, the selected variables will be only attached to each segment without any further aggregation. Default is FALSE.


```

import_as = "graph")

# Subset the graph and get a smaller catchment
my_graph <- get_catchment_graph(g = my_graph, subc_id = 513866048, mode = "in",
                              outlet = FALSE, as_graph = TRUE, n_cores = 1)

# Get a vector of all segment IDs
library(igraph)
subc_id <- as_ids(V(my_graph))

# Get all (up-and downstream) directly adjacent neighbours of each segment
get_segment_neighbours(g = my_graph, subc_id = subc_id,
                      order = 1, mode = "all")

# Get the upstream segment neighbours in the 5th order
# and report the length and source elevation
# for the neighbours of each input segment
get_segment_neighbours(g = my_graph, subc_id = subc_id,
                      order = 5, mode = "in", n_cores = 1,
                      var_layer = c("length", "source_elev"),
                      attach_only = TRUE)

# Get the downstream segment neighbours in the 5th order
# and calculate the median length and source elevation
# across the neighbours of each input segment
get_segment_neighbours(g = my_graph, subc_id = subc_id,
                      order = 2, mode = "out", n_cores = 1,
                      var_layer = c("length", "source_elev"),
                      stat = median)

# Get the up-and downstream segment neighbours in the 5th order
# and report the median length and source elevation
# for the neighbours of each input segment
get_segment_neighbours(g = my_graph, subc_id = subc_id, order = 2,
                      mode = "all", n_cores = 1,
                      var_layer = c("length", "source_elev"),
                      stat = mean, attach_only = TRUE)

```

get_tile_id

Get the Hydrography90m 20°x20° tile ID

Description

Identifies the 20°x20° tile IDs of the Hydrography90m data in which the input points are located. The IDs can then be used to download the data using [download_tiles\(\)](#). The input is a data frame with point coordinates. For orientation, please also see the tiles at the https://hydrography.org/hydrography90m/hydrography90m_layers

Usage

```
get_tile_id(data, lon, lat)
```

Arguments

data	a data.frame or data.table that contains the columns regarding the longitude / latitude coordinates in WGS84.
lon	character. The name of the column with the longitude coordinates.
lat	character. The name of the column with the latitude coordinates.

Author(s)

Afroditi Grigoropoulou

Examples

```
# Download test data into the temporary R folder
# or define a different directory
my_directory <- tempdir()
download_test_data(my_directory)

# Load species occurrence data
species_occurrence <- read.table(paste0(my_directory,
                                         "/hydrography90m_test_data",
                                         "/spdata_1264942.txt"),
                                header = TRUE)

# Get the tile ID
get_tile_id(data = species_occurrence,
            lon = "longitude", lat = "latitude")
```

```
get_upstream_catchment
```

Delineate the upstream catchment

Description

Delineates the upstream catchment of a given point, where the point is considered as the outlet of the catchment.

Usage

```
get_upstream_catchment(
  data,
  id,
  lon,
  lat,
```

```

    direction_layer = NULL,
    out_dir = NULL,
    n_cores = NULL,
    compression = "low",
    bigtiff = TRUE,
    quiet = TRUE
)

```

Arguments

data	a data.frame or data.table that contains the columns regarding the longitude / latitude coordinates in WGS84. Note that the points need to be snapped to the stream network with snap_to_network() or snap_to_subc_segment() .
id	character. The name of a column containing unique IDs for each row of "data" (e.g., occurrence or site IDs). The unique IDs need to be numeric and less than 10 characters long.
lon	character. The name of the column with the longitude coordinates.
lat	character. The name of the column with the latitude coordinates.
direction_layer	character. Full path to the flow direction raster file.
out_dir	Full path to the directory where the output(s) will be stored. The output file name includes the "id" which helps identifying the upstream corresponding catchment.
n_cores	numeric. Number of cores used for parallelisation. If NULL, available cores - 1 will be used. Default is NULL.
compression	character. Compression of the written output file. Compression levels can be defined as "none", "low", or "high". Default is "low", referring to compression type "DEFLATE" and compression level 2. "high" refers to compression level 9.
bigtiff	logical. Define whether the output file is expected to be a BIGTIFF (file size larger than 4 GB). If FALSE and size > 4GB no file will be written. Default is TRUE.
quiet	logical. If FALSE, the standard output will be printed. Default is TRUE.

