Package 'od'

November 15, 2023

Title Manipulate and Map Origin-Destination Data

Version 0.4.3

Description The aim of 'od' is to provide tools and example datasets for working with origin-destination ('OD') datasets of the type used to describe aggregate urban mobility patterns (Carey et al. 1981) <doi:10.1287/trsc.15.1.32>. The package builds on functions for working with 'OD' data in the package 'stplanr', (Lovelace and Ellison 2018) <doi:10.32614/RJ-2018-053> with a focus on computational efficiency and support for the 'sf' class system (Pebesma 2018) <doi:10.32614/RJ-2018-009>. With few dependencies and a simple class system based on data frames, the package is intended to facilitate efficient analysis of 'OD' datasets and to provide a place for developing new functions. The package enables the creation and analysis of geographic entities representing large scale mobility patterns, from daily travel between zones in cities to migration between countries.

License GPL-3

URL https://github.com/itsleeds/od, https://itsleeds.github.io/od/

BugReports https://github.com/itsleeds/od/issues

Encoding UTF-8

LazyData true

Depends R (>= 3.4.0)

Imports sfheaders, methods, vctrs

Suggests sf, knitr, rmarkdown, tinytest, covr, lwgeom

RoxygenNote 7.2.3

VignetteBuilder knitr

NeedsCompilation no

Author Robin Lovelace [aut, cre] (<https://orcid.org/0000-0001-5679-6536>), David Cooley [ctb]

Maintainer Robin Lovelace <rob00x@gmail.com>

Repository CRAN

Date/Publication 2023-11-15 09:40:07 UTC

R topics documented:

`coords_to_od` *Convert coordinates into a data frame of origins and destinations*

Description

Takes geographic coordinates and converts them into a data frame representing the potential flows, or 'spatial interaction', between every combination of points.

Usage

```
coords_to_od(p, interzone_only = FALSE, ids_only = FALSE)
```

Arguments

p	A spatial points object or a matrix of coordinates representing points
interzone_only	Should the result only include interzonal OD pairs, in which the ID of the origin is different from the ID of the destination zone? FALSE by default
ids_only	Should a data frame with only 2 columns (origin and destination IDs) be returned? The default is FALSE, meaning the result should also contain the coordinates of the start and end points of each OD pair.

Value

A data frame object with O and D codes and origin and destination coordinates.

Examples

```
p = sf::st_coordinates(od_data_centroids[1:3, ])
od = points_to_od(p)
(od = coords_to_od(p, interzone_only = TRUE))
l = odc_to_sf(od[3:6], d = od[1:2])
l$v = 1
(l_oneway = od_oneway(l))
plot(l_oneway)
```

---

odc_to_sf *Convert origin-destination coordinates into geographic desire lines*

Description

Convert origin-destination coordinates into geographic desire lines

Usage

```
odc_to_sf(odc, d = NULL, crs = 4326)
```

Arguments

odc	A matrix containing coordinates representing line start and end points
d	An optional data frame to add to the geometry column
crs	The coordinate reference system of the output, if not known in z. 4326 by default.

Examples

```
(odc = od_coordinates(od_data_df, p = od_data_zones, sfnames = TRUE))
(l = odc_to_sf(odc))
plot(l)
lsfc = odc_to_sfc(odc)
```

odc_to_sfc *Convert origin-destination coordinates into geographic desire lines*

Description

Convert origin-destination coordinates into geographic desire lines

Usage

```
odc_to_sfc(odc)
```

Arguments

odc	A matrix containing coordinates representing line start and end points

Examples

```
(odc = od_coordinates(od_data_df, p = od::od_data_zones, sfnames = TRUE))
(l = odc_to_sfc(odc))
plot(l)
```

odmatrix_to_od *Convert origin-destination data from wide to long format*

Description

This function takes a matrix representing travel between origins (with origin codes in the `rownames` of the matrix) and destinations (with destination codes in the `colnames` of the matrix) and returns a data frame representing origin-destination pairs.

Usage

```
odmatrix_to_od(odmatrix)
```

Arguments

odmatrix	A matrix with row and columns representing origin and destination zone codes and cells representing the flow between these zones.

Details

The function returns a data frame with rows ordered by origin and then destination zone code values and with names `orig`, `dest` and `flow`.

