Package ‘SSNbler’

March 26, 2025

Title Assemble 'SSN' Objects

Version 1.1.0

Description Import, create and assemble data needed to fit spatial-statistical stream-network models using the 'SSN2' package for 'R'. Streams, observations, and prediction locations are represented as simple features and specific tools provided to define topological relationships between features; calculate the hydrologic distances (with flow-direction preserved) and the spatial additive function used to weight converging stream segments; and export the topological, spatial, and attribute information to an `SSN` (spatial stream network) object, which can be efficiently stored, accessed and analysed in 'R'. A detailed description of methods used to calculate and format the spatial data can be found in Peterson, E.E. and Ver Hoef, J.M., (2014) <doi:10.18637/jss.v056.i02>.

License GPL (>= 3)

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Depends R (>= 3.5.0)

Imports sf, dplyr, pdist, stats, utils, igraph, RSQLite, withr, SSN2, doParallel, foreach

Suggests ggplot2, knitr, rmarkdown, testthat (>= 3.0.0)

URL `https://github.com/pet221/SSNbler`

BugReports `https://github.com/pet221/SSNbler/issues`

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Erin Peterson [aut, cre] (<`https://orcid.org/0000-0003-2992-0372`>), Michael Dumelle [aut] (<`https://orcid.org/0000-0002-3393-5529`>), Alan Pearse [aut] (<`https://orcid.org/0000-0002-4133-8548`>), Dan Teleki [ctb], Jay M. Ver Hoef [ctb] (<`https://orcid.org/0000-0003-4302-6895`>)

Maintainer Erin Peterson <`erin@peterson-consulting.com`>

Repository CRAN

Date/Publication 2025-03-26 03:40:02 UTC

Contents

`accum_edges` *Accumulate edge values downstream*

Description

Accumulate (sum) edge values downstream in a Landscape Network (LSN)

Usage

```
accum_edges(
  edges,
  lsn_path,
  sum_col,
  acc_col,
  save_local = TRUE,
  overwrite = FALSE,
  verbose = TRUE
)
```

Arguments

`edges`	An `sf` object with LINESTRING geometry created using `lines_to_lsn`.
`lsn_path`	Local pathname to a directory in character format specifying where relationships.csv resides, which is created using `link{lines_to_lsn}`.
`sum_col`	Name of an existing column in `edges` (character format) that contains the value to sum downstream.

acc_col	Name of the new column in edges (character format) where the accumulated sum_col values will be stored (see details).
save_local	Logical indicating whether the updated edges should be saved to lsn_path in GeoPackage format. Defaults to TRUE.
overwrite	A logical indicating whether results should be overwritten if acc_col already exists in edges or edges.gpkg already exists in lsn_path and save_local = TRUE.
verbose	Logical. Indicates whether messages about the function progress should be printed to the console. Defaults to TRUE.

Details

accum_edges sums (i.e. accumulates) numeric values in edges downstream, from headwater or source features to each network outlet feature (i.e. most downstream line feature in each subnetwork). Missing values are not allowed in sum_col and should be replaced with 0, or any other meaningful numeric value, prior to running accum_edges. The acc_col returned contains the sum of sum_col found in upstream edges, in addition the the sum_col for the edge itself. As such, acc_col represents the cumulative upstream sum of sum_col for the downstream node of each line feature in edges.

Value

An sf object representing edges in the LSN, with a new acc_col column. If save_local = TRUE, edges is saved to lsn_path in GeoPackage (.gpkg) format.

Examples

```
# Get temporary directory, where the example LSN will be stored
# locally.
temp_dir <- tempdir()

# Build the LSN. When working with your own data, lsn_path will be
# a local folder of your choice rather than a temporary directory.
edges <- lines_to_lsn(
  streams = MF_streams,
  lsn_path = temp_dir,
  snap_tolerance = 1,
  check_topology = FALSE,
  overwrite = TRUE,
  verbose = FALSE
)

# Accumulate RCA area (rcaAreaKm2) downstream and store in a new
# column named WArea_km2
edges <- accum_edges(
  edges = edges,
  lsn_path = temp_dir,
  sum_col = "rcaAreaKm2",
  acc_col = "WArea_km2",
  save_local = FALSE,
  overwrite = TRUE,
  verbose = FALSE
)

summary(edges$WArea_km2)
```

afv_edges *Calculate the additive function value for edges in a LSN*

Description

Calculate the additive function value for each edge feature in a Landscape Network (LSN)

Usage

```
afv_edges(
  edges,
  lsn_path,
  infl_col,
  segpi_col,
  afv_col,
  save_local = TRUE,
  overwrite = TRUE
)
```

Arguments

edges	An sf object with LINESTRING geometry created using lines_to_lsn.
lsn_path	Local pathname to a directory in character format specifying where relationships.csv resides, which is created using lines_to_lsn.
infl_col	Name of the existing column in the edges data.frame that will be used to generate the segment proportional influence (segment PI) for each line feature, in character format.
segpi_col	Name of the new column created in edges that will store the new segment proportional influence values for each feature, in character format.
afv_col	Name of the new column created in edges to store the additive function value for each feature, in character format.
save_local	Logical indicating whether the updated edges should be saved to lsn_path in GeoPackage format. Defaults to TRUE.
overwrite	A logical indicating whether results should be overwritten if segpi_col and/or afv_col already exists in edges, or edges.gpkg already exists in lsn_path and save_local = TRUE. Default = TRUE.

Details

Spatial weights are used when fitting statistical models with 'SSN2' to split the tail-up covariance function upstream of network confluences, which allows for the disproportionate influence of one upstream edge over another (e.g., a large stream channel converges with a smaller one) on downstream values. Calculating the spatial weights is a four-step process:

1. calculating the segment proportional influence (PI) values for the edges,
2. calculating the additive function values (AFVs) for the edges,
3. calculating the AFVs for the observed and prediction sites, and
4. calculating the spatial weights for observed and prediction sites.

Steps 1) and 2) are undertaken in `afv_edges`, Step 3) is calculated in `afv_sites()`, and Step 4) is calculated in the package 'SSN2' when spatial stream-network models that include the tail-up covariance function are fit using `ssn_lm` or `ssn_glm`.