Author(s)

Jaime Garcia Marquez, Afroditi Grigoropoulou, Marlene Schürz

References

- <https://grass.osgeo.org/grass82/manuals/r.water.outlet.html>
- <https://grass.osgeo.org/grass82/manuals/r.region.html>

See Also

- [snap_to_network](#) to snap the data points to the next stream segment within a given radius and/or a given flow accumulation threshold value.

- `snap_to_subc_segment` to snap the data points to the next stream segment within the sub-catchment the point is located.
- `extract_ids` to extract basin and sub-catchment IDs.

Examples

```
# Download test data into temporary R folder
# or define a different directory
my_directory <- tempdir()
download_test_data(my_directory)

# Before running the function get_upstream_catchment(), snap the points to
# to the stream segment. There are multiple ways to snap the points. Here is
# one example:

# Load occurrence data
species_occurrence <- read.table(paste0(my_directory,
                                         "/hydrography90m_test_data",
                                         "/spdata_1264942.txt"),
                                header = TRUE)

# Define full path to the basin and sub-catchments raster layer
basin_raster <- paste0(my_directory,
                      "/hydrography90m_test_data/basin_1264942.tif")
subc_raster <- paste0(my_directory,
                     "/hydrography90m_test_data/subcatchment_1264942.tif")

# Define full path to the vector file of the stream network
stream_vector <- paste0(my_directory,
                       "/hydrography90m_test_data/order_vect_59.gpkg")

# Automatically extract the basin and sub-catchment IDs and
# snap the data points to the stream segment
snapped_coordinates <- snap_to_subc_segment(data = species_occurrence,
                                           lon = "longitude",
                                           lat = "latitude",
                                           id = "occurrence_id",
                                           basin_layer = basin_raster,
                                           subc_layer = subc_raster,
                                           stream_layer = stream_vector,
                                           n_cores = 2)

# Define full path to the direction .tif
direction_raster <- paste0(my_directory,
                          "/hydrography90m_test_data/direction_1264942.tif")

# Define the path for the output file(s)
output_folder <- paste0(my_directory, "/upstream_catchments")
if(!dir.exists(output_folder)) dir.create(output_folder)
# Get the upstream catchment for each point location
get_upstream_catchment(snapped_coordinates,
                      lon = "lon_snap",
                      lat = "lat_snap",
```

```

id = "occurrence_id",
direction_layer = direction_raster,
out_dir = output_folder,
n_cores = 2)

```

get_upstream_variable *Calculate upstream variables for each sub-catchment*

Description

For each input sub-catchment, the function calculates the upstream mean, median, min, max, sd or sum for one or more variables. Note that the stream segment and sub-catchment IDs are identical. The input graph can be created with [read_geopackage\(\)](#) and [get_catchment_graph\(\)](#).

This function can be used to obtain all upstream stream segments / sub-catchments for species distribution modelling, or for spatial prioritization analyses (e.g. Marxan/Gurobi) to specify the connectivity (and in this case the "length" attribute can be used).

The function accepts as an input a graph with one or more outlets (e.g. an entire tile with many drainage basins). The variable that should be aggregated upstream should be prepared as a data.table, e.g. with [extract_zonal_stat\(\)](#).

Usage

```

get_upstream_variable(
  g,
  variable_table = NULL,
  subc_id = NULL,
  var_layer = NULL,
  upstream_stat = NULL,
  include_focal = FALSE,
  save_up_conn = NULL,
  load_up_conn = NULL,
  n_cores = 1,
  max_size = 3000
)

```

Arguments

<code>g</code>	igraph object. A directed graph.
<code>variable_table</code>	a data.table that includes the stream column (corresponding to the subc_id) as well as the attributes that should be aggregated across the upstream network subc_id. Default is NULL.
<code>subc_id</code>	A vector of subc_id for which the upstream variables should be calculated. Optional; default is to use all subc_id of the input graph.
<code>var_layer</code>	character vector. One or more attributes (variable layers) of the variable_table should be reported for each output subc_id. Default is NULL.

