Package 'hydrographr'

January 30, 2023

Type Package

Title Scalable Hydrographic Data Processing in R

Date 2023

Version 1.0.2

Maintainer Maria Üblacker <maria.ueblacker@igb-berlin.d>

Description A collection of R function wrappers for GDAL and GRASS-GIS functions to efficiently work with Hydrography90m spatial data.

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)

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_catchment_graph.R'
'get_regional_unit_id.R'
'get_segment_neighbours.R'
'get_tile_id.R'
'get_upstream_catchment.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'

R topics documented:

`crop_to_extent` *Crop raster to extent*

Description

This function crops an input raster layer directly on disk, i.e. the input layer does not need to be loaded into R. The raster .tif is cropped to a polygon border line if a vector layer (cutline source) is provided, otherwise if a bounding box is provided (xmin, ymin, xmax, ymax coordinates or a spatial object from which to extract a bounding box), the raster is cropped to the extent of the bounding box. At least a cutline source (vector_layer) or a bounding box (bounding_box) should be provided. 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,
  read = TRUE,
  quiet = TRUE
)
```

Arguments

`raster_layer`	character. Full path to the input raster .tif layer.
`vector_layer`	character. Full path to a vector layer that is used as a cutline data source (similar to a mask operation).
`bounding_box`	numeric vector of the coordinates of the corners of a bounding box (xmin, ymin, xmax, ymax), SpatRaster, SpatVector, or other spatial object.
`out_dir`	character. The directory where the output will be stored.
`file_name`	character. Name of the cropped output raster .tif file.
`read`	logical. If TRUE, the cropped raster .tif layer gets read into R. If FALSE, the layer is only stored on disk. Default is TRUE.
`quiet`	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 ans 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` *Download test data*

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 is available at https://drive.google.com/file/d/1kYNWXmtVm6X7MZLISOePGpvxB1pk1scD/view?usp=share_
and can be automatically downloaded and unzipped with this function to a desired path.

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 for 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` *Download files of the Hydrography90m dataset*

Description

The function downloads files of the Hydrography90m dataset, available at https://public.igb-berlin.de/index.php/s/agciopgzXj
The files will be stored in the folder architecture of the above domain. Multiple regular tile or regional unit files can be requested in a single call of the function. The tile or regional unit IDs can be obtained using the functions "get_tile_id" and "get_regional_unit_id" respectively.

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.
`file_format`	character. Format of the requested file ("tif" or "gpkg").
`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.

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)
```

---

extract_from_gpkg *Extract values from the stream order .gpkg files.*

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

data_dir	character. Path to the directory containing all input data.
subc_id	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 extract_ids() function, by sub-setting the resulting data.frame.
subc_layer	character. Full path to the sub-catchment ID .tif layer
var_layer	character vector of .gpkg files on disk, e.g. "order_vect_point_h18v04.gpkg".
out_dir	character. The directory where the output will be stored. If the out_dir 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.
file_name	character. Name of the .csv file where the output table will be stored. out_dir should also be specified for this purpose.
n_cores	numeric. Number of cores used for parallelization, in case multiple .gpkg files are provided to var_layer. If NULL, available cores - 1 will be used.
quiet	logical. If FALSE, the standard output will be printed. Default is TRUE.

Author(s)

Afroditi Grigoropoulou, Jaime Garcia Marquez, Maria M. Üblacker

References

https://grass.osgeo.org/grass82/manuals/v.in.ogr.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 a given point location.

Usage

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

Arguments

`data`	a data.frame or data.table with lat/lon 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 basin ID .tif layer.
`subc_layer`	character. Full path to the sub-catchment ID .tif layer.
`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 of the WGS84 reference system. The function can also be used to extract any value from a given raster layer with a WGS84 projection, such as e.g. environmental information that is stored in the input raster file.

Note

Duplicated rows will be removed.

Author(s)

Afroditi Grigoropoulou, Maria Üblacker

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 *Calculate zonal statistics*

Description

Calculate zonal statistics based on one or more environmental variable raster .tif layers. This function can be used to aggregate data across a set (or all) sub-catchments. The sub-catchment raster (.tif) input file is stored on disk. The output is a data.table which is loaded into R.

Usage

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

Arguments

data_dir	character. Path to the directory containing all input data.
subc_id	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 extract_ids() function, and by subsetting the resulting data.frame.
subc_layer	character. Full path to the sub-catchment ID .tif layer.
var_layer	character vector of variable raster layers on disk, e.g. "slope_grad_dw_cel_h00v00.tif". Variable names should remain intact in file names, even after file processing, i.e., slope_grad_dw_cel should appear in the file name. The files should be cropped to the extent of the sub-catchment layer to speed up the computation.
out_dir	character. The directory where the output will be stored. If the out_dir and file_name 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.
file_name	character. Name of the .csv file where the output table will be stored. out_dir should also be specified for this purpose.
n_cores	numeric. Number of cores used for parallelization, in case multiple .tif files are provided to var_layer.
quiet	logical. If FALSE, the standard output will be printed. Default is TRUE.

