Package: pavement (via r-universe)

September 18, 2024

Title Analyzing Spatial Events on Roadways

Version 0.0.1

Description Pavement is a package designed to analyze spatial events occurring on roadways. It provides a comprehensive toolkit for working with spatial data, empowering users to understand patterns and trends in road-related phenomena.

License MIT + file LICENSE

URL `https://github.com/NONONOexe/pavement`, `https://nononoexe.github.io/pavement/`

BugReports `https://github.com/NONONOexe/pavement/issues`

Depends R (>= 3.3.0)

Imports graphics, grDevices, igraph, lwgeom, osmdata, sf

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

BuildManual TRUE

Repository https://nononoexe.r-universe.dev

RemoteUrl https://github.com/NONONOexe/pavement

RemoteRef HEAD

RemoteSha eff01bda1c53b511799cfa57f354883e9fbe95e7

Contents

`compute_epanechnikov` *Compute the Epanechnikov function*

Description

The Epanechnikov function is one of the quadratic kernels used in kernel density estimation. It is defined as follows:

$$K(x) = \frac{3}{4}(1 - x^2) \quad \text{if} \quad |x| \leq 1.$$

Usage

```
compute_epanechnikov(x)
```

Arguments

`x`	A numeric vector.

Value

A numeric vector of weights of the kernel for each input points.

Examples

```
x <- seq(-3, 3, 0.1)
y <- compute_epanechnikov(x)
plot(x, y, xlim = c(-3, 3), ylim = c(0, 1), type = "l")
```

`compute_gaussian` *Compute the Gaussian function*

Description

The Gaussian function is a bell-shaped function defined as follows:

$$K(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{x^2}{2\sigma^2}\right).$$

Usage

```
compute_gaussian(x, sigma = 1)
```

Arguments

`x`	A numric vector.
`sigma`	The standard deviation of the Gaussian function.

Value

A numeric vector of weights of the kernel for each input points.

Examples

```
x <- seq(-3, 3, 0.1)
y <- compute_gaussian(x, sigma = 1)
plot(x, y, xlim = c(-3, 3), ylim = c(0, 1), type = "l")
```

`convolute_segmented_network` *Convolute segmented road network*

Description

This function convolves a segmented network using a specified kernel function, typically for traffic modeling or network analysis. It computes weights densities based on the distance between links in the network and the number of events assigned to each link. Optionally, it can adjust for branching in the network.

Usage

```
convolute_segmented_network(
  segmented_network,
  kernel = compute_epanechnikov,
  bandwidth = 3,
  esd = FALSE,
  ...
)
```

Arguments

`segmented_network`	A `segmented_network` object assigned with events.
`kernel`	A kernel function to use for convolution (default is Epanechnikov kernel).
`bandwidth`	Numeric value representing the bandwidth for the kernel function (default is 3).
`esd`	If `TRUE`, considers branching in the kernel using the Equal Split Discontinous kernel (ESD). ESD follows the method described in Okabe et al.
`...`	Additional arguments passed to the kernel function.

Value

The segmented network with updated link densities.

References

Okabe, A., Satoh, T., & Sugihara, K. (2009). A kernel density estimation method for networks, its computational method and a GIS-based tool. *International Journal of Geographical Information Science*, 23(1), 7-32. doi:10.1080/13658810802475491

Examples

```
# Create a road network
road_network <- create_road_network(sample_roads)

# Assign sample accidents data
```

```
road_network <- set_events(road_network, sample_accidents)

# Segment the road network
segmented_network <- create_segmented_network(
  road_network,
  segment_length = 0.5
)

# Check the segmented road network after assigning events
segmented_network

# Apply the convilution to calculate link densities using
# the kernel function
convoluted_network <- convolute_segmented_network(segmented_network)

# Check the convoluted network with the computed densities
convoluted_network

# Plot the convoluted network showing the density distribution
plot(convoluted_network, mode = "density")
```

---

create_bbox *Create a bounding box*

Description

This function creates a bounding box from either cardinal coordinates (north, south, east and west) or center coordinates and dimensions (center longitude and latitude, and size of width and height).

Usage

```
create_bbox(
  north = NULL,
  south = NULL,
  east = NULL,
  west = NULL,
  center_lon = NULL,
  center_lat = NULL,
  width = NULL,
  height = NULL
)
```

Arguments

`north`	The northernmost latitude.
`south`	The southernmost latitude.
`east`	The easternmost longitude.
`west`	The westernmost longitude.

