Package ‘nhdplusTools’

March 10, 2023

Type Package
Title NHDPlus Tools
Version 0.6.2
Description Tools for traversing and working with National Hydrography Dataset Plus (NHD-Plus) data. All methods implemented in 'nhdplusTools' are available in the NHDPlus documentation available from the US Environmental Protection Agency <https://www.epa.gov/waterdata/basic-information>.

URL https://doi-usgs.github.io/nhdplusTools/
https://github.com/doi-usgs/nhdplusTools/

BugReports https://github.com/doi-usgs/nhdplusTools/issues/

Depends R (>= 4.0)
Imports dplyr, rlang, sf, RANN, units, magrittr, jsonlite, htrr, xml2, R.utils, utils, tidyr, methods, rosm, prettymapr, fst, dataRetrieval, tools, zip, pbapply
Suggests testthat, knitr, rmarkdown, markdown, ggmap, ggplot2, sp, lwgeom, devtools, codetools, data.table, parallel, s2, gifski, leaflet

License CC0
Encoding UTF-8
RoxygenNote 7.2.3
VignetteBuilder knitr
Config/testthat/parallel true
Config/testthat/edition 3
LazyData true
NeedsCompilation no
Author David Blodgett [aut, cre] (<https://orcid.org/0000-0001-9489-1710>), Mike Johnson [aut] (<https://orcid.org/0000-0002-5288-8350>), Marc Weber [ctb] (<https://orcid.org/0000-0002-9742-4744>), Josh Erickson [ctb]
Maintainer David Blodgett <dblodgett@usgs.gov>
Repository  CRAN
Date/Publication  2023-03-10 09:40:14 UTC

R topics documented:

add_plus_network_attributes ........................................ 3
align_nhdplus_names ..................................................... 5
calculate_arbolate_sum .................................................. 5
calculate_total_drainage_area ......................................... 6
disambiguate_flowline_index ........................................... 7
discover_nhdplus_id ..................................................... 8
discover_nldi_characteristics .......................................... 9
download_nhdplusr ....................................................... 10
download_nhdplusv2 ..................................................... 11
download_rf1 ............................................................... 12
download_vaa .............................................................. 12
download_wbd .............................................................. 13
fix_flowdir ................................................................. 14
get_boundaries ............................................................. 15
get_DD ......................................................................... 15
get_DM ....................................................................... 16
get_elev_along_path ...................................................... 17
get_flowline_index ....................................................... 18
get_gagesII ................................................................. 20
get_hr_data ................................................................. 21
get_huc ...................................................................... 21
get_huc12 ................................................................. 22
get_huc8 ..................................................................... 23
get_hydro_location ....................................................... 24
get_levelpaths ............................................................. 24
get_nhdarea ................................................................. 25
get_nhdplus ............................................................... 26
get_nhdplusr .............................................................. 28
get_nldi_basin ............................................................ 29
get_nldi_characteristics ............................................... 30
get_nldi_feature .......................................................... 31
get_nldi_index ............................................................ 32
get_node ................................................................. 32
get_nwis ................................................................. 33
get_partial_length ....................................................... 34
get_pathlength ............................................................ 35
get_path_lengths .......................................................... 36
get_path_members .......................................................... 37
get_pfaf ................................................................. 38
get_raindrop_trace ...................................................... 39
get_sorted ................................................................. 40
get_split_catchment ...................................................... 41
add_plus_network_attributes

Add NHDPlus Network Attributes to a provided network.

Description

Given a river network with required base attributes, adds the NHDPlus network attributes: hydrosequence, levelpath, terminalpath, pathlength, down levelpath, down hydroseq, total drainage area, and terminalflag. The function implements two parallelization schemes for small and large basins respectively. If a number of cores is specified, parallel execution will be used.
add_plus_network_attributes

Usage

add_plus_network_attributes(
  net,
  override = 5,
  cores = NULL,
  split_temp = NULL,
  status = TRUE
)

Arguments

net  data.frame containing comid, tocomid, nameID, lengthkm, and areasqkm. Additional attributes will be passed through unchanged. tocomid == 0 is the convention used for outlets. If a “weight” column is provided, it will be used in get_levelpaths otherwise, arbolate sum is calculated for the network and used as the weight.

override  numeric factor to be passed to get_levelpaths

cores  integer number of processes to spawn if run in parallel.

split_temp  character path to optional temporary copy of the network split into independent sub-networks. If it exists, it will be read from disk rather than recreated.

status  logical should progress be printed?

Value

data.frame with added attributes

Examples

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

test_flowline <- prepare_nhdplus(walker_flowline, 0, 0, FALSE)

test_flowline <- data.frame(
  comid = test_flowline$COMID,
  tocomid = test_flowline$toCOMID,
  nameID = walker_flowline$GNIS_ID,
  lengthkm = test_flowline$LENGTHKM,
  areasqkm = walker_flowline$AreaSqKM)

add_plus_network_attributes(test_flowline)
align_nhdplus_names  

**Align NHD Dataset Names**

**Description**

this function takes any NHDPlus dataset and aligns the attribute names with those used in nhdplus-Tools.

**Usage**

```r
align_nhdplus_names(x)
```

**Arguments**

- `x`: a sf object of nhdplus flowlines

**Value**

data.frame renamed sf object

**Examples**

```r
source(system.file("extdata/new_hope_data.R", package = "nhdplusTools"))
names(new_hope_flowline)
names(new_hope_flowline) <- tolower(names(new_hope_flowline))
new_hope_flowline <- align_nhdplus_names(new_hope_flowline)
names(new_hope_flowline)
```

calculate_arbolate_sum

**Calculate Arbolate Sum**

**Description**

Calculates arbolate sum given a dendritic network and incremental lengths. Arbolate sum is the total length of all upstream flowlines.

**Usage**

```r
calculate_arbolate_sum(x)
```
calculate_total_drainage_area

Arguments

- `x` data.frame with ID, toID, and length columns.

Value

numeric with arbolate sum.

Examples

```r
library(dplyr)
source(system.file("extdata", "walker_data.R", package = "nhdplusTools"))
catchment_length <- select(walker_flowline, COMID, AreaSqKM) %>%
  right_join(prepare_nhdplus(walker_flowline, 0, 0,
    purge_non_dendritic = FALSE, warn = FALSE), by = "COMID") %>%
  select(ID = COMID, toID = toCOMID, length = LENGTHKM)

arb_sum <- calculate_arbolate_sum(catchment_length)
catchment_length$arb_sum <- arb_sum
catchment_length$nhd_arb_sum <- walker_flowline$ArbolateSu

mean(abs(catchment_length$arb_sum - catchment_length$nhd_arb_sum))
max(abs(catchment_length$arb_sum - catchment_length$nhd_arb_sum))
```

calculate_total_drainage_area

**Total Drainage Area**

Description

Calculates total drainage area given a dendritic network and incremental areas.

Usage

`calculate_total_drainage_area(x)`

Arguments

- `x` data.frame with ID, toID, and area columns.

Value

numeric with total area.
Examples

```r
library(dplyr)
source(system.file("extdata", "walker_data.R", package = "nhdplusTools"))
catchment_area <- select(walker_flowline, COMID, AreaSqKM) %>%
  right_join(prepare_nhdplus(walker_flowline, 0, 0,
    purge_non_dendritic = FALSE, warn = FALSE), by = "COMID") %>%
  select(ID = COMID, toID = toCOMID, area = AreaSqKM)

new_da <- calculate_total_drainage_area(catchment_area)
catchment_area$totda <- new_da
catchment_area$nhdptotda <- walker_flowline$TotDASqKM

mean(abs(catchment_area$totda - catchment_area$nhdptotda))
max(abs(catchment_area$totda - catchment_area$nhdptotda))
```

disambiguate_flowline_indexes

**Disambiguate Flowline Indexes**

**Description**

Given a set of flowline indexes and numeric or ascii criteria, return closest match. If numeric criteria are used, the minimum difference in the numeric attribute is used for disambiguation. If ascii criteria are used, the `adist` function is used with the following algorithm: `1 - adist_score / max_string_length`. Comparisons ignore case.

**Usage**

disambiguate_flowline_indexes(indexes, flowpath, hydro_location)

**Arguments**

- **indexes** data.frame as output from `get_flowline_index` with more than one hydrologic location per indexed point.
- **flowpath** data.frame with two columns. The first should join to the COMID field of the indexes and the second should be the numeric or ascii metric such as drainage area or GNIS Name. Names of this data.frame are not used.
- **hydro_location** data.frame with two columns. The first should join to the id field of the indexes and the second should be the numeric or ascii metric such as drainage area or GNIS Name. Names of this data.frame are not used.

**Value**

data.frame indexes deduplicated according to the minimum difference between the values in the metric columns. If two or more result in the same "minimum" value, duplicates will be returned.
Examples

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

hydro_location <- sf::st_sf(id = c(1, 2, 3),
  geom = sf::st_sfc(list(sf::st_point(c(-76.86934, 39.49328)),
  sf::st_point(c(-76.91711, 39.40884)),
  sf::st_point(c(-76.88081, 39.36354))),
  crs = 4326),
  totda = c(23.6, 7.3, 427.9),
  nameid = c("Patapsco", ",", "Falls Run River"))

flowpath <- dplyr::select(sample_flines,
  comid = COMID,
  totda = TotDASqKM,
  nameid = GNIS_NAME,
  REACHCODE,
  ToMeas,
  FromMeas)

indexes <- get_flowline_index(flowpath, hydro_location, search_radius = 0.2, max_matches = 10)

disambiguate_flowline_indexes(indexes, dplyr::select(flowpath, comid, totda), dplyr::select(hydro_location, id, totda))

result <- disambiguate_flowline_indexes(indexes, dplyr::select(flowpath, comid, nameid), dplyr::select(hydro_location, id, nameid))

result[result$id == 1, ]
result[result$id == 2, ]
result[result$id == 3, ]

discover_nhdplus_id

Discover NHDPlus ID

Description

Multipurpose function to find a COMID of interest.

Usage

discover_nhdplus_id(point = NULL, nldi_feature = NULL, raindrop = FALSE)
**discover_nldi_characteristics**

**Arguments**

- **point**
  - `sf` POINT including crs as created by: `sf::st_sfc(sf::st_point(. , .), crs)`

- **nldi_feature**
  - list with names 'featureSource' and 'featureID' where 'featureSource' is derived from the "source" column of the response of `get_nldi_sources` and the 'featureSource' is a known identifier from the specified 'featureSource'.

- **raindrop**
  - logical if `TRUE` will call a raindrop trace web service and return will be the same as `get_raindrop_trace` with direction "none".

**Value**

integer COMID or list containing COMID and raindrop trace.

**Examples**

```r
point <- sf::st_sfc(sf::st_point(c(-76.874, 39.482)), crs = 4326)
discover_nhdplus_id(point)

discover_nhdplus_id(point, raindrop = TRUE)
nldi_nwis <- list(featureSource = "nwissite", featureID = "USGS-08279500")
discover_nhdplus_id(nldi_feature = nldi_nwis)
```

---

**discover_nldi_characteristics**  
*Discover Characteristics Metadata*

**Description**

Provides access to metadata for characteristics that are returned by `get_nldi_characteristics()`.

**Usage**

```r
discover_nldi_characteristics(type = "all")
```

**Arguments**

- **type**
  - character "all", "local", "total", or "divergence_routed".

**Value**

data.frame containing available characteristics
Examples

chars <- discover_nldi_characteristics()
names(chars)
head(chars$local, 10)

download_nhdplusr(nhd_dir, hu_list, download_files = TRUE)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nhd_dir</td>
<td>character directory to save output into</td>
</tr>
<tr>
<td>hu_list</td>
<td>character vector of hydrologic region(s) to download. Use get_huc8 to find HU codes of interest. Accepts two digit and four digit codes.</td>
</tr>
<tr>
<td>download_files</td>
<td>boolean if FALSE, only URLs to files will be returned can be hu02s and/or hu04s</td>
</tr>
</tbody>
</table>

Value

character Paths to geodatabases created.

Examples

hu <- nhdplusTools::get_huc8(sf::st_sfc(sf::st_point(c(-73, 42)), crs = 4326))
(hu <- substr(hu$huc8, 1, 2))
download_nhdplusr(tempdir(), c(hu, "0203"), download_files = FALSE)
download_nhdplusv2

Description

This function downloads and decompresses staged seamless NHDPlusV2 data. The following requirements are needed: p7zip (MacOS), 7zip (windows) Please see: https://www.epa.gov/waterdata/get-nhdplus-national-hydrography-dataset-plus-data for more information and metadata about this data.

