Package 'flowcluster'

July 5, 2025

Title Cluster Origin-Destination Flow Data

Version 0.1.0

Description Provides functionality for clustering origin-destination (OD) pairs, representing desire lines (or flows). This includes creating distance matrices between OD pairs and passing distance matrices to a clustering algorithm. See the academic paper Tao and Thill (2016) <doi:10.1111/gean.12100> for more details on spatial clustering of flows.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

URL https://hussein-mahfouz.github.io/flowcluster/

Depends R (>= 4.1.0)

Imports sf, dbscan, dplyr, glue, lwgeom, tibble, units, tidyr, tidyselect

LazyData true

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Author Hussein Mahfouz [aut, cre] (ORCID: <https://orcid.org/0000-0003-1706-7802>), Robin Lovelace [aut] (ORCID: <https://orcid.org/0000-0001-5679-6536>)

Maintainer Hussein Mahfouz <husseinmahfouz93@gmail.com>

Repository CRAN

Date/Publication 2025-07-05 18:50:11 UTC

Contents

`add_flow_length` *Add Length Column to Flow Data*

Description

Also checks that 'origin' and 'destination' columns are present.

Usage

```
add_flow_length(x)
```

Arguments

x	sf object of flows (LINESTRING, projected CRS)

Value

sf object with an additional length_m column (od length in meters)

Examples

```
flows <- sf::st_transform(flows_leeds, 3857)
flows <- add_flow_length(flows)
```

`add_xyuv` *Add Start/End Coordinates & Flow IDs*

Description

Add Start/End Coordinates & Flow IDs

Usage

```
add_xyuv(x)
```

Arguments

x	sf object of flows

Value

tibble with x, y, u, v, flow_ID columns

Examples

```
flows <- sf::st_transform(flows_leeds, 3857)
flows <- add_flow_length(flows)
flows <- add_xyuv(flows)
```

`cluster_flows_dbscan` *Cluster Flows using DBSCAN*

Description

See dbscan for details on the DBSCAN algorithm.

Usage

```
cluster_flows_dbscan(dist_mat, w_vec, x, eps, minPts)
```

Arguments

`dist_mat`	distance matrix
`w_vec`	weight vector
`x`	flows tibble with flow_ID
`eps`	DBSCAN epsilon parameter
`minPts`	DBSCAN minPts parameter

Value

flows tibble with an additional cluster column

Examples

```
flows <- sf::st_transform(flows_leeds, 3857)
flows <- head(flows, 100) # for testing
# Add flow lengths and coordinates
flows <- add_flow_length(flows)
# filter by length
flows <- filter_by_length(flows, length_min = 5000, length_max = 12000)
flows <- add_xyuv(flows)
# Calculate distances
distances <- flow_distance(flows, alpha = 1.5, beta = 0.5)
dmat <- distance_matrix(distances)
wvec <- weight_vector(dmat, flows, weight_col = "count")
clustered <- cluster_flows_dbscan(dmat, wvec, flows, eps = 8, minPts = 70)
```

dbscan_sensitivity *Sensitivity analysis of DBSCAN parameters for flow clustering. The function allows you to test different combinations of epsilon and minPts parameters for clustering flows using DBSCAN. It can be used to determine what parameter values make sense for your data*

Description

Sensitivity analysis of DBSCAN parameters for flow clustering. The function allows you to test different combinations of epsilon and minPts parameters for clustering flows using DBSCAN. It can be used to determine what parameter values make sense for your data

Usage

```
dbscan_sensitivity(
  dist_mat,
  flows,
  options_epsilon,
  options_minpts,
  w_vec = NULL
)
```

Arguments

dist_mat	a precalculated distance matrix between desire lines (output of distance_matrix())
flows	the original flows tibble (must contain flow_ID and 'count' column)
options_epsilon	a vector of options for the epsilon parameter
options_minpts	a vector of options for the minPts parameter
w_vec	Optional precomputed weight vector (otherwise computed internally from 'count' column)

Value

a tibble with columns: id (to identify eps and minpts), cluster, size (number of desire lines in cluster), count_sum (total count per cluster)

Examples

```
flows <- sf::st_transform(flows_leeds, 3857)
flows <- head(flows, 1000) # for testing
# Add flow lengths and coordinates
flows <- add_flow_length(flows)
# filter by length
flows <- filter_by_length(flows, length_min = 5000, length_max = 12000)
# Add x, y, u, v coordinates to flows
flows <- add_xyuv(flows)
```

```
# Calculate distance matrix
distances <- flow_distance(flows, alpha = 1.5, beta = 0.5)
dmat <- distance_matrix(distances)
# Generate weight vector
w_vec <- weight_vector(dmat, flows, weight_col = "count")

# Define the parameters for sensitivity analysis
options_epsilon <- seq(1, 10, by = 2)
options_minpts <- seq(10, 100, by = 10)
# # Run the sensitivity analysis
results <- dbscan_sensitivity(
  dist_mat = dmat,
  flows = flows,
  options_epsilon = options_epsilon,
  options_minpts = options_minpts,
  w_vec = w_vec
)
```