See Also

Other od: `od_id`, `od_to_odmatrix()`

Examples

```
x = od_data_df
x[1:3]
odmatrix = od_to_odmatrix(od_data_df)
odmatrix
odmatrix_to_od(odmatrix)
```

`od_aggregate` *Aggregate od pairs based on aggregating zones*

Description

This function is for aggregating OD pairs. It generally decreases the number of rows in an OD dataset, while aiming to keep the amount of travel represented in the data the same.

Usage

```
od_aggregate(od, aggzones = NULL, FUN = sum)

od_group(od, aggzones = NULL, FUN = sum)
```

Arguments

`od`	An origin-destination data frame
`aggzones`	Points within the zones defining the OD data
`FUN`	The aggregating function to use

Details

An alias for the function is `od_group()`.

Examples

```
od_aggregated = od_data_df[1:2, c(1, 2, 9)]
aggzones = od::od_data_zones_min
subzones = od_data_zones_small
plot(aggzones$geometry)
plot(subzones$geometry, add = TRUE)
od = od_disaggregate(od_aggregated, aggzones, subzones)
od_agg = od_aggregate(od, aggzones)
names(od_agg)[1:(ncol(od_agg) - 1)] = names(od_aggregated)
attr(od_aggregated, "spec") = NULL
identical(sf::st_drop_geometry(od_agg), od_aggregated)
```

od_coordinates *Create matrices representing origin-destination coordinates*

Description

This function takes a wide range of input data types (spatial lines, points or text strings) and returns a data frame of coordinates representing origin (ox, oy) and destination (dx, dy) points.

Usage

```
od_coordinates(x, p = NULL, pd = NULL, silent = TRUE, sfnames = FALSE)
```

Arguments

x	A data frame in which the first two columns are codes representing points/zones of origin and destination
p	Points representing origins and destinations
pd	Points representing destinations, if different from origin points
silent	Hide messages? FALSE by default.
sfnames	Should output column names be compatible with the sf package?

Value

A data frame with origin and destination coordinates

Examples

```
x = od_data_df
p = od_data_centroids
res = od_coordinates(x, p)[1:2, ]
class(res)
res
od_coordinates(x, p, sfnames = TRUE)[1:2, ]
od_coordinates(x, p, silent = FALSE)[1:2, ]
od_coordinates(x, p)
x = od_data_df2[1:3, ]
p = od_data_centroids2
pd = od_data_destinations
od_coordinates(x, p, pd)
```

od_coordinates_ids *Interleave origin and destination coordinates*

Description

This function takes a matrix with 4 columns representing origin and destination coordinates and returns a data frame with 3 columns with the ID of each linestring, plus the coordinates representing origin and destination coordinates. Essentially the function is a pivot, converting from wide to long format, to feed into other functions for creating geographic desire lines.

Usage

```
od_coordinates_ids(odc)
```

Arguments

odc	A matrix containing coordinates representing line start and end points

Examples

```
od_coordinates_ids(od_coordinates(od_data_df, p = od_data_zones, sfnames = TRUE))
```

od_data_buildings *Simple buildings dataset*

Description

Building data from OSM for testing od_disaggregate.

Examples

```
nrow(od_data_buildings)
head(od_data_buildings)
plot(od_data_buildings$geometry)
plot(od_data_zones_min$geometry, lwd = 3, col = NULL, add = TRUE)
```

od_data_centroids *Datasets reporesenting zone centroids*

Description

These are provided as a geographic (sf) object and a simple data frame with longitude (X) and latitude (Y) columns.

Note

The schema data can be (re-)generated using code in the data-raw directory.

Examples

```
head(od_data_coordinates)
```

od_data_centroids2 *Output area centroids*

Description

This dataset represents geographic centroids of Output Areas in Leeds, UK.

This dataset represents geographic centroids of Output Areas in Leeds, UK.

Note

The schema data can be (re-)generated using code in the data-raw directory.

The schema data can be (re-)generated using code in the data-raw directory.

Examples

```
head(od_data_centroids2)
head(od_data_centroids2)
```

od_data_destinations *Workplace zone (destination) centroids*

Description

This dataset represents geographic centroids of Output Areas in Leeds, UK.

Note

The schema data can be (re-)generated using code in the `data-raw` directory.