<code>upstream_stat</code>	one of the functions mean, median, min, max, sd, sum (without quotes). The function will be used to aggregate (or summarize) the upstream variables for each <code>subc_id</code> (e.g., the average land cover across the entire upstream area). Default is NULL.
<code>include_focal</code>	Whether the focal <code>subc_id</code> should be included in the aggregation with <code>include_focal = TRUE</code> which is the default. Set to <code>FALSE</code> if the focal <code>subc_id</code> should not be included in the upstream aggregation of the given sub-catchment.
<code>save_up_conn</code>	character. Provide a name of the .RData file that will be written to disk (to <code>getwd()</code>), and which includes the intermediate result consisting of a <code>data.table</code> that includes all upstream connections for each <code>subc_id</code> . Useful for large study areas as it avoids re-running the possibly time-consuming pre-processing to obtain other metrics (e.g. mean, sum) or other variables. The <code>data.table</code> is called <code>upstream_dt</code> and has the columns <code>stream</code> and <code>base</code> . Default is <code>FALSE</code> .
<code>load_up_conn</code>	Optional, and if used, it should be <code>upstream_dt</code> . In case the file with the upstream connections was previously written to disk (using e.g. <code>save_up_conn = "my_file"</code>), then this <code>data.table</code> can be first loaded with <code>load(paste0(getwd(), "/my_file.RData"))</code> , which loads the <code>upstream_dt</code> <code>data.table</code> with the columns <code>c("stream", "base")</code> . Use <code>load_up_conn = upstream_dt</code> to then skip the pre-processing. Optional, default is NULL.
<code>n_cores</code>	numeric. Number of cores used for parallelisation. In case the graph is very large, and many segments are used as an input, setting <code>n_cores</code> to a higher value can speed up the computation. Note however that the parallelisation process requires copying the input graph to each core. This may result in possible RAM limitations and even slower processing. Hence consider testing first with a lower number of cores. Default is <code>n_cores = 1</code> . Optional.
<code>max_size</code>	numeric. Specifies the maximum size of the data passed to the parallel back-end in MB. Default is 1500 (1.5 GB). Consider a higher value for large study areas (more than one 20°x20° tile). Optional.

Value

A `data.table` indicating the "stream" (=subc_id) and the upstream variables which have the same column names as the `var_layer` argument. The intermediate output can be saved to disk (requires setting `save_up_conn = "my_file"`).

Author(s)

Sami Domisch

References

Csardi G, Nepusz T: The igraph software package for complex network research, InterJournal, Complex Systems 1695. 2006. <https://igraph.org>

See Also

`read_geopackage()` and `get_catchment_graph()` to create the input graph. Alternatively, see `get_segment_neighbours()` to obtain the upstream variables for a specified neighbourhood.


```

        upstream_stat=sum,
        include_focal = TRUE,
        n_cores = 4,
        save_up_conn = "my_file")

## Alternatively, load the previously generated upstream connections
## and use it directly, skipping the pre-processing
load(paste0(getwd(), "/my_file.RData"))

result <- get_upstream_variable(my_graph,
                               variable_table = variable_table,
                               var_layer = var_layer,
                               upstream_stat=max,
                               include_focal = TRUE,
                               load_up_conn = upstream_dt)

## Map the new variable across the network

## Specify tif-layer for reclassification
subc_raster <- paste0(my_directory, "/hydrography90m_test_data/subcatchment_1264942.tif")
recl_raster <- paste0(my_directory, "/upstream_sum.tif")

## Set columns as integer
result <- result[, names(result) := lapply(.SD, as.integer)]

### Create raster - select the "to" column which represents the unique subc_id
r <- reclass_raster(data = result,
                    rast_val = "stream",
                    new_val = "flow_accum",
                    raster_layer = subc_raster,
                    recl_layer = recl_raster,
                    read = TRUE)

## Plot the map
terra::plot(r, background = "grey")

```

merge_tiles

Merge raster or vector objects

Description

Merge multiple raster or spatial vector objects from disk to form a new raster or spatial vector object with a larger spatial extent. A directory with at least two raster .tif or spatial vector geopackage files should be provided. Depending on the input, the output is a .tif or a .gpkg file (saved under out_dir). If read = TRUE, the output is read into R as a SpatRaster (terra package) object in case of .tif files, or as a SpatVector (terra package) object in case of .gpkg files.

