Package: hydrofab (via r-universe)

December 6, 2024

Type Package

Title Hydrologic Network Refactoring and Aggregation Tools

Version 0.6.1

Description A collection of tools for manipulating hydrologic and hydraulic networks

URL https://github.com/NOAA-OWP/hydrofab

BugReports https://github.com/NOAA-OWP/hydrofab/issues

Depends R (>= 3.5.0)

Imports data.table, dplyr, glue, httr, hydroloom, igraph, logger, lwgeom, methods, nhdplusTools, parallel, pbapply, rlang, rmapshaper, rvest, sf, stats, terra, tibble, tidyr, units, utils, yyjsonr

Suggests testthat

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

VignetteBuilder knitr

Config/pak/sysreqs cmake libgdal-dev gdal-bin libgeos-dev libglpk-dev libicu-dev libpng-dev libxml2-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev libnode-dev libx11-dev

Repository https://owp-spatial.r-universe.dev

RemoteUrl https://github.com/NOAA-OWP/hydrofab

RemoteRef HEAD

RemoteSha 9c2c1d8bcd8c065e3ea12477fc360e42954644d1

Contents

`add_areasqkm` *Compute km2 area Short hand for safely computing area in sqkm and returning as numeric vector.*

Description

Compute km2 area Short hand for safely computing area in sqkm and returning as numeric vector.

Compute area in square kilometers

Usage

```
add_areasqkm(x)

add_areasqkm(x)
```

Arguments

x	POLYGON sf object

Value

numeric vector

numeric vector

Examples

```
library(sf)
nc = st_read(system.file("shape/nc.shp", package="sf"))
add_areasqkm(nc[1,])
```

add_areasqkm_to_crosswalk

Add small area to crosswalk

Description

Add small area to crosswalk

Usage

```
add_areasqkm_to_crosswalk(crosswalk, comid = "hf_id")
```

Arguments

crosswalk	an existing crosswalk table
comid	the shared ID

Value

data.frame

add_flowpath_edge_list

Generate Catchment Network Table

Description

Generate Catchment Network Table

Usage

```
add_flowpath_edge_list(gpkg)
```

Arguments

gpkg	a hydrofabric gpkg

Value

data.frame with ID, toID, length, area, and levelpath

`add_hydroseq` *Add hydrosequence*

Description

Add hydrosequence

Usage

```
add_hydroseq(flowpaths)
```

Arguments

`flowpaths`	sf object (LINESTRING)

Value

sf object

`add_lengthkm` *Compute length in kilometers*

Description

Compute length in kilometers

Usage

```
add_lengthkm(x)
```

Arguments

`x`	LINESTRING sf object

Value

numeric vector

add_lengthmap *Add Length Map to Refactored Network*

Description

This function replicates the member_COMID column of a refactored network but adds a new notation Following each COMID is '.' which is proceeded by the fraction of that COMID making up the new flowpath. For example 101.1 would indicate 100 Equally 101.05 would indicate 50

Usage

```
add_lengthmap(flowpaths, length_table)
```

Arguments

`flowpaths`	a refactored flowpath network containing an member_COMID column
`length_table`	a table of NHDPlus COMIDs and LENGTH to use as weights. Can be found with `nhdplusTools::get_vaa("lengthkm")`

Value

sf object

Examples

```
## Not run:
path <- system.file("extdata/walker_reconcile.gpkg", package = "hydrofab")
fps  <- add_lengthmap(flowpaths = sf::read_sf(path),
length_table = nhdplusTools::get_vaa("lengthkm"))

## End(Not run)
```

add_lookup_table *Generate Lookup table for refactored or aggregated network*

Description

Generate Lookup table for refactored or aggregated network

Usage

```
add_lookup_table(
  gpkg = NULL,
  refactored_gpkg = NULL,
  reconciled_layer = "flowpaths"
)
```

Arguments

`gpkg` character path to gpkg containing aggregated network. Omit for refactored network lookup table creation.

`refactored_gpkg` character path to the gpkg for the refactored network used to create the aggregate network. If no aggregatedd gpkg is passed in, a lookup table will be added to this gpkg.

`reconciled_layer` character path layer name containing fully reconciled flowpaths. Ignored for aggregated network lookup table creation.

Value

file path to modified gpkg

`add_mapped_hydrolocations`

Add a mapped_POI layer to network_list

Description

Add a mapped_POI layer to network_list

Usage

```
add_mapped_hydrolocations(
  network_list,
  type = c("HUC12", "Gages", "TE", "NID", "WBIn", "WBOut"),
  refactored_gpkg = NULL,
  verbose = TRUE
)
```

Arguments

`network_list` a list with flowpath and catchment data

`refactored_gpkg` a (optional) path to

`verbose` should messages be emited?

Value

list()

add_measures *Add/sync/update length and area measures*

Description

Add/sync/update length and area measures

Usage

```
add_measures(flowpaths, divides)
```

Arguments

cat	sf object (POLYGON)

Value

list

add_nonnetwork_divides *Add Non Network Divides to Aggregated Fabric The refactoring process intentionally drop catchments without a flowpath. In cases where a seamless discritization of the landscape is needed, these area must be reintroduced from the reference dataset.*

Description

Add Non Network Divides to Aggregated Fabric The refactoring process intentionally drop catchments without a flowpath. In cases where a seamless discritization of the landscape is needed, these area must be reintroduced from the reference dataset.

Usage

```
add_nonnetwork_divides(
  gpkg = NULL,
  vpu = NULL,
  divides = NULL,
  huc12 = NULL,
  reference_gpkg = NULL,
  reference_divides = NULL,
  verbose = TRUE
)
```

Arguments

`gpkg` a path to a gpkg

`huc12` huc12 to COMID crosswalk

`reference_gpkg` A path to the reference VPU geopackage (see get_hydrofabric(..., type = "reference"))

`verbose` Should messages be emitted?

`divide` If gpkg is NULL, then an sf data.frame, otherwise a the layer name. See details.

Value

gpkg path

---

add_nonnetwork_nexus_location

Add Non-Network Nexus Locations

Description

This function generates spatial points for non-network nexus locations (coastal and internal) based on the provided divides. It assigns unique identifiers and links them to waterbody identifiers.

Usage

```
add_nonnetwork_nexus_location(
  divides,
  coastal_nexus_prefix = "cnx-",
  internal_nexus_prefix = "inx-",
  waterbody_prefix = "wb-"
)
```

Arguments

`divides` A spatial data frame (e.g., ‘sf‘ object) containing polygons representing divides. Must include columns ‘type‘, ‘divide_id‘, and ‘toid‘.

`coastal_nexus_prefix`
A character string to prefix ‘divide_id‘ values for coastal nexus locations. Default is ‘"cnx-"‘.