`distance_matrix` *Convert Long-Format Distance Tibble to Matrix*

Description

Convert Long-Format Distance Tibble to Matrix

Usage

```
distance_matrix(distances, distance_col = "fds")
```

Arguments

`distances`	tibble with columns flow_ID_a, flow_ID_b, and distance
`distance_col`	column name for distance (default "fds")

Value

distance matrix (tibble with rownames). The matrix has flow_ID_a as rownames and flow_ID_b as column names. This function converts the output of `flow_distance()` into a format suitable for the dbscan clustering algorithm.

Examples

```
flows <- sf::st_transform(flows_leeds, 3857)
flows <- head(flows, 100) # for testing
# Add flow lengths and coordinates
flows <- add_flow_length(flows)
flows <- add_xyuv(flows)
# Calculate distances
distances <- flow_distance(flows, alpha = 1.5, beta = 0.5)
dmat <- distance_matrix(distances)
```

`filter_by_length` *Filter Flows by Length*

Description

Filter Flows by Length

Usage

```
filter_by_length(x, length_min = 0, length_max = Inf)
```

Arguments

`x`	sf object with length_m
`length_min`	minimum length (default 0)
`length_max`	maximum length (default Inf)

Value

filtered sf object. Flows with length_m outside the specified range are removed.

Examples

```
flows <- sf::st_transform(flows_leeds, 3857)
flows <- add_flow_length(flows)
flows <- filter_by_length(flows, length_min = 5000, length_max = 12000)
```

`flows_leeds` *Example flow data for Leeds. It is from the 2021 census, and it contains all Origin - Destination flows at the MSOA level. For more info on census flow data, see the* R*hrefhttps://www.ons.gov.uk/census/aboutcensus/censusproducts/origindestinationflowdataONS documentation See data-raw/flows_leeds.R for how this data was created.*

Description

Example flow data for Leeds. It is from the 2021 census, and it contains all Origin - Destination flows at the MSOA level. For more info on census flow data, see the ONS documentation See data-raw/flows_leeds.R for how this data was created.

Usage

```
flows_leeds
```

Format

An object of class sf with LINESTRING geometry. It has the following columns:

origin MSOA code of origin zone

destination MSOA code of destination zone

count number of people moving from origin to destination

geometry desire line between origin and destination

Source

https://www.nomisweb.co.uk/sources/census_2021_od

---

flow_distance *Calculate Flow Distance and Dissimilarity*

Description

This function calculates flow distance and dissimilarity measures between all pairs of flows based on the method described in @tao2016spatial.

Usage

```
flow_distance(x, alpha = 1, beta = 1)
```

Arguments

x	tibble with flow_ID, x, y, u, v, length_m
alpha	numeric, origin weight
beta	numeric, destination weight

Value

tibble of all OD pairs with fd, fds columns

References

Tao, R., Thill, J.-C., 2016. Spatial cluster detection in spatial flow data. Geographical Analysis 48, 355–372. https://doi.org/10.1111/gean.12100

Examples

```
flows <- sf::st_transform(flows_leeds, 3857)
flows <- head(flows, 100) # for testing
# Add flow lengths and coordinates
flows <- add_flow_length(flows)
flows <- add_xyuv(flows)
# Calculate distances
distances <- flow_distance(flows, alpha = 1.5, beta = 0.5)
```

`weight_vector` *Generate Weight Vector from Flows*

Description

Generate Weight Vector from Flows

Usage

```
weight_vector(dist_mat, x, weight_col = "count")
```

Arguments

`dist_mat`	distance matrix
`x`	flows tibble with flow_ID and weight_col
`weight_col`	column to use as weights (default = "count")

Value

numeric weight vector. Each element corresponds to a flow in the distance matrix, and is used as a weight in the DBSCAN clustering algorithm.

Examples

```
flows <- sf::st_transform(flows_leeds, 3857)
flows <- head(flows, 100) # for testing
# Add flow lengths and coordinates
flows <- add_flow_length(flows)
flows <- add_xyuv(flows)
# Calculate distances
distances <- flow_distance(flows, alpha = 1.5, beta = 0.5)
dmat <- distance_matrix(distances)
wvec <- weight_vector(dmat, flows, weight_col = "count")
```

Index