Author(s)

Afroditi Grigoropoulou, Jaime Garcia Marquez, Maria M. Üblacker

References

https://grass.osgeo.org/grass82/manuals/r.univar.html

See Also

report_no_data to check the defined NoData value. set_no_data to define a NoData value.

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

# Define the directory where the output will be stored
output_folder <- paste0(my_directory, "/hydrography90m_test_data/output")
# Create output folder if it doesn't exist
if(!dir.exists(output_folder)) dir.create(output_folder)

# Calculate the zonal statistics for all sub-catchments for two variables
stat <- extract_zonal_stat(data_dir = paste0(my_directory,
                                             "/hydrography90m_test_data"),
                           subc_id = c(513837216, 513841103,
                                       513850467, 513868394,
                                       513870312),
                           subc_layer = subc_raster,
                           var_layer = c("spi_1264942.tif",
                                         "sti_1264942.tif"),
                           out_dir = output_folder,
                           file_name = "zonal_statistics.csv",
                           n_cores = 2)
# Show output table
stat
```

---

get_catchment_graph *Get catchment from graph*

Description

Subset the 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.

Usage

```
get_catchment_graph(
  g,
  subc_id = NULL,
  outlet = FALSE,
```

```
  mode = NULL,
  as_graph = FALSE,
  n_cores = 1,
  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 which to delineate the upstream drainage area. If empty, then outlets will be used as sub-catchment IDs (with outlet = TRUE). 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.
outlet	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.
mode	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.
as_graph	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.
n_cores	numeric. Number of cores used for parallelization in the case of multiple stream segments / outlets. Default is 1. Currently, the parallelization 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.
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 graph or data.table that reports all subc_ids. In case of multiple input segments, the results are stored in a list.

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

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 = "513855877"
# Get the upstream catchment as a graph
g_up <- get_catchment_graph(g = my_graph, subc_id = subc_id, mode = "in",
                            outlet = FALSE, as_graph = TRUE, n_cores = 1)

# Get the downstream segments as a data.table,
g_down <- get_catchment_graph(g = my_graph, subc_id = subc_id, mode = "out",
                              outlet = FALSE, as_graph = FALSE, n_cores = 1)

# Get the catchments of all outlets in the study area as a graph
g_all <- get_catchment_graph(g = my_graph, mode = "in", outlet = TRUE,
                             as_graph = TRUE, n_cores = 1)
```

---

get_regional_unit_id *Get Hydrography90m regional unit IDs*

Description

Identifies the IDs of the regional units within the Hydrography90m data in which the input points are located. The IDs are required to then download the data using download_tiles(). Input is a point data frame.

Usage

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

Arguments

data	a data.frame or data.table with lat/lon 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

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_occurence <- read.table(paste0(my_directory,
                                       "/hydrography90m_test_data",
                                       "/spdata_1264942.txt"),
                                header = TRUE)

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

`get_segment_neighbours` *Get stream segment neighbours*

Description

For each segment, reports those upstream, downstream, or up-and downstream segments that are connected to one or multiple input segments 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".

This function can also be used to create the connectivity table for Marxan by using var_layer="length" and attach_only=TRUE. The resulting table reports the connectivity from each segment, along with the stream length for all connected segments.

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

g	igraph object. A directed graph.
subc_id	numeric vector of the input sub-catchment IDs (=stream segment IDs) for which to search the connected segments.
var_layer	character vector. One or more attributes (variable layers) of the input graph that should be reported for each output segment_id ("to_stream"). Optional.
stat	one of the functions mean, median, min, max, sd (without quotes). Aggregates (or summarizes) the variables for the neighbourhood of each input segment ("stream", e.g., the average land cover in the next five upstream segments or sub-catchments). Default is NULL.
attach_only	logical. If TRUE, the selected variables will be only attached to each segment without any further aggregation. Default is FALSE.
order	numeric. The neighbouring order as in igraph::ego. Order = 1 would be immediate neighbours of the input sub-catchment IDs, order = 2 would be the order 1 plus the immediate neighbours of those sub-catchment IDs in order 1, and so on.
mode	character. One of "in", "out", or "all". "in" returns only upstream neighbouring segments, "out" returns only the downstream segments, and "all" returns both.
n_cores	numeric. Number of cores used for parallelization in the case of multiple stream segments / outlets. Default is 1. Currently, the parallelization 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.
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.

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

Examples

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

# Load the stream network as graph
my_graph <- read_geopackage(gpkg= paste0(my_directory,
```

```
                                        "/hydrography90m_test_data",
                                        "/order_vect_59.gpkg"),
                              import_as = "graph")

# 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 regular tile ID*

Description

Identifies the ids of the tiles within the Hydrography90m data in which the given points are located. The IDs are required to then download the data using download_tiles(). Input is a point data frame.

Usage

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

Arguments

`data`	a data.frame or data.table with lat/lon 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` *Calculate upstream basin*

Description

Calculates the upstream basin of a given point, considering the point as the outlet.

Usage