The segment PI for each edge, ω_j, is defined as the relative influence of the j-th edge feature on the edge directly downstream. ω_j is often based on cumulative watershed area for the downstream node of each edge, which is used as a surrogate for flow volume. However, simpler measures could be used, such as Shreve's stream order (Shreve 1966) or equal weighting, as long as a value exists for every line feature in `edges` (i.e., missing data are not allowed). It is also preferable to use a column that does not contain values equal to zero, which is explained below.

The segment PI values produced in `afv_edges()` are ratios. Therefore, the sum of the PI values for edges directly upstream of a single node always sum to one. Also note that $\omega_j = 0$ when $A_j = 0$.

The AFVs for the j-th edge, AFV_j, is equal to the product of the segment PIs found in the path between the edge and the network outlet (i.e., most downstream edge in the network), including edge j itself. Therefore, $0 \leq AFV \leq 1$. If $\omega_j = 0$, the AFV values for edges upstream of the j-th edge will also be equal to zero. This may not be problematic if the j-th edge is a headwater segment without an observed site. However, it can have a significant impact on the covariance structure of the tail-up model when the j-th edge is found lower in the stream network.

A more detailed description of the segment PIs and the steps used to calculate AFVs are provided in Peterson and Ver Hoef (2010; Appendix A).

Value

An `sf` object representing edges in the LSN, with new `segpi_col` and `afv_col` columns. If `save_local = TRUE`, the updated version of `edges` will be saved as `edges.gpkg` in `lsn_path`.

References

Peterson, E.E. and Ver Hoef, J.M. (2010) A mixed-model moving-average approach to geostatistical modeling in stream networks. Ecology 91(3), 644–651.

Examples

```
# Get temporary directory, where the example LSN will be stored
# locally.
temp_dir <- tempdir()
```

```
# Build the LSN. When working with your own data, lsn_path will be
# a local folder of your choice rather than a temporary directory.
edges <- lines_to_lsn(
  streams = MF_streams,
  lsn_path = temp_dir,
  snap_tolerance = 1,
  check_topology = FALSE,
  overwrite = TRUE,
  verbose = FALSE
)

# Calculate the AFV for the edges using an existing column
# representing watershed area for the downstream node of each line
# feature (h2oAreaKm2).
edges <- afv_edges(
  edges = edges,
  infl_col = "h2oAreaKm2",
  segpi_col = "areaPI",
  lsn_path = temp_dir,
  afv_col = "afvArea",
  overwrite = TRUE,
  save_local = FALSE
)

# Check AFVs stored in new column afvArea to ensure that 0 <= AFV
# <= 1 and that there are no NULL values.
summary(edges$afvArea)
```

afv_sites *Calculate additive function values for sites in a Landscape Network (LSN)*

Description

Calculate additive function values for sites in a Landscape Network (LSN)

Usage

```
afv_sites(
  sites,
  edges,
  afv_col,
  save_local = TRUE,
  lsn_path = NULL,
  overwrite = TRUE
)
```

Arguments

`sites`	A named list of one or more `sf` objects with POINT geometry that have been snapped to the LSN using `sites_to_lsn`.
`edges`	`sf` object with LINESTRING geometry created using `lines_to_lsn`.
`afv_col`	Name of the column in `edges` containing the additive function value for each feature, in character format. Created using `afv_edges`.
`save_local`	Logical indicating whether the updated `sites` should be saved to `lsn_path` in GeoPackage format. File basenames are taken from the names assigned to the `sites` list. Default is `TRUE`.
`lsn_path`	Optional. Local pathname to a directory in character format specifying where the LSN resides, which is created using `link[SSNbler]{lines_to_lsn}`. Must be specified if `save_local = TRUE`.
`overwrite`	A logical indicating whether results should be overwritten if `afv_col` already exists in `sites` or sites.gpkg already exists in `lsn_path` and `save_local = TRUE`. Default = TRUE.

Details

Spatial weights are used when fitting statistical models with 'SSN2' to split the tail up covariance function upstream of network confluences, which allows for the disproportionate influence of one upstream edge over another (e.g., a large stream channel converges with a smaller one) on downstream values. Calculating the spatial weights is a four step process:

1. calculating the segment proportional influence (PI) values for the edges,
2. calculating the additive function values (AFVs) for the edges,
3. calculating the AFVs for the observed and prediction sites, and
4. calculating the spatial weights for observed and prediction sites.

Steps 1) and 2) are undertaken in `afv_edges()`, Step 3) is calculated in `afv_sites()`, and Step 4) is calculated in the package 'SSN2' when spatial stream network models that include the tail up covariance function are fit using `ssn_lm` or `ssn_glm`.

The additive function value (AFV) for an observed or prediction site is equal to the AFV of the edge the site resides on. Therefore, $0 \leq AFV \leq 1$. See Peterson and Ver Hoef (2010) for a more detailed description of AFVs, how they are calculated, and how they are used in the tail up covariance function.

Value

One or more `sf` object(s) with all the original data from `sites`, along with a new `afv_col` column in each `sites` `sf` object. A named list is returned. If `save_local = TRUE`, a GeoPackage for each `sf` object is saved in `lsn_path`. Output file names are assigned based on the input `sites` attribute `names`.

References

Peterson, E.E. and Ver Hoef, J.M. (2010) A mixed model moving average approach to geostatistical modeling in stream networks. Ecology 91(3), 644–651.

Examples

```
#' # Get temporary directory, where the example LSN will be stored
# locally.
temp_dir <- tempdir()
# Build the LSN. When working with your own data, lsn_path will be
# a local folder of your choice rather than a temporary directory.
edges<- lines_to_lsn(
   streams = MF_streams,
   lsn_path = temp_dir,
   snap_tolerance = 1,
   check_topology = FALSE,
   overwrite = TRUE,
   verbose = FALSE
)

# Incorporate observed sites into the LSN
obs<- sites_to_lsn(
   sites = MF_obs,
   edges = edges,
   save_local = FALSE,
   snap_tolerance = 100,
   overwrite = TRUE,
   verbose = FALSE
)

# Incorporate the prediction dataset, preds, into the LSN
preds<- sites_to_lsn(sites = MF_preds,
   edges = edges,
   save_local = FALSE,
   snap_tolerance = 1,
   overwrite = TRUE,
   verbose = FALSE
)

# Calculate the AFVs for the edges using a column representing
# watershed area (h2oAreaKm2) for the downstream node of each edge
# feature.
edges<- afv_edges(
   edges=edges,
   infl_col = "h2oAreaKm2",
   segpi_col = "areaPI",
   lsn_path = temp_dir,
   afv_col = "afvArea",
   overwrite = TRUE,
   save_local = FALSE
)

# Calculate AFVs for observed sites (obs) and the prediction
# dataset, preds.
site.list<- afv_sites(
   sites = list(obs = obs,
                preds = preds),
    edges=edges,
    afv_col = "afvArea",
    save_local = FALSE,
    overwrite = TRUE
)

# Get names of sites in site.list
names(site.list)

# Check AFVs stored in new column afvArea to ensure that 0 <= AFV
# <= 1 and that there are no NULL values.
summary(site.list$obs$afvArea)
summary(site.list$preds$afvArea)
```

`copy_streams_to_temp` *Copy streams data to temporary directory*

Description

Copies the streams data directory to R's temporary directory so the examples in SSNbler do not write to the local library or any other places.