`internal_nexus_prefix`
A character string to prefix ‘divide_id‘ values for internal nexus locations. Default is ‘"inx-"‘.

`waterbody_prefix`
A character string to prefix ‘toid‘ values for non-network nexus locations. Default is ‘"wb-"‘.

Details

- For each divide of type ‘coastal‘ or ‘internal‘, the function calculates a representative point using ‘st_point_on_surface‘. - Prefixes are applied to ‘divide_id‘ and ‘toid‘ values using ‘flush_prefix‘ and ‘mutate‘. - The output retains only the columns ‘id‘, ‘toid‘, ‘type‘, and geometry.

Value

A spatial data frame (e.g., ‘sf‘ object) containing the non-network nexus locations. The resulting data frame includes the columns: - ‘id‘: Unique identifier for each nexus location (prefixed with ‘coastal_nexus_prefix‘ or ‘internal_nexus_prefix‘). - ‘toid‘: Linked waterbody identifier (prefixed with ‘waterbody_prefix‘). - ‘type‘: The type of nexus (‘coastal‘ or ‘internal‘). - ‘geometry‘: The spatial location of each nexus.

add_prefix *Add Prefixes to Topological Data*

Description

This function adds specified prefixes to the ‘id‘ and ‘toid‘ columns of a topological data frame based on the type of topology and its context (e.g., network or nexus).

Usage

```
add_prefix(
  topo,
  hf_prefix = "cat-",
  nexus_prefix = "nex-",
  terminal_nexus_prefix = "tnx-",
  coastal_nexus_prefix = "cnx-",
  internal_nexus_prefix = "inx-"
)
```

Arguments

`topo` A data frame containing topological data. Must include columns ‘topo_type‘, ‘type‘, ‘id‘, and ‘toid‘.

`hf_prefix` A character string to prefix ‘id‘ values in rows where ‘topo_type == "network"‘. Default is ‘"cat-"‘.

`nexus_prefix` A character string to prefix ‘toid‘ values in rows where ‘type‘ does not match specific cases (‘terminal‘, ‘coastal‘, or ‘internal‘). Default is ‘"nex-"‘.

`terminal_nexus_prefix` A character string to prefix ‘toid‘ values where ‘type == "terminal"‘. Default is ‘"tnx-"‘.

`coastal_nexus_prefix` A character string to prefix ‘toid‘ values where ‘type == "coastal"‘. Default is ‘"cnx-"‘.

`internal_nexus_prefix`
A character string to prefix ‘toid‘ values where ‘type == "internal"‘. Default is ‘"inx-"‘.

Value

A data frame with updated ‘id‘ and ‘toid‘ values, including appropriate prefixes based on the type of topology and the ‘type‘ column. The resulting data frame contains the columns ‘id‘, ‘toid‘, ‘type‘, and any additional columns with names containing ‘"vpu"‘.

`aggregate_along_mainstems` *Aggregate along network mainstems*

Description

Given a set of ideal catchment sizes, plus the minimum allowable catchment size and segment length, aggregate the network along mainstems.

Usage

```
aggregate_along_mainstems(
  network_list,
  ideal_size_sqkm,
  min_area_sqkm,
  min_length_km,
  verbose = TRUE,
  cache_file = NULL
)
```

Arguments

`network_list`	a list containing flowline and catchment ‘sf‘ objects
`min_area_sqkm`	The minimum allowable size of the output hydrofabric catchments
`min_length_km`	The minimum allowable length of the output hydrofabric flowlines
`ideal_size`	The ideal size of output hydrofabric catchments
`term_cut`	cutoff integer to define terminal IDs

Value

a list containing aggregated and validated flowline and catchment ‘sf‘ objects

aggregate_network_to_outlets *Aggregate Network*

Description

Aggregates a catchment network according to a set of outlet.

Usage

```
aggregate_network_to_outlets(
  flowpath,
  outlets,
  da_thresh = NA,
  only_larger = FALSE,
  post_mortem_file = NA
)
```

Arguments

`flowpath` sf data.frame Flowpaths with ID, toID, LevelPathID, and Hydroseq attributes.

`outlets` data.frame with "ID" and "type" columns. "ID" must be identifiers from flowpath and divide data.frames. "type" should be "outlet", or "terminal". "outlet" will include the specified ID. "terminal" will be treated as a terminal node with nothing downstream.

`da_thresh` numeric Defaults to NA. A threshold total drainage area in the units of the TotDASqKM field of the flowpath data.frame. When automatically adding confluences to make the network valid, tributary catchments under this threshold will be lumped with the larger tributaries rather than being added to the set of output catchments.

`only_larger` boolean Defaults to TRUE. If TRUE when adding confluences to make the network valid, only tributaries larger than the one with an upstream outlet will be added. e.g. if a tributary is required in the model this will add main stems that the tributary contributes to. Note that the NHDPlus treats divergences as part of the main stem, so the da_thresh may still be needed to eliminate small tributary catchments introduced by divergences near confluences.

`post_mortem_file` rda file to dump environment to in case of error

Details

This function operates on the catchment network as a node-edge graph. The outlet types are required to ensure that graph searches start from the appropriate nodes and includes the appropriate catchments. Outlets such as gages should be treated as "outlet" outlets. While it may be possible for the algorithm to determine terminal outlets, at this time, it is required that they be specified explicitely as "terminal" outlet types.

The function checks supplied outlets to make sure they connect downstream. Checks verify that the outlet of the levelpath (main stem of a total catchment) of each supplied outlet is in the supplied outlet set. If the outlet of a levelpath is not in the supplied set, it is added along with other catchments that contribute to the same receiving catchment. These checks ensure that all output catchments have one and only one input and output nexus and that all catchments are well-connected.

Examples

```
source(system.file("extdata", "walker_data.R", package = "nhdplusTools"))

fline <- dplyr::right_join(dplyr::select(walker_flowline, COMID),
                           nhdplusTools::prepare_nhdplus(walker_flowline, 0, 0, 0, FALSE))

fline <- dplyr::select(fline, ID = COMID, toID = toCOMID,
                       LevelPathID = LevelPathI, Hydroseq)

outlets <- data.frame(ID = c(5329357, 5329317, 5329365, 5329303, 5329435, 5329817),
                      type = c("outlet", "outlet", "outlet", "terminal", "outlet", "outlet"),
                          stringsAsFactors = FALSE)

aggregated <- aggregate_network_to_outlets(fline, outlets)

aggregated <- aggregate_network_to_outlets(fline, outlets)

outlets <- dplyr::filter(fline, ID %in% outlets$ID)

outlets <- nhdplusTools::get_node(outlets)

plot(aggregated$fline_sets$geom, lwd = 3, col = "red")
plot(walker_flowline$geom, lwd = .7, col = "blue", add = TRUE)
plot(outlets$geometry, add = TRUE)
```

aggregate_sets *Aggregate Sets by Index Table*

Description

Aggregate Sets by Index Table

Usage

```
aggregate_sets(network_list, index_table)
```

Arguments

network_list	a list of flowpaths and catchments
index_table	index table to aggregate with