Usage

```
merge_tiles(
  tile_dir,
  tile_names,
  out_dir,
  file_name,
  name = "stream",
  compression = "low",
  bigtiff = TRUE,
  read = FALSE,
  quiet = TRUE
)
```

Arguments

tile_dir	character. The directory containing the raster or spatial vectors tiles, which should be merged.
tile_names	character. The names of the files to be merged, including the file extension (.tif or .gpkg).
out_dir	character. The directory where the output will be stored.
file_name	character. Name of the merged output file, including the file extension (.tif or .gpkg).
name	character. The attribute table column name of the stream segment ("stream"), sub-catchment ("ID"), basin ("ID") or outlet ("ID") column which is used for merging GeoPackages. Default is "stream".
compression	character. Compression of the written output file. Compression levels can be defined as "none", "low", or "high". Default is "low", referring to compression type "DEFLATE" and compression level 2. "high" refers to compression level 9.
bigtiff	logical. Define whether the output file is expected to be a BIGTIFF (file size larger than 4 GB). If FALSE and size > 4GB no file will be written. Default is TRUE.
read	logical. If TRUE, the merged layer gets read into R. If FALSE, the layer is only stored on disk. Default is FALSE.
quiet	logical. If FALSE, the standard output will be printed. Default is TRUE.

Value

A .tif raster file or .gpkg spatial vector object that is always written to disk, and optionally loaded into R.

Author(s)

Thomas Tomiczek, Jaime Garcia Marquez, Afroditi Grigoropoulou

References

<https://gdal.org/programs/gdalbuildvrt.html>
https://gdal.org/programs/gdal_translate.html
<https://gdal.org/programs/ogrmerge.html#ogrmerge>
<https://gdal.org/programs/ogr2ogr.html>

Examples

```

# Download tiles into the temporary R folder
# or define a different directory
my_directory <- tempdir()
download_tiles(variable = "basin",
               file_format = "tif",
               tile_id = c("h22v08", "h22v10"),
               download_dir = my_directory)

# Define folder containing only the tiles, which should be merged
tiles_folder <- paste0(my_directory, "/r.watershed/basin_tiles20d")
# Define output folder
output_folder <- paste0(my_directory, "/merged_tiles")
# Create output folder if it doesn't exist
if(!dir.exists(output_folder)) dir.create(output_folder)

# Merge tiles
merged_tiles <- merge_tiles(tile_dir = tiles_folder,
                           tile_names = c("basin_h22v08.tif", "basin_h22v10.tif"),
                           out_dir = output_folder,
                           file_name = "basin_merged.tif",
                           read = TRUE)

```

read_geopackage

Read a GeoPackage file

Description

Reads an entire, or a subset of a GeoPackage vector file from disk either as a table (data.table), as a directed graph object (igraph), a spatial dataframe (sf) or a SpatVect object (terra).

Usage

```

read_geopackage(
  gpkg,
  import_as = "data.table",
  layer_name = NULL,
  subc_id = NULL,
  name = "stream"
)

```

Arguments

gpkg	character. Full path of the GeoPackage file.
import_as	character. "data.table", "graph", "sf", or "SpatVect". "data.table" imports data as a data.table. "graph" imports the layer as a directed graph (igraph object). This option is only possible for a network layer (e.g. the stream network) and it needs to contain the attributes "stream" and "next_stream". "sf" imports the layer as a spatial data frame (sf object). "SpatVect" imports the layer as a SpatVector (terra object). Default is "data.table".
layer_name	character. Name of the specific data layer to import from the GeoPackage. A specific data layer only needs to be defined if the GeoPackage contains multiple layers. To see the available layers the function st_layers() from the R package 'sf' can be used. Optional. Default is NULL.
subc_id	numeric. Vector of the sub-catchment (or stream segment) IDs in the form of (c(ID1, ID2, ...)) for which the spatial objects or attributes of the GeoPackage should be imported. Optional. Default is NULL.
name	character. The attribute table column name of the stream segment ("stream"), sub-catchment ("ID"), basin ("ID") or outlet ("ID") column which is used for subsetting the GeoPackage prior importing. Optional. Default is "stream".