Usage

```
copy_streams_to_temp()
```

Details

Copies the Middle Fork data to R's temporary directory

Value

A copy of the Middle Fork streams data residing in R's temporary directory

Examples

```
copy_streams_to_temp()
# getwd()
# setwd(tempdir())
# getwd()
# if unix-alike, list temporary directory contents using: system('ls')
# if windows, list temporary directory contents using: shell('dir')
```

lines_to_lsn *Convert lines to a landscape network (LSN)*

Description

Convert an sf object containing features with LINESTRING geometry to a landscape network (LSN), which is a topological data model of streams/rivers represented as a directional graph embedded in 2-D geographic space. Relationship tables are created and topological relationships are checked.

Usage

```
lines_to_lsn(
  streams,
  lsn_path,
  check_topology = TRUE,
  snap_tolerance = 0,
  topo_tolerance = 0,
  remove_ZM = FALSE,
  overwrite = FALSE,
  use_parallel = FALSE,
  no_cores = NULL,
  verbose = TRUE
)
```

Arguments

streams	An sf object with LINESTRING geometry representing streams.
lsn_path	Pathname to a directory in character format specifying where to store the outputs. The directory will be created if it does not already exist.
check_topology	Logical. If TRUE, the edges will be checked for topological errors and potential node errors will be flagged in an output file called node_errors.gpkg. Defaults to TRUE.
snap_tolerance	Distance in map units >= 0. Two nodes separated by a distance less than or equal to snap_tolerance are assumed to be connected. Defaults to 0.
topo_tolerance	Distance in map units >= 0. Only used if check_topology = TRUE. Two nodes on the network separated by a distance less than or equal to topo_tolerance are flagged as potential topological errors in the network. Defaults to 0.
remove_ZM	Logical. If TRUE, Z and M dimensions are removed from streams, if they exist. Default is FALSE.
overwrite	Logical. If TRUE, output files will overwrite existing files with the same names. If FALSE and files sharing the same names as the outputs exist in the lsn_path, the function will exit early with an error.
use_parallel	Logical. if TRUE parallel processing will be used. Default is FALSE.
no_cores	Integer representing the number of cores used when use_parallel = TRUE.

`verbose`	Logical. If `TRUE`, messages describing function progress will be printed to the console. Default is `TRUE`.

Details

`lines_to_lsn` converts an `sf` object representing streams to a landscape network (LSN), which is a directional graph used to represent the topological and geographic relationships between line features, along with additional attribute information. `streams` must have LINESTRING geometry and a projected coordinate system, rather than geographic coordinate system (i.e. not Longitude and Latitude).

The LSN is saved to a local directory defined by `lsn_path` and has 5 components:

- nodes.gpkg: A GeoPackage of features with POINT geometry representing topologic breaks in the stream network such as pseudonodes, confluences, stream sources, or stream outlets. A column containing a unique node identifier, pointid, is included.
- edges.gpkg: A GeoPackage of features with LINESTRING geometry representing flow paths (i.e. line features) from node to node. A new column named rid is added that contains a unique identifier for each line feature.
- nodexy.csv: a table with three columns: the pointid (unique node identifier), and the x-, y-coordinates for each node.
- noderelationships.csv: a table describing the relationship between the nodes and their associated edges, as well as directionality in the LSN. The table contains three columns: the rid (for each edge), fromnode (pointid for the upstream node of each edge), and tonode (pointid for the downstream node of each edge). Directionality is defined based on the digitized direction of the line features.
- relationships.csv: a table representing the downstream flow path from edge to edge. The table contains two columns: fromedge (upstream edge) and toedge (downstream edge).

Topological errors are common in spatial data and must be corrected to ensure that the LSN accurately represents direction and connectivity within the stream network. When `check_topology = TRUE`, the edges are checked for topological errors. Two nodes on the network separated by a distance less than or equal to `topo_tolerance` are flagged as potential topological errors in the network saved in an output file called `node_errors.gpkg`, which is also saved to `lsn_path`. In addition to the pointid, `node_errors.gpkg` contains a column containing the node class (nodecat), which can take on values of Pseudonode, Confluence, Source, or Outlet. A second column (error) defines the potential error type and can take on values of Complex Confluence, Converging Node, Dangling Node, Intersection Without Node, Downstream Divergence and Unsnapped Node. The nodecat column is also added to nodes.gpkg. A node_errors.gpkg file is not produced if no obvious errors are identified. There is no guarantee that all topological errors will be identified and included in node_errors.gpkg. Therefore, potential errors *and* node classes found in node_errors.gpkg and nodes.gpkg must be checked in a GIS and topological errors in `streams` corrected before rebuilding the LSN using `lines_to_lsn()`. This process is iterative and must continue until the LSN is free of topological errors.

The arguments `snap_tolerance` and `topo_tolerance` are both provided in map units and must be >= 0. It is generally a good idea to set the `snap_tolerance` to a relatively small value compared to the `topo_tolerance` argument. Nodes separated by a Euclidean distance <= `snap_tolerance` are assumed to be connected. If this distance <= (snap_tolerance/10000), the nodes are automatically snapped when `check_topology = TRUE`. When (snap_tolerance/10000) <= distance <=

snap_tolerance, the nodes are not snapped and are instead flagged as potential errors for the user to check and correct. Similarly, when snap_tolerance < distance <= topo_tolerance, nodes are flagged as potential errors. Note that `snap_tolerance` must always be < the length of the shortest line feature found in `streams`. Use the `st_length` to obtain and check the length of each line feature.