Value

a list of catchments and flowpaths that have been validated

---

aggregate_to_distribution *Aggregate Network to Uniform Size*

Description

This function aggregates a network to a desired size distribution while enforcing minimum flowpath legnths and catchment areas. Additionally a set of explicit nexus locations can be provided over which the network cannot be aggregated (see poi_to_outlet)

Usage

```
aggregate_to_distribution(
  gpkg = NULL,
  vpu = NULL,
  flowpath = NULL,
  divide = NULL,
  crs = 5070,
  pois = NULL,
  ideal_size_sqkm = 10,
  min_length_km = 1,
  min_area_sqkm = 3,
  outfile = NULL,
  log = TRUE,
  overwrite = FALSE,
  cache = FALSE,
  verbose = TRUE
)
```

Arguments

`gpkg`	a path to a gpkg
`flowpath`	If gpkg is NULL, then an sf data.frame, otherwise a the layer name. See details.
`divide`	If gpkg is NULL, then an sf data.frame, otherwise a the layer name. See details.
`ideal_size_sqkm`	The ideal size of catchments (default = 10 sqkm)
`min_length_km`	The minimum allowable length of flowpath features (default = 1 km)
`min_area_sqkm`	The minimum allowable area of catchment features (default = 3 sqkm)
`outfile`	of not NULL, where to write the output files
`log`	a filepath to write messages to or Boolean (TRUE = print to console; FALSE = no messages)

`overwrite` overwrite existing gf file. Default is FALSE

`verbose` print status updates. Default = TRUE

`outlets` data.frame with mandatory "ID" column and optional "POI_ID" column. "ID" must be identifiers from flowpath and divide data.frames and POI ID must be unique.

`nexus_locations`
a data.frame with columns specifying the ID, and the nexus type.

Details

If gpkg is not NULL, divide and flowpath can be left NULL as well. The code attempts to infer the correct layers. The divides layer will be the one including the word "divide" or "catchment" and the flowpath layer will be the one including 'flowpath' or 'flowline'. If no layers, or more then one layer are deemed possible for each input, then the function will stop and ask for explicit names.

Value

if outfile = TRUE, a file path, else a list object

`aggregate_to_outlets` *Aggregate Catchments*

Description

Aggregates catchments according to a set of outlet catchments. Network aggregation is completed using: See `aggregate_network_to_outles`.

Usage

```
aggregate_to_outlets(
  gpkg = NULL,
  flowpath = NULL,
  divide = NULL,
  outlets = NULL,
  zero_order = NULL,
  coastal_cats = NULL,
  da_thresh = NA,
  only_larger = FALSE,
  post_mortem_file = NA,
  keep = NULL
)
```

Arguments

`flowpath` sf data.frame Flowpaths as generated by ‘refactor_nhdplus‘

`divide` sf data.frame Reconciled catchment divides as generated by ‘reconcile_catchment_divides‘

`outlets` data.frame with "ID" and "type" columns. "ID" must be identifiers from flowpath and divide data.frames. "type" should be "outlet", or "terminal". "outlet" will include the specified ID. "terminal" will be treated as a terminal node with nothing downstream.

`zero_order` list of vectors containing IDs to be aggregated into 0-order catchments.

`coastal_cats` sf data.frame with coastal catchments to be used with zero order.

`da_thresh` numeric See `aggregate_network_to_outles`

`only_larger` boolean See `aggregate_network_to_outles`

`post_mortem_file` rda file to dump environment to in case of error

`keep` logical passed along to `clean_geometry`

Details

See `aggregate_network_to_outlets`

Examples

```
source(system.file("extdata", "walker_data.R", package = "hydrofab"))
outlets <- data.frame(ID = c(31, 3, 5, 1, 45, 92),
 type = c("outlet", "outlet", "outlet", "terminal", "outlet", "outlet"),
 stringsAsFactors = FALSE)
aggregated <- aggregate_to_outlets(flowpath = walker_fline_rec,
                                   divide = walker_catchment_rec,
                                   outlets = outlets)
plot(aggregated$cat_sets$geom, lwd = 3, border = "red")
plot(walker_catchment_rec$geom, lwd = 1.5, border = "green", col = NA, add = TRUE)
plot(walker_catchment$geom, lwd = 1, add = TRUE)
plot(walker_flowline$geom, lwd = .7, col = "blue", add = TRUE)
plot(aggregated$cat_sets$geom, lwd = 3, border = "black")
plot(aggregated$fline_sets$geom, lwd = 3, col = "red", add = TRUE)
plot(walker_flowline$geom, lwd = .7, col = "blue", add = TRUE)
```

`agg_length_area` *Enforces area and length grouping*

Description

This function takes a vector of area’s and length’s and returns a grouping vector that enforces the grouping of lengths and areas less then defined thresholds

Usage

```
agg_length_area(l, a, lthres, athres)
```

Arguments

`l`	a vector of lengths
`a`	a vector of areas
`lthres`	a minimum length that must be achieved
`athres`	a minimum length that must be achieved

Value

a vector of length(a) containing grouping indexes

`append_style` *Append a hydrofabric style to a hydrofabric GeoPackage*

Description

Append a hydrofabric style to a hydrofabric GeoPackage

Usage

```
append_style(gpkg_path, layer_names)
```

Arguments

`gpkg_path`	Path to GeoPackage
`layer_names`	character vector of names to append styles for. These names must be in the package QML directory.

`apply_nexus_topology` *Apply Nexus Topology*

Description

This function enforces the nexus–>flowpath topology and adds nexus locations, a catchment edge list, a flowpath edge list, and a lookup_table to the network_list object.

Usage

```
apply_nexus_topology(
  gpkg,
  catchments = NULL,
  flowpaths = NULL,
  vpu = NA,
  nexus_prefix = "nex-",
  terminal_nexus_prefix = "tnx-",
  coastal_nexus_prefix = "cnx-",
  internal_nexus_prefix = "inx-",
  catchment_prefix = "cat-",
  waterbody_prefix = "wb-",
  term_add = 1e+09,
  term_filter = NULL,
  verbose = TRUE,
  enforce_dm = FALSE,
  export_gpkg = NULL
)
```

Arguments

nexus_prefix character prefix for nexus IDs

terminal_nexus_prefix
character prefix for terminal nexus IDs

coastal_nexus_prefix
character prefix for coastal nexus IDs

internal_nexus_prefix
character prefix for internal nexus IDs

catchment_prefix
character prefix for catchment IDs

waterbody_prefix
character prefix for catchment IDs

enforce_dm should the data model be validated prior to writing?

export_gpkg file path to write new data. If NULL list object is returned

network_list list containing flowpath and catchment ‘sf‘ objects

Value

list or file path

assign_global_identifiers

Update Hydrofabric Identifiers For a given set of hydrofabric geopackages, update the ID and toID values to be globally unique.