Author(s)

Sami Domisch, Marlene Schürz

Examples

```
# Download test data into the temporary R folder
# or define a different directory
my_directory <- tempdir()
download_test_data(my_directory)

# Read the stream network as a graph
my_graph <- read_geopackage(gpkg = paste0(my_directory,
                                           "/hydrography90m_test_data",
                                           "/order_vect_59.gpkg"),
                           import_as = "graph")

# Read the stream network as a data.table
my_dt <- read_geopackage(gpkg = paste0(my_directory,
                                        "/hydrography90m_test_data",
                                        "/order_vect_59.gpkg"))

# Read the stream network as a data.table for specific IDs
my_dt <- read_geopackage(gpkg = paste0(my_directory,
                                        "/hydrography90m_test_data",
                                        "/order_vect_59.gpkg"),
                        subc_id = c(513833203, 513833594))

# Read the sub-catchments as a SF-object
```

```

my_sf <- read_geopackage(gpkg = paste0(my_directory,
                                       "/hydrography90m_test_data",
                                       "/sub_catchment_59.gpkg"),
                        import_as = "sf",
                        layer_name = "sub_catchment")

# Read a subset of sub-catchments as a SF-object
my_sf <- read_geopackage(gpkg = paste0(my_directory,
                                       "/hydrography90m_test_data",
                                       "/sub_catchment_59.gpkg"),
                        import_as = "sf",
                        subc_id = c(513833203, 513833594),
                        name = "ID")

# Read the basin as SpatVect object
my_sv <- read_geopackage(gpkg = paste0(my_directory,
                                       "/hydrography90m_test_data",
                                       "/basin_59.gpkg"),
                        import_as = "SpatVect")

```

reclass_raster

Reclassify a raster layer

Description

Reclassifies a raster .tif layer based on a look-up table, such that the output raster contains the new values. The function uses the `r.reclass` function of GRASS GIS.

Note that the input raster needs to be of type integer. If the input raster layer has floating point values, you can multiply it by some factor (e.g. 1000) to achieve integer values, otherwise the GRASS GIS `r.reclass` will round the raster values down to the next integer which is not always desired.

Usage

```

reclass_raster(
  data,
  rast_val,
  new_val,
  raster_layer,
  recl_layer,
  no_data = -9999,
  type = "Int32",
  compression = "low",
  bigtiff = TRUE,
  read = FALSE,
  quiet = TRUE
)

```

Arguments

<code>data</code>	a data.frame or data.table with the original and new values to be written to the raster.
<code>rast_val</code>	character. The name of the column with the original raster values.
<code>new_val</code>	character. The name of the column with the new raster values, which need to be integer values. In case of floating point values, consider multiplying the values e.g. by 1000 to keep three decimals.
<code>raster_layer</code>	Full path to the input raster .tif layer.
<code>recl_layer</code>	character. Full path of the output .tif layer, i.e., the reclassified raster file.
<code>no_data</code>	numeric. The no_data value of the new .tif layer. Default is -9999.
<code>type</code>	character. Data type; Options are Byte, Int16, UInt16, Int32, UInt32, CInt16, CInt32. Default is Int32. Note that only integer values are allowed.
<code>compression</code>	character. Compression of the written output file. Compression levels can be defined as "none", "low", or "high". Default is "low", referring to compression type "DEFLATE" and compression level 2. "high" refers to compression level 9.
<code>bigtiff</code>	logical. Define whether the output file is expected to be a BIGTIFF (file size larger than 4 GB). If FALSE and size > 4GB no file will be written. Default is TRUE.
<code>read</code>	logical. If TRUE, then the reclassified raster .tif layer gets read into R as a SpatRaster (terra object). If FALSE, the layer is only stored on disk. Default is FALSE.
<code>quiet</code>	logical. If FALSE, the standard output will be printed. Default is TRUE.