Value

An `sf` object representing edges in the LSN. The LSN, including edges.gpkg, nodes.gpkg, nodexy.csv, noderelationships.csv, and relationships.csv files, are saved locally to a directory defined by `lsn_path`. If `check_topology = TRUE` and topological errors are identified, then node_errors.gpkg is also saved to `lsn_path`.

`lines_to_lsn_temp` *Convert lines to a landscape network*

Description

Convert an `sf` object containing features with LINESTRING geometry to a landscape network (LSN), which is a topological data model of streams/rivers represented as a directional graph embedded in 2-D geographic space. Relationship tables are created and topological relationships are checked.

Usage

```
lines_to_lsn_temp(
  streams,
  lsn_path,
  snap_tolerance = 0,
  topo_tolerance = 0,
  check_topology = TRUE,
  remove_ZM = FALSE,
  overwrite = FALSE,
  verbose = TRUE
)
```

Arguments

`streams`	An `sf` object with LINESTRING geometry representing streams.
`lsn_path`	Pathname to a directory in character format specifying where to store the outputs. The directory will be created if it does not already exist.
`snap_tolerance`	Distance in map units. Two nodes separated by a distance less than or equal to `snap_tolerance` are assumed to be connected. Defaults to 0.
`topo_tolerance`	Distance in map units. Only used if `check_topology = TRUE`. Two nodes on the network separated by a distance less than or equal to `topo_tolerance` are flagged as potential topological errors in the network. Defaults to 0.

`check_topology`	Logical. If `TRUE`, the edges will be checked for topological errors and potential node errors will be flagged in an output file called `node_errors.gpkg`. Defaults to `TRUE`.
`remove_ZM`	Logical. If `TRUE`, Z and M dimensions are removed from streams, if they exist. Default is `FALSE`.
`overwrite`	Logical. If `TRUE`, output files will overwrite existing files with the same names. If `FALSE` and files sharing the same names as the outputs exist in the `lsn_path`, the function will exit early with an error.
`verbose`	Logical. If `TRUE`, messages describing function progress will be printed to the console. Default is `TRUE`.

Details

`lines_to_lsn` converts an `sf` object representing streams to a landscape network (LSN), which is a directional graph used to represent the topological and geographic relationships between line features, along with additional attribute information. `streams` must have LINESTRING geometry and a projected coordinate system, rather than geographic coordinate system (i.e. not Longitude and Latitude).

The LSN is saved to a local directory defined by `lsn_path` and has 5 components:

- nodes.gpkg: A GeoPackage of features with POINT geometry representing topologic breaks in the stream network such as pseudonodes, confluences, stream sources, or stream outlets. A column containing a unique node identifier, pointid, is included.
- edges.gpkg: A GeoPackage of features with LINESTRING geometry representing flow paths (i.e. line features) from node to node. A new column named rid is added that contains a unique identifier for each line feature.
- nodexy.csv: a table with three columns: the pointid (unique node identifier), and the x-, y-coordinates for each node.
- noderelationships.csv: a table describing the relationship between the nodes and their associated edges, as well as directionality in the LSN. The table contains three columns: the rid (for each edge), fromnode (pointid for the upstream node of each edge), and tonode (pointid for the downstream node of each edge). Directionality is defined based on the digitized direction of the line features.
- relationships.csv: a table representing the downstream flow path from edge to edge. The table contains two columns: fromedge (upstream edge) and toedge (downstream edge).

Topological errors are common in spatial data and must be corrected to ensure that the LSN accurately represents direction and connectivity within the stream network. When `check_topology = TRUE`, the edges are checked for topological errors. Two nodes on the network separated by a distance less than or equal to `topo_tolerance` are flagged as potential topological errors in the network saved in an output file called `node_errors.gpkg`, which is also saved to `lsn_path`. In addition to the pointid, `node_errors.gpkg` contains a column containing the node class (nodecat), which can take on values of Pseudonode, Confluence, Source, or Outlet. A second column (error) defines the potential error type and can take on values of Complex Confluence, Converging Node, Dangling Node, Intersection Without Node, Downstream Divergence and Unsnapped Node. The nodecat column is also added to nodes.gpkg. A node_errors.gpkg file is not produced if no obvious errors are identified. There is no guarantee that all topological errors will be identified and included in node_errors.gpkg. Therefore, potential errors *and* node classes found in node_errors.gpkg and nodes.gpkg must be checked in a GIS and topological errors in `streams` corrected before rebuilding the LSN using `lines_to_lsn()`. This process is iterative and must continue until the LSN is free of topological errors.

Value

An `sf` object representing edges in the LSN. The LSN, including edges.gpkg, nodes.gpkg, nodexy.csv, noderelationships.csv, and relationships.csv files, are saved locally to a directory defined by `lsn_path`. If `check_topology = TRUE` and topological errors are identified, then node_errors.gpkg is also saved to `lsn_path`.

`MF_CapeHorn` *MF_CapeHorn: Prediction locations on Cape Horn Creek, in the Middle Fork Basin, Idaho.*

Description

`MF_CapeHorn` is an `sf` object with POINT geometry representing prediction locations and covariates on Cape Horn Creek, Middle Fork Basin, Idaho, USA.

Usage

```
MF_CapeHorn
```

Format

An object of class `sf` (inherits from `data.frame`) with 654 rows and 10 columns.

Details

The `sf data.frame` contains 654 point features and 9 columns:

- COMID: Common identifier of an NHD feature or relationship
- AREAWTMAP: Area weighted mean annual precipitation (mm) at lowermost location on the line segment where the site resides
- SLOPE: Slope of the line segment (cm/cm) where the site resides
- ELEV_DEM: Elevation at the site based on a 30m DEM
- Source: Source of the data - relates to the ID field of the source table
- FlowCMS: Average stream flow (cubic meters per sec) for August, by year, from 1950-2010 across 9 USGS gauges in the region
- AirMEANc: Average mean air temperature (C) from July 15 - August 31, from 1980-2009 across 10 COOP air stations within the domain
- AirMWMTc: Average maximum air temperature (C) from July 15 - August 31, from 1980-2009 across 10 COOP air stations within the domain. MWMT = maximum 7-day moving average of the maximum daily temperature (i.e. maximum of all the 7-day maximums)

- rcaAreaKm2: Reach contributing area (km2) for the downstream node of the edge feature the site resides on. RCA area is the land area draining directly into each line segment.
- h2oAreaKm2: Watershed area (km2) for the downstream node of the edge feature the site resides on

Source

`MF_CapeHorn` are unpublished United States Forest Service data.