Description

Update Hydrofabric Identifiers For a given set of hydrofabric geopackages, update the ID and toID values to be globally unique.

Usage

```
assign_global_identifiers(
  gpkgs = NULL,
  outfiles = NULL,
  flowpath_layer = "flowpaths",
  divide_layer = "divides",
  network_layer = "network",
  overwrite = FALSE,
  term_add = 1e+09,
  modifications = NULL,
  verbose = TRUE
)
```

Arguments

`gpkgs`	a vecotor of file.paths to that define the global network
`outfiles`	a vector of file.paths to write to
`flowpath_layer`	the layer name containing flowpaths
`divide_layer`	the layer name containing divides
`network_layer`	the name of layer containing the hydrologic network
`overwrite`	overwrite existing files?
`term_add`	value to be added to all terminal IDs
`verbose`	emit messages

Value

a data.frame

`assign_id` *Index a Vector by Cumulative Sum*

Description

Index a Vector by Cumulative Sum

Usage

```
assign_id(x, athres)
```

Arguments

`x`	a vector of values
`athres`	the ideal cumulative size of each group Cumulative sums will get as close to this value without exceeding it

Value

a vector of length(a)

`build_collapse_table` *Build Headwater Collapse Table*

Description

Identifies small (pathlength or area) headwater catchments and returns a data.frame with the current ID and the feature ID it should collapse into (becomes). Headwaters are segments in which there are no inflows (!ID

Usage

```
build_collapse_table(network_list, min_area_sqkm = 3, min_length_km = 1)
```

Arguments

`network_list`	a list containing flowpath and catchment 'sf' objects
`min_area_sqkm`	The minimum allowable size of the output hydrofabric catchments
`min_length_km`	The minimum allowable length of the output hydrofabric flowlines

Value

a 2 column data.frame with id, becomes

build_new_id_table *Build a new ID table*

Description

Build a new ID table

Usage

```
build_new_id_table(
  meta,
  index,
  network_layer = "network",
  term_add = 1e+09,
  modifications = NULL
)
```

Arguments

network_layer	the name of layer containing the hydrologic network
gpkgs	a row of network metadata built with ‘network_metadata‘
flowpath_layer	the layer name containing flowpaths
divide_layer	the layer name containing divides

Value

data.frame

clean_geometry *Clean Catchment Geometry*

Description

Fixes geometry issues present in catchments derived from DEMs. These include, but are not limited to disjoint polygon fragments, artifacts from the DEM used to generate the catchments, and non-valid geometry topologies. A secondary goal of this functions is to provide a way to reduce the data column of the catchments by offering a topology preserving simplification through `ms_simplify`. Generally a "keep" parameter of .9 seems appropriate for the resolution of the data but can be modified in function

Usage

```
clean_geometry(
  catchments,
  flowlines = NULL,
  fl_ID = NULL,
  ID = "ID",
  keep = NULL,
  crs = 5070,
  grid = 9e-04,
  gb = 8,
  force = FALSE,
  sys = NULL
)
```

Arguments

`catchments` catchments geometries to fix

`flowlines` flowlines geometries to filter largest unit (optional)

`fl_ID` flowlines unique identifier

`ID` name of uniquely identifying column

`keep` proportion of points to retain in geometry simplification (0-1; default 0.05). See `ms_simplify`. If NULL, then no simplification will be executed.

`crs` integer or object compatible with sf::st_crs coordinate reference. Should be a projection that supports area-calculations.

`gb` The amount of heap memory to be allocated when force = TRUE

`force` should the mapshaper/mapshaper-xl binaries be used directly for simplification?

`sys` logical should the mapshaper system library be used. If NULL the system library will be used if available.

Value

sf object

`collapse_flowlines` *Collapse NHDPlus Network*

Description

Refactors the NHDPlus flowline network, eliminating short and non-confluence flowlines. The aim of this function is to create flowpaths that describe a network of catchments that combines complex hydrology near confluences into upstream catchments and removes very short flowlines along mainstem flow-paths.

Usage

```
collapse_flowlines(
  flines,
  thresh,
  add_category = FALSE,
  mainstem_thresh = NULL,
  exclude_cats = NULL,
  warn = TRUE
)
```

Arguments

`flines`	data.frame with COMID, toCOMID, LENGTHKM, Hydroseq, and LevelPathI columns
`thresh`	numeric a length threshold (km). Flowlines shorter than this will be collapsed with up or downstream flowlines.
`add_category`	boolean if combination category is desired in output, set to TRUE
`mainstem_thresh`	numeric threshold for combining inter-confluence mainstems
`exclude_cats`	integer vector of COMIDs to be excluded from collapse modifications.
`warn`	boolean controls whether warning an status messages are printed

Value

A refactored network with merged up and down flowlines.

See Also

The `refactor_nhdplus` function implements a complete workflow using ‘collapse_flowlines()‘.

`collapse_headwaters` *Collapse Headwaters*

Description

This function identifies small (pathlength or area) headwater catchments and collapses them into the existing network until none remain. Headwaters are those segments in which there are no inflows (!ID

Usage

```
collapse_headwaters(
  network_list,
  min_area_sqkm = 3,
  min_length_km = 1,
  verbose = TRUE,
  cache_file = NULL
)
```

Arguments

`network_list` a list containing flowpath and catchment ‘sf‘ objects

`min_area_sqkm` The minimum allowable size of the output hydrofabric catchments

`min_length_km` The minimum allowable length of the output hydrofabric flowlines

`verbose` should messages be emitted?

`cache_file` If not NULL results will be written to a provide path (.gpkg)

Value

a list containing flowpath and catchment ‘sf‘ objects

`cs_group` *Cumulative sum area grouping*

Description

This function takes a vector of areas and lengths and returns a index vector that combines them towards an ideal aggregate area (ideal_size_sqkm). While enforcing a minimum area (amin) and length (lmin). Additionally, this function can take a set of indexes to exclude over which the network cannot be aggregated.