Default downloads lower-48 only. See examples for islands. No Alaska data are available.

Usage

download_nhdplusv2(
  outdir,
  url = paste0("https://edap-ow-data-commons.s3.amazonaws.com/NHDPlusV21/",
    "Data/NationalData/NHDPlusV21_NationalData_Seamless", ", Geodatabase_Lower48_07.7z"),
  progress = TRUE
)

Arguments

outdir The folder path where data should be downloaded and extracted
url the location of the online resource
progress boolean display download progress?

Value

character path to the local geodatabase

Examples

## Not run:
download_nhdplusV2("./data/nhd/")

download_nhdplusv2(outdir = "./inst/",
  url = paste0("https://edap-ow-data-commons.s3.amazonaws.com/NHDPlusV21/",
    "Data/NationalData/NHDPlusV21_NationalData_Seamless",
    ", Geodatabase_HI_PR_VI_PI_03.7z"))

## End(Not run)
**download_rf1**  
*Download the seamless Reach File (RF1) Database*

**Description**

This function downloads and decompresses staged RF1 data. See: https://water.usgs.gov/GIS/metadata/usgswrd/XML/erf1_2.xml for metadata.

**Usage**

```r
download_rf1(  
  outdir,  
  url = "https://water.usgs.gov/GIS/dsdl/erf1_2.e00.gz",  
  progress = TRUE  
)
```

**Arguments**

- `outdir`: The folder path where data should be downloaded and extracted
- `url`: the location of the online resource
- `progress`: boolean display download progress?

**Value**

character path to the local e00 file

**Examples**

```r
## Not run:  
download_wbd("./data/rf1/")
## End(Not run)
```

---

**download_vaa**  
*Download nhdplusVAA data from HydroShare*

**Description**

downloads and caches nhdplusVAA data on your computer

**Usage**

```r
download_vaa(  
  path = get_vaa_path(updated_network),  
  force = FALSE,  
  updated_network = FALSE  
)
```

**Arguments**

- `path`: The folder path where data should be downloaded and extracted
- `force`: boolean force download?
- `updated_network`: boolean update network?

**Value**

character path to the local vaa file

**Examples**

```r
## Not run:  
download_vaa("./data/vaa/")
## End(Not run)
```
download_wbd

Arguments

path character path where the file should be saved. Default is a persistent system data as retrieved by nhdplusTools_data_dir. Also see: get_vaa_path
force logical. Force data re-download. Default = FALSE
updated_network logical default FALSE. If TRUE, updated network attributes from E2NHD and National Water Model retrieved from doi:10.5066/P9W79I7Q.

Details

The VAA data is a aggregate table of information from the NHDPlusV2 elevslope.dbf(s), PlusFlowlineVAA.dbf(s); and NHDFlowlines. All data originates from the EPA NHDPlus Homepage here. To see the location of cached data on your machine use get_vaa_path. To view aggregate data and documentation, see here

Value

character path to cached data

download_wbd Download the seamless Watershed Boundary Dataset (WBD)

Description

This function downloads and decompresses staged seamless WBD data. Please see: https://prd-tnm.s3.amazonaws.com/StagedProducts/Hydrography/WBD/National/GDB/WBD_National_GDB.xml for metadata.

Usage

download_wbd(
  outdir,  
  progress = TRUE    
)

Arguments

outdir The folder path where data should be downloaded and extracted
url the location of the online resource
progress boolean display download progress?

Value

character path to the local geodatabase
Examples

```r
## Not run:
  download_wbd("./data/wbd/")

## End(Not run)
```

`fix_flowdir`  
*Fix flow direction*

Description

If flowlines aren’t digitized in the expected direction, this will reorder the nodes so they are.

Usage

```r
fix_flowdir(comid, network = NULL, fn_list = NULL)
```

Arguments

- `comid` integer The COMID of the flowline to check
- `network` sf data.frame The entire network to check from. Requires a "toCOMID" field.
- `fn_list` list containing named elements ‘flowline’, ‘network’, and ‘check_end’, where ‘flowline’ is the flowline to be checked and ‘network’ the feature up or downstream of the flowline to be checked, and ‘check_end’ is “start” or “end” depending if the ‘network’ input is upstream (“start”) or downstream (“end”) of the flowline to be checked. This option allows pre-compilation of pairs of features which may be useful for very large numbers of flow direction checks.

Value

a geometry for the feature that has been reversed if needed.

Examples

```r
source(system.file("extdata/sample_data.R", package = "nhdplusTools"))

fline <- sf::read_sf(sample_data, "NHDFlowline_Network")

# We add a tocomid with prepare_nhdplus
fline <- sf::st_sf(prepare_nhdplus(fline, 0, 0, 0, FALSE), geom = sf::st_zm(sf::st_geometry(fline)))

# Look at the end node of the 10th line.
(n1 <- get_node(fline[10, ], position = "end"))

# Break the geometry by reversing it.
sf::st_geometry(fline)[10] <- sf::st_reverse(sf::st_geometry(fline)[10])
```
# Note that the end node is different now.
(n2 <- get_node(fline[10, ], position = "end"))

# Pass the broken geometry to fix_flowdir with the network for toCOMID
sf::st_geometry(fline)[10] <- fix_flowdir(fline$COMID[10], fline)

# Note that the geometry is now in the right order.
(n3 <- get_node(fline[10, ], position = "end"))

plot(sf::st_geometry(fline)[10])
plot(n1, add = TRUE)
plot(n2, add = TRUE, col = "blue")
plot(n3, add = TRUE, cex = 2, col = "red")

---

**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_DD**

*Navigate Downstream with Diversions*

**Description**

Traverse NHDPlus network downstream with diversions NOTE: This algorithm may not scale well in large watersheds. For reference, the lower Mississippi will take over a minute.

**Usage**

get_DD(network, comid, distance = NULL)
get_DM

Navigate Downstream Mainstem

Description
Traverse NHDPlus network downstream main stem

Usage
get_DM(network, comid, distance = NULL, sort = FALSE, include = TRUE)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>network</td>
<td>data.frame NHDPlus flowlines including at a minimum: COMID, LENGTHKM, DnHydroseq, and Hydroseq.</td>
</tr>
<tr>
<td>comid</td>
<td>integer identifier to start navigating from.</td>
</tr>
<tr>
<td>distance</td>
<td>numeric distance in km to limit how many COMIDs are returned. The COMID that exceeds the distance specified is returned.</td>
</tr>
<tr>
<td>sort</td>
<td>if TRUE, the returned COMID vector will be sorted in order of distance from the input COMID (nearest to farthest)</td>
</tr>
<tr>
<td>include</td>
<td>if TRUE, the input COMID will be included in the returned COMID vector</td>
</tr>
</tbody>
</table>

Examples

```r
library(sf)
start_COMID <- 11688818

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

DD_COMIDs <- get_DD(sample_flines, start_COMID, distance = 4)
plot(dplyr::filter(sample_flines, COMID %in% DD_COMIDs)$geom, col = "red", lwd = 2)

DM_COMIDs <- get_DM(sample_flines, start_COMID, distance = 4)
plot(dplyr::filter(sample_flines, COMID %in% DM_COMIDs)$geom, col = "blue", add = TRUE, lwd = 2)
```
get_elev_along_path

Value

integer vector of all COMIDs downstream of the starting COMID along the mainstem

Examples

library(sf)

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

plot(sample_flines$geom)
start_COMID <- 11690092
DM_COMIDs <- get_DM(sample_flines, start_COMID)
plot(dplyr::filter(sample_flines, COMID %in% DM_COMIDs)$geom,
  col = "red", add = TRUE, lwd = 3)

DM_COMIDs <- get_DM(sample_flines, start_COMID, distance = 40)
plot(dplyr::filter(sample_flines, COMID %in% DM_COMIDs)$geom,
  col = "blue", add = TRUE, lwd = 2)

get_elev_along_path  Get Elevation Along Path (experimental)

Description

Uses a cross section retrieval web services to retrieve elevation along a path.

Usage

get_elev_along_path(points, num_pts, res = 1, status = TRUE)

Arguments

points  sf data.frame containing a point column.
num_pts numeric number of points to retrieve along the cross section.
res  integer resolution of 3D Elevation Program data to request. Must be one of: 1, 3, 5, 10, 30, 60.
status  logical

Value

sf data.frame containing points retrieved. Names include "id", "distance_m", "elevation_m", "spatial_ref", "geometry", and ".group". .group tracks which input point each set of output points belongs to.
get_flowline_index

Examples

```r
point1 <- sf::st_sfc(sf::st_point(x = c(-105.9667, 36.17602)), crs = 4326)
point2 <- sf::st_sfc(sf::st_point(x = c(-105.97768, 36.17526)), crs = 4326)
point3 <- sf::st_sfc(sf::st_point(x = c(-105.98869, 36.17450)), crs = 4326)

points <- sf::st_as_sf(c(point1, point2, point3))
(xs <- get_elev_along_path(points, 100))
if(inherits(xs, "sf")) {
  bbox <- sf::st_bbox(xs) + c(-0.005, -0.005, 0.005, 0.005)
  nhdplusTools::plot_nhdplus(bbox = bbox, cache_data = FALSE)
  plot(sf::st_transform(sf::st_sfc(point1, crs = 4326), 3857), add = TRUE, col = "red")
  plot(sf::st_transform(sf::st_sfc(point2, crs = 4326), 3857), add = TRUE)
  plot(sf::st_transform(sf::st_sfc(point3, crs = 4326), 3857), add = TRUE)
  plot(xs$distance_m, xs$elevation_m)
}
```

get_flowline_index

Get Flowline Index

Description

given an sf point geometry column, return COMID, reachcode, and measure for each.

Usage

```r
get_flowline_index(
  flines,
  points,
  search_radius = NULL,
  precision = NA,
  max_matches = 1
)
```

Arguments

- **flines**: sf data.frame of type LINESTRING or MULTILINESTRING including COMID, REACHCODE, ToMeas, and FromMeas. Can be "download_nhdplusv2" and remote nhdplusv2 data will be downloaded for the bounding box surround
get_flowline_index
the submitted points. NOTE: The download option may not work for large areas,
use with caution.

points sf or sfc of type POINT in analysis projection. NOTE: flines will be projected
to the projection of the points layer.

search_radius units distance for the nearest neighbor search to extend in analysis projection.
If missing or NULL, and points are in a lon lat projection, a default of 0.01
degree is used, otherwise 200 m is used. Conversion to the linear unit used by
the provided crs of points is attempted. See RANN nn2 documentation for more
details.

precision numeric the resolution of measure precision in the output in meters.

max_matches numeric the maximum number of matches to return if multiple are found in
search_radius

Details

Note 1: Inputs are cast into LINESTRINGS. Because of this, the measure output of inputs that are
true multipart lines may be in error.

Note 2: This algorithm finds the nearest node in the input flowlines to identify which flowline the
point should belong to. As a second pass, it can calculate the measure to greater precision than the
nearest flowline geometry node.

Note 3: Offset is returned in units consistent with the projection of the input points.

Note 4: See ‘dfMaxLength’ input to sf::st_segmentize() for details of handling of precision param-
eter.

Note 5: “from” is downstream – 0 is the outlet “to” is upstream – 100 is the inlet

Value
data.frame with five columns, id, COMID, REACHCODE, REACH_meas, and offset. id is the row
or list element in the point input.

Examples

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

point <- sf::st_sfc(sf::st_point(c(-76.87479, 39.48233)),
                         crs = 4326)

get_flowline_index(sample_flines, point)

point <- sf::st_transform(point, 5070)

get_flowline_index(sample_flines, point,
                         search_radius = units::set_units(200, "m"))

get_flowline_index("download_nhdplusv2", point)
get_flowline_index(sample_flines, point, precision = 30)

get_flowline_index(sample_flines,
  sf::st_sfc(list(sf::st_point(c(-76.86934, 39.49328)),
               sf::st_point(c(-76.91711, 39.40884)),
               sf::st_point(c(-76.88081, 39.36354))),
  crs = 4326),
  search_radius = units::set_units(0.2, "degrees"),
  max_matches = 10)

get_gagesII

Find GAGESII Features

Description
Subsets the gagesII dataset by location (POINT), area (POLYGON), or set of IDs.