`MF_obs` *MF_obs: Water temperature observations in the Middle Fork Basin, Idaho in 2004.*

Description

`MF_obs` is an `sf` object with POINT geometry representing water temperature observations and covariates in the Middle Fork Basin, Idaho, USA collected in 2004.

Usage

```
MF_obs
```

Format

An object of class `sf` (inherits from `data.frame`) with 45 rows and 17 columns.

Details

The `sf data.frame` contains 45 point features and 16 columns:

- STREAMNAME: Stream name
- COMID: Common identifier of an NHD feature or relationship
- AREAWTMAP: Area weighted mean annual precipitation (mm) at lowermost location on the line segment where the site resides
- SLOPE: Slope of the line segment (cm/cm) where the site resides
- ELEV_DEM: Elevation at the site based on a 30m DEM
- Source: Source of the data - relates to the ID field of the source table
- Summer_mn: Overall summer mean temperature (C) of the deployment
- MaxOver20: Binary variable: 1 represents the maximum summer temperature was greater than 20C and 0 indicates that it was less than 20C
- C16: Number of times daily stream temperature exceeded 16C
- C20: Number of times daily stream temperature exceeded 20C
- C24: Number of times daily stream temperature exceeded 24C
- FlowCMS: Average stream flow (cubic meters per sec) for August, by year, from 1950-2010 across 9 USGS gauges in the region

- AirMEANc: Average mean air temperature (C) from July 15 - August 31, from 1980-2009 across 10 COOP air stations within the domain
- AirMWMTc: Average maximum air temperature (C) from July 15 - August 31, from 1980-2009 across 10 COOP air stations within the domain. MWMT = maximum 7-day moving average of the maximum daily temperature (i.e. maximum of all the 7-day maximums)
- rcaAreaKm2: Reach contributing area (km2) for the downstream node of the edge feature the site resides on. RCA area is the land area draining directly into each line segment.
- h2oAreaKm2: Watershed area (km2) for the downstream node of the edge feature the site resides on

Source

`MF_obs` are unpublished United States Forest Service data.

`MF_pred1km` *MF_pred1km: Prediction locations at 1km intervals throughout the Middle Fork Basin, Idaho.*

Description

`MF_pred1km` is an `sf` object with POINT geometry representing prediction locations and covariates distributed at 1km intervals throughout the Middle Fork Basin, Idaho, USA.

Usage

```
MF_pred1km
```

Format

An object of class `sf` (inherits from `data.frame`) with 175 rows and 10 columns.

Details

The `sf data.frame` contains 175 point features and 9 columns:

- COMID: Common identifier of an NHD feature or relationship
- AREAWTMAP: Area weighted mean annual precipitation (mm) at lowermost location on the line segment where the site resides
- SLOPE: Slope of the line segment (cm/cm) where the site resides
- ELEV_DEM: Elevation at the site based on a 30m DEM
- Source: Source of the data - relates to the ID field of the source table
- FlowCMS: Average stream flow (cubic meters per sec) for August, by year, from 1950-2010 across 9 USGS gauges in the region
- AirMEANc: Average mean air temperature (C) from July 15 - August 31, from 1980-2009 across 10 COOP air stations within the domain

- AirMWMTc: Average maximum air temperature (C) from July 15 - August 31, from 1980-2009 across 10 COOP air stations within the domain. MWMT = maximum 7-day moving average of the maximum daily temperature (i.e. maximum of all the 7-day maximums)
- rcaAreaKm2: Reach contributing area (km2) for the downstream node of the edge feature the site resides on. RCA area is the land area draining directly into each line segment.
- h2oAreaKm2: Watershed area (km2) for the downstream node of the edge feature the site resides on

Source

`MF_pred1km` are unpublished United States Forest Service data.

`MF_preds` *MF_preds: A small set of prediction locations found in the Middle Fork Basin, Idaho.*

Description

`MF_preds` is an `sf` object with POINT geometry representing prediction locations and covariates distributed at 1km intervals throughout the Middle Fork Basin, Idaho, USA.

Usage

```
MF_preds
```

Format

An object of class `sf` (inherits from `data.frame`) with 43 rows and 10 columns.

Details

The `sf data.frame` contains 43 point features and 9 columns:

- COMID: Common identifier of an NHD feature or relationship
- AREAWTMAP: Area weighted mean annual precipitation (mm) at lowermost location on the line segment where the site resides
- SLOPE: Slope of the line segment (cm/cm) where the site resides
- ELEV_DEM: Elevation at the site based on a 30m DEM
- FlowCMS: Average stream flow (cubic meters per sec) for August, by year, from 1950-2010 across 9 USGS gauges in the region
- AirMEANc: Average mean air temperature (C) from July 15 - August 31, from 1980-2009 across 10 COOP air stations within the domain
- AirMWMTc: Average maximum air temperature (C) from July 15 - August 31, from 1980-2009 across 10 COOP air stations within the domain. MWMT = maximum 7-day moving average of the maximum daily temperature (i.e. maximum of all the 7-day maximums)

- rcaAreaKm2: Reach contributing area (km2) for the downstream node of the edge feature the site resides on. RCA area is the land area draining directly into each line segment.
- h2oAreaKm2: Watershed area (km2) for the downstream node of the edge feature the site resides on

Source

`MF_preds` are unpublished United States Forest Service data.

`MF_streams` *MF_streams: Middle Fork streams*

Description

`MF_streams` is an `sf` object with LINESTRING geometry representing a subset of streams and rivers in the Middle Fork Basin, Idaho, USA.

Usage

```
MF_streams
```

Format

An object of class `sf` (inherits from `data.frame`) with 163 rows and 10 columns.

Details

The `sf data.frame` contains a set of 163 features and 9 columns:

- COMID: Common identifier of an NHD feature or relationship
- GNIS_NAME: Feature name as found in the Geographic Names Information System
- REACHCODE: Unique identifier for a reach. The first 8 digits contain the identifier for the HUC8 and the last 6 digits are a unique within-HUC8 identifier for the reach
- FTYPE: three-digit integer used to classify hydrography features in the NHD and define subtypes
- FCODE: Numeric code that contains the feature type and its attributes as found in the NHDFCode lookup table
- AREAWTMAP: Area weighted mean annual precipitation (mm) at the lowermost location on the edge
- SLOPE: Slope of the edge (cm/cm)
- rcaAreaKm2: Reach contributing area (km2), which is the land area draining directly into each line segment.
- h2oAreaKm2: Watershed area (km2) for the lowermost location (downstream end node) on the line segment

Source

`MF_streams` are a modified version of the United States National Hydrography Dataset (http://nhd.usgs.gov/).