Author(s)

Marlene Schürz

References

<https://grass.osgeo.org/grass82/manuals/r.reclass.html>

Examples

```
# Download test data into the temporary R folder
# or define a different directory
my_directory <- tempdir()
download_test_data(my_directory)

# Read the stream order for each sub-catchment as a data.table
my_dt <- read_geopackage(gpkg= paste0(my_directory,
                                     "/hydrography90m_test_data",
                                     "/order_vect_59.gpkg"),
                        import_as = "data.table")

# Select the stream segment ID and and the Strahler stream order
```

```

str_ord <- my_dt[,c("stream", "strahler")]

# Define input and output raster layer
stream_raster <- paste0(my_directory,
                        "/hydrography90m_test_data/stream_1264942.tif")

recl_raster <- paste0(my_directory,
                     "/hydrography90m_test_data/reclassified_raster.tif")

# Reclassify the stream network to obtain the Strahler stream order raster
str_ord_rast <- reclass_raster(data = str_ord,
                              rast_val = "stream",
                              new_val = "strahler",
                              raster_layer = stream_raster,
                              recl_layer = recl_raster)

```

report_no_data	<i>Report NoData value</i>
----------------	----------------------------

Description

This function reports the defined NoData value of a raster layer. The NoData value of a raster layer represents the absence of data. In computations the NoData value can be treated in different ways. Either the NoData value will be reported or the Nodata value will be ignored and a value is computed from the available values of a specified location.

Usage

```
report_no_data(data_dir, var_layer, n_cores = NULL)
```

Arguments

data_dir	character. Path to the directory containing all input data.
var_layer	character vector of variable raster layers on disk, e.g. "slope_grad_dw_cel_h00v00.tif".
n_cores	numeric. Number of cores used for parallelization, in case multiple .tif files are provided to var_layer.

Author(s)

Afroditi Grigoropoulou, Marlene Schürz

References

<https://gdal.org/programs/gdalinfo.html>

Examples

```
# Download test data into the temporary R folder
# or define a different directory
my_directory <- tempdir()
download_test_data(my_directory)

# Report the NoData value
report_no_data(data_dir = paste0(my_directory, "/hydrography90m_test_data"),
               var_layer = c("subcatchment_1264942.tif", "flow_1264942.tif",
                             "spi_1264942.tif"),
               n_core = 2)
```

set_no_data	<i>Set no data value</i>
-------------	--------------------------

Description

Change or set the NoData value for a raster layer. The change happens in-place, meaning that the original file is overwritten on disk.

Usage

```
set_no_data(data_dir, var_layer, no_data, quiet = TRUE)
```

Arguments

- data_dir character. Path to the directory containing all input data.
- var_layer character vector of variable layers on disk, e.g. c("sti_h16v02.tif", "slope_grad_dw_cel_h00v00.tif"). The original files will be overwritten.
- no_data numeric. The desired NoData value.
- quiet logical. If FALSE, the standard output will be printed. Default is TRUE.

Author(s)

Afroditi Grigoropoulou, Marlene Schürz

References

https://gdal.org/programs/gdal_edit.html

Examples

```
# Download test data into the temporary R folder
# or define a different directory
my_directory <- tempdir()
download_test_data(my_directory)

# Define no data value
set_no_data(data_dir = paste0(my_directory, "/hydrography90m_test_data"),
            var_layer = "cti_1264942.tif",
            no_data = -9999)
```

snap_to_network	<i>Snap points to stream segment based on distance or flow accumulation</i>
-----------------	---

Description

Snap points to the next stream segment within a defined radius (in map pixels) or a minimum flow accumulation.

Usage

```
snap_to_network(
  data,
  lon,
  lat,
  id,
  stream_layer,
  accu_layer = NULL,
  method = "distance",
  distance = 500,
  accumulation = 0.5,
  quiet = TRUE
)
```

Arguments

data	a data.frame or data.table that contains the columns regarding the longitude / latitude coordinates in WGS84.
lon	character. The name of the column with the longitude coordinates.
lat	character. The name of the column with the latitude coordinates.
id	character. The name of a column containing unique IDs for each row of "data" (e.g., occurrence or site IDs). The unique IDs need to be numeric and less than 10 characters long.
stream_layer	character. Full path of the stream network .tif file

accu_layer	character. Full path of the flow accumulation .tif file. Needed if the point should be snapped to the next stream segment having an accumulation value higher than the flow accumulation threshold (set by 'accumulation'). This prevents points from being snapped to small stream tributaries. Optional. Default is NULL.
method	character. One of "distance", "accumulation", or "both". Defines if the points are snapped using the distance or flow accumulation (see "Details" for more information). If method is set to "both" the output will contain the new coordinates for both calculations. Default is "distance" (in map pixels).
distance	numeric. Maximum radius in map pixels. The points will be snapped to the next stream within this radius. Default is 500.
accumulation	numeric. Minimum flow accumulation. Points will be snapped to the next stream with a flow accumulation equal or higher than the given value. Default is 0.5.
quiet	logical. If FALSE, the standard output will be printed. Default is TRUE.