Usage
get_gagesII(AOI = NULL, id = NULL, t_srs = NULL, buffer = 0.5, basin = FALSE)

Arguments

AOI sf (MULTI)POINT or (MULTI)POLYGON. An 'area of interest' can be provided as either a location (sf POINT) or area (sf POLYGON) in any Spatial Reference System.

id character NWIS Gage ID(s)

t_srs character (PROJ string or EPSG code) or numeric (EPSG code). A user specified target Spatial Reference System (SRS/CRS) for returned objects. Will default to the CRS of the input AOI if provided, and to 4326 for ID requests.

buffer numeric. The amount (in meters) to buffer a POINT AOI by for an extended search. Default = 0.5

basin logical should the gagesII basin also be returned? If True, return value will be a list with "site" and "basin" elements.

Details
The returned object(s) will have the same Spatial Reference System (SRS) as the input AOI. If a individual or set of IDs are used to query, then the default geoserver CRS of EPSG:4326 is preserved. In all cases, a user-defined SRS can be passed to t_srs which will override all previous SRS’s (either input or default). All buffer and distance operations are handled internally using in EPSG:5070 Albers Equal Area projection

Value
a simple features (sf) object
**get_hr_data**  
*Get NHDPlus HiRes Data*

**Description**

Use to remove unwanted detail NHDPlusHR data See get_nhdplushr for examples.

**Usage**

```r
get_hr_data(
  gdb,
  layer = NULL,
  min_size_sqkm = NULL,
  simp = NULL,
  proj = NULL,
  rename = TRUE
)
```

**Arguments**

- `gdb` character path to geodatabase to get data from.
- `layer` character layer name from geodatabase found with `st_layers`
- `min_size_sqkm` numeric minimum basin size to be included in the output
- `simp` numeric simplification tolerance in units of projection
- `proj` a projection specification compatible with `st_crs`
- `rename` boolean if TRUE, nhdplusTools standard attribute values will be applied.

**Value**

`sf` data.frame containing requested data

**get_huc**  
*Find WBD HUC 12 unit subsets*

**Description**

Subsets the WBD level 12 features by location (POINT), area (POLYGON), or set of IDs.

**Usage**

```r
get_huc(AOI = NULL, id = NULL, t_srs = NULL, buffer = 0.5, type = "huc12")
```
get_huc12

Arguments

AOI sf (MULTI)POINT or (MULTI)POLYGON. An ‘area of interest’ can be provided as either a location (sf POINT) or area (sf POLYGON) in any Spatial Reference System.

id WBD HUC ID(s)

t_srs character (PROJ string or EPSG code) or numeric (EPSG code). A user specified - target -Spatial Reference System (SRS/CRS) for returned objects. Will default to the CRS of the input AOI if provided, and to 4326 for ID requests.

buffer numeric. The amount (in meters) to buffer a POINT AOI by for an extended search. Default = 0.5


Details

The returned object(s) will have the same Spatial Reference System (SRS) as the input AOI. If an individual or set of IDs are used to query, then the default geoserver CRS of EPSG:4326 is preserved. In all cases, a user-defined SRS can be passed to t_srs which will override all previous SRS’s (either input or default). All buffer and distance operations are handled internally using in EPSG:5070 Albers Equal Area projection

Value

a simple features (sf) object

get_huc12 Find WBD HUC 12 unit subsets (DEPRECATED)

Description

Subsets the WBD level 12 features by location (POINT), area (POLYGON), or set of IDs. Derived from a static snapshot of HUC 12s from: https://doi.org/10.5066/P9BTKP3T

Usage

get_huc12(AOI = NULL, id = NULL, t_srs = NULL, buffer = 0.5)

Arguments

AOI sf (MULTI)POINT or (MULTI)POLYGON. An ‘area of interest’ can be provided as either a location (sf POINT) or area (sf POLYGON) in any Spatial Reference System.

id WBD HUC12 ID(s) See /linkdownload_nhdplusv2 for documentation of that dataset.
get_huc8  

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>t_srs</td>
<td>character (PROJ string or EPSG code) or numeric (EPSG code). A user specified target Spatial Reference System (SRS/CRS) for returned objects. Will default to the CRS of the input AOI if provided, and to 4326 for ID requests.</td>
</tr>
<tr>
<td>buffer</td>
<td>numeric. The amount (in meters) to buffer a POINT AOI by for an extended search. Default = 0.5</td>
</tr>
</tbody>
</table>

**Details**

The returned object(s) will have the same Spatial Reference System (SRS) as the input AOI. If an individual or set of IDs are used to query, then the default geoserver CRS of EPSG:4326 is preserved. In all cases, a user-defined SRS can be passed to t_srs which will override all previous SRS’s (either input or default). All buffer and distance operations are handled internally using in EPSG:5070 Albers Equal Area projection.

**Value**

a simple features (sf) object

---

### Description

Subsets the WBD level 08 features by location (POINT), area (POLYGON), or set of IDs.

### Usage

get_huc8(AOI = NULL, id = NULL, t_srs = NULL, buffer = 0.5)

### Arguments

- **AOI**: sf (MULTI)POINT or (MULTI)POLYGON. An ’area of interest’ can be provided as either a location (sf POINT) or area (sf POLYGON) in any Spatial Reference System.
- **id**: WBD HUC08 ID(s)
- **t_srs**: character (PROJ string or EPSG code) or numeric (EPSG code). A user specified target Spatial Reference System (SRS/CRS) for returned objects. Will default to the CRS of the input AOI if provided, and to 4326 for ID requests.
- **buffer**: numeric. The amount (in meters) to buffer a POINT AOI by for an extended search. Default = 0.5

### Details

The returned object(s) will have the same Spatial Reference System (SRS) as the input AOI. If an individual or set of IDs are used to query, then the default geoserver CRS of EPSG:4326 is preserved. In all cases, a user-defined SRS can be passed to t_srs which will override all previous SRS’s (either input or default). All buffer and distance operations are handled internally using in EPSG:5070 Albers Equal Area projection.
Value

a simple features (sf) object

get_hydro_location  
Get Hydro Location

Description

given a flowline index, returns the hydrologic location (point) along the specific linear element referenced by the index.

Usage

get_hydro_location(indexes, flowpath)

Arguments

indexes  
data.frame as output from get_flowline_index.

flowpath  
data.frame with three columns: COMID, FromMeas, and ToMeas as well as geometry.

Examples

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

indexes <- get_flowline_index(sample_flines,

 sf::st_sfc(sf::st_sfc(list(sf::st_point(c(-76.86934, 39.49328)),
 sf::st_point(c(-76.91711, 39.40884)),
 sf::st_point(c(-76.88081, 39.36354))),
 crs = 4326))

get_hydro_location(indexes, sample_flines)

get_levelpaths  
Get Level Paths

Description

Calculates level paths using the stream-leveling approach of NHD and NHDPlus. In addition to a levelpath identifier, a topological sort and levelpath outlet identifier is provided in output. If arbolate sum is provided in the weight column, this will match the behavior of NHDPlus. Any numeric value can be included in this column and the largest value will be followed when no nameID is available.

Usage

get_levelpaths(x, override_factor = NULL, status = FALSE, cores = NULL)
get_nhdarea

Arguments

x  data.frame with ID, toID, nameID, and weight columns.
override_factor  numeric factor to use to override nameID. If ‘weight’ is ‘numeric_factor’ times larger on a path, it will be followed regardless of the nameID indication.
status  boolean if status updates should be printed.
cores  numeric number of cores to use in initial path ranking calculations.

details

1. levelpath provides an identifier for the collection of flowlines that make up the single mainstem flowpath of a total upstream aggregate catchment.
2. outletID is the catchment ID (COMID in the case of NHDPlus) for the catchment at the outlet of the levelpath the catchment is part of.
3. topo_sort is similar to Hydroseq in NHDPlus in that large topo_sort values are upstream of small topo_sort values. Note that there are many valid topological sort orders of a directed graph.

Value

data.frame with ID, outletID, topo_sort, and levelpath columns. See details for more info.

Examples

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

test_flowline <- prepare_nhdplus(walker_flowline, 0, 0, FALSE)

test_flowline <- data.frame(
  ID = test_flowline$COMID,
  toID = test_flowline$toCOMID,
  nameID = walker_flowline$GNIS_ID,
  weight = walker_flowline$ArbolateSu,
  stringsAsFactors = FALSE)

get_levelpaths(test_flowline)

Description

Subsets NHD Area features by location (POINT), area (POLYGON), or set of IDs.
Usage

get_nhdarea(AOI = NULL, id = NULL, t_srs = NULL, buffer = 0.5)

Arguments

AOI sf (MULTI)POINT or (MULTI)POLYGON. An 'area of interest' can be provided as either a location (sf POINT) or area (sf POLYGON) in any Spatial Reference System.
id NHD Area COMID(s)
t_srs character (PROJ string or EPSG code) or numeric (EPSG code). A user specified - target -Spatial Reference System (SRS/CRS) for returned objects. Will default to the CRS of the input AOI if provided, and to 4326 for ID requests.
buffer numeric. The amount (in meters) to buffer a POINT AOI by for an extended search. Default = 0.5

Details

The returned object(s) will have the same Spatial Reference System (SRS) as the input AOI. If a individual or set of IDs are used to query, then the default geoserver CRS of EPSG:4326 is preserved. In all cases, a user-defined SRS can be passed to t_srs which will override all previous SRS’s (either input or default). All buffer and distance operations are handled internally using in EPSG:5070 Albers Equal Area projection

Value

a simple features (sf) object

get_nhdplus Get National Hydrography Dataset V2 Subsets (Multirealization)

Description

Subsets NHDPlusV2 features by location (POINT), area (POLYGON), or set of COMIDs. Multi realizations are supported allowing you to query for flowlines, catchments, or outlets.

Usage

get_nhdplus(
  AOI = NULL,
  comid = NULL,
  nwis = NULL,
  realization = "flowline",
  streamorder = NULL,
  t_srs = NULL
)
get_nhdplus

Arguments

AOI sf (MULTI)POINT or (MULTI)POLYGON. An ‘area of interest’ can be provided as either a location (sf POINT) or area (sf POLYGON) in any Spatial Reference System.

comid numeric or character. Search for NHD features by COMID(s)

nwis numeric or character. Search for NHD features by collocated NWIS identifiers

realization character. What realization to return. Default is flowline and options include: outlet, flowline, catchment, and all

streamorder numeric or character. Only return NHD flowlines with a streamorder greater than or equal to this value for input value and higher. Only usable with AOI and flowline realizations.

t_srs character (PROJ string or EPSG code) or numeric (EPSG code). A user specified target -Spatial Reference System (SRS/CRS) for returned objects. Will default to the CRS of the input AOI if provided, and to 4326 for ID requests.

Details

The returned object(s) will have the same Spatial Reference System (SRS) as the input AOI. If a individual or set of IDs are used to query, then the default geoserver CRS of EPSG:4326 is preserved. In all cases, a user-defined SRS can be passed to t_srs which will override all previous SRS’s (either input or default). All buffer and distance operations are handled internally using in EPSG:5070 Albers Equal Area projection

Value

sfc a single, or list, of simple feature objects

Examples

```
point <- sf::st_sfc(sf::st_point(c(-119.845, 34.4146)), crs = 4326)
get_nhdplus(point)
get_nhdplus(point, realization = "catchment")
get_nhdplus(point, realization = "all")
get_nhdplus(comid = 101)
get_nhdplus(nwis = c(11120000, 11120500))
area <- sf::st_as_sfc(sf::st_bbox(c(xmin = -119.8851, xmax = -119.8361, ymax = 34.42439, ymin = 34.40473), crs = 4326))
get_nhdplus(area)
get_nhdplus(area, realization = "flowline", streamorder = 3)
```
get_nhdplushr Get NHDPlus HiRes

Description
Get NHDPlus HiRes

Usage
get_nhdplushr(
  hr_dir,
  out_gpkg = NULL,
  layers = c("NHDFlowline", "NHDPlusCatchment"),
  pattern = ".*GDB.gdb$",
  check_terminals = TRUE,
  overwrite = FALSE,
  keep_cols = NULL,
  ...
)

Arguments
hr_dir character directory with geodatabases (gdb search is recursive)
out_gpkg character path to write output geopackage
layers character vector with desired layers to return. c("NHDFlowline", "NHDPlus-
Catchment") is default. Choose from: c("NHDFlowline", "NHDPlusCatch-
ment", "NHDWaterbody", "NHDArea", "NHDLine", "NHDPlusSink", "NHD-
PlusWall", "NHDPoint", "NHDPlusBurnWaterbody", "NHDPlusBurnLineEvent",
"HYDRO_NET_Junctions", "WBDHU2", "WBDHU4", "WBDHU6", "WBDHU8"
"WBDHU10", "WBDHU12", "WBDLine") Set to NULL to get all available.
pattern character optional regex to select certain files in hr_dir
check_terminals boolean if TRUE, run make_standalone on output.
overwrite boolean should the output overwrite? If false and the output layer exists, it will
be read and returned so this function will always return data even if called a
second time for the same output. This is useful for workflows. Note that this
will NOT delete the entire Geopackage. It will overwrite on a per layer basis.
keep_cols character vector of column names to keep in the output. If NULL, all will be
kept.
...
parameters passed along to get_hr_data for "NHDFlowline" layers.