Examples

```
nrow(od_data_destinations)
head(od_data_destinations)
```

od_data_df *Origin-destination datasets*

Description

Datasets representing top commuter desire lines in Leeds based on the 2011 Census. The first two variables of the data frame are the zone code of origin and destination, respectively. The other columns record the number of people who travel by different modes, including `all`, `train`, `bus`, `bicycle` and by `foot`.

Details

`od_data_df_medium` is a larger dataset with the same variables, with around 10k rows.

Note

The schema data can be (re-)generated using code in the `data-raw` directory.

Examples

```
od_data_df
```

od_data_df2 *Origin-destination data with destinations in a different layer than origins*

Description

This dataset represents commuter flows between Output Areas and Workplace Zones, the most detailed open OD data in the UK. See https://wicid.ukdataservice.ac.uk/ and the script `data-raw/od_wpz.R` in the od package's GitHub repo.

Details

The dataset reports (in the 3rd column) the number of people travelling between origins and destinations.

Note

The schema data can be (re-)generated using code in the `data-raw` directory.

Examples

```
head(od_data_df2)
```

od_data_network *Route network data for Leeds*

Description

Route network data for Leeds

Note

The schema data can be (re-)generated using code in the `data-raw` directory.

Examples

```
head(od_data_network)
```

od_data_zones *Example OD data*

Description

Zone datasets for packages examples

Note

The schema data can be (re-)generated using code in the `data-raw` directory.

od_data_zones_small *Small zones dataset*

Description

This dataset represents geographic zones of Lower Super Output Areas in Leeds, UK. They fit completely within the `od_data_zones_min` dataset.

Note

The schema data can be (re-)generated using code in the `data-raw` directory.

Examples

```
nrow(od_data_zones_small)
head(od_data_zones_small)
plot(od_data_zones_small$geometry)
plot(od_data_zones_min$geometry, lwd = 3, col = NULL, add = TRUE)
```

od_disaggregate *Split-up each OD pair into multiple OD pairs based on sub-points/subzones*

Description

This function is for splitting-up OD pairs. It increases the number of rows in an OD dataset, while aiming to keep the amount of travel represented in the data the same. To take an analogy from another package, it's roughly equivalent to `tidyr::pivot_longer()`.

Usage

```
od_disaggregate(
  od,
  z,
  subpoints = NULL,
  code_append = "_ag",
  population_column = 3,
  max_per_od = 5,
  keep_ids = TRUE,
  integer_outputs = FALSE
)

od_split(
  od,
  z,
  subpoints = NULL,
  code_append = "_ag",
  population_column = 3,
  max_per_od = 5,
  keep_ids = TRUE,
  integer_outputs = FALSE
)
```

Arguments

od	An origin-destination data frame
z	Zones representing origins and destinations
subpoints	Points, lines or polygons within the zones. These define the OD data start/end points.
code_append	The name of the column containing aggregate zone names
population_column	The column containing the total population (if it exists)
max_per_od	Maximum flow in the population_column to assign per OD pair. This only comes into effect if there are enough subpoints to choose from.
keep_ids	Should the origin and destination ids be kept? TRUE by default, meaning 2 extra columns are appended, with the names o_agg and d_agg containing IDs from the original OD data.
integer_outputs	Should integer outputs be returned? FALSE by default. Note: there is a known issue when integer results are generated. See https://github.com/ITSLeeds/od/issues/31 for details.

Details

An alias for the function is od_split().