`center_lon`	The center longitude.
`center_lat`	The center latitude.
`width`	The width of the bounding box.
`height`	The height of the bounding box.

Value

A 2x2 matrix representing the bounding box.

Examples

```
# Create a bounding box from cardinal coordinates
create_bbox(
  north =  35.1899,
  south =  35.1399,
  east  = 136.9524,
  west  = 136.8524
)

# Create a bounding box from center coordinates and dimensions
create_bbox(
  center_lon = 136.9024,
  center_lat =  35.1649,
  width      = 0.10,
  height     = 0.05
)
```

`create_coordinates` *Create a collection of coordinates.*

Description

This function creates a collection of coordinates from a sequence x and y coordinates.

Usage

```
create_coordinates(...)
```

Arguments

`...`	A sequence of x and y coordinates.

Value

A `coordinates` object which is a matrix with x and y columns.

Examples

```
# Create a `coordinates` object from a sequence of x and y coordinates
create_coordinates(1, 2, 3, 4)

# Create a `coordinates` object from an with a vector of x and y
# coordinates
create_coordinates(c(1, 2, 3, 4))
```

`create_graph` *Create a graph from nodes and links*

Description

This function creates a graph object from a set of nodes and links.

Usage

```
create_graph(nodes, links, directed = FALSE)
```

Arguments

`nodes`	A `sf` object representing nodes.
`links`	A `sf` object representing links between nodes.
`directed`	A logical indicating whether the graph is directed.

Value

A `igraph` object.

Examples

```
# Create nodes and links
nodes <- extract_road_network_nodes(sample_roads)
links <- extract_road_network_links(sample_roads, nodes)

# Create the graph
graph <- create_graph(nodes, links)
graph

# Plot the graph
plot(graph)
```

create_graph_links *Create graph links from source and target nodes*

Description

This function creates a data frame of graph links from a set of source and target nodes.

Usage

```
create_graph_links(
  source_nodes,
  target_nodes,
  directed = FALSE,
  unique_links = TRUE
)
```

Arguments

`source_nodes`	A vector of source node IDs.
`target_nodes`	A vector of target node IDs.
`directed`	If `FALSE`, the default, the links are undirected. If `TRUE`, the links are directed.
`unique_links`	If `TRUE`, the default, the links are unique. Duplicate links are removed. If `FALSE`, duplicate links are retained.

Value

A data frame of graph links with columns `from` and `to`.

Examples

```
# Create source and target nodes
source_nodes <- c("jn_000001", "jn_000002", "jn_000002")
target_nodes <- c("jn_000002", "jn_000001", "jn_000003")

# Create graph links (duplicate links are removed)
create_graph_links(source_nodes, target_nodes)
```

create_graph_nodes *Create a data frame of nodes for a graph*

Description

This function creates a data frame of nodes for a graph from a set of node IDs and geometries.

Usage

```
create_graph_nodes(node_ids, node_geometries)
```

Arguments

`node_ids`	A vector of node IDs.
`node_geometries`	A `sfc_POINT` object containing node geometries.

Value

A data frame of nodes with columns `id`, `x`, and `y`.

Examples

```
# Create node IDs and geometries
node_ids <- c("jn_000001", "jn_000002", "jn_000003")

# Create some node geometries
node_geometries <- create_points(0, 1, 1, 0, 2, 1)

# Create a data frame of nodes
create_graph_nodes(node_ids, node_geometries)
```

create_linestring *Create a linestring geometry*

Description

This function creates a simple feature linestring object from a series of x, y coordinates.

Usage

```
create_linestring(...)

## S3 method for class 'numeric'
create_linestring(..., crs = NULL)

## S3 method for class 'coordinates'
create_linestring(coordinates, crs = NULL, ...)
```

Arguments

`...`	A series of x, y coordinates.
`crs`	The coordinate reference system of the points.
`coordinates`	A `coordinates` object.

Value

A simple feature linestring object.