Details

The function makes use of the `r.stream.snap` function available in GRASS GIS to simultaneously snap a number of points to the stream network. A distance threshold can be specified and points will be snapped to any stream segment within this distance radius (in map pixels). However, to avoid snapping to small tributaries, an accumulation threshold can be used and the snapping occurs on stream segment with equal or higher accumulation threshold and within the given distance radius.

Value

Returns a data.frame with the snapped coordinates and the sub-catchment ID of the snapped stream segment. If the sub-catchment ID is NA, no stream segment was found within the given distance (method = "distance") or no stream segment was found within the given distance and a flow accumulation equal or higher than the given threshold (method = "accumulation"). "out-bbox" means that the provided coordinates are not within the extend (bounding box) of the provided stream network layer.

Note

Duplicated rows will be removed from the input data.

Author(s)

Marlene Schürz, Jaime Garcia Marquez

References

<https://grass.osgeo.org/grass78/manuals/addons/r.stream.snap.html>

Examples

```
# Download test data into the temporary R folder
# or define a different directory
my_directory <- tempdir()
```

```

download_test_data(my_directory)

# Load occurrence data
species_occurrence <- read.table(paste0(my_directory,
                                         "/hydrography90m_test_data/spdata_1264942.txt"),
                                header = TRUE)

# Define full path to stream network and flow accumulation
stream_raster <- paste0(my_directory,
                        "/hydrography90m_test_data/stream_1264942.tif")
flow_raster <- paste0(my_directory,
                      "/hydrography90m_test_data/flow_1264942.tif")

# To calculate the new (snapped) coordinates for a radius and a flow
snapped_coordinates <- snap_to_network(data = species_occurrence,
                                       lon = "longitude",
                                       lat = "latitude",
                                       id = "occurrence_id",
                                       stream_layer = stream_raster,
                                       accu_layer = flow_raster,
                                       method = "both",
                                       distance = 300,
                                       accumulation = 0.8)

# Show head of output table
head(snapped_coordinates)

```

snap_to_subc_segment *Snap points to stream segment within the sub-catchment*

Description

Move points to the stream segment within the sub-catchment where the point is located.

Usage

```

snap_to_subc_segment(
  data,
  lon,
  lat,
  id,
  basin_id = NULL,
  subc_id = NULL,
  basin_layer,
  subc_layer,
  stream_layer,
  n_cores = 1,
  quiet = TRUE
)

```

Arguments

<code>data</code>	a data.frame or data.table that contains the columns regarding the longitude / latitude coordinates in WGS84.
<code>lon</code>	character. The name of the column with the longitude coordinates.
<code>lat</code>	character. The name of the column with the latitude coordinates.
<code>id</code>	character. The name of a column containing unique IDs for each row of "data" (e.g., occurrence or site IDs). The unique IDs need to be numeric and less than 10 characters long.
<code>basin_id</code>	character. The name of the column with the basin IDs. If NULL, the basin IDs will be extracted automatically. Optional. Default is NULL
<code>subc_id</code>	character. The name of the column with the sub-catchment IDs. If NULL, the sub-catchment IDs will be extracted automatically. Optional. Default is NULL.
<code>basin_layer</code>	character. Full path to the basin ID .tif layer.
<code>subc_layer</code>	character. Full path to the sub-catchment ID .tif layer.
<code>stream_layer</code>	character. Full path of the stream network .gpkg file.
<code>n_cores</code>	numeric. Number of cores used for parallelisation. Default is 1.
<code>quiet</code>	logical. If FALSE, the standard output will be printed. Default is TRUE.

Details

The function uses the network module of GRASS GIS (v.net), to connect a vector line map (stream network) with a point map (occurrence/sampling points). After masking the stream segment and the sub-catchment where the target point is located, the connect operation snaps the point to the stream segment using a distance threshold. This threshold is automatically calculated as the longest distance between two points within the sub-catchment. In this way the snapping will always take place. From the new location, the function extracts the new snapped coordinates.

Value

A data.table of the original and new coordinates, along with the sub-catchment ID.