Examples

```
od = od_data_df[1:2, c(1, 2, 9)]
od
zones = od::od_data_zones_min
od_sf = od_to_sf(od, zones)
set.seed(2021) # for reproducibility
od_disag = od_disaggregate(od, zones)
od_disag2 = od_disaggregate(od, zones, max_per_od = 11)
plot(zones$geometry)
plot(od_sf$geometry, lwd = 9, add = TRUE)
plot(od_disag$geometry, col = "grey", lwd = 1, add = TRUE)
plot(od_disag2$geometry, col = "green", lwd = 1, add = TRUE)
table(od_disag$o_agg, od_disag$d_agg)
# integer results
od_disaggregate(od, zones, integer_outputs = TRUE)

# with more trips per disaggregated OD pair:
disag = od_disaggregate(od_data_df[1:2, ], z = zones, max_per_od = 50)
plot(disag[0])

# with subpoints
subpoints = sf::st_sample(zones, 100)
od_disag_subpoints = od_disaggregate(od, zones, subpoints = subpoints)
plot(subpoints)
plot(od_disag_subpoints$geometry, add = TRUE)

# with buildings data
od_disag_buildings = od_disaggregate(od, zones, od_data_buildings)
summary(od_disag_buildings)
plot(od_data_buildings$geometry)
plot(od_disag_buildings[3], add = TRUE)
# mapview::mapview(od_disag_buildings)

od = od_data_df[1:2, 1:4]
subzones = od_data_zones_small
try(od_disaggregate(od, zones, subzones))
od_disag = od_disaggregate(od, zones, subzones, max_per_od = 500)
ncol(od_disag) -3 == ncol(od) # same number of columns, the same...
# Except disag data gained geometry and new agg ids:
sum(od_disag[[3]]) == sum(od[[3]])
sum(od_disag[[4]]) == sum(od[[4]])
plot(od_disag)
# test with road network dataset (don't run as time consuming):
## Not run:
od_disag_net = od_disaggregate(od, zones, od_road_network, max_per_od = 500)
plot(zones$geometry)
plot(od_road_network$geometry, add = TRUE, col = "green")
plot(od_disag_net$geometry, add = TRUE)
mapview::mapview(zones) + od_disag_net + od_road_network

## End(Not run)
```

od_filter *Filter OD datasets*

Description

This function takes and OD dataset and a character vector of codes and returns an OD dataset with rows matching origin and destinations zones present in the codes.

Usage

```
od_filter(x, codes, silent = FALSE)
```

Arguments

x	A data frame in which the first two columns are codes representing points/zones of origin and destination
codes	The zone codes that must be in origins and destination
silent	Hide messages? FALSE by default.

Value

A data frame

Examples

```
x = od_data_df
z = od_data_zones
codes = z[[1]]
z_in_x_o = codes %in% x[[1]]
z_in_x_d = codes %in% x[[2]]
sum(z_in_x_d)
sum(z_in_x_o)
z = z[which(z_in_x_o | z_in_x_d)[-1], ]
z[[1]]
unique(c(x[[1]], x[[2]]))
try(od_to_sf(x, z)) # fails
nrow(x)
x = od_filter(x, z[[1]])
nrow(x)
od_to_sf(x, z)
```

od_id *Combine two ID values to create a single ID number*

Description

Combine two ID values to create a single ID number

Usage

```
od_id_szudzik(x, y, ordermatters = FALSE)

od_id_max_min(x, y)

od_id_character(x, y)
```

Arguments

x	a vector of numeric, character, or factor values
y	a vector of numeric, character, or factor values
ordermatters	logical, does the order of values matter to pairing, default = FALSE

Details

In OD data it is common to have many 'oneway' flows from "A to B" and "B to A". It can be useful to group these an have a single ID that represents pairs of IDs with or without directionality, so they contain 'twoway' or bi-directional values.
od_id* functions take two vectors of equal length and return a vector of IDs, which are unique for each combination but the same for twoway flows.

- the Szudzik pairing function, on two vectors of equal length. It returns a vector of ID numbers.

This function superseeds od_id_order as it is faster on large datasets

See Also

od_oneway

Other od: od_to_odmatrix(), odmatrix_to_od()

Examples

```
(d = od_data_df[2:9, 1:2])
(id = od_id_character(d[[1]], d[[2]]))
duplicated(id)
od_id_szudzik(d[[1]], d[[2]])
od_id_max_min(d[[1]], d[[2]])
```

od_id_order *Generate ordered ids of OD pairs so lowest is always first This function is slow on large datasets, see szudzik_pairing for faster alternative*

Description

Generate ordered ids of OD pairs so lowest is always first This function is slow on large datasets, see szudzik_pairing for faster alternative

Usage

```
od_id_order(x, id1 = names(x)[1], id2 = names(x)[2])
```

Arguments

x	A data frame representing OD pairs
id1	Optional (it is assumed to be the first column) text string referring to the name of the variable containing the unique id of the origin
id2	Optional (it is assumed to be the second column) text string referring to the name of the variable containing the unique id of the destination

Examples

```
x = data.frame(id1 = c(1, 1, 2, 2, 3), id2 = c(1, 2, 3, 1, 4))
od_id_order(x) # 4th line switches id1 and id2 so oneway_key is in order
```

od_interzone *Return only interzonal (io intrazonal) OD pairs*

Description

This function takes an OD dataset and returns only the rows corresponding to movements in which the origin is different than the destination.