Examples

```
# Create a linestring from individual coordinates
linestring_1 <- create_linestring(0, 1, 1, 0, 2, 1)
linestring_1
plot(linestring_1)

# Create a linestring from a numeric vector
linestring_2 <- create_linestring(
  c(0, 1, 1, 1, 1, 0, 0, 0, 0, 0.5, 1, 0.5)
)
linestring_2
plot(linestring_2)
```

`create_line_graph` *Create a line graph from a network*

Description

This function creates a line graph where nodes represent the midpoints of links in the input network.

Usage

```
create_line_graph(network)
```

Arguments

`network`	A `road_network` or `segmented_network` object.

Value

A `igraph` object representing the midpoint graph.

Examples

```
# Create a road network
network <- create_road_network(sample_roads)

# Plot the road network
plot(network$graph)

# Create a midpoint graph from a road network
graph <- create_line_graph(network)

# Plot the midpoint graph
plot(graph)
```

`create_points` *Create points geometry*

Description

This function creates a simple feature collection of points from a series of x, y coordinates.

Usage

```
create_points(...)

## S3 method for class 'numeric'
create_points(..., crs = NULL)

## S3 method for class 'coordinates'
create_points(coordinates, crs = NULL, ...)
```

Arguments

`...`	A series of x, y coordinates.
`crs`	The coordinate reference system of the points.
`coordinates`	A `coordinates` object.

Value

A simple feature points object.

Examples

```
# Create a points from individual coordinates
points_1 <- create_points(0, 1, 1, 0, 2, 1)
points_1
plot(points_1)

# Create a linestring from a numeric vector
```

```
points_2 <- create_points(
  c(0, 1, 1, 1, 1, 0, 0, 0, 0, 0.5, 1, 0.5)
)
points_2
plot(points_2)
```

---

create_polygon *Create polygon from bounding box*

Description

This function creates a rectangle polygon from a bounding box.

Usage

```
create_polygon(bbox)
```

Arguments

bbox	A numeric matrix with two columns and two rows. bbox represents the minimum and maximum coordinates.

Value

A rectangle polygon with the same size as bbox.

---

decompose_linestring *Decompose a linestring into a list of line segments*

Description

This function decomposes a linestring into a list of line segments, where each segment is a linestring connecting two consecutive points of the input linestring.

Usage

```
decompose_linestring(linestring)
```

Arguments

linestring	A linestring object.

Value

A list of linestring objects, each representing a segment of the input linestring.

Examples

```
# Create a linestring object
linestring <- create_linestring(0, -1, 0, 1, 2, 1, 2, 0, 0, 0)
plot(linestring)

# Decompose the linestring into line segments
segments <- decompose_linestring(linestring)
plot(segments, col = c("#E69F00", "#56B4E9", "#009E73", "#F0E442"))
```

`download_roads` *Download and process OpenStreetMap road data*

Description

This function downloads OpenStreetMap highway data within the specified bounding box. It corrects the geometries of reversed one-way roads. Optionally, it crops the data to the bounding box.

Usage

```
download_roads(bbox, crop = TRUE)
```

Arguments

`bbox`	A bounding box in the form of a matrix with two rows and two columns.
`crop`	A logical value indicating whether to crop the data to the bounding box. Defaults to `TRUE`.

Value

A data frame containing OpenStreetMap road data.

Examples

```
## Not run:
# Download road data
bbox <- create_bbox(
  north =  35.17377,
  south =  35.16377,
  east  = 136.91590,
  west  = 136.90090
)
roads <- download_roads(bbox)

# Plot the roads
plot(roads$geometry)

## End(Not run)
```

`exclude_points` *Exclude points from a points*

Description

This function excludes points from a points.

Usage

```
exclude_points(input_points, points_to_exclude)
```

Arguments

`input_points`	A `sfc` object containing points.
`points_to_exclude`	A `sfc` object containing points to exclude.

Value

A `sfc` object containing the points from `input_points` that are not in `points_to_exclude`.

Examples

```
# Create a set of points and a set of points to exclude
points <- create_points(0, 0, 1, 0, 0, 1, 1, 1)
points_to_exclude <- create_points(0, 0, 1, 1)
plot(points)
plot(points_to_exclude, add = TRUE, pch = 4)

# Exclude points in `points_to_exclude` from `points`
remain_points <- exclude_points(points, points_to_exclude)
plot(remain_points)
```

`extract_road_endpoints` *Extract road endpoints*

Description

This function extracts the endpoints of a set of roads. It identifies and counts the roads that intersect at each endpoint.