`sites_to_lsn` *Incorporate sites into a Landscape Network*

Description

Incorporates an `sf` object containing features with POINT geometry into a landscape network (LSN), which is a topological data model of streams/rivers represented as a directional graph embedded in 2-D geographic space. Point locations are 'snapped' to the closest edge location and new information is generated describing the geographic and topological location relative to other features in the LSN.

Usage

```
sites_to_lsn(
  sites,
  edges,
  snap_tolerance,
  save_local = TRUE,
  lsn_path = NULL,
  file_name = NULL,
  overwrite = FALSE,
  verbose = TRUE
)
```

Arguments

`sites`	An `sf` object with POINT geometry, typically representing observed data or prediction locations.
`edges`	An `sf` object with LINESTRING geometry created using `lines_to_lsn`.
`snap_tolerance`	Numeric distance in map units >= 0. Sites that are <= this distance from an edge feature will snapped to the closest location on the edge. When the distance between the site and all edges is > `snap_tolerance`, the point feature is not snapped or included in the lsn_path.
`save_local`	Logical indicating whether the outputs should be saved to a local directory in GeoPackage format. Defaults to `TRUE`.
`lsn_path`	Pathname to the LSN. This is typically a directory created by `lines_to_lsn`. Required if `save_local = TRUE`.
`file_name`	Filename for output sites, which are saved to `lsn_path` in GeoPackage format (must include the .gpkg extension). Required if `save_local = TRUE`.
`overwrite`	Logical indicating whether the outputs saved to `lsn_path` should overwrite existing files if they exist. Defaults to `FALSE`.
`verbose`	Logical indicating whether progress messages should be printed to the console. Defaults to `TRUE`

Details

The `sites_to_lsn` function is used to incorporate observed and prediction sites into the LSN. The output is an `sf` object with POINT geometry, which contains only the point features from `sites` that were found less than the `snap_tolerance` distance from the closest edge feature. When a `sites` point feature meets these requirements, it is moved (i.e. snapped) to the closest location on an edge feature. Three new columns are also added: rid, ratio, and snapdist. The rid column contains the rid for the edge the site has been snapped to. The second column, ratio, represents the proportional length of the edge found between the downstream end node for the edge and the updated/snapped site location. The snapdist is the distance in map units that the site was moved. If the distance between a point feature and the closest edge is greater than or equal to the `snap_tolerance`, the feature is not included in the output.

The `snap_tolerance` must always be ≥ 0 and must be large enough to snap all of the point features to the edges. Using `snap_tolerance = 0` is not recommended, even when the `sites` features intersect the edge features. Instead, a very small `snap_tolerance` value is recommended to ensure that differences in the precision of the x and y coordinates and the line location do not result in unsnapped point features.

Note that the `sites` and `edges` must have the same projection.

Value

An `sf` object with POINT geometry containing the features from `sites` that were found within the `snap_tolerance` distance to the closest edge. In addition to the original columns in `sites`, three new columns are added: rid, ratio, and snapdist (see Details). If `save_local = TRUE`, the `sf` object is also saved to `lsn_path`.

Examples

```
# Get temporary directory, where the example LSN will be stored
# locally.
temp_dir <- tempdir()
# Build the LSN. When working with your own data, lsn_path will be
# a local folder of your choice rather than a temporary directory.
edges<- lines_to_lsn(
   streams = MF_streams,
   lsn_path = temp_dir,
   snap_tolerance = 1,
   check_topology = FALSE,
   overwrite = TRUE,
   verbose = FALSE
)

# # Incorporate observed sites, MF_obs, into LSN
obs<- sites_to_lsn(
   sites = MF_obs,
   edges = edges,
   save_local = FALSE,
   snap_tolerance = 100,
   overwrite = TRUE,
   verbose = FALSE
)
```

```
# Notice that 3 new columns have been added to obs
names(obs)

# Incorporate prediction dataset, MF_preds, into LSN
preds<- sites_to_lsn(sites = MF_preds,
   edges = edges,
   save_local = FALSE,
   snap_tolerance = 1,
   overwrite = TRUE,
   verbose = FALSE
)
```

---

ssn_assemble *Assemble an* SSN *object from an LSN*

Description

Create an SSN (spatial stream network) object from a Landscape Network (LSN).

Usage

```
ssn_assemble(
  edges,
  lsn_path = NULL,
  obs_sites = NULL,
  preds_list = NULL,
  ssn_path,
  import = TRUE,
  check = TRUE,
  afv_col = NULL,
  overwrite = FALSE,
  verbose = TRUE
)
```

Arguments

edges	An sf object with LINESTRING geometry created using lines_to_lsn (see Details).
lsn_path	Local pathname to a directory in character format specifying where relationships.csv resides, which is created using link{lines_to_lsn}.
obs_sites	Optional. A single sf object with POINT geometry created using link{sites_to_lsn} that represents the observation locations (i.e. where data were collected). Default = NULL (see Details).
preds_list	Optional. A list of one or more sf objects representing prediction sites.
ssn_path	Pathname to an output directory where output files will be stored. A .ssn extension will be added if it is not included.

`import`	Logical indicating whether the output files should be returned as an `SSN` object. Defaults to `TRUE`.
`check`	Logical indicating whether the validity of the `SSN` should be checked using `[ssn_check]` when both `import = TRUE` and `verbose = TRUE`. Default = `TRUE`.
`afv_col`	Character vector containing the name(s) of the additive function value column(s) that will be checked when `check = TRUE`. Columns must be present in `edges`, `obs_sites` and `preds_list`, if all are included. Default is `NULL`.
`overwrite`	Logical. If `TRUE` and `ssn_path` already exists, the contents of `ssn_path` will be overwritten. Defaults to `FALSE`.
`verbose`	Logical. Indicates whether messages about the function progress and object validity check (when `check = TRUE`) should be printed to the console. Defaults to `TRUE`.