Details
NHDFlowline is joined to value added attributes prior to being returned. Names are not modified
from the NHDPlusHR geodatabase. Set layers to "NULL" to get all layers.
get_nldi_basin

Value

sf data.frames containing output that may also be written to a geopackage for later use.

Examples

```r
## Not run:
# Note this will download a lot of data to a temp directory.
# Change /quotesingle.Var temp_dir/quotesingle.Var to your directory of choice.
temp_dir <- file.path(nhdplusTools_data_dir(), "temp_hr_cache")

download_dir <- download_nhdplushr(temp_dir, c("0302", "0303"))

get_nhdplushr(download_dir, file.path(download_dir, "nhdplus_0302-03.gpkg"))

get_nhdplushr(download_dir,
  file.path(download_dir, "nhdplus_0302-03.gpkg"),
  layers = NULL, overwrite = TRUE)

get_nhdplushr(download_dir,
  file.path(download_dir, "nhdplus_0302-03.gpkg"),
  layers = "NHDFlowline", overwrite = TRUE,
  min_size_sqkm = 10, simp = 10, proj = "+init=epsg:5070")

# Cleanup
unlink(temp_dir, recursive = TRUE)
```

## End(Not run)

get_nldi_basin

Get NLDI Basin Boundary

Description

Get a basin boundary for a given NLDI feature.

Usage

```r
get_nldi_basin(nldi_feature, simplify = TRUE, split = FALSE)
```

Arguments

- **nldi_feature**: list with names ‘featureSource’ and ‘featureID’ where ‘featureSource’ is derived from the "source" column of the response of dataRetrieval::get_nldi_sources() and the ‘featureID’ is a known identifier from the specified ‘featureSource’.
- **simplify**: logical should response geometry be simplified for visualization and performance?
- **split**: logical should response resolve precisely to the location of the ‘nldi_feature’? Setting ‘TRUE’ calls an additional service and will be slower and less robust.
get_nldi_characteristics

Details

Only resolves to the nearest NHDPlus catchment divide. See: https://waterdata.usgs.gov/blog/nldi-intro/ for more info on the nldi.

Value

sf data.frame with result basin boundary

Examples

```r
library(sf)
library(dplyr)

nldi_nwis <- list(featureSource = "nwissite", featureID = "USGS-05428500")

site <- get_nldi_feature(nldi_nwis)
basin <- get_nldi_basin(nldi_feature = nldi_nwis)
plot(st_geometry(basin))

basin

basin2 <- get_nldi_basin(nldi_feature = nldi_nwis,
simplify = FALSE, split = TRUE)

if(inherits(basin, "sf") & inherits(basin2, "sf")) {
  length(st_coordinates(basin))
  length(st_coordinates(basin2))
  plot(st_geometry(st_buffer(site, units::set_units(3000, "m"))), border = NA)
  plot(st_geometry(site), add = TRUE)
  plot(st_geometry(basin2), add = TRUE)
  plot(st_geometry(basin), border = "red", add = TRUE)
}
```

get_nldi_characteristics

Get Catchment Characteristics

Description

Retrieves catchment characteristics from the Network Linked Data Index. Metadata for these characteristics can be found using `discover_nldi_characteristics()`.
get_nldi_feature

Usage

get_nldi_characteristics(nldi_feature, type = "local")

Arguments

nldi_feature list with names ‘featureSource’ and ‘featureID’ where ‘featureSource’ is derived from the "source" column of the response of dataRetrieval::get_nldi_sources() and the ‘featureID’ is a known identifier from the specified ‘featureSource’.

type character "all", "local", "total", or "divergence_routed".

Value
data.frame containing requested characteristics

Examples

chars <- get_nldi_characteristics(list(featureSource = "nwissite", featureID = "USGS-05429700"))
names(chars)
head(chars$local, 10)

get_nldi_feature

Get NLDI Feature

Description

Get a single feature from the NLDI

Usage

get_nldi_feature(nldi_feature)

Arguments

nldi_feature list with names ‘featureSource’ and ‘featureID’ where ‘featureSource’ is derived from the "source" column of the response of dataRetrieval::get_nldi_sources() and the ‘featureID’ is a known identifier from the specified ‘featureSource’.

Value

sf data.frame with one feature

Examples

get_nldi_feature(list("featureSource" = "nwissite", featureID = "USGS-05428500"))
get_nldi_index  
Get NLDI Index

Description
uses the Network Linked Data Index to retrieve and estimated network location for the given point. If not within a grid cell of a flowline, will use a raindrop trace service to find the nearest downslope flowline location.

Usage
get_nldi_index(location)

Arguments
location numeric WGS84 lon/lat pair (X, Y)

Examples
index <- get_nldi_index(c(-89.276, 42.988))
if(inherits(index, "sf")) {
  plot_nhdplus(bbox = sf::st_bbox(sf::st_buffer(index[1,], units::set_units(1000, "m"))))
  plot(sf::st_geometry(sf::st_transform(index, 3857)), add = TRUE)
}

get_node  
Get Flowline Node

Description
Given one or more flowlines, returns a particular node from the flowline.

Usage
get_node(x, position = "end")

Arguments
x sf data.frame with one or more flowlines
position character either "start" or "end"
**Value**

sf data.frame containing requested nodes

**Examples**

```r
source(system.file("extdata/sample_data.R", package = "nhdplusTools"))

fline <- sf::read_sf(sample_data, "NHDFlowline_Network")

start <- get_node(fline, "start")
end <- get_node(fline, "end")

plot(sf::st_zm(fline$geom),
     lwd = fline$StreamOrde, col = "blue")
plot(sf::st_geometry(start), add = TRUE)
plot(sf::st_zm(fline$geom),
     lwd = fline$StreamOrde, col = "blue")
plot(sf::st_geometry(end), add = TRUE)
```

---

**get_nwis**

*Discover USGS NWIS Stream Gages*

**Description**

Returns a POINT feature class of active, stream network, NWIS gages for an Area of Interest. If a POINT feature is used as an AOI, then the returned sites within the requested buffer, are sorted by distance (in meters) from that POINT.

**Usage**

```r
get_nwis(AOI = NULL, t_srs = NULL, buffer = 20000)
```

**Arguments**

- **AOI**: sf (MULTI)POINT or (MULTI)POLYGON. An 'area of interest' can be provided as either a location (sf POINT) or area (sf POLYGON) in any Spatial Reference System.
- **t_srs**: character (PROJ string or EPSG code) or numeric (EPSG code). A user specified - target - Spatial Reference System (SRS/CRS) for returned objects. Will default to the CRS of the input AOI if provided, and to 4326 for ID requests.
- **buffer**: numeric. The amount (in meters) to buffer a POINT AOI by for an extended search. Default = 20,000. Returned results are arranged by distance from POINT AOI.
Details

The returned object(s) will have the same Spatial Reference System (SRS) as the input AOI. If a individual or set of IDs are used to query, then the default geoserver CRS of EPSG:4326 is preserved. In all cases, a user-defined SRS can be passed to `t_srs` which will override all previous SRS’s (either input or default). All buffer and distance operations are handled internally using in EPSG:5070 Albers Equal Area projection.

Value

a simple features (sf) object

---

**get_partial_length**  
*Get Partial Flowline Length*

Description

Finds the upstream and downstream lengths along a given flowpath (flowline in nhdplus terminology). Internally, the function rescales the reach measure to a flowpath measure and applies that rescaled measure to the length of the flowpath.

Usage

`get_partial_length(hl, net = NULL, fl = NULL)`

Arguments

- `hl` list containing a hydrologic location with names reachcode and reach_meas.
- `net` data.frame containing a flowpath network with reachcode, frommeas, tomeas, and lengthkm attributes. Not required if `fl` is provided.
- `fl` data.frame containing one flowline that corresponds to the reachcode and measure of `hl`. Not required if `hl` is provided.

Value

list containing ‘up’ and ‘dn’ elements with numeric length in km.

Examples

```r
source(system.file("extdata", "walker_data.R", package = "nhdplusTools"))
hydro_location <- list(comid = 5329339,
    reachcode = "18050005000078",
    reach_meas = 30)

(pl <- get_partial_length(hydro_location, walker_flowline))

hydro_location <- sf::st_sf(hydro_location,
```
get_pathlength


```r
geom = nhdplusTools::get_hydro_location(data.frame(hydro_location),
  walker_flowline))

net <- navigate_network(hydro_location,
  mode = "DM", network = walker_flowline,
  distance_km = 4, trim_start = TRUE)

plot(sf::st_geometry(walker_flowline[walker_flowline$COMID == hydro_location$comid,]))
plot(sf::st_geometry(hydro_location), add = TRUE)
plot(sf::st_geometry(net), add = TRUE, col = "blue", lwd = 2)

sf::st_length(net)
pl$dn
```

---

**get_pathlength**  

*Get Path Length*

**Description**

Generates the main path length to a basin’s terminal path.

**Usage**

```r
get_pathlength(x)
```

**Arguments**

- `x`  
  data.frame with ID, toID, length columns.

**Value**

data.frame containing levelpaths for each ID

**Examples**

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

fl <- dplyr::select(prepare_nhdplus(walker_flowline, 0, 0),
  ID = COMID, toID = toCOMID, length = LENGTHKM)

get_pathlength(fl)
```
**get_path_lengths**  

**Get Path Lengths**

**Description**

Given a network and set of IDs, finds path lengths between all identified flowpath outlets. This algorithm finds distance between outlets regardless of flow direction.

**Usage**

```r
get_path_lengths(outlets, network, cores = 1, status = FALSE)
```

**Arguments**

- **outlets**: vector of IDs from data.frame  
- **network**: data.frame with ID, toID, and lengthkm attributes.  
- **cores**: integer number of cores to use for parallel computation.  
- **status**: logical print status and progress bars?

**Value**

data.frame containing the distance between pairs of network outlets. For a network with one terminal outlet, the data.frame will have `nrow(network)^2` rows.

**Examples**