Usage

```
extract_road_endpoints(roads)
```

Arguments

`roads`	A linestring object representing roads. It should be have a column named `road_id`.

Value

A points object representing road endpoints. Each endpoint has the following columns:

- parent_road: A list of road IDs that intersect at the endpoint.
- num_overlaps: The number of roads that intersect at the endpoint.

`extract_road_intersections` *Extract road intersections*

Description

This function identifies and extracts intersections between roads. It considers only intersections where multiple roads on the same layer.

Usage

```
extract_road_intersections(roads)
```

Arguments

`roads`	A linestring object representing roads. It should be have a column named `layer` and a column named `road_id`.

Value

A data frame with the following columns:

- parent_road: A list of road IDs that intersect at the intersection.
- num_overlaps: The number of roads that intersect at the intersection.

Examples

```
# Extract road intersections
intersections <- extract_road_intersections(sample_roads)
intersections

# Plot the intersections
plot(sample_roads$geometry)
plot(intersections$geometry, pch = 16, col = "#E69F00", add = TRUE)
```

`extract_road_network_links` *Extract road network links*

Description

This function extracts the road network links from a set of road geometries. It splits the road geometries at the nodes and creates a new sf object representing the road network links.

Usage

```
extract_road_network_links(roads, nodes)
```

Arguments

`roads`	A linestring object representing the roads.
`nodes`	A point object representing the road network nodes.

Value

A linestring object representing the road network links.

See Also

To create the nodes, use the `extract_road_network_nodes()`

`extract_road_network_nodes` *Extract road network nodes*

Description

This function extracts nodes from roads represented as linestring objects downloaded from Open-StreetMap. Nodes are defined as intersections between roads and endpoints of road segments.

Usage

```
extract_road_network_nodes(roads)
```

Arguments

`roads`	A linestring object representing roads.

Value

A points object representing road network nodes.

`extract_segmented_network_links`
Extract segmented network links from a road network

Description

This function extracts the segmented network links from a road network based on a specified set of segmented network nodes. It splits the linestrings of the road network links at the points of the segmented network nodes to create the segmented network links.

Usage

```
extract_segmented_network_links(
  road_network,
  segmented_network_nodes,
  tolerance = 1e-05
)
```

Arguments

`road_network`	A `road_network` object created with `create_road_network()`
`segmented_network_nodes`	A data frame representing the segmented network nodes.
`tolerance`	A numeric value representing the maximum distance allowed between a linestring of a road link and a point of a segmented network node. If the distance exceeds this value, the point will not be used to split the linestring.

Value

A data frame representing the segmented network links.

Examples

```
# Create a road network
road_network <- create_road_network(sample_roads)

# Extract segmented network nodes by length of 1
segmented_network_nodes <- extract_segmented_network_nodes(road_network, 1)

# Extract segmented network links
segmented_network_links <- extract_segmented_network_links(
  road_network,
  segmented_network_nodes
)
segmented_network_links

# Plot the segmented network nodes and links
plot(segmented_network_links$geometry, col = "#E69F00")
plot(segmented_network_nodes$geometry, pch = 16, add = TRUE)
```

`extract_segmented_network_nodes` *Extract nodes of segmented network from a road network*

Description

This function extracts nodes at regular intervals along each link in a road network

Usage

```
extract_segmented_network_nodes(road_network, segment_length)
```

Arguments

`road_network`	A `road_network` object created with `create_network()`.
`segment_length`	The length of each segment to sample along the links.

Value

An `sf` object with the sampled points.

Examples

```
# Create a road network
road_network <- create_road_network(sample_roads)

# Extract nodes with a segment length of 1
segmented_network_nodes <- extract_segmented_network_nodes(road_network, 1)
segmented_network_nodes

# Plot the segmented network nodes
plot(road_network)
plot(segmented_network_nodes$geometry, add = TRUE, pch = 16, col = "#E69F00")
```

`filter_points_within_tolerance` *Filter points within a tolerance distance*

Description

This function filters points that are within a specified tolerance distance from a reference linestring.

Usage

```
filter_points_within_tolerance(points, linestring, tolerance = 0.01)
```

Arguments

`points`	A `sfc` object containing points to filter.
`linestring`	A linestring object.
`tolerance`	A numeric value representing the maximum allowable distance between points and the linestring. Points within this distance from the linestring will be included in the filtered set.