Details

The `SSNbler` package is used to generate the spatial, topological, and attribute information needed to fit spatial stream-network models using the 'SSN2' package. The `ssn_assemble` function will often be the final step in the 'SSNbler' data-processing workflow and it is important that the previous processing steps have been followed. Prior to running `ssn_assemble`, the `edges` must be processed using `link{lines_to_lsn}`, `link{updist_edges}`, and `link{afv_edges}`. The `obs_sites` and prediction site datasets in `preds_list` must be processed with `link{sites_to_lsn}`, `link{updist_sites}`, and `link{afv_sites}`. In addition, the `edges`, `obs_sites`, and all of the `sf` objects in `preds_list` must be part of the same LSN.

The `obs_sites` and `preds_list` are optional arguments, with the Default = NULL. If `obs_sites = NULL`, an `SSN` object will be returned with NA stored in `ssn.object$obs` and a warning returned that `ssn.object$obs` is required for fitting spatial statistical models in 'SSN2'.

`ssn_assemble` stores the output locally in `ssn_path`. If `ssn_path` does not include the .ssn extension, it is added before the new directory is created. This directory contains:

- edges.gpkg: edges in GeoPackage format. A new network identifier, netID, is added that is unique to each subnetwork.
- sites.gpkg: observed sites in GeoPackage format (if present). Three new ID columns are added that are unique to the measurement (pid), the location (locID), and the network (netID).
- prediction datasets in GeoPackage format (if present). The prediction sites also contain pid, locID, and netID. The naming convention is taken from the names provided in `preds_list`.
- netID.dat files for each distinct network, which store the binaryID values for line segments in edges.

A more detailed description of the .ssn directory and its contents is provided in Peterson and Ver Hoef (2014).

Value

The components of an `SSN` object are written to `ssn_path` (see Details). When `import = TRUE`, the function also returns an object of class `SSN`. If `check = TRUE` and `verbose = TRUE`, the validity of the returned `SSN` object is checked using `[ssn_check]` and results are printed to the console.

Examples

```
# Get temporary directory, where the example LSN will be stored
# locally.
temp_dir <- tempdir()
# Build the LSN. When working with your own data, lsn_path will be
# a local folder of your choice rather than a temporary directory.
edges<- lines_to_lsn(
   streams = MF_streams,
   lsn_path = temp_dir,
   snap_tolerance = 1,
   check_topology = FALSE,
   overwrite = TRUE,
   verbose = FALSE
)

# Incorporate observed sites, MF_obs, into LSN
obs<- sites_to_lsn(
   sites = MF_obs,
   edges = edges,
   save_local = FALSE,
   snap_tolerance = 100,
   overwrite = TRUE,
   verbose = FALSE
)

# Incorporate prediction dataset, MF_preds, into LSN
preds<- sites_to_lsn(sites = MF_preds,
   edges = edges,
   save_local = FALSE,
   snap_tolerance = 1,
   overwrite = TRUE,
   verbose = FALSE
)

# Calculate the AFV for the edges using
# a column representing watershed area (h2oAreaKm2).
edges<- afv_edges(
   edges=edges,
   infl_col = "h2oAreaKm2",
   segpi_col = "areaPI",
   lsn_path = temp_dir,
   afv_col = "afvArea",
   overwrite = TRUE,
   save_local = FALSE
)

# Calculate the AFV for observed sites (obs) and prediction
# dataset, preds.
site.list<- afv_sites(
   sites = list(obs = obs,
                preds = preds),
   edges=edges,
   afv_col = "afvArea",
   save_local = FALSE,
   overwrite = TRUE
)

# Calculate upstream distance for edges
edges<- updist_edges(
   edges = edges,
   lsn_path = temp_dir,
   calc_length = TRUE,
   length_col = "Length",
   overwrite = TRUE,
   save_local = FALSE,
   verbose = FALSE
)

# Calculate upstream distance for observed sites (obs) and one
# prediction dataset (preds)
site.list<- updist_sites(
   sites = site.list,
   edges = edges,
   length_col= "Length",
   lsn_path = temp_dir,
   save_local = FALSE,
   overwrite = TRUE
)

# Assemble SSN object
ssn.obj<- ssn_assemble(
   edges = edges,
   lsn_path = temp_dir,
   obs_sites = site.list[["obs"]],
   preds_list = site.list[c("preds")],
   ssn_path = paste0(temp_dir, "/example.ssn"),
   import = TRUE,
   overwrite = TRUE
)

# Summarise SSN object
summary(ssn.obj)
```

`ssn_check` *Check an* `SSN` *object*

Description

Check an `SSN` (spatial stream network) object to ensure that it contains valid spatial, topological, and attribute information needed to fit spatial statistical stream network models using the 'SSN2' package.

Usage

```
ssn_check(ssn.object, check_obs = TRUE, afv_col = NULL, verbose = TRUE)
```

Arguments

`ssn.object`	An `SSN` object created using `ssn_assemble` or imported using `ssn_import`.
`check_obs`	A logical indicating whether the observations should be checked. Default = `TRUE`.
`afv_col`	Character vector containing names of columns containing additive function values. The Default = `NULL`.
`verbose`	Logical `TRUE/FALSE` indicating whether details describing the checks are printed to the console. If `verbose` = `FALSE`, a logical `TRUE/FALSE` is returned indicating whether the `SSN` object passed all of the checks. Default = `TRUE`.

Value

Boolean indicating whether the `SSN` object is valid. If `verbose` = `TRUE`, additional messages are printed to the console describing potential issues with the `SSN` object.

Examples

```
## Create local temporary copy of MiddleFork04.ssn found in
## the SSN2 package. Only necessary for this example.
SSN2::copy_lsn_to_temp()

# Import the SSN object with prediction points, pred1km
mf04 <- SSN2::ssn_import(
  paste0(tempdir(), "/MiddleFork04.ssn"),
  predpts = c("pred1km"),
  overwrite = TRUE
)

# Check the SSN object, including the additive function column,
# afvArea
ssn_check(mf04, afv_col = "afvArea")
```

`updist_edges` *Get upstream distance for edges in a Landscape Network (LSN)*

Description

Calculate the distance from the stream outlet (i.e. the most downstream location on the stream network) to the upstream node of each edge feature (i.e. upstream distance) in the Landscape Network (LSN)

Usage

```
updist_edges(
  edges,
  lsn_path = NULL,
  calc_length = FALSE,
  length_col = NULL,
  save_local = TRUE,
  overwrite = TRUE,
  verbose = TRUE
)
```