Author(s)

Jaime Garcia Marquez, Marlene Schürz

References

<https://grass.osgeo.org/grass82/manuals/v.net.html>

See Also

- `snap_to_network()` to snap the data points to the next stream segment within a given radius and/or a given flow accumulation threshold value.
- `extract_ids()` to extract basin and sub-catchment IDs.

Examples

```

# Download test data into the temporary R folder
# or define a different directory
my_directory <- tempdir()
download_test_data(my_directory)

# Load occurrence data
species_occurrence <- read.table(paste0(my_directory,
                                          "/hydrography90m_test_data/spdata_1264942.txt"),
                                header = TRUE)

basin_rast <- paste0(my_directory,
                    "/hydrography90m_test_data/basin_1264942.tif")
subc_rast <- paste0(my_directory,
                   "/hydrography90m_test_data/subcatchment_1264942.tif")

# Define full path to the vector file of the stream network
stream_vect <- paste0(my_directory,
                     "/hydrography90m_test_data/order_vect_59.gpkg")

hydrography90m_ids <- extract_ids(data = species_occurrence,
                                lon = "longitude",
                                lat = "latitude",
                                id = "occurrence_id",
                                subc_layer = subc_rast,
                                basin_layer = basin_rast)

# Snap data points to the stream segment of the provided sub-catchment ID
snapped_coordinates <- snap_to_subc_segment(data = hydrography90m_ids,
                                           lon = "longitude",
                                           lat = "latitude",
                                           id = "occurrence_id",
                                           basin_id = "basin_id",
                                           subc_id = "subcatchment_id",
                                           basin_layer = basin_rast,
                                           subc_layer = subc_rast,
                                           stream_layer = stream_vect,
                                           n_cores = 2)

# Show head of output table
head(snapped_coordinates)

# OR
# Automatically extract the basin and sub-catchment IDs and
# snap the data points to the stream segment
snapped_coordinates <- snap_to_subc_segment(data = species_occurrence,
                                           lon = "longitude",
                                           lat = "latitude",
                                           id = "occurrence_id",
                                           basin_layer = basin_rast,
                                           subc_layer = subc_rast,
                                           stream_layer = stream_vect,
                                           n_cores = 2)

# Show head of output table

```

snap_to_subc_segment

55

```
head(snapped_coordinates)
```

Index

1, [25](#)

crop_to_extent, [3](#)
crop_to_extent(), [14](#), [15](#)

download_test_data, [5](#)
download_tiles, [6](#)
download_tiles(), [32](#), [35](#)

extract_from_gpkg, [9](#)
extract_ids, [11](#), [38](#)
extract_ids(), [28](#), [53](#)
extract_zonal_stat, [13](#)
extract_zonal_stat(), [39](#)

get_all_upstream_distances, [16](#)
get_catchment_graph, [17](#)
get_catchment_graph(), [17](#), [24](#), [25](#), [30](#), [33](#),
[34](#), [39](#), [40](#)
get_catchment_graph. (), [31](#), [34](#)
get_centrality, [20](#)
get_distance, [21](#)
get_distance(), [24](#), [28](#)
get_distance_graph, [24](#)
get_distance_parallel, [27](#)
get_distance_parallel(), [22](#), [23](#)
get_pfafstetter_basins, [29](#)
get_regional_unit_id, [32](#)
get_segment_neighbours, [33](#)
get_segment_neighbours(), [17](#), [40](#)
get_tile_id, [35](#)
get_upstream_catchment, [36](#)
get_upstream_variable, [39](#)
get_upstream_variable(), [17](#)

merge_tiles, [42](#)

read_geopackage, [44](#)
read_geopackage(), [17–21](#), [24](#), [25](#), [30](#), [31](#),
[33](#), [34](#), [39](#), [40](#)
reclass_raster, [46](#)

report_no_data, [48](#)
report_no_data(), [15](#)

set_no_data, [49](#)
set_no_data(), [15](#)
snap_to_network, [37](#), [50](#)
snap_to_network(), [21](#), [23](#), [27](#), [28](#), [37](#), [53](#)
snap_to_subc_segment, [38](#), [52](#)
snap_to_subc_segment(), [21](#), [23](#), [27](#), [28](#), [37](#)