Usage

```
od_interzone(x)

od_intrazone(x)
```

Arguments

x	A data frame in which the first two columns are codes representing points/zones of origin and destination

Examples

```
od_data = points_to_od(od_data_centroids)
nrow(od_data)
nrow(od_interzone(od_data))
nrow(od_intrazone(od_data))
```

`od_jitter` *Move desire line end points within zone to avoid all trips going to a single centroid*

Description

These functions tackle the problem associated with OD data representing movement to and from large zones. Typically the associated desire lines start and end in one point per zone. This function produces desire lines that can start and end anywhere (or at predefined points) within each zone. See issue #11 for details.

Usage

```
od_jitter(
  od,
  z,
  subpoints = NULL,
  code_append = "_ag",
  population_column = 3,
  max_per_od = 1e+05,
  keep_ids = TRUE,
  integer_outputs = FALSE,
  zd = NULL,
  subpoints_o = NULL,
  subpoints_d = NULL,
  disag = FALSE
)
```

Arguments

`od`	An origin-destination data frame
`z`	Zones representing origins and destinations
`subpoints`	Points, lines or polygons within the zones. These define the OD data start/end points.
`code_append`	The name of the column containing aggregate zone names
`population_column`	The column containing the total population (if it exists)
`max_per_od`	Maximum flow in the population_column to assign per OD pair. This only comes into effect if there are enough subpoints to choose from.

keep_ids	Should the origin and destination ids be kept? TRUE by default, meaning 2 extra columns are appended, with the names o_agg and d_agg containing IDs from the original OD data.
integer_outputs	Should integer outputs be returned? FALSE by default. Note: there is a known issue when integer results are generated. See https://github.com/ITSLeeds/od/issues/31 for details.
zd	Zones with ids matching the destination codes in input OD data
subpoints_o	Points within origin zones representing possible destinations
subpoints_d	Points within destination zones representing possible destinations
disag	Should the od_disaggregate function be used as a 'back end' where possible? FALSE by default. See https://github.com/ITSLeeds/od/issues/39.

Value

An sf data frame

Examples

```
# Basic example
od = od_data_df
z = od_data_zones_min
dlr = od_jitter(od, z) # desire_lines_random
desire_lines = od_to_sf(od, z)
plot(z$geometry)
plot(dlr["all"], add = TRUE, lwd = 3)
dlr$all
desire_lines$all
plot(desire_lines["all"], add = TRUE, lwd = 5)

# Example showing use of subpoints
subpoints_o = sf::st_sample(z, 200)
subpoints_d = sf::st_sample(z, 100)
dlr_d = od_jitter(od, z, subpoints_o = subpoints_o, subpoints_d = subpoints_d)
plot(z$geometry)
plot(dlr_d$geometry, add = TRUE)
plot(subpoints_o, add = TRUE)
plot(subpoints_d, col = "red", add = TRUE)
plot(desire_lines, add = TRUE, lwd = 5)
# mapview::mapview(desire_lines) + dlr + z # interactive map
sp = sf::st_sample(z, 100)
dlr2 = od_jitter(desire_lines, z, subpoints_o = sp, subpoints_d = sp)
plot(z$geometry)
plot(sp, add = TRUE)
plot(dlr2, add = TRUE, lwd = 3)
plot(desire_lines, add = TRUE, lwd = 5)

# Example showing jittering with origin and destination zones
od = od_data_df2
```

```
z = sf::st_buffer(od_data_centroids2, dist = 1000)
zd = sf::st_buffer(od_data_destinations, dist = 300)
zd = zd[zd[[1]] %in% od[[2]], ]
desire_lines = od_to_sf(od, od_data_centroids2, zd = od_data_destinations)
dlr = od_jitter(od, z, zd = zd)
plot(z$geometry)
plot(od_data_centroids2$geometry, add = TRUE)
plot(od_data_destinations$geometry, add = TRUE)
plot(zd$geometry, add = TRUE)
plot(dlr, add = TRUE, lwd = 3)
plot(desire_lines, add = TRUE, lwd = 5)

# Larger example with only subset of matching zones
# od = od_data_df_medium
# od_sf = od_to_sf(od, od_data_zones)
# dlr3 = od_jitter(od_sf, od_data_zones)
# plot(od_sf[od$all > 200, 1])
# plot(dlr3[od$all > 200, 1])
# mapview::mapview(od_sf$geometry[od$all > 200])
```

---

od_oneway *Aggregate OD pairs they become non-directional*

Description

For example, sum total travel in both directions.