```r
source(system.file("extdata", "walker_data.R", package = "nhdplusTools"))  
fline <- walker_flowline  
outlets <- c(5329303, 5329357, 5329317, 5329365, 5329435, 5329817)  
# Add toCOMID  
fline <- nhdplusTools::get_tocomid(fline, add = TRUE)  
fl <- dplyr::select(fline, ID = comid, toID = tocomid, lengthkm)  
path_lengths <- get_path_lengths(outlets, fl)  
outlet_geo <- sf::st_sf(
  dplyr::left_join(data.frame(ID = outlets),
    dplyr::select(fline, ID = comid), by = "ID"))  
sf::st_geometry(outlet_geo) <- sf::st_geometry(nhdplusTools::get_node(outlet_geo))  
plot(sf::st_geometry(fl))  
plot(sf::st_geometry(outlet_geo), add = TRUE)
```
get_path_members  

Description

Given a network and set of IDs, finds paths between all identified flowpath outlets. This algorithm finds members between outlets regardless of flow direction.

Usage

get_path_members(outlets, network, cores = 1, status = FALSE)

Arguments

outlets vector of IDs from data.frame
network data.frame with ID, toID, and lengthkm attributes.
cores integer number of cores to use for parallel computation.
status logical print status and progress bars?

Value

list of lists containing flowpath identifiers along path that connect outlets.

Examples

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

outlets <- c(5329303, 5329357, 5329317, 5329365, 5329435, 5329817)

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

fl <- dplyr::select(fline, ID = comid, toID = tocomid, lengthkm)

get_path_members(outlets, fl)
Get Pfafstetter Codes (Experimental)

Description
Determines Pfafstetter codes for a dendritic network with total drainage area, levelpath, and topo_sort attributes.

Usage
get_pfaf(x, max_level = 2, status = FALSE)

Arguments
- x: sf data.frame with ID, toID, totda, outletID, topo_sort, and levelpath attributes.
- max_level: integer number of pfaf levels to attempt to calculate. If the network doesn’t have resolution to support the desired level, unexpected behavior may occur.
- status: boolean print status or not

Value
data.frame with ID and pfaf columns.

Examples
```r
library(dplyr)
source(system.file("extdata/nhdplushr_data.R", package = "nhdplusTools"))
hr_flowline <- align_nhdplus_names(hr_data$NHDFlowline)

fl <- select(hr_flowline, COMID, AreaSqKM) %>%
  right_join(prepare_nhdplus(hr_flowline, 0, 0,
    purge_non_dendritic = FALSE,
    warn = FALSE),
    by = "COMID") %>%
  sf::st_sf() %>%
  select(ID = COMID, toID = toCOMID, area = AreaSqKM)
fl$nameID = ""
fl$totda <- calculate_total_drainage_area(sf::st_set_geometry(fl, NULL))
fl <- left_join(fl, get_levelpaths(rename(sf::st_set_geometry(fl, NULL),
  weight = totda)), by = "ID")

pfaf <- get_pfaf(fl, max_level = 3)
fl <- left_join(fl, pfaf, by = "ID")
plot(fl["pf_level_3"], lwd = 2)
```
pfaf <- get_pfaf(fl, max_level = 4)

hr_catchment <- left_join(hr_data$NHDPlusCatchment, pfaf, by = c("FEATUREID" = "ID"))

colors <- data.frame(pf_level_4 = unique(hr_catchment$pf_level_4),  
                      color = sample(terrain.colors(length(unique(hr_catchment$pf_level_4)))))

hr_catchment <- left_join(hr_catchment, colors, by = "pf_level_4")
plot(hr_catchment["color"], border = NA, reset = FALSE)
plot(sf::st_geometry(hr_flowline), col = "blue", add = TRUE)

generate_raindrop_trace()
**get_sorted**

**Value**

sf data.frame containing raindrop trace and requested portion of flowline.

**Examples**

```r
point <- sf::st_sfc(sf::st_point(x = c(-89.2158, 42.9561)), crs = 4326)

(trace <- get_raindrop_trace(point))

if(inherits(trace, "sf")) {
  bbox <- sf::st_bbox(trace) + c(-0.005, -0.005, 0.005, 0.005)

  nhdplusTools::plot_nhdplus(bbox = bbox, cache_data = FALSE)

  plot(sf::st_transform(sf::st_sfc(point, crs = 4326), 3857), add = TRUE)
  plot(sf::st_transform(sf::st_geometry(trace)[1], 3857), add = TRUE, col = "red")
  plot(sf::st_transform(sf::st_geometry(trace)[2], 3857), add = TRUE, col = "black")
}
```

---

### get_sorted

**Get Sorted Network**

**Description**

given a tree with an id and and toid in the first and second columns, returns a sorted and potentially split set of output.

Can also be used as a very fast implementation of upstream with tributaries navigation. The full network from each outlet is returned in sorted order.

**Usage**

```r
get_sorted(x, split = FALSE, outlets = NULL)
```

**Arguments**

- **x**
  
data.frame with an identifier and to identifier in the first and second columns.

- **split**
  
  logical if TRUE, the result will be split into independent networks identified by the id of their outlet. The outlet id of each independent network is added as a "terminalID" attribute.

- **outlets**
  
  same as id in x; if specified only the network emanating from these outlets will be considered and returned.

**Value**

data.frame containing a topologically sorted version of the requested network and optionally a terminal id.
get_split_catchment

Examples

source(system.file("extdata/new_hope_data.R", package = "nhdplusTools"))

fpath <- get_tocomid(
  dplyr::select(new_hope_flowline, COMID, FromNode, ToNode, Divergence, FTYPE,
                AreaSqKM, LENGTHKM, GNIS_ID)
)

head(fpath <- get_sorted(fpath, split = TRUE))

fpath['sort_order'] <- 1:nrow(fpath)

plot(fpath['sort_order'])

get_split_catchment

Get split catchment

Description

Uses catchment splitting web service to retrieve the portion of a catchment upstream of the point provided.

Usage

get_split_catchment(point, upstream = TRUE)

Arguments

point scf POINT including crs as created by: sf::st_sfc(sf::st_point(..., ...), crs)

upstream logical If TRUE, the entire drainage basin upstream of the point provided is returned in addition to the local catchment.

Value

sf data.frame containing the local catchment, the split portion and optionally the total drainage basin.

Examples

point <- sf::st_sfc(sf::st_point(x = c(-89.2158, 42.9561)), crs = 4326)

trace <- get_raindrop_trace(point)

if(inherits(trace, "sf")) {
  (snap_point <- sf::st_sfc(sf::st_point(trace$intersection_point[[1]]),
                            crs = 4326))
get_streamlevel

Get Streamlevel

Description

Applies a topological sort and calculates stream level. Algorithm: Terminal level paths are assigned level 1 (see note 1). Paths that terminate at a level 1 are assigned level 2. This pattern is repeated until no paths remain.

If a TRUE/FALSE coastal attribute is included, coastal terminal paths begin at 1 and internal terminal paths begin at 4 as is implemented by the NHD stream leveling rules.

Usage

get_streamlevel(x)
**get_streamorder**

**Arguments**

- **x**
  data.frame with levelpathi, dnlevelpat, and optionally a coastal flag. If no coastal flag is included, all terminal paths are assumed to be coastal.

**Value**

numeric stream order in same order as input

**Examples**

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

test_flowline <- data.frame(
  levelpathi = walker_flowline$LevelPathI,
  dnlevelpat = walker_flowline$DnLevelPat)

test_flowline$dnlevelpat[1] <- 0

(level <- get_streamlevel(test_flowline))
walker_flowline$level <- level

plot(sf::st_geometry(walker_flowline), lwd = walker_flowline$level, col = "blue")

test_flowline$coastal <- rep(FALSE, nrow(test_flowline))

(level <- get_streamlevel(test_flowline))

test_flowline$coastal[!test_flowline$dnlevelpat %in% test_flowline$levelpathi] <- TRUE

(level <- get_streamlevel(test_flowline))
```

**Description**

Applies a topological sort and calculates strahler stream order. Algorithm: If more than one upstream flowpath has an order equal to the maximum upstream order then the downstream flowpath is assigned the maximum upstream order plus one. Otherwise it is assigned the max upstream order.

**Usage**

`get_streamorder(x, status = TRUE)`

**Arguments**

- **x**
  data.frame with dendritic ID and toID columns.
- **status**
  logical show progress update messages?
get_terminal

Value
numeric stream order in same order as input

Examples

```r
global::get_terminal::get_terminal
```

get_terminal(x, outlets)

Arguments

- `x`: two column data.frame with IDs and toIDs. Names are ignored.
- `outlets`: IDs of outlet flowlines

Value
data.frame containing the terminal ID for each outlet

Examples

```r
global::get_terminal::source(system.file("extdata", "walker_data.R", package = "nhdplusTools"))
global::get_terminal::fl <- dplyr::select(prepare_nhdplus(walker_flowline, 0, 0),
  ID = COMID, toID = toCOMID)
global::get_terminal::outlet <- fl$ID[which(!fl$toID %in% fl$ID)]
global::get_terminal::plot(sf::st_geometry(walker_flowline), lwd = walker_flowline$order, col = "blue")
```
get_tocomid

get_terminal(fl, outlet)

---

get_tocomid  Get tocomid

Description

Given flowlines with fromnode and tonode attributes, will return a toid attribute that is the result of joining tonode and fromnode attributes. In the case that a terminalpa attribute is included, the join is executed by terminalpa group. This is done grouped by terminalpathID because duplicate node ids have been encountered across basins in some datasets. If `remove_coastal` is ‘TRUE’ (the default) either ftype or fcode are required.

Usage

```r
get_tocomid(
  x,
  return_dendritic = TRUE,
  missing = 0,
  remove_coastal = TRUE,
  add = TRUE
)
```

Arguments

- **x**  
  data.frame with comid, tonode, fromnode, and (optionally) divergence and terminalpa attributes.
- **return_dendritic**  
  logical if TRUE, a divergence attribute is required (2 indicates diverted path, 1 is main) and diverted paths will be treated as headwaters. If this is FALSE, the return value is a data.frame including the comid and tocomid attributes.
- **missing**  
  integer value to use for terminal nodes.
- **remove_coastal**  
  logical remove coastal features prior to generating tocomid values? ftype or fcode are required if ‘TRUE’. fcode == 56600 or fcode == ”Coastline” will be fremoved.
- **add**  
  logical if TRUE, a tocomid column will be added, otherwise a data.frame with two columns will be returned.

Value

data.frame containing comid and tocomid attributes or all attributes provided with comid and tocomid in the first and second columns.
Examples

```
source(system.file("extdata", "sample_flines.R", package = "nhdplusTools"))
tocomid <- get_tocomid(sample_flines)
tocomid <- get_tocomid(sample_flines, return_dendritic = FALSE)
```

---

**get_UM**

**Navigate Upstream Mainstem**

Description

Traverse NHDPlus network upstream main stem

Usage

```
get_UM(network, comid, distance = NULL, sort = FALSE, include = TRUE)
```

Arguments

- `network` : data.frame NHDPlus flowlines including at a minimum: COMID, Pathlength, LevelPathI, and Hydroseq.
- `comid` : integer identifier to start navigating from.
- `distance` : numeric distance in km to limit how many COMIDs are
- `sort` : if TRUE, the returned COMID vector will be sorted in order of distance from the input COMID (nearest to farthest)
- `include` : if TRUE, the input COMID will be included in the returned COMID vector returned. The COMID that exceeds the distance specified is returned.

Value

integer vector of all COMIDs upstream of the starting COMID along the mainstem

Examples

```
library(sf)

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

plot(sample_flines$geom)
start_COMID <- 11690196
UM_COMIDs <- get_UM(sample_flines, start_COMID)
plot(dplyr::filter(sample_flines, COMID %in% UM_COMIDs)$geom, col = "red", add = TRUE, lwd = 3)
UM_COMIDs <- get_UM(sample_flines, start_COMID, distance = 50)
```
get_UT

plot(dplyr::filter(sample_flines, COMID %in% UM_COMIDs)$geom,
     col = "blue", add = TRUE, lwd = 2)

get_UT

Navigate Upstream with Tributaries

Description

Traverse NHDPlus network upstream with tributaries

Usage

get_UT(network, comid, distance = NULL)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>network</td>
<td>data.frame NHDPlus flowlines including at a minimum: COMID, Pathlength, LENGTHKM, and Hydroseq.</td>
</tr>
<tr>
<td>comid</td>
<td>integer Identifier to start navigating from.</td>
</tr>
<tr>
<td>distance</td>
<td>numeric distance in km to limit how many COMIDs are returned. The COMID that exceeds the distance specified is returned.</td>
</tr>
</tbody>
</table>

Value

integer vector of all COMIDs upstream with tributaries of the starting COMID.

Examples

library(sf)
source(system.file("extdata", "sample_flines.R", package = "nhdplusTools"))
plot(sample_flines$geom)
start_COMID <- 11690196
UT_COMIDs <- get_UT(sample_flines, start_COMID)
plot(dplyr::filter(sample_flines, COMID %in% UT_COMIDs)$geom,
     col = "red", add = TRUE)

UT_COMIDs <- get_UT(sample_flines, start_COMID, distance = 50)
plot(dplyr::filter(sample_flines, COMID %in% UT_COMIDs)$geom,
     col = "blue", add = TRUE)
get_vaa  

**NHDPlusV2 Attribute Subset**

**Description**

Return requested NHDPlusv2 Attributes.