Usage

```
cs_group(areas, lengths, exclude_dn, exclude_un, ideal_size_sqkm, amin, lmin)
```

Arguments

`areas` a vector of areas

`lengths` a vector of lengths

`exclude_dn` a vector of equal length to areas and lengths. Any non NA value will be used to enforce an aggregation break on the outflow node of a flowpath

`exclude_un` a vector of equal length to areas and lengths. Any non NA value will be used to enforce an aggregation break on the inflow node of a flowpath

`ideal_size_sqkm`
a vector of areas

`amin` a threshold, or target, cumulative size

`lmin` a threshold, or target, cumulative size

Value

a vector of length(areas) containing grouping indexes

define_touch_id *Identify intersection types and downstream topology*

Description

Identify intersection types and downstream topology

Usage

```
define_touch_id(flowpaths, term_cut = 1e+09)
```

Arguments

`flowpaths`	sf LINESTRING

Value

data.frame with id, type, touches, touches_toID columns

describe_hydrofabric *Describe Hydrofabric Describes a hydrofabric in terms of flowpath and catchment count. If they are unequal, FALSE is returned. If equal TRUE is returned. Messages can optionally be emitted.*

Description

Describe Hydrofabric Describes a hydrofabric in terms of flowpath and catchment count. If they are unequal, FALSE is returned. If equal TRUE is returned. Messages can optionally be emitted.

Usage

```
describe_hydrofabric(network_list, verbose = TRUE)
```

Arguments

`network_list`	a list containing flowpaths and catchments
`verbose`	should messages be emitted?

Value

boolean condition

`download_elev` *Download Elevation and Derivatives*

Description

Download Elevation and Derivatives

Usage

```
download_elev(product, out_dir, regions = NULL)
```

Arguments

`product`	character DEM, hydroDEM, or FDRFAC.
`out_dir`	path to directory to store output.
`regions`	character vector of two digit hydrologic

`download_fdr_fac` *Download FDR FAC*

Description

Download FDR FAC

Usage

```
download_fdr_fac(out_dir, regions = NULL)
```

Arguments

`out_dir`	path to directory to store output.
`regions`	character vector of two digit hydrologic

drop_extra_features *Remove non-coincident Network Features Remove non-coincident flowlines and catchment pairs from a network list*

Description

Remove non-coincident Network Features Remove non-coincident flowlines and catchment pairs from a network list

Usage

```
drop_extra_features(network_list, verbose)
```

Arguments

`network_list`	a list containing flowpaths and catchments
`verbose`	should message be emitted?

Value

a list containing flowpaths and catchments

flowpaths_to_linestrings *Convert MULITLINESTINGS to LINESTRINGS*

Description

Convert MULITLINESTINGS to LINESTRINGS

Usage

```
flowpaths_to_linestrings(flowpaths)
```

Arguments

`flowpaths`	a flowpath ‘sf‘ object

Value

a ‘sf‘ object

`flush_prefix` *Flush existing ID prefixes Given a data object and column, remove a prefix and adjoining "-"*

Description

This function removes prefixes from specified columns in a data frame by extracting the numeric portion of the values after the last '-' character.

Usage

```
flush_prefix(input, col)

flush_prefix(input, col)
```

Arguments

`input`	A data frame containing the columns to be processed.
`col`	A character vector specifying the names of the columns from which to remove prefixes.

Details

- The function processes each specified column by removing the prefix up to and including the last '-' character using a regular expression. - The updated columns are converted to numeric values.

Value

data object with updated column

The input data frame with the specified columns updated. The values in these columns are converted to numeric, retaining only the portion after the last '-' character.

`get_boundaries` *Return RPU or VPU boundaries*

Description

Return RPU or VPU boundaries

Usage

```
get_boundaries(type = "vpu")
```

Arguments

`type`	character. Either "RPU" or "VPU"

Value

An object of class "sf"

get_minimal_network *Get Minimal Network*

Description

Given a set of outlets, will generate a minimal network by calling aggregate_network_to_outlets and adding nhdplus attributes to the result.

If geometry is included with the network, it will be merged and returned.

Usage

```
get_minimal_network(flowpath, outlets)
```

Arguments

flowpath	sf data.frame Flowpaths with ID, toID, LevelPathID, Hydroseq and LENGTHKM and AreaSqKM attributes.
outlets	data.frame with "ID" and "type" columns. "ID" must be identifiers from flowpath and divide data.frames. "type" should be "outlet", or "terminal". "outlet" will include the specified ID. "terminal" will be treated as a terminal node with nothing downstream.

Value

a data.frame (potentially including an sfc list column) with attributes generated by add_plus_network_attributes and a list column "set" containing members of each output flowpath.

Examples

```
source(system.file("extdata", "walker_data.R", package = "nhdplusTools"))
fline <- walker_flowline

outlets <- data.frame(ID = c(5329357, 5329317, 5329365, 5329435, 5329817),
                      type = c("outlet", "outlet", "outlet", "outlet", "outlet"))

#' Add toCOMID
fline <- nhdplusTools::get_tocomid(fline, add = TRUE)

# get attributes set
fline <- dplyr::select(fline, ID = comid, toID = tocomid,
                       LevelPathID = levelpathi, hydroseq = hydroseq,
                       AreaSqKM = areasqkm, LENGTHKM = lengthkm)

min_net <- get_minimal_network(fline, outlets)
```

```
plot(sf::st_geometry(fline), col = "blue")
plot(sf::st_geometry(min_net), lwd = 2, add = TRUE)
plot(sf::st_geometry(nhdplusTools::get_node(min_net)), add = TRUE)
```

`get_row_col` *Get Row and Column*

Description

Get Row and Column

Usage

```
get_row_col(fdr, start, fac_matrix)
```

Arguments

`fdr`	flow direction grid
`start`	matrix (row, col)
`fac_matrix`	flow accumulation matrix

`hl_to_outlet` *Extract nexus locations for Reference POIs*

Description

Extract nexus locations for Reference POIs

Usage

```
hl_to_outlet(
  gpkg,
  type = c("HUC12", "Gages", "TE", "NID", "WBIn", "WBOut"),
  verbose = TRUE
)
```

Arguments

`gpkg`	a reference hydrofabric gpkg
`type`	the type of desired POIs
`verbose`	should messages be emitted?

Value

data.frame with ID, type columns

hyaggregate_log *Logging shorthand Log a message with given log level, and optional verbosity.*

Description

Logging shorthand Log a message with given log level, and optional verbosity.

Usage

```
hyaggregate_log(level, message, verbose = TRUE)
```

Arguments

`level`	log level, see logger::log_levels for more details
`message`	R objects that can be converted to a character vector via the active message formatter function
`verbose`	should message be emitted?