Usage

```
od_oneway(
  x,
  attrib = names(x[-c(1:2)])[vapply(x[-c(1:2)], is.numeric, TRUE)],
  FUN = sum,
  ...,
  id1 = names(x)[1],
  id2 = names(x)[2],
  oneway_key = NULL
)
```

Arguments

x	A data frame or SpatialLinesDataFrame, representing an OD matrix
attrib	A vector of column numbers or names, representing variables to be aggregated. By default, all numeric variables are selected.
FUN	The aggregating function such as sum (the default) and mean
...	Further arguments passed to or used by methods
id1	Optional (it is assumed to be the first column) text string referring to the name of the variable containing the unique id of the origin

id2	Optional (it is assumed to be the second column) text string referring to the name of the variable containing the unique id of the destination
oneway_key	Optional key of unique OD pairs regardless of the order, e.g., as generated by od_id_max_min() or od_id_szudzik()

Details

Flow data often contains movement in two directions: from point A to point B and then from B to A. This can be problematic for transport planning, because the magnitude of flow along a route can be masked by flows the other direction. If only the largest flow in either direction is captured in an analysis, for example, the true extent of travel will by heavily under-estimated for OD pairs which have similar amounts of travel in both directions. Flows in both direction are often represented by overlapping lines with identical geometries which can be confusing for users and are difficult to plot.

Value

oneway outputs a data frame (or sf data frame) with rows containing results for the user-selected attribute values that have been aggregated.

Examples

```
(od_min = od_data_df[c(1, 2, 1), 1:4])
od_min[3, 1:2] = rev(od_min[3, 1:2])
od_min[3, 3:4] = od_min[3, 3:4] - 5
(od_oneway = od_oneway(od_min))
nrow(od_oneway) < nrow(od_min) # result has fewer rows
sum(od_min$all) == sum(od_oneway$all) # but the same total flow
(od_oneway = od_oneway(od_min, FUN = mean))
od_oneway(od_min, attrib = "all")
od_min$all[3] = NA
(od_oneway = od_oneway(od_min, FUN = mean, na.rm = TRUE))
```

---

od_road_network *Simple road network dataset*

Description

Road network data from OSM for testing od_disaggregate.

Examples

```
library(sf)
nrow(od_road_network)
head(od_road_network)
plot(od_road_network$geometry)
plot(od_data_zones_min$geometry, lwd = 3, col = NULL, add = TRUE)
```

od_sample_vertices *Create a subsample of points from a route network for jittering*

Description

Todo: export this at some point

Usage

```
od_sample_vertices(x, fraction = 1)
```

Arguments

x	An sf object representing a road network
fraction	What percent of the network to sample?

Examples

```
## Not run:
u = "https://github.com/ITSLeeds/od/releases/download/v0.3.1/road_network_min.Rds"
f = basename(u)
if(!file.exists(f)) download.file(u, f)
road_network_min = readRDS(f)
od_sample_vertices(road_network_min)

## End(Not run)
```

od_to_network *Convert OD data into lines with start and end points sampled on a network*

Description

Convert OD data into lines with start and end points sampled on a network

Usage

```
od_to_network(
  x,
  z,
  zd = NULL,
  silent = TRUE,
  package = "sf",
  crs = 4326,
  network = NULL
)
```

Arguments

x	A data frame in which the first two columns are codes representing points/zones of origin and destination
z	Zones representing origins and destinations
zd	Zones representing destinations
silent	Hide messages? FALSE by default.
package	Which package to use to create the sf object? sfheaders is the default.
crs	The coordinate reference system of the output, if not known in z. 4326 by default.
network	An sf object representing a transport network

Examples

```
x = od_data_df
z = od_data_zones_min
network = od_data_network
(lines_to_points_on_network = od_to_network(x, z, network = network))
(lines_to_points = od_to_sf(x, z))
```

---

od_to_odmatrix *Convert origin-destination data from long to wide format*

Description

This function takes a data frame representing travel between origins (with origin codes in name_orig, typically the 1st column) and destinations (with destination codes in name_dest, typically the second column) and returns a matrix with cell values (from attrib, the third column by default) representing travel between origins and destinations.