```
get_upstream_catchment(
  data,
  id,
  lon,
  lat,
  direction_layer = NULL,
  out_dir = NULL,
  n_cores = NULL,
  quiet = TRUE
)
```

Arguments

`data`	a data.frame or data.table with lat/lon coordinates in WGS84, which have been snapped to the stream network. The snapping can be done using the function 'snap_to_network'.
`id`	character. The name of a column containing unique IDs for each row of "data" (e.g., occurrence or site IDs).
`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 raster file with the direction variable.

out_dir
: Full path to the directory where the output(s) will be stored. To identify the upstream catchment the output file name includes the site id.

n_cores
: numeric. Number of cores used for parallelization. If NULL, available cores - 1 will be used.

quiet
: logical. If FALSE, the standard output will be printed. Default is TRUE.

Author(s)

Jaime Garcia Marquez, Afroditi Grigoropoulou, Maria M. Üblacker

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_occurence <- 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_occurence,
                                            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)
```

`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, read = FALSE)
```

Arguments

`tile_dir`	character. The directory containing only 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).
read	logical. If TRUE, the merged layer gets read into R. If FALSE, the layer is only stored on disk. Default is FALSE.

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 me 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("h22v08", "h22v10"),
                            out_dir = output_folder,
                            file_name = "basin_merged.tif",
                            read = TRUE)
```

read_geopackage *Read a GeoPackage file*

Description

Reads an entire, or only 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). "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, Maria M.Üblacker

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 an integer raster layer*

Description

Reclassifies an integer raster .tif layer using the r.reclass function of GRASS GIS. To reclassify the raster layer the present raster values and the new raster values have to be defined.

If the input raster layer has floating point values, you should multiply the input data 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,
  read = TRUE,
  no_data = -9999,
  type = "Int32",
  compress = "DEFLATE",
  quiet = TRUE
)
```

Arguments

data	a data.frame or data.table with the present and new raster values.
rast_val	character. The name of the column with the present raster values.
new_val	character. The name of the column with the new raster values.
raster_layer	Full path to the input raster .tif layer.
recl_layer	character. Full path of the output .tif layer of the reclassified raster file.
read	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 TRUE.
no_data	numeric. The no_data value of the new .tif layer. Default is -9999.
type	character. Data type; Options are Byte, Int16, UInt16, Int32, UInt32,CInt16, CInt32. Default is Int32.
compress	Compression type: DEFLATE or LZW. Default is DEFLATE.

Author(s)

Maria M. Üblacker

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` *Report NoData value*

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, Maria M. Üblacker

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 *Set no data value*

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, Maria M. Üblacker

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` *Snap points to stream segment based on distance or flow accumulation*

Description

Snap points to the next stream segment within a defined radius 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 with lat/lon 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).
`stream_layer`	character. Full path of the stream network .gpkg 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.

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".
distance	numeric. Maximum radius in meters. 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 command available in GRASS GIS to simultaneously a number of points to a stream network. A distance threshold can be specified and points will be snapped to any stream segment within this distance radius. 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.

Note

Duplicated rows will be removed.

Author(s)

Maria M. Üblacker, Jaime Garcia Marquez

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_occurence <- read.table(paste0(my_directory,
                           "/hydrography90m_test_data/spdata_1264942.txt"),
                             header = TRUE)

# Define full path to flow accumulation
stream_rast <- paste0(my_directory,
                      "/hydrography90m_test_data/stream_1264942.tif")
flow_rast <- 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_vect,
                                    accu_layer = flow_rast,
                                    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

Snaps data points to the stream segment of the sub-catchment the data 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

`data`	a data.frame or data.table with lat/lon 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_id`	character. The name of the column with the basin IDs. If NULL, the basin IDs will be extracted automatically. Default is NULL
`subc_id`	character. The name of the column with the sub-catchment IDs. If NULL, the sub-catchment IDs will be extracted automatically. Default is NULL.
`basin_layer`	character. Full path to the basin ID .tif layer.
`subc_layer`	character. Full path to the sub-catchment ID .tif layer.

stream_layer	character. Full path of the stream network .gpkg file.
n_cores	numeric. Number of cores used for parallelization. Default is 1.
quiet	logical. If FALSE, the standard output will be printed. Default is TRUE.

Details

The function uses the network preparation and maintenance module of GRASS GIS (v.net), to connect a vector lines map (stream network) with a points 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.This operation creates a new node on the vector line (i.e. stream segment) from which the new snapped coordinates can be extracted.

Author(s)

Jaime Garcia Marquez, Maria M. Üblacker

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_occurence <- 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_occurence,
                                  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_occurence,
                                            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)
```

Index