Value

A `sfc` object containing the filtered points.

Examples

```
# Create a points
points <- create_points(
  0.000, 1.000, 0.500, 0.600, 1.000, 0.010, 1.500, 0.501, 2.000, 0.990
)

# Create a linestring
linestring <- create_linestring(0, 1, 1, 0, 2, 1)

# Plot the points
plot(linestring, col = "gray")
plot(points, add = TRUE)

# Filter points within a tolerance distance (default: 0.01)
filtered_points <- filter_points_within_tolerance(points, linestring)

# Plot the filtered points
plot(linestring, col = "gray")
plot(filtered_points, add = TRUE)
```

`generate_ids` *Generate unique IDs based on parent IDs*

Description

This function generates unique IDs based on parent IDs. It first extracts the first ID from each parent ID, then ranks these IDs and uses the ranks to format the IDs.

Usage

```
generate_ids(parent_list, id_format)
```

Arguments

`parent_list`	A list of parent IDs.
`id_format`	A format string for the IDs.

Value

A vector of unique IDs.

Examples

```
parent_list <- c("rd_0001", "rd_0002", "rd_0003")
id_format <- "jn_%06x"
generate_ids(parent_list, id_format)
```

---

get_adjacent_links *Get adjacent links of a specified link*

Description

This function returns a vector of link IDs that are adjacent to a specified link in a road network.

Usage

```
get_adjacent_links(network, link_ids, reachable_only = FALSE)
```

Arguments

`network`	A `road_network` or `segmented_network` object.
`link_ids`	The ID of the link to find adjacent links for.
`reachable_only`	If `FALSE`, the default, all adjacent links are returned. If `TRUE`, only reachable links are returned.

Value

A vector of link ID adjacent to the specified link.

Examples

```
# Create a road network
road_network <- create_road_network(sample_roads)
target_link <- road_network$links$id[7]
target_link

# Get adjacent links
adjacent_links <- get_adjacent_links(road_network, target_link)
adjacent_links

# Plot the target link and the adjacent links
is_adjacent <- road_network$links$id %in% adjacent_links
adjacent_links_geom <- road_network$links$geometry[is_adjacent]
plot(road_network, col = "gray")
plot(road_network$links$geometry[7], add = TRUE, pch = 19)
plot(adjacent_links_geom, add = TRUE, col = "#E69F00", lwd = 2)
```

`get_connected_links` *Get connected links from a specified node*

Description

This function retrieves all links connected to a specified node in a road network.

Usage

```
get_connected_links(network, node_ids)
```

Arguments

`network`	A `road_network` or `segmented_network` object.
`node_ids`	A vector of node IDs to find connected links for.

Value

A vector of link IDs connected to the specified node.

Examples

```
# Create a road network
road_network <- create_road_network(sample_roads)
target_node <- road_network$nodes[3,]$id
target_node

# Get connected links
connected_links <- get_connected_links(road_network, target_node)
connected_links

# Plot the target node and the connected links
is_connected <- road_network$links$id %in% connected_links
connected_links_geom <- road_network$links$geometry[is_connected]
plot(road_network, col = "gray")
plot(connected_links_geom, add = TRUE, col = "#E69F00", lwd = 2)
plot(road_network$nodes[3,]$geometry, add = TRUE, pch = 19)
```

`osm_highway_values` *Values for the 'highway' tag in OpenStreetMap data*

Description

`osm_highway_values` contains a character vector of values for the 'highway' tag in OpenStreetMap data.

Usage

```
osm_highway_values
```

Format

A character vector of length 12.

Examples

```
osm_highway_values
```

---

plot.road_network *Plot a road network*

Description

This function plots a road network.

Usage

```
## S3 method for class 'road_network'
plot(x, y, mode = c("default", "event", "graph"), ...)
```

Arguments

x	A road network object.
y	This argument is not used.
mode	The plotting mode. "event" mode shows the location events assigned to the road network. "graph" mode shows the direction of links when the network is directed. This utilizes the `plot.igraph` function.
...	Additional arguments passed to or from other methods.