Arguments

`edges`	An `sf` object with LINESTRING geometry created using `lines_to_lsn`.
`lsn_path`	Local pathname to a directory in character format specifying where relationships.csv resides, which is created using `lines_to_lsn`.
`calc_length`	A logical indicating whether a column representing line length should be calculated and added to `edges`. Default = `FALSE`.
`length_col`	Optional. If `calc_length` = `FALSE`, `length_col` is the name of the column in the `edges` attribute table that contains the length of the edge feature. When `calc_length` = `FALSE`, `length_col` is required. If `calc_length` = `TRUE`, `length_col` is the name of the new column created in `edges` that will store the new length values for each feature, in character format. When `calc_length` = `TRUE` and `length_col` = `NULL`, the default for `length_col` is `"Length"`.
`save_local`	Logical indicating whether the updated `edges` should be saved to `lsn_path` in GeoPackage format. Defaults to `TRUE`.
`overwrite`	A logical indicating whether results should be overwritten if the upDist column already exists in `edges` or edges.gpkg already exists in `lsn_path` and `save_local` = `TRUE`. Default = TRUE
`verbose`	Logical. Indicates whether messages about the function progress should be printed to the console. Defaults to `TRUE`.

Details

`updist_edges()` calculates the total hydrologic distance from the uppermost location on each edge feature (upstream node) to the stream outlet (i.e. the most downstream location in the stream network), when movement is restricted to the stream network. We refer to this as the upstream distance. Upstream distances are measured in the map projection units for the `sf` object edges and stored in a new column in edges named `upDist`.

The upstream distances stored in `upDist` are used to calculate the upstream distance for sites in `updist_sites()` and the pairwise hydrologic distances used to fit spatial stream network models in the 'SSN2' package. Do not modify the name of the column in any way or the values the `upDist` column contains.

Value

An `sf` object representing edges in the LSN, with a new `upDist` column. When `calc_length = TRUE` an additional column named `length_col`

Examples

```
# Get temporary directory, where the example LSN will be stored
# locally.
temp_dir <- tempdir()
# Build the LSN. When working with your own data, lsn_path will be
# a local folder of your choice rather than a temporary directory.
edges<- lines_to_lsn(
   streams = MF_streams,
   lsn_path = temp_dir,
   snap_tolerance = 1,
   check_topology = FALSE,
   overwrite = TRUE,
   verbose = FALSE
)

# Calculate upstream distance for edges
edges<- updist_edges(
   edges = edges,
   lsn_path = temp_dir,
   calc_length = TRUE,
   length_col = "Length",
   overwrite = TRUE,
   save_local = FALSE,
   verbose = FALSE
)

# Notice that two new columns have been added to edges containing
# line feature length (Length) and the upstream distance (upDist)
names(edges)
summary(edges[,c("Length", "upDist")])
```

`updist_sites` *Get upstream distance for sites in a Landscape Network (LSN)*

Description

Get upstream distance for sites in a Landscape Network (LSN)

Usage

```
updist_sites(
  sites,
  edges,
  length_col,
  lsn_path,
  save_local = TRUE,
  overwrite = TRUE
)
```

Arguments

sites	A named list of one or more sf objects with POINT geometry that have been snapped to the LSN using sites_to_lsn.
edges	An sf object with LINESTRING geometry created using lines_to_lsn and link[SSNbler]{updist_edges}.
length_col	The name of the column in edges that contains the length of each edge feature.
lsn_path	Local pathname to a directory in character format specifying where the LSN resides, which is created using link[SSNbler]{lines_to_lsn}. Must be specified if save_local = TRUE.
save_local	Logical indicating whether the updated sites should be saved to lsn_path in GeoPackage format. File basenames are taken from the names assigned to the sites list. Default is TRUE.
overwrite	A logical indicating whether results should be overwritten if the upDist or length_col columns already exist in sites or sites.gpkg already exists in lsn_path. Default = TRUE. #' @details updist_sites() calculates the total hydrologic distance from each observed or prediction point feature to the stream outlet (i.e. the most downstream location in the stream network), when movement is restricted to the stream network. We refer to this as the upstream distance. Upstream distances are measured in the map projection units for the sf object containing the point features and stored in a new column named upDist. The upstream distances stored in upDist are used to calculate the pairwise hydrologic distances used to fit spatial stream network models in the 'SSN2' package. Do not modify the name of the column in any way or the values the upDist column contains. If the upDist or length_col columns already exist in sites and overwrite = TRUE, the columns will be deleted from all sites.,

Value

One or more sf object(s) with all the original data from sites, along with a new upDist column in each sites sf object. A named list is returned. If save_local = TRUE, a GeoPackage for each sf object is saved in lsn_path. Output file names are assigned based on the input sites attribute names.

Examples

```
# Get temporary directory, where the example LSN will be stored
# locally.
temp_dir <- tempdir()
# Build the LSN. When working with your own data, lsn_path will be
# a local folder of your choice rather than a temporary directory.
```

```
edges<- lines_to_lsn(
   streams = MF_streams,
   lsn_path = temp_dir,
   snap_tolerance = 1,
   check_topology = FALSE,
   overwrite = TRUE,
   verbose = FALSE
)

# Incorporate observed sites, MF_obs, into LSN
obs<- sites_to_lsn(
   sites = MF_obs,
   edges = edges,
   save_local = FALSE,
   snap_tolerance = 100,
   overwrite = TRUE,
   verbose = FALSE
)

# Incorporate prediction dataset, MF_preds, into LSN
preds<- sites_to_lsn(
   sites = MF_preds,
   edges = edges,
   save_local = FALSE,
   snap_tolerance = 1,
   overwrite = TRUE,
   verbose = FALSE
)

# Calculate upstream distance for edges
edges<- updist_edges(
   edges = edges,
   lsn_path = temp_dir,
   calc_length = TRUE,
   length_col = "Length",
   overwrite = TRUE,
   save_local = FALSE,
   verbose = FALSE
)

# Calculate upstream distance for observed sites (obs) and one
# prediction dataset (preds)
site.list<- updist_sites(
   sites = list(obs = obs,
                preds = preds),
   edges = edges,
   length_col= "Length",
   lsn_path = temp_dir,
   save_local = FALSE,
   overwrite = TRUE
)

# Summarize the new column upDist in obs
```

```
summary(site.list$obs$upDist)

# Summarize the new column upDist in preds
summary(site.list$preds$upDist)
```

Index