**Usage**

```r
get_vaa(
  atts = NULL,
  path = get_vaa_path(),
  download = TRUE,
  updated_network = FALSE
)
```

**Arguments**

- `atts` character: The variable names you would like, always includes `comid`
- `path` character: path where the file should be saved. Default is a persistent system data as retrieved by `nhdplusTools_data_dir`. Also see: `get_vaa_path`
- `download` logical: if TRUE, the default, will download VAA table if not found at path.
- `updated_network` logical: default FALSE. If TRUE, updated network attributes from E2NHD and National Water Model retrieved from doi:10.5066/P9W79I7Q.

**Details**

The VAA data is a aggregate table of information from the NHDPlusV2 elevslope.dbf(s), PlusFlowlineVAA.dbf(s); and NHDFlowlines. All data originates from the EPA NHDPlus Homepage [here](#). To see the location of cached data on your machine use `get_vaa_path`. To view aggregate data and documentation, see [here](#)

**Value**

data.frame containing requested VAA data

**Examples**

```r
## Not run:
# This will download the vaa file to the path from get_vaa_path()

get_vaa("slope")
get_vaa(c("slope", "lengthkm"))

get_vaa(updated_network = TRUE)
get_vaa("reachcode", updated_network = TRUE)
```
get_vaa_names

#cleanup if desired
unlink(dirname(get_vaa_path()), recursive = TRUE)

## End(Not run)

---

get_vaa_names  

Available NHDPlusV2 Attributes

Description

Find variables available from the NHDPlusV2 attribute data.frame

Usage

get_vaa_names(updated_network = FALSE)

Arguments

updated_network

logical default FALSE. If TRUE, updated network attributes from E2NHD and National Water Model retrieved from doi:10.5066/P9W7917Q.

Details

The VAA data is a aggregate table of information from the NHDPlusV2 eleve slope.dbf(s), PlusFlowlineVAA.dbf(s); and NHDFlowlines. All data originates from the EPA NHDPlus Homepage here. To see the location of cached data on your machine use get_vaa_path. To view aggregate data and documentation, see here

Value

character vector

Examples

## Not run:
# This will download the vaa file to the path from get_vaa_path()
get_vaa_names()

#cleanup if desired
unlink(dirname(get_vaa_path()), recursive = TRUE)

## End(Not run)
get_vaa_path

File path to value added attribute (vaa) Cache

**Description**

nhdplusTools will download and cache an ‘fst’ file with NHDPlusV2 attribute data sans geometry. This function returns the file path to the cached file. Will use the user data dir indicated by nhdplusTools_data_dir.

**Usage**

get_vaa_path(updated_network = FALSE)

**Arguments**

updated_network

logical default FALSE. If TRUE, returns path to updated network parameters. See get_vaa for more.

**Details**

The VAA data is a aggregate table of information from the NHDPlusV2 elevslope.dbf(s), PlusFlowlineVAA.dbf(s); and NHDFlowlines. All data originates from the EPA NHDPlus Homepage [here](#). To see the location of cached data on your machine use get_vaa_path. To view aggregate data and documentation, see [here](#)

**Value**

character file path

**Examples**

get_vaa_path()

get_vaa_path(updated_network = TRUE)

get_waterbodies

Find NHD Water Bodies

**Description**

Subsets NHD waterbody features by location (POINT), area (POLYGON), or set of IDs.

**Usage**

get_waterbodies(AOI = NULL, id = NULL, t_srs = NULL, buffer = 0.5)
get_waterbody_index

Arguments

AOI sf (MULTI)POINT or (MULTI)POLYGON. An 'area of interest' can be provided as either a location (sf POINT) or area (sf POLYGON) in any Spatial Reference System.

id NHD Waterbody COMID(s)

t_srs character (PROJ string or EPSG code) or numeric (EPSG code). A user specified - target -Spatial Reference System (SRS/CRS) for returned objects. Will default to the CRS of the input AOI if provided, and to 4326 for ID requests.

buffer numeric. The amount (in meters) to buffer a POINT AOI by for an extended search. Default = 0.5

Details

The returned object(s) will have the same Spatial Reference System (SRS) as the input AOI. If a individual or set of IDs are used to query, then the default geoserver CRS of EPSG:4326 is preserved. In all cases, a user-defined SRS can be passed to t_srs which will override all previous SRS’s (either input or default). All buffer and distance operations are handled internally using in EPSG:5070 Albers Equal Area projection

Value

a simple features (sf) object

get_waterbody_index  Get Waterbody Index

Description

given an sf point geometry column, return waterbody id, and COMID of dominant artificial path

Usage

get_waterbody_index(waterbodies, points, flines = NULL, search_radius = NULL)

Arguments

waterbodies sf data.frame of type POLYGON or MULTIPOLYGON including COMID attributes.

points sfc of type POINT

flines sf data.frame of type LINESTRING or MULTILINESTRING including COMID, WBAREACOMI, and Hydroseq attributes

search_radius units class with a numeric value indicating how far to search for a waterbody boundary in units of provided projection. Set units with set_units.
get_wb_outlet

Value
data.frame with two columns, COMID, in_wb_COMID, near_wb_COMID, near_wb_dist, and outlet_fline_COMID. Distance is in units of provided projection.

Examples

```r
source(system.file("extdata/sample_data.R", package = "nhdplusTools"))

waterbodies <- sf::st_transform(
    sf::read_sf(sample_data, "NHDWaterbody"), 5070)

points <- sf::st_transform(
    sf::st_sfc(sf::st_point(c(-89.356086, 43.079943)),
    crs = 4326), 5070)

gtwb <- get_waterbody_index(waterbodies, points,
    search_radius = units::set_units(500, "m"))
```
**get_xs_point**

wtbdy <- sf::read_sf(sample_data, "NHDWaterbody")

lake_COMID <- wtbdy$COMID[wtbdy$GNIS_NAME=='Lake Mendota 254']

get_wb_outlet(13293262, fline)

---

**get_xs_point**  
*Get Cross Section From Point (experimental)*

**Description**

Uses a cross section retrieval web services to retrieve a cross section given a point and specified width. Orientation is determined based on direction of a the flowline found near point. This function uses a 10m National Elevation Dataset request on the back end.

**Usage**

```r
get_xs_point(point, width, num_pts)
```

**Arguments**

- **point**: sfc POINT including crs as created by: `sf::st_sfc(sf::st_point(..., ...), crs)`
- **width**: Cross section width in meters.
- **num_pts**: numeric number of points to retrieve along the cross section.

**Value**

sf data.frame containing points retrieved.

**Examples**

```r
point <- sf::st_sfc(sf::st_point(x = c(-105.97218, 36.17592)), crs = 4326)
(xs <- get_xs_point(point, 300, 100))

if(inherits(xs, "sf") { 
  bbox <- sf::st_bbox(xs) + c(-0.005, -0.005, 0.005, 0.005)
  nhdplusTools::plot_nhdplus(bbox = bbox, cache_data = FALSE)
  plot(sf::st_transform(sf::st_geometry(xs), 3857), pch = ".", add = TRUE, col = "red")
  plot(sf::st_transform(sf::st_sfc(point, crs = 4326), 3857), add = TRUE)
  plot(xs$distance_m, xs$elevation_m)
```
get_xs_points

get_xs_points  Get Cross Section Endpoints (experimental)

Description

Uses a cross section retrieval web services to retrieve a cross section between two endpoints.

Usage

get_xs_points(point1, point2, num_pts, res = 1)

Arguments

- **point1**: sfc POINT including crs as created by: sf::st_sfc(sf::st_point(., .), crs)
- **point2**: sfc POINT including crs.
- **num_pts**: numeric number of points to retrieve along the cross section.
- **res**: integer resolution of 3D Elevation Program data to request. Must be one of: 1, 3, 5, 10, 30, 60.

Value

sf data.frame containing points retrieved.

Examples

point1 <- sf::st_sfc(sf::st_point(x = c(-105.9667, 36.17602)), crs = 4326)
point2 <- sf::st_sfc(sf::st_point(x = c(-105.97768, 36.17526)), crs = 4326)
(xs <- get_xs_points(point1, point2, 100))

if(inherits(xs, "sf")) {
  bbox <- sf::st_bbox(xs) + c(-0.005, -0.005, 0.005, 0.005)
  nhdplusTools::plot_nhdplus(bbox = bbox, cache_data = FALSE)
  plot(sf::st_transform(sf::st_geometry(xs), 3857), pch = ".", add = TRUE, col = "red")
  plot(sf::st_transform(sf::st_sfc(point1, crs = 4326), 3857), add = TRUE)
  plot(sf::st_transform(sf::st_sfc(point2, crs = 4326), 3857), add = TRUE)
  plot(xs$distance_m, xs$elevation_m)
}
**Description**

creates a node topology table from an edge topology

**Usage**

```r
make_node_topology(x, add_div = NULL, add = TRUE)
```

**Arguments**

- **x**: data.frame with an identifier and to identifier in the first and second columns.
- **add_div**: data.frame containing id and toid diverted paths to add. Should have id and toid fields in the first and second columns. Names are not used.
- **add**: logical if TRUE, a tocomid column will be added, otherwise a data.frame with two columns will be returned.

**Value**

data.frame containing id, fromnode, and tonode attributes or all attributes provided with id, fromnode and tonode in the first three columns.

**Examples**

```r
source(system.file("extdata/new_hope_data.R", package = "nhdplusTools"))
x <- dplyr::select(get_tocomid(
  dplyr::select(new_hope_flowline, COMID, FromNode, ToNode, Divergence, FTYPE, 
  AreaSqKM, LENGTHKM, GNIS_ID)
), -tonode, -fromnode)
head(y <- make_node_topology(x))

# just the divergences which have unique fromids in x but don't in new hope.
div <- get_tocomid(dplyr::select(new_hope_flowline, COMID, FromNode, ToNode),
  return_dendritic = FALSE, remove_coastal = FALSE)
div <- div[div$tocomid %in%
  new_hope_flowline$COMID[new_hope_flowline$Divergence == 2],]
y <- make_node_topology(x, div)
```
make_standalone

Description

Cleans up and prepares NHDPlusHR regional data for use as complete NHDPlus data. The primary modification applied is to ensure that any flowpath that exits the domain is labeled as a terminal path and attributes are propagated upstream such that the domain is independently complete.

Usage

make_standalone(flowlines)

Arguments

flowlines sf data.frame of NHDPlusHR flowlines.

Value

sf data.frame containing standalone network

Examples

library(dplyr)
library(sf)
source(system.file("extdata/nhdplushr_data.R", package = "nhdplusTools"))

(outlet <- filter(hr_data$NHDFlowline, Hydroseq == min(Hydroseq)))
nrow(filter(hr_data$NHDFlowline, TerminalPa == outlet$Hydroseq))

hr_data$NHDFlowline <- make_standalone(hr_data$NHDFlowline)

(outlet <- filter(hr_data$NHDFlowline, Hydroseq == min(Hydroseq)))
nrow(filter(hr_data$NHDFlowline, TerminalPa == outlet$Hydroseq))

source(system.file("extdata/nhdplushr_data.R", package = "nhdplusTools"))

# Remove mainstem and non-dendritic stuff.
subset <- filter(hr_data$NHDFlowline,
    StreamLeve > min(hr_data$NHDFlowline$StreamLeve) &
    StreamOrde == StreamCalc)

subset <- subset_nhdplus(subset$COMID, nhdplus_data = hr_gpkg)$NHDFlowline

plot(sf::st_geometry(hr_data$NHDFlowline))

flowline_mod <- make_standalone(subset)

terminals <- unique(flowline_mod$TerminalPa)
colors <- sample(hcl.colors(length(terminals), palette = "Zissou 1"))

for(i in 1:length(terminals)) {
  fl <- flowline_mod[flowline_mod$TerminalPa == terminals[i], ]
  plot(st_geometry(fl), col = colors[i], lwd = 2, add = TRUE)
}

ol <- filter(flowline_mod, TerminalFl == 1 & TerminalPa %in% terminals)

plot(st_geometry(ol), lwd = 2, add = TRUE)

---

**map_nhdplus**  
*Make Interactive Map of NHDPlus*

**Description**

Given a list of outlets, get their basin boundaries and network and return a leaflet map in EPSG:4326.