Value

log message

layer_exists *Check if a geopackage and layer exists This function checks if a layer exists in a geopackage*

Description

Check if a geopackage and layer exists This function checks if a layer exists in a geopackage

Usage

```
layer_exists(gpkg, name)
```

Arguments

`gpkg`	path to geopackage
`name`	name of layer to check

Value

logical

make_hf_gpkg_from_refactor

Convert Refactor Output to HF gpkg This is a temporary function as changes get pushed upstream

Description

Convert Refactor Output to HF gpkg This is a temporary function as changes get pushed upstream

Usage

```
make_hf_gpkg_from_refactor(gpkg)
```

Arguments

gpkg	gpkg file path

Value

file.path

make_hf_gpkg_from_reference

Convert Reference Output to HF gpkg This is a temporary function as changes get pushed upstream

Description

Convert Reference Output to HF gpkg This is a temporary function as changes get pushed upstream

Usage

```
make_hf_gpkg_from_reference(gpkg)
```

Arguments

gpkg	gpkg file path

Value

file.path

make_hf_gpkg_from_uniform_aggregate

Convert Target Size Aggregate output to HF gpkg This is a temporary function as changes get pushed upstream

Description

Convert Target Size Aggregate output to HF gpkg This is a temporary function as changes get pushed upstream

Usage

```
make_hf_gpkg_from_uniform_aggregate(gpkg)
```

Arguments

`gpkg`	gpkg file path

Value

file.path

map_outlet_ids

Map outlets from COMID to ID for aggregate catchments

Description

given reconciled flowlines and a set of source outlets, returns a set of outlets with reconciled IDs suitable for use with aggregate_catchments.

Usage

```
map_outlet_ids(source_outlets, reconciled)
```

Arguments

`source_outlets`	data.frame with COMID and type columns
`reconciled`	data.frame as returned by refactor workflow

`middle_massage` *Re-index the interior of vector by threshold*

Description

Merges the interior values of a vector if they are less then the provided threshold. Merging will look "up" and "down" the vector and merge into the smaller of the two.

Usage

```
middle_massage(x, index_values, threshold)
```

Arguments

`x`	vector of values
`index_values`	current index values
`threshold`	threshold to evaluate x

Value

a vector of length(x) containing grouping indexes

`network_metadata` *Capture Network Metadata This function assumes that files are names *_VPU.gpkg*

Description

Capture Network Metadata This function assumes that files are names *_VPU.gpkg

Usage

```
network_metadata(
  gpkgs,
  flowpath_layer = "flowpaths",
  divide_layer = "divides",
  network_layer = "network"
)
```

Arguments

`gpkgs`	a vector of file.paths to attribute
`flowpath_layer`	the layer name containing flowpaths
`divide_layer`	the layer name containing divides
`network_layer`	the name of layer containing the hydrologic network

Value

data.frame

`pack_set` *pack set*

Description

pack set

Usage

```
pack_set(x, y = "set")
```

Arguments

`x`	data.frame containing "set" list column to be packed

Value

data.frame containing comma seperated character column

`pinch_sides` *Re-index the edges of vector by threshold Merge the outside edges of a vector if they are less then the provides threshold.*

Description

Re-index the edges of vector by threshold Merge the outside edges of a vector if they are less then the provides threshold.

Usage

```
pinch_sides(x, ind, thres)
```

Arguments

`x`	vector of values
`ind`	current index values
`thres`	threshold to evaluate x

Value

a vector of length(x) containing grouping indexes

prepare_network *Prepare Hydrologic Network*

Description

Prepare Hydrologic Network

Usage

```
prepare_network(network_list)
```

Arguments

network_list	a list with flowpath and catchment data

Details

This function adds an area, length, hydrosequence, streamorder and contributing drainage area metric to the flowpath list element of network_list.

tot_drainage_areasqkm can only be added when there are no NA areas

Value

a list containing flowpath and catchment ‘sf‘ objects

prep_cat_fdr_fac *Prep catchment with FDR/FAC*

Description

Prep catchment with FDR/FAC

Usage

```
prep_cat_fdr_fac(cat, fdr, fac)
```

Arguments

cat	catchment (sf object)
fdr	flow direction grid
fac	flow accumulation grid

prep_split_events *Prep Split Events*

Description

Prep Split Events

Usage

```
prep_split_events(pois, fline, divides, threshold = 25)
```

Arguments

pois	a set of POIs with a poi_id, X and (in 5070)
divides	a set of divides geometries (EPSG:5070)
flines	a set of flowlines geometries (EPSG:5070)
theshold	a percentage (0-100) a POI must be upstream before splitting

Value

sf POINT object

read_hydrofabric *Read Catchments and Flowpaths from Geopackage Convenience function for reading two layers into a list*

Description

Read Catchments and Flowpaths from Geopackage Convenience function for reading two layers into a list

Usage

```
read_hydrofabric(
  gpkg = NULL,
  catchments = NULL,
  flowpaths = NULL,
  realization = "all",
  crs = NULL,
  verbose = Sys.getenv("hydrofab_verbose") != "false"
)
```

Arguments

`gpkg` path to geopackage

`realization` what layers to read? Options: "catchemnts", "flowpaths", "all"

`crs` desired CRS, if NULL they stay as read. If all CRS layers arenot

`catchment_name` name of catchment layer. If NULL, attempts to find divides layer

`flowpath_name` name of flowpath layer. If NULL, attempts to find flowpath layer

`vebose` should message be emitted?

Value

list

`realign_topology` *Realign Topology to a nexus network*

Description

Realign Topology to a nexus network

Usage

```
realign_topology(
  network_list,
  nexus_prefix = NULL,
  terminal_nexus_prefix = NULL,
  coastal_nexus_prefix = NULL,
  internal_nexus_prefix = NULL,
  catchment_prefix = NULL,
  waterbody_prefix = NULL,
  term_add = 1e+09,
  term_filter = NULL
)
```

Arguments

`network_list` list containing flowpath and catchment ‘sf‘ objects

`nexus_prefix` character prefix for nexus IDs

`terminal_nexus_prefix` character prefix for terminal nexus IDs

`coastal_nexus_prefix` character prefix for coastal nexus IDs

`internal_nexus_prefix` character prefix for internal nexus IDs

`catchment_prefix` character prefix for catchment IDs

`waterbody_prefix` character prefix for catchment IDs

Value

list

reconcile_catchment_divides

Reconcile Catchment Divides

Description

Reconciles catchment divides according to the output of `reconcile_collapsed_flowlines` and `refactor_nhdplus`

Usage

```
reconcile_catchment_divides(
  catchment,
  fline_ref,
  fline_rec,
  fdr = NULL,
  fac = NULL,
  para = 2,
  cache = NULL,
  min_area_m = 800,
  snap_distance_m = 100,
  simplify_tolerance_m = 40,
  vector_crs = 5070,
  fix_catchments = TRUE,
  keep = NULL
)
```

Arguments

`catchment`	sf data.frame NHDPlus Catchment or CatchmentSP layers for included COMIDs
`fline_ref`	sf data.frame flowlines as returned by `refactor_nhdplus` and `reconcile_collapsed_flowlines`
`fline_rec`	sf data.frame flowpaths as returned by `reconcile_collapsed_flowlines`
`fdr`	character path to D8 flow direction
`fac`	character path to flow accumulation
`para`	integer numer of cores to use for parallel execution
`cache`	path .rda to cache incremental outputs
`min_area_m`	minimum area in m^2 to filter out slivers (caution, use with care!!)
`snap_distance_m`	distance in meters to snap SpatRaster generated geometry to polygon geometry

`simplify_tolerance_m`
: dTolerance in meters for simplification of grid-cell based polygons

`vector_crs`
: integer or object compatible with sf::st_crs coordinate reference. Should be a projection that supports area-calculations.