Usage

```
od_to_odmatrix(x, attrib = 3, name_orig = 1, name_dest = 2)
```

Arguments

x	A data frame representing flows between origin and destinations
attrib	A number or character string representing the column containing the attribute data of interest from the flow data frame
name_orig	A number or character string representing the zone of origin
name_dest	A number or character string representing the zone of destination

See Also

Other od: od_id, odmatrix_to_od()

Examples

```
x = od_data_df[1:4, ]
x_matrix = od_to_odmatrix(x)
class(x_matrix)
od_to_odmatrix(x, attrib = "bicycle")
```

---

od_to_sf *Convert OD data into geographic 'desire line' objects*

Description

Convert OD data into geographic 'desire line' objects

Usage

```
od_to_sf(
  x,
  z,
  zd = NULL,
  odc = NULL,
  silent = FALSE,
  filter = TRUE,
  package = "sfheaders",
  crs = 4326
)

od_to_sfc(
  x,
  z,
  zd = NULL,
  silent = TRUE,
  package = "sfheaders",
  crs = 4326,
  filter = TRUE
)
```

Arguments

x	A data frame in which the first two columns are codes representing points/zones of origin and destination
z	Zones representing origins and destinations
zd	Zones representing destinations
odc	A matrix containing coordinates representing line start and end points
silent	Hide messages? FALSE by default.
filter	Remove rows with no matches in z? TRUE by default

package	Which package to use to create the sf object? sfheaders is the default.
crs	The coordinate reference system of the output, if not known in z. 4326 by default.

Examples

```
x = od_data_df
z = od_data_zones
desire_lines = od_to_sf(x, z)
desire_lines[1:3]
plot(desire_lines)
desire_lines_d = od_to_sf(od_data_df2, od_data_centroids2, od_data_destinations)
o1 = od_data_centroids2[od_data_centroids2[[1]] == od_data_df2[[1]][1], ]
d1 = od_data_destinations[od_data_destinations[[1]] == od_data_df2[[2]][1], ]
plot(desire_lines_d$geometry)
plot(od_data_centroids2$geometry, add = TRUE, col = "green")
plot(od_data_destinations$geometry, add = TRUE)
plot(o1, add = TRUE)
plot(d1, add = TRUE)
plot(desire_lines_d$geometry[1], lwd = 3, add = TRUE)
n = 7
on = od_data_centroids2[od_data_centroids2[[1]] == od_data_df2[[1]][n], ]
dn = od_data_destinations[od_data_destinations[[1]] == od_data_df2[[2]][n], ]
plot(desire_lines_d$geometry)
plot(on, add = TRUE)
plot(dn, add = TRUE)
plot(desire_lines_d$geometry[n], lwd = 3, add = TRUE)
```

---

points_to_od *Convert a series of points into a dataframe of origins and destinations*

Description

Takes a series of geographical points and converts them into a data.frame representing the potential flows, or 'spatial interaction', between every combination of points.

Usage

```
points_to_od(p, pd = NULL, interzone_only = FALSE, ids_only = FALSE)

points_to_odl(
  p,
  pd = NULL,
  interzone_only = FALSE,
  ids_only = FALSE,
  crs = 4326
)
```

Arguments

p	A spatial points object or a matrix of coordinates representing points
pd	Optional spatial points object or matrix objects representing destinations
interzone_only	Should the result only include interzonal OD pairs, in which the ID of the origin is different from the ID of the destination zone? FALSE by default
ids_only	Should a data frame with only 2 columns (origin and destination IDs) be returned? The default is FALSE, meaning the result should also contain the coordinates of the start and end points of each OD pair.
crs	The coordinate reference system of the output, if not known in z. 4326 by default.

Details

points_to_odl() generates the same output but returns a geographic object representing desire lines in the class sf.

Examples

```
library(sf)
p = od_data_centroids[1:3, ]
points_to_od(p)
points_to_od(p, ids_only = TRUE)
(l = points_to_odl(p, interzone_only = TRUE))
plot(l)
library(