**Usage**

```r
map_nhdplus(
  outlets = NULL,
  bbox = NULL,
  streamorder = NULL,
  nhdplus_data = NULL,
  gpkg = NULL,
  flowline_only = NULL,
  plot_config = NULL,
  overwrite = TRUE,
  cache_data = NULL,
  return_map = FALSE
)
```

**Arguments**

- **outlets** list of nldi outlets. Other inputs are coerced into nldi outlets, see details.
- **bbox** object of class bbox with a defined crs. See examples.
- **streamorder** integer only streams of order greater than or equal will be returned
- **nhdplus_data** geopackage containing source nhdplus data (omit to download)
- **gpkg** path and file with .gpkg ending. If omitted, no file is written.
- **flowline_only** boolean only subset and plot flowlines only, default=FALSE
- **plot_config** list containing plot configuration, see details.
- **overwrite** passed on the `subset_nhdplus`.
map_nhdplus

cache_data  character path to rds file where all plot data can be cached. If file doesn’t exist, it will be created. If set to FALSE, all caching will be turned off – this includes basemap tiles.

return_map  if FALSE (default), a data.frame of plot data is returned invisibly in NAD83 Lat/Lon, if TRUE the leaflet object is returned

Details

map_nhdplus supports several input specifications. An unexported function "as_outlet" is used to convert the outlet formats as described below.

1. if outlets is omitted, the bbox input is required and all nhdplus data in the bounding box is plotted.
2. If outlets is a list of integers, it is assumed to be NHDPlus IDs (comids) and all upstream tributaries are plotted.
3. if outlets is an integer vector, it is assumed to be all NHDPlus IDs (comids) that should be plotted. Allows custom filtering.
4. If outlets is a character vector, it is assumed to be NWIS site ids.
5. if outlets is a list containing only characters, it is assumed to be a list of nldi features and all upstream tributaries are plotted.
6. if outlets is a data.frame with point geometry, a point in polygon match is performed and upstream with tributaries from the identified catchments is plotted.

See plot_nhdplus for details on plot configuration.

Value

data.frame or leaflet map (see return_map)

Examples

map_nhdplus("05428500")
map_nhdplus("05428500", streamorder = 2)
map_nhdplus(list(13293970, 13293750))
source(system.file("extdata/sample_data.R", package = "nhdplusTools"))
map_nhdplus(list(13293970, 13293750), streamorder = 3, nhdplus_data = sample_data)
#return leaflet object
map_nhdplus("05428500", return_map = TRUE)
navigate_network

navigate_network

Navigate Network

Description

Provides a full feature network navigation function that will work with local or web service data. Parameter details provide context.

Usage

navigate_network(
  start, 
  mode = "UM", 
  network = NULL, 
  output = "flowlines", 
  distance_km = 10, 
  trim_start = FALSE, 
  trim_stop = FALSE, 
  trim_tolerance = 5
)

Arguments

start list, integer, sf, or sfc if list must be a valid NLDI feature if integer must be a valid comid. If sf, must contain a "comid" field.

mode character chosen from c(UM, DM, UT, or DD)

network sf should be compatible with network navigation functions If NULL, network will be derived from requests to the NLDI

output character flowline or a valid NLDI data source

distance_km numeric distance to navigate in km

trim_start logical should start be trimmed or include entire catchment?

trim_stop logical should stop(s) be trimmed or include entire catchment(s)? # Not supported

trim_tolerance numeric from 0 to 100 percent of flowline length. If amount to trim is less than this tolerance, no trim will be applied.

Examples

navigate_network(list(featureSource = "nwissite", featureID = "USGS-06287800"), 
  "UM", 
  output = "flowlines", 
  trim_start = TRUE)
navigate_nldi

Navigate NLDI

Description

Navigate the Network Linked Data Index network.

Usage

navigate_nldi(
  nldi_feature,
  mode = "upstreamMain",
  data_source = "flowlines",
  distance_km = 10
)

Arguments

nldi_feature  list with names 'featureSource' and 'featureID' where 'featureSource' is derived from the "source" column of the response of dataRetrieval::get_nldi_sources() and the 'featureID' is a known identifier from the specified 'featureSource'.

mode  character chosen from ("UM", "UT", "DM", "DD"). See examples.

data_source  character chosen from 'source' column of the response of dataRetrieval::get_nldi_sources() or empty string for flowline geometry.

distance_km  numeric distance in km to stop navigating.

Value

sf data.frame with result
Examples

```r
library(sf)
library(dplyr)
nldi_nwis <- list(featureSource = "nwissite", featureID = "USGS-05428500")
navigate_nldi(nldi_feature = nldi_nwis,
mode = "upstreamTributaries")$UT %>%
st_geometry() %>%
plot()
navigate_nldi(nldi_feature = nldi_nwis,
mode = "UM")$UM %>%
st_geometry() %>%
plot(col = "blue", add = TRUE)

nwissite <- navigate_nldi(nldi_feature = nldi_nwis,
mode = "UT",
data_source = "nwissite")$UT_nwissite
st_geometry(nwissite) %>%
plot(col = "green", add = TRUE)
nwissite
```

---

**nhdplusTools_data_dir**  
Get or set nhdplusTools data directory

**Description**

If left unset, will return the user data dir as returned by `tools::R_user_dir` for this package.

**Usage**

```r
nhdplusTools_data_dir(dir = NULL)
```

**Arguments**

- `dir`  
  path of desired data directory

**Value**

character path of data directory (silent when setting)
nhdplus_path

**Examples**

nhdplusTools_data_dir()

nhdplusTools_data_dir("demo")

nhdplusTools_data_dir(tools::R_user_dir("nhdplusTools"))

---

**nhdplus_path** | **NHDPlus Data Path**

**Description**

Allows specification of a custom path to a source dataset. Typically this will be the national seamless dataset in geodatabase or geopackage format.

**Usage**

nhdplus_path(path = NULL, warn = FALSE)

**Arguments**

- **path** character path ending in .gdb or .gpkg
- **warn** boolean controls whether warning an status messages are printed

**Value**

0 (invisibly) if set successfully, character path if no input.

**Examples**

nhdplus_path("/data/NHDPlusV21_National_Seamless.gdb")

nhdplus_path("/data/NHDPlusV21_National_Seamless.gdb", warn=FALSE)

nhdplus_path()
**Description**

Given a list of outlets, get their basin boundaries and network and return a plot in EPSG:3857 Web Mercator Projection.

**Usage**

```r
plot_nhdplus(
  outlets = NULL,
  bbox = NULL,
  streamorder = NULL,
  nhdplus_data = NULL,
  gpkg = NULL,
  plot_config = NULL,
  basemap = "cartolight",
  add = FALSE,
  actually_plot = TRUE,
  overwrite = TRUE,
  flowline_only = NULL,
  cache_data = NULL,
  ...
)
```

**Arguments**

- **outlets**: list of nldi outlets. Other inputs are coerced into nldi outlets, see details.
- **bbox**: object of class bbox with a defined crs. See examples.
- **streamorder**: integer only streams of order greater than or equal will be returned
- **nhdplus_data**: geopackage containing source nhdplus data (omit to download)
- **gpkg**: path and file with .gpkg ending. If omitted, no file is written.
- **plot_config**: list containing plot configuration, see details.
- **basemap**: character indicating which basemap type to use. Chose from: `osm.types`
- **add**: boolean should this plot be added to an already built map.
- **actually_plot**: boolean actually draw the plot? Use to get data subset only.
- **overwrite**: passed on the `subset_nhdplus`
- **flowline_only**: boolean only subset and plot flowlines only, default=FALSE
- **cache_data**: character path to rds file where all plot data can be cached. If file doesn’t exist, it will be created. If set to FALSE, all caching will be turned off – this includes basemap tiles.
- **...**: parameters passed on to rosm.
plot_nhdplus supports several input specifications. An unexported function "as_outlet" is used to convert the outlet formats as described below.

1. If outlets is omitted, the bbox input is required and all nhdplus data in the bounding box is plotted.
2. If outlets is a list of integers, it is assumed to be NHDPlus IDs (comids) and all upstream tributaries are plotted.
3. If outlets is an integer vector, it is assumed to be all NHDPlus IDs (comids) that should be plotted. Allows custom filtering.
4. If outlets is a character vector, it is assumed to be NWIS site ids.
5. If outlets is a list containing only characters, it is assumed to be a list of nldi features and all upstream tributaries are plotted.
6. If outlets is a data.frame with point geometry, a point in polygon match is performed and upstream with tributaries from the identified catchments is plotted.

The plot_config parameter is a list with names "basin", "flowline", "outlets", "network_wtbd", and "off_network_wtbd". The following shows the defaults that can be altered.

1. basin
   list(lwd = 1, col = NA, border = "black")
2. flowline
   list(lwd = 1, col = "blue")
3. outlets
   list(default = list(col = "black", border = NA, pch = 19, cex = 1),
        nwissite = list(col = "grey40", border = NA, pch = 17, cex = 1),
        huc12pp = list(col = "white", border = "black", pch = 22, cex = 1),
        wqp = list(col = "red", border = NA, pch = 20, cex = 1))
4. network_wtbd list(lwd = 1, col = "lightblue", border = "black")
5. off_network_wtbd list(lwd = 1, col = "darkblue", border = "black")

If adding additional layers to the plot, data must be projected to EPSG:3857 with 'sf::st_transform(x, 3857)' prior to adding to the plot.

Value

data.frame plot data is returned invisibly in NAD83 Lat/Lon.

Examples

options("rgdal_show_exportToProj4_warnings"="none")
# Beware plot_nhdplus caches data to the default location.
# If you do not want data in "user space" change the default.
old_dir <- nhdplusTools::nhdplusTools_data_dir()
nhdplusTools_data_dir(tempdir())
plot_nhdplus("05428500")

plot_nhdplus("05428500", streamorder = 2)

plot_nhdplus(list(13293970, 13293750))

source(system.file("extdata/sample_data.R", package = "nhdplusTools"))

plot_nhdplus(list(13293970, 13293750), streamorder = 3, nhdplus_data = sample_data)

plot_nhdplus(list(list("comid", "13293970"),
    list("nwissite", "USGS-05428500"),
    list("huc12pp", "070900020603"),
    list("huc12pp", "070900020602"),
    streamorder = 2,
    nhdplus_data = sample_data)

plot_nhdplus(sf::st_as_sf(data.frame(x = -89.36083,
    y = 43.08944),
    coords = c("x", "y"), crs = 4326),
    streamorder = 2,
    nhdplus_data = sample_data)

bbox <- sf::st_bbox(c(xmin = -89.43, ymin = 43, xmax = -89.28, ymax = 43.1),
    crs = "+proj=longlat +datum=WGS84 +no_defs")
fline <- sf::read_sf(sample_data, "NHDFlowline_Network")
comids <- nhdplusTools::get_UT(fline, 13293970)

plot_nhdplus(comids)

#' # With Local Data
plot_nhdplus(bbox = bbox, nhdplus_data = sample_data)

# With downloaded data
plot_nhdplus(bbox = bbox, streamorder = 3)

# Can also plot on top of the previous!
plot_nhdplus(bbox = bbox, nhdplus_data = sample_data,
    plot_config = list(flowline = list(lwd = 0.5)))
plot_nhdplus(comids, nhdplus_data = sample_data, streamorder = 3, add = TRUE,
prepare_nhdplus

prep NHDPlus Data

Description

Function to prep NHDPlus data for use by nhdplusTools functions

Usage

```r
prepare_nhdplus(
  flines,
  min_network_size = 0,
  min_path_length = 0,
  min_path_size = 0,
  purge_non_dendritic = TRUE,
  warn = TRUE,
  error = TRUE,
  skip_toCOMID = FALSE,
  align_names = TRUE
)
```

Arguments

- `flines` data.frame NHDPlus flowlines including: COMID, LENGTHKM, FTYPE (or FCODE), TerminalFl, FromNode, ToNode, TotDASqKM, StartFlag, StreamOrde, StreamCalc, TerminalPa, Pathlength, and Divergence variables.
- `min_network_size` numeric Minimum size (sqkm) of drainage network to include in output.
- `min_path_length` numeric Minimum length (km) of terminal level path of a network.
- `min_path_size` numeric Minimum size (sqkm) of outlet level path of a drainage basin. Drainage basins with an outlet drainage area smaller than this will be removed.
- `purge_non_dendritic` logical Should non dendritic paths be removed or not.
- `warn` logical controls whether warning an status messages are printed
- `error` logical controls whether to return potentially invalid data with a warning rather than an error
- `skip_toCOMID` logical if TRUE, toCOMID will not be added to output.
- `align_names` logical
Value

data.frame ready to be used with the refactor_flowlines function.