`fix_catchments`
: logical. should catchment geometries be rectified?

`keep`
: Only applicable if fix_catchments = TRUE. Defines the proportion of points to retain in geometry simplification (0-1; default 0.05). See `ms_simplify`. Set to NULL to skip simplification.

Details

Note that all inputs must be passed in the same projection.

Value

Catchment divides that have been split and collapsed according to input flowpaths

See Also

The `refactor_nhdplus` function implements a complete workflow using ‘reconcile_collapsed_flowlines()‘ and can be used in prep for this function.

`reconcile_collapsed_flowlines` *Reconcile Collapsed Flowlines*

Description

Reconciles output of collapse_flowlines giving a unique ID to each new flowpath and providing a mapping to NHDPlus COMIDs.

Usage

```
reconcile_collapsed_flowlines(flines, geom = NULL, id = "COMID")
```

Arguments

`flines`
: data.frame with COMID, toCOMID, LENGTHKM, LevelPathI, Hydroseq, and TotDASqKM columns

`geom`
: sf data.frame for flines

`id`
: character id collumn name.

Value

reconciled flowpaths with new ID, toID, LevelPathID, and Hydroseq identifiers. Note that all the identifiers are new integer IDs. LevelPathID and Hydroseq are consistent with the LevelPathID and Hydroseq from the input NHDPlus flowlines.

See Also

The refactor_nhdplus function implements a complete workflow using ‘reconcile_collapsed_flowlines()‘.

refactor *Refactoring Wrapper*

Description

A wrapper around refactor_nhdplus and reconcile_catchment_divides

Usage

```
refactor(
  gpkg = NULL,
  flowpaths = NULL,
  catchments = NULL,
  pois = NULL,
  avoid = NULL,
  split_flines_meters = 10000,
  collapse_flines_meters = 1000,
  collapse_flines_main_meters = 1000,
  threshold = 25,
  min_area_m = 800,
  snap_distance_m = 100,
  simplify_tolerance_m = 40,
  cores = 1,
  fac = NULL,
  fdr = NULL,
  purge_non_dendritic = TRUE,
  keep = NULL,
  outfile = NULL
)
```

Arguments

`gpkg` a starting GPKG

`flowpaths` Reference flowline features

`catchments` Reference catchment features

`avoid` integer vector of COMIDs to be excluded from collapse modifications.

`split_flines_meters`
numeric the maximum length flowpath desired in the output.

`collapse_flines_meters`
numeric the minimum length of inter-confluence flowpath desired in the output.

`collapse_flines_main_meters`
numeric the minimum length of between-confluence flowpaths.

`cores` integer number of cores to use for parallel execution

`fac` path to flow accumulation grid. If NULL (default) then catchments are NOT reconciled.

`fdr` path to flow direction grid. If NULL (default) then catchments are NOT reconciled.

`keep` proportion of points to retain in geometry simplification (0-1; default 0.05). See ms_simplify. If NULL, then no simplification will be executed.

`outfile` path to geopackage to write refactored_flowlines, and if facfdr != NULL, refactored catchments.

`events` data.frame containing events

Value

data to the specified gpkg

`refactor_nhdplus` *Refactor NHDPlus*

Description

A complete network refactor workflow has been packaged into this function. Builds a set of normalized catchment-flowpaths from input flowline features. See details and vignettes for more information.

Usage

```
refactor_nhdplus(
  nhdplus_flines,
  split_flines_meters,
  split_flines_cores,
  collapse_flines_meters,
  collapse_flines_main_meters,
  out_refactored,
  out_reconciled,
  three_pass = FALSE,
  purge_non_dendritic = TRUE,
  exclude_cats = NULL,
  events = NULL,
  warn = TRUE
)
```

Arguments

`nhdplus_flines` data.frame raw nhdplus flowline features as derived from the national seamless geodatabase.

`split_flines_meters`
numeric the maximum length flowpath desired in the output.

`split_flines_cores`
numeric the number of processing cores to use while splitting flowlines.

`collapse_flines_meters`
numeric the minimum length of inter-confluence flowpath desired in the output.

`collapse_flines_main_meters`
numeric the minimum length of between-confluence flowpaths.

`out_refactored` character where to write a geopackage containing the split and collapsed flowlines.

`out_reconciled` character where to write a geopackage containing the reconciled flowpaths.

`three_pass` boolean whether to perform a three pass collapse or single pass.

`purge_non_dendritic`
boolean passed on to prepare_nhdplus

`exclude_cats` integer vector of COMIDs to be excluded from collapse modifications.

`events` data.frame containing events as generated by nhdplusTools::get_flowline_index()

`warn` boolean controls whether warning an status messages are printed

Details

This is a convenient wrapper function that implements three phases of the network refactor workflow: split, collapse, reconcile. See the NHDPlus Refactor vignette for details of these three steps by running: `vignette("refactor_nhdplus", package = "hydrofab")`

See Also

In addition to ‘prepare_nhdplus‘ from the nhdplusTools package, The following three functions are used in the ‘refactor_nhdplus‘ workflow.

1. `split_flowlines`
2. `collapse_flowlines`
3. `reconcile_collapsed_flowlines`

Examples

```
source(system.file("extdata",
                   "sample_flines.R",
                   package = "nhdplusTools"))

nhdplus_flowlines <- sf::st_zm(sample_flines)

refactor_nhdplus(nhdplus_flines = nhdplus_flowlines,
                 split_flines_meters = 2000,
                 split_flines_cores = 2,
                 collapse_flines_meters = 500,
                 collapse_flines_main_meters = 500,
                 out_refactored = "temp.gpkg",
                 out_reconciled = "temp_rec.gpkg",
                 three_pass = TRUE,
                 purge_non_dendritic = FALSE,
                 warn = FALSE)

unlink("temp.gpkg")
unlink("temp_rec.gpkg")
```

rpu_boundaries *RPU Boundaries Raster Processing Unit boundaries*

Description

RPU Boundaries Raster Processing Unit boundaries

Usage

```
rpu_boundaries
```

Format

An object of class "sf"

sb_id *Return ScienceBase ID for hydrofabric This function checks if a layer exists in a geopackage*

Description

Return ScienceBase ID for hydrofabric This function checks if a layer exists in a geopackage

Usage

```
sb_id(type)
```

Arguments

gpkg	path to geopackage
name	name of layer to check

Value

character

split_catchment_divide *Split Catchment Divides*

Description

A catchment-divide splitting algorithm that works with a D8 flow direction grid and the output of nhdplus_refactor. See Vignette for examples.

