

Package ‘rcoins’

July 23, 2025

Title Identify Naturally Continuous Lines in a Spatial Network

Version 0.3.2

Description Provides functionality to group lines that form naturally continuous lines in a spatial network. The algorithm implemented is based on the Continuity in Street Networks (COINS) method from Tripathy et al. (2021) <[doi:10.1177/2399808320967680](https://doi.org/10.1177/2399808320967680)>, which identifies continuous “strokes” in the network as the line strings that maximize the angles between consecutive segments.

License Apache License (>= 2)

URL <https://cityriverspaces.github.io/rcoins/>,
<https://doi.org/10.5281/zenodo.14501805>,
<https://github.com/CityRiverSpaces/rcoins>

BugReports <https://github.com/CityRiverSpaces/rcoins/issues>

Encoding UTF-8

RoxygenNote 7.3.2

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

Config/testthat/edition 3

Imports dplyr, rlang, sf, sfheaders

VignetteBuilder knitr

Depends R (>= 4.1.0)

NeedsCompilation no

Author Francesco Nattino [aut, cre, cph] (ORCID:
<<https://orcid.org/0000-0003-3286-0139>>),
Claudiu Forgaci [aut] (ORCID: <<https://orcid.org/0000-0003-3218-5102>>),
Fakhreh Alidoost [aut] (ORCID:
<<https://orcid.org/0000-0001-8407-6472>>),
Thijs van Lankveld [ctb] (ORCID:
<<https://orcid.org/0009-0001-1147-4813>>),
Netherlands eScience Center [fnd]

Maintainer Francesco Nattino <f.nattino@esciencecenter.nl>

Repository CRAN
Date/Publication 2025-05-29 18:20:02 UTC

Contents

get_example_data	2
stroke	2
Index	5

get_example_data	<i>Get example data</i>
------------------	-------------------------

Description

This function retrieves example OpenStreetMap (OSM) data for the city of Bucharest, Romania, from a persistent URL on the 4TU.ResearchData data repository. The dataset includes the street network and the geometry of the Dâmbovița river.

Usage

```
get_example_data()
```

Value

A list of sf objects containing the OSM data.

Examples

```
get_example_data()
```

stroke	<i>Identify naturally continuous lines in a spatial network</i>
--------	---

Description

Provides functionality to group lines that form naturally continuous lines in a spatial network. The algorithm implemented is based on the Continuity in Street Networks (COINS) method [doi:10.1177/2399808320967680](https://doi.org/10.1177/2399808320967680), which identifies continuous "strokes" in the network as the line strings that maximize the angles between consecutive segments.

Usage

```
stroke(
  edges,
  angle_threshold = 0,
  attributes = FALSE,
  flow_mode = FALSE,
  from_edge = NULL
)
```

Arguments

<code>edges</code>	An object of class <code>sfc</code> (or compatible), including the network edge geometries (should be of type <code>LINESTRING</code>).
<code>angle_threshold</code>	Consecutive line segments can be considered part of the same stroke if the internal angle they form is larger than <code>angle_threshold</code> (in degrees). It should fall in the range $0 \leq \text{angle_threshold} < 180$.
<code>attributes</code>	If <code>TRUE</code> , return a label for each edge, representing the groups each edge belongs to. Only possible for <code>flow_mode = TRUE</code> .
<code>flow_mode</code>	If <code>TRUE</code> , line segments that belong to the same edge are not split across strokes (even if they form internal angles smaller than <code>angle_threshold</code>).
<code>from_edge</code>	Only look for the continuous strokes that include the provided edges or line segments.

Value

An object of class `sfc` (if `attributes = FALSE`), a vector with the same length as `edges` otherwise.

Examples

```
library(sf)

# Setup a simple network

p1 <- st_point(c(0, 3))
p2 <- st_point(c(2, 1))
p3 <- st_point(c(3, 0))
p4 <- st_point(c(1, 4))
p5 <- st_point(c(3, 2))
p6 <- st_point(c(4, 1))
p7 <- st_point(c(4, 3))
p8 <- st_point(c(5, 3))

l1 <- st_linestring(c(p1, p2, p5))
l2 <- st_linestring(c(p2, p3))
l3 <- st_linestring(c(p4, p5))
l4 <- st_linestring(c(p5, p6))
l5 <- st_linestring(c(p5, p7))
l6 <- st_linestring(c(p7, p8))
```

```
network_edges <- st_sfc(l1, l2, l3, l4, l5, l6)

# Identify strokes in the full network with default settings
stroke(network_edges)

# Set a threshold to the angle between consecutive segments
stroke(network_edges, angle_threshold = 150)

# Identify strokes in flow mode (do not break initial edges)
stroke(network_edges, flow_mode = TRUE)

# Instead of returning stroke geometries, return stroke labels
stroke(network_edges, flow_mode = TRUE, attributes = TRUE)

# Identify strokes that continue one (or a subset) of edges
stroke(network_edges, from_edge = 2)
stroke(network_edges, from_edge = c(2, 3))
```

Index

`get_example_data`, [2](#)

`sfc`, [3](#)

`stroke`, [2](#)