Examples

```r
source(system.file("extdata", "sample_flines.R", package = "nhdplusTools"))
prepare_nhdplus(sample_flines,
  min_network_size = 10,
  min_path_length = 1,
  warn = FALSE)
```

Description

correctly renames the geometry column of a sf object.

Usage

```r
rename_geometry(g, name)
```

Arguments

g sf data.table
name character name to be used for geometry

Examples

```r
(g <- sf::st_sf(a=3, geo = sf::st_sfc(sf::st_point(1:2))))
rename_geometry(g, "geometry")
```
rescale_measures  
Rescale reachcode measure to comid flowline measure

Description
Given a reachcode measure and the from and to measure for a comid flowline, returns the measure along the comid flowline. This is a utility specific to the NHDPlus data model where many comid flowlines make up a single reachcode / reach. "Measures" are typically referenced to reaches. Flowlines have a stated from-measure / to-measure. In some cases it is useful to rescale the measure such that it is relative only to the flowline.

from is downstream – 0 is the outlet to is upstream – 100 is the inlet

Usage
rescale_measures(measure, from, to)

Arguments
measure  
numeric reach measure between 0 and 100
from  
numeric flowline from-measure relative to the reach
to  
numeric flowline to-measure relative to the reach

Value
numeric rescaled measure

Examples
rescale_measures(40, 0, 50)
rescale_measures(60, 50, 100)

rpu_boundaries  
RPU Boundaries Raster Processing Unit boundaries

Description
RPU Boundaries Raster Processing Unit boundaries

Usage
rpu_boundaries

Format
An object of class "sf"
**stage_national_data**  
*Stage NHDPlus National Data (deprecated)*

**Description**  
Breaks down the national geo database into a collection of quick to access R binary files.

**Usage**  
```
stage_national_data(  
  include = c("attribute", "flowline", "catchment"),  
  output_path = NULL,  
  nhdplus_data = NULL,  
  simplified = TRUE  
)
```

**Arguments**  
- **include** character vector containing one or more of: "attributes", "flowline", "catchment".
- **output_path** character path to save the output to defaults to the directory of the nhdplus_data.
- **nhdplus_data** character path to the .gpkg or .gdb containing the national seamless dataset. Not required if `nhdplus_path` has been set.
- **simplified** boolean if TRUE (the default) the CatchmentSP layer will be included.

**Details**  
"attributes" will save ‘NHDFlowline_Network’ attributes as a separate data.frame without the geometry. The others will save the ‘NHDFlowline_Network’ and ‘Catchment’ or ‘CatchmentSP’ (per the ‘simplified’ parameter) as sf data.frames with superfluous Z information dropped.  
The returned list of paths is also added to the nhdplusTools_env as "national_data".

**Value**  
list containing paths to the .rds files.

**Examples**  
```
source(system.file("extdata/sample_data.R", package = "nhdplusTools"))

stage_national_data(nhdplus_data = sample_data, output_path = tempdir())
```
st_compatibalize  make spatial inputs compatible

Description

makes sf1 compatible with sf2 by projecting into the projection of 2 and ensuring that the geometry columns are the same name.

Usage

st_compatibalize(sf1, sf2)

Arguments

sf1 sf data.frame
sf2 sf data.frame

Examples

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

(one <- dplyr::select(sample_flines))
(two <- sf::st_transform(one, 5070))

attr(one, "sf_column") <- "geotest"
names(one)[names(one) == "geom"] <- "geotest"

st_compatibalize(one, two)

subset_nhdplus  Subset NHDPlus

Description

Saves a subset of the National Seamless database or other nhdplusTools compatible data based on a specified collection of COMIDs. This function uses get_nhdplus for the "download" data source but returns data consistent with local data subsets in a subset file.
subset_nhdplus

Usage

subset_nhdplus(
  comids = NULL,
  output_file = NULL,
  nhdplus_data = NULL,
  bbox = NULL,
  simplified = TRUE,
  overwrite = FALSE,
  return_data = TRUE,
  status = TRUE,
  flowline_only = NULL,
  streamorder = NULL,
  out_prj = 4269
)

Arguments

comids integer vector of COMIDs to include.
output_file character path to save the output to defaults to the directory of the nhdplus_data.
nhdplus_data character path to the .gpkg or .gdb containing the national seamless database, a subset of NHDPlusHR, or "download" to use a web service to download NHD-PlusV2.1 data. Not required if nhdplus_path has been set or the default has been adopted. See details for more.
bbox object of class "bbox" as returned by sf::st_bbox in Latitude/Longitude. If no CRS is present, will be assumed to be in WGS84 Latitude Longitude.
simplified boolean if TRUE (the default) the CatchmentSP layer will be included. Not relevant to the "download" option or NHDPlusHR data.
overwrite boolean should the output file be overwritten
return_data boolean if FALSE path to output file is returned silently otherwise data is returned in a list.
status boolean should the function print status messages
flowline_only boolean WARNING: experimental if TRUE only the flowline network and attributes will be returned
streamorder integer only streams of order greater than or equal will be downloaded. Not implemented for local data.
out_prj character override the default output CRS of NAD83 lat/lon (EPSG:4269)

Details

This function relies on the National Seamless Geodatabase or Geopackage. It can be downloaded here.

The "download" option of this function should be considered preliminary and subject to revision. It does not include as many layers and may not be available permanently.
Value

character path to the saved subset geopackage

Examples

```r
source(system.file("extdata/sample_data.R", package = "nhdplusTools"))

nhdplus_path(sample_data)

staged_nhdplus <- stage_national_data(output_path = tempdir())

sample_flines <- readRDS(staged_nhdplus$flowline)

geom_col <- attr(sample_flines, "sf_column")

plot(sample_flines[[geom_col]], lwd = 3)

start_point <- sf::st_sfc(sf::st_point(c(-89.362239, 43.090266)), crs = 4326)

plot(start_point, cex = 1.5, lwd = 2, col = "red", add = TRUE)

start_comid <- discover_nhdplus_id(start_point)

comids <- get_UT(sample_flines, start_comid)

plot(dplyr::filter(sample_flines, COMID %in% comids)[[geom_col]], add = TRUE, col = "red", lwd = 2)

output_file <- tempfile(fileext = ".gpkg")

subset_nhdplus(comids = comids,
               output_file = output_file,
               nhdplus_data = sample_data,
               overwrite = TRUE,
               status = TRUE)

sf::st_layers(output_file)

catchment <- sf::read_sf(output_file, "CatchmentSP")

plot(catchment[[attr(catchment, "sf_column")]], add = TRUE)

waterbody <- sf::read_sf(output_file, "NHDWaterbody")

plot(waterbody[[attr(waterbody, "sf_column")]],
     col = rgb(0, 0, 1, alpha = 0.5), add = TRUE)

# Cleanup temp
```
sapply(staged_nhdplus, unlink)
unlink(output_file)

# Download Option:
subset_nhdplus(comids = comids,
    output_file = output_file,
    nhdplus_data = "download",
    overwrite = TRUE,
    status = TRUE, flowline_only = FALSE)

sf::st_layers(output_file)

# NHDPlusHR
source(system.file("extdata/nhdplushr_data.R", package = "nhdplusTools"))

up_ids <- get_UT(hr_data$NHDFlowline, 15000500028335)

sub_gpkg <- file.path(work_dir, "sub.gpkg")
sub_nhdhr <- subset_nhdplus(up_ids, output_file = sub_gpkg,
    nhdplus_data = hr_gpkg, overwrite = TRUE)

sf::st_layers(sub_gpkg)
names(sub_nhdhr)

plot(sf::st_geometry(hr_data$NHDFlowline), lwd = 0.5)
plot(sf::st_geometry(sub_nhdhr$NHDFlowline), lwd = 0.6, col = "red", add = TRUE)

unlink(output_file)
unlink(sub_gpkg)

---

**subset_rpu**

**Subset by Raster Processing Unit**

**Description**

Given flowlines and an rpu_code, performs a network-safe subset such that the result can be used in downstream processing. Has been tested to work against the entire NHDPlusV2 domain and satisfies a number of edge cases.

**Usage**

subset_rpu(fline, rpu, run_make_standalone = TRUE, strict = FALSE)

**Arguments**

- **fline**: sf data.frame NHD Flowlines with comid, pathlength, lengthkm, hydseq, levelpathi, rpuid, and arbolatesu (dnhydseq is required if tocomid is not provided).
subset_vpu

rpu character e.g. "01a"
run_make_standalone logical default TRUE should the run_make_standalone function be run on result?
strict logical if TRUE, paths that extend outside the RPU but have no tributaries in the upstream RPU will be included in the output.

Value
data.frame containing subset network

Examples

source(system.file("extdata/sample_data.R", package = "nhdplusTools"))
sample_flines <- sf::read_sf(sample_data, "NHDFlowline_Network")
subset_rpu(sample_flines, rpu = "07b")

subset_vpu  Subset by Vector Processing Unit

Description
Calls subset_rpu for all raster processing units for the requested vector processing unit.

Usage
subset_vpu(fline, vpu, include_null_rpuid = TRUE, run_make_standalone = TRUE)

Arguments
fline sf data.frame NHD Flowlines with comid, pathlength, lengthkm, hydoseq, lengthpathi, rpuid, vpuid, and arbolatesu (dnhydoseq is required if tocomid is not provided).
vpu character e.g. "01"
include_null_rpuid logical default TRUE. Note that there are some flowlines that may have a NULL rpuid but be included in the vector processing unit.
run_make_standalone logical default TRUE should the run_make_standalone function be run on result?

Value
data.frame containing subset network
Examples

```r
source(system.file("extdata/sample_data.R", package = "nhdplusTools"))

sample_flines <- sf::read_sf(sample_data, "NHDFlowline_Network")

subset_vpu(sample_flines, "07")
```

---

### Description

VPU Boundaries Vector Processing Unit boundaries

### Usage

`vpu_boundaries`

### Format

An object of class "sf"
Index

* data
  rpu_boundaries, 68
  vpu_boundaries, 75

add_plus_network_attributes, 3
adist, 7
align_nhdplus_names, 5

calculate_arbolute_sum, 5
calculate_total_drainage_area, 6

disambiguatate_flowline_indexes, 7
discover_nhdplus_id, 8
discover_nldi_characteristics, 9
download_nhdplusr, 10
download_nhdplusv2, 11
download_rf1, 12
download_vaa, 12
download_wbd, 13

fix_flowdir, 14

get_boundaries, 15
get_DD, 15
get_DM, 16
get_elev_along_path, 17
get_flowline_index, 7, 18, 24
get_gagesII, 20
get_hr_data, 21, 28
get_huc, 21
get_huc12, 22
get_huc8, 10, 23
get_hydro_location, 24
get_levelpaths, 4, 24
get_nhdarea, 25
get_nhdplus, 26, 70
get_nhdplusr, 21, 28
get_nldi_basin, 29
get_nldi_characteristics, 30
get_nldi_feature, 31
get_nldi_index, 32
get_nldi_sources, 9
get_node, 32
get_nwis, 33
get_partiial_length, 34
get_path_lengths, 36
get_path_members, 37
get_pathlength, 35
get_pfaf, 38
get_raindrop_trace, 9, 39
get_sorted, 40
get_split_catchment, 41
get_streamlevel, 42
get_streamorder, 43
get_terminal, 44
get_tocomid, 45
get_ULM, 46
get_UT, 47
get_vaa, 48, 50
get_vaa_names, 49
get_vaa_path, 13, 48–50, 50
get_waterbodies, 50
get_waterbody_index, 51
get_wb_outlet, 52
get_xs_point, 53
get_xs_points, 54

make_node_topology, 55
make_standalone, 28, 56
map_nhdplus, 57

navigate_network, 59
navigate_nldi, 60
nhdplus_path, 62, 69, 71
nhdplusTools_data_dir, 13, 48, 50, 61

osm.types, 63

plot_nhdplus, 58, 63
prepare_nhdplus, 66
rename_geometry, 67
rescale_measures, 68
rpu_boundaries, 68

set_units, 51
st_compatibalize, 70
st_crs, 21
st_layers, 21
stage_national_data, 69
subset_nhdplus, 57, 63, 70
subset_rpu, 73, 74
subset_vpu, 74

vpu_boundaries, 75