Usage

```
split_catchment_divide(
  catchment,
  fline,
  fdr,
  fac,
  lr = FALSE,
  min_area_m = 800,
  snap_distance_m = 100,
  simplify_tolerance_m = 40,
  vector_crs = NULL
)
```

Arguments

`catchment` sf data.frame with one catchment divide

`fline` sf data.frame with one or more flowline segments in upstream downstream order.

`fdr` character path to flow direction that fully covers the catchment

`fac` character path to flow accumulation that fuller covers the catchment

`lr` boolean should catchments be split along the left/right bank?

`min_area_m` minimum area in m^2 to filter out slivers (caution, use with care!!)

`snap_distance_m`
distance in meters to snap SpatRaster generated geometry to polygon geometry

`simplify_tolerance_m`
dTolerance in meters for simplification of grid-cell based polygons

`vector_crs` any object compatible with sf::st_crs. Used for vector-based calculations in case that fdr projection is not suitable (e.g. lon/lat) – must result in units of meters.

Value

Split catchment divides as an sfc geometry.

split_flowlines *Split Flowlines*

Description

A wrapper for split_lines that works on nhdplus attributes

Usage

```
split_flowlines(flines, max_length = NULL, events = NULL, para = 0, avoid = NA)
```

Arguments

`flines`	data.frame with COMID, toCOMID, LENGTHKM and LINESTRING sf column in "meters" projection
`max_length`	maximum segment length to return
`events`	data.frame containing events as generated by nhdplusTools::get_flowline_index() if an ‘identifier‘ attribute is included, it will be passed through in the output table.
`para`	numeric how many threads to use in parallel computation
`avoid`	vector of ids to avoid

Value

All the flowlines with some split apart. COMIDs are returned as strings with a semantic part number appended. That is .1, .2, ... .10, .11, etc. are appended and must be treated as one would treat a semantic version. .1 is the most upstream and the sequence increases in the downstream direction.

See Also

The `refactor_nhdplus` function implements a complete workflow using ‘split_flowlines()‘.

Examples

```
source(system.file("extdata", "new_hope_data.R", package = "hydrofab"))

new_hope_flowline <-
  dplyr::right_join(dplyr::select(new_hope_flowline, COMID, REACHCODE, FromMeas, ToMeas),
                    suppressWarnings(nhdplusTools::prepare_nhdplus(
                      new_hope_flowline, 0, 0, 0, FALSE, warn = FALSE)),
                    by = "COMID")

split <- split_flowlines(suppressWarnings(sf::st_cast(sf::st_transform(
  new_hope_flowline, 5070), "LINESTRING")),
                         max_length = 2000, events = new_hope_events)
```

`st_rename` *Rename simple features layer*

Description

Rename simple features layer

Usage

```
st_rename(dsn, layer, new_layer)
```

Arguments

`dsn`	data source name. Interpretation varies by driver: can be a filename, a folder, a database name, or a Database Connection (we officially test support for `RPostgres::Postgres()` connections).
`layer`	layer name. Varies by driver, may be a file name without extension; for database connection, it is the name of the table. If layer is missing, the `basename` of `dsn` is taken.
`new_layer`	new layer name

Value

dsn

`trace_upstream` *Trace Upstream*

Description

Trace Upstream

Usage

```
trace_upstream(start_point, cat, fdr, fac_matrix, fdr_matrix)
```

Arguments

`start_point`	row col index
`cat`	catchment
`fdr`	flow direction grid
`fac_matrix`	flow accumulation matrix
`fdr_matrix`	flow direction matrix

Value

sfc

union_linestrings *DEPRECATED: Fast LINESTRING union*

Description

Wayyyy faster then either data.table, or sf based line merging

Usage

```
union_linestrings(lines, ID)
```

Arguments

lines	lines to merge
ID	ID to merge over

Value

an sf object

union_linestrings_geos *DEPRECATED: Fast LINESTRING union*

Description

Wayyyy faster then either data.table, or sf based line merging

Usage

```
union_linestrings_geos(lines, ID)
```

Arguments

lines	lines to merge
ID	ID to merge over

Value

an sf object

union_polygons *Fast POLYGON Union*

Description

This is significantly faster then sf::st_union or summarize

Usage

```
union_polygons(poly, ID)
```

Arguments

`poly`	sf POLYGON object
`ID`	the column name over which to union geometries

Value

sf object

unpack_set *unpack set*

Description

unpack set

Usage

```
unpack_set(x, y = "set")
```

Arguments

`x`	data.frame containing comma separated "set" column to be unpacked

Value

data.frame containing a list column

update_network_identifiers *Update Network Identifiers Given a data.frame of sf object, the id and toid values are undated based on a provided lookup table (produced with build_new_id_table), and a vpu_topo list if there are cross VPU flows. In the vpu_topo is NULL or has 0 rows, no vpu correction is applied.*

Description

Update Network Identifiers Given a data.frame of sf object, the id and toid values are undated based on a provided lookup table (produced with build_new_id_table), and a vpu_topo list if there are cross VPU flows. In the vpu_topo is NULL or has 0 rows, no vpu correction is applied.

Usage

```
update_network_identifiers(x, lookup, term_add = 1e+09, connections = NULL)
```

Arguments

`x`	a data.frame or sf object with id and/or toid columns
`lookup`	a lookup table of new ID values
`vpu_topo`	a VPU lookup correction table

Value

data.frame

vpu_boundaries *VPU Boundaries Vector Processing Unit boundaries*

Description

VPU Boundaries Vector Processing Unit boundaries

Usage

```
vpu_boundaries
```

Format

An object of class `"sf"`

`write_hydrofabric` *Write a hydrofabric gpkg A hydrofabric consists of a flowpath, catchment, and topology layer written to a self contained geopackage*

Description

Write a hydrofabric gpkg A hydrofabric consists of a flowpath, catchment, and topology layer written to a self contained geopackage

Usage

```
write_hydrofabric(network_list, outfile, verbose = TRUE, enforce_dm = TRUE)
```

Arguments

`network_list`	a list containing flowpaths and catchments
`outfile`	a file (gpkg) where layers should be written
`verbose`	should messages be emitted?
`catchment_name`	the layer name for divides
`flowpath_name`	the layer name for flowpaths

Value

file path

Index