Examples

```
# Create the road network
road_network <- create_road_network(sample_roads, events = sample_accidents)

# Plot the road network
plot(road_network)

# Plot the road network with events
plot(road_network, mode = "event")

# Plot the road network as a graph
plot(road_network, mode = "graph")
```

remove_points_near_endpoints

Remove points near endpoints of a linestring

Description

This function removes points from a set of points that are within a specified tolerance distance of the start or end point of a linestring.

Usage

```
remove_points_near_endpoints(points, linestring, tolerance = 0.01)
```

Arguments

`points`	A `sfc` object containing points to filter.
`linestring`	A linestring object.
`tolerance`	A numeric value representing the maximum distance from the line's ends to consider points for filtering. Points within this distance will be excluded.

Value

A `sfc` object containing the filtered points.

Examples

```
# Create a points
points <- create_points(
  0.000, 1.000, 0.500, 0.600, 1.000, 0.010, 1.500, 0.501, 2.000, 0.990
)

# Create a linestring
linestring <- create_linestring(0, 1, 1, 0, 2, 1)

# Plot the points
plot(linestring, col = "gray")
plot(points, add = TRUE)

# Filter points within a tolerance distance (default: 0.01)
filtered_points <- remove_points_near_endpoints(points, linestring)

# Plot the filtered points
plot(linestring, col = "gray")
plot(filtered_points, add = TRUE)
```

road_network *Create a road network from roads*

Description

This function constructs a road network from a set of roads. The road network is represented nodes and links between nodes. Nodes are defined as intersections between roads and endpoints of road segments.

Usage

```
create_road_network(roads, directed = FALSE, events = NULL, ...)
```

Arguments

`roads`	A linestring object representing roads.
`directed`	Logical indicating whether the road network is directed.
`events`	A `sf` object representing events.
`...`	Additional arguments passed to or from other methods.

Value

A road network object.

Examples

```
# Create the road network
road_network <- create_road_network(sample_roads)

# Print the road network summary
road_network

# Plot the road network
plot(road_network)
```

sample_accidents *Sample accidents data*

Description

A sample dataset containing information about 10 accidents. The dataset is for demonstration and testing and does not represent real-world events.

Usage

```
sample_accidents
```

Format

A `sf` object with 10 rows and 4 columns:

id Accident ID

time Time of the accident (in hours)

weather Weather condition at the time of the accident

severity Severity of the accident

geometry Point geometry of the accident location

Examples

```
# Show information about the accidents
sample_accidents

# Plot the locations of the accidents
plot(sample_roads$geometry)
plot(sample_accidents$geometry, pch = 4, col = "red", add = TRUE)
```

`sample_points_along_linestring`
Sample points along a linestring

Description

This function samples points at regular intervals along a linestring object.

Usage

```
sample_points_along_linestring(linestrings, segment_length)
```

Arguments

`linestrings`	A `sfc` object containing linestrings.
`segment_length`	The desired length of each segment between sampled points.

Value

A `sfc` object containing the sampled points.

Examples

```
library(sf)

# Create a linestrings
linestrings <- c(
  create_linestring(0, 1, 2, 1),
  create_linestring(1, 1.3, 1, 0, 2, 0.5)
)

# Sample points along the linestrings
sampled_points <- sample_points_along_linestring(linestrings, 0.5)

# Plot the sampled points
plot(linestrings)
plot(sampled_points, add = TRUE, pch = 16, col = c("#E69F00", "#56B4E9"))
```

`sample_roads` *Sample roads data*

Description

A sample dataset representing simple roads. It intended for demonstration and testing and does not represent real-world roads.

Usage

```
sample_roads
```

Format

A `sf` linestring object with 6 rows and 2 columns:

id Road ID

layer Layer of the road (indicates a road on an elevated structure such as a bridge, tunnel, or highway)

geometry Linestring geometry of the road

Examples

```
# Show information about the roads
sample_roads

# Shape of the roads
plot(sample_roads$geometry)
```

segmented_network *Create a segmented road network*

Description

This function creates a segmented road network by dividing each link of the input road network into segments of a specified length.

Usage

```
create_segmented_network(road_network, segment_length = 1, events = NULL, ...)

## S3 method for class 'road_network'
create_segmented_network(road_network, segment_length = 1, events = NULL, ...)
```

Arguments

road_network	A road_network object representing the input road network.
segment_length	A numeric value specifying the length of each segment.
events	A sf object representing events.
...	Additional arguments passed to or from other methods.

Value

A segmented_network object.

Examples

```
# Create a road network
road_network <- create_road_network(sample_roads)

# Create a segmented road network
segmented_network <- create_segmented_network(
  road_network,
  segment_length = 0.5
)
segmented_network

# Plot the segmented road network
plot(segmented_network)
```

`set_events` *Set events on a space.*

Description

This function sets events on a space.

Usage

```
set_events(x, events, ...)

## S3 method for class 'road_network'
set_events(x, events, ...)

## S3 method for class 'segmented_network'
set_events(x, events, ...)
```

Arguments

x	A `road_network` or `segmented_network` object.
events	A `sf` object representing events.
...	Additional arguments passed to or from other methods.

Value

The input object with the events added.

Examples

```
# Create the road network
road_network <- create_road_network(sample_roads)

# Set accidents on the road network
road_network <- set_events(road_network, sample_accidents)
```

`spatiotemporal_event` *Create a spatiotemporal event collection*

Description

This function creates a spatiotemporal event collection object from a given `sf` object, allowing spatial and temporal data to be handled together.

Usage

```
create_spatiotemporal_event(x, ...)

## S3 method for class 'sf'
create_spatiotemporal_event(
  x,
  time_column_name = "time",
  time_format = "%H",
  ...
)
```

Arguments

`x`	A `sf` object representing the spatial data.
`...`	Additional arguments passed to or from other methods.
`time_column_name`	A character string specifying the column name in `x` that contains time-related data. Defaults to `"time"`
`time_format`	A character string specifying the format of the time data in the `time_column_name`. For example, you can use formats like `"%Y-%m-%d"` for dates or `"%H:%M:%S"` for time. Defaults to `"%H"`.

Value

An `sf` object with class `spatiotemporal_event` added, representing a spatiotemporal event collection.

Examples

```
# Simple feature collection object representing accidents
sample_accidents

# Create a spatiotemporal event collection
create_spatiotemporal_event(sample_accidents)
```

`split_linestring` *Split a linestring into multiple segments*

Description

Split a linestring into multiple segments

Usage

```
split_linestring(linestring, split_points, tolerance = 0.01)
```

Arguments

`linestring`	A linestring object.
`split_points`	A `sfc` object containing points to split the linestring.
`tolerance`	A numeric value representing the maximum distance allowed between the linestring and the split points. If the distance between a split point and the linestring exceeds this value, the point will not be used to split the linestring.

Value

A `sfc` object containing the split linestrings.

Examples

```
# Create a linestring
linestring <- create_linestring(0, -1, 0, 1, 2, 1, 2, 0, 0, 0)

# Create a set of split points
split_points <- create_points(0, 0, 1, 1, 2, 0)

# Plot the linestring and split points
plot(linestring)
plot(split_points, add = TRUE)

# Split the linestring
segments <- split_linestring(linestring, split_points)

# Plot the split linestrings
plot(segments, col = c("#E69F00", "#56B4E9", "#009E73", "#F0E442"), lwd = 2)
```

`transform_to_cartesian` *Transform to Cartesian coordinates*

Description

This function transforms a spatial object to Cartesian coordinates (plane rectangular coordinate system). Currently, it supports the coordinate system used in Aichi Prefecture (JDG2011, EPSG:6675).

Usage

```
transform_to_cartesian(x)
```

Arguments

`x`	A spatial object.

Value

A spatial object transformed to Cartesian coordinates.

See Also

`transform_to_geographic()` to transform to geographic coordinates.

Examples

```
# Create points
points <- create_points(136.9024, 35.1649, crs = 4326)
points

# Transform to Cartesian coordinates
transformed <- transform_to_cartesian(points)
transformed
```

---

transform_to_geographic

Transform to geographic coordinates

Description

This function transforms a spatial object to geographic coordinates (WGS84, EPSG:4326).

Usage

```
transform_to_geographic(x)
```

Arguments

`x`	A spatial object.

Value

A spatial object transformed to geographic coordinates.

See Also

`transform_to_cartesian()` to transform to Cartesian coordinates.

Examples

```
# Create points
points <- create_points(-24073.54, -92614.18, crs = 6675)
points

# Transform to geographic coordinates
transformed <- transform_to_geographic(points)
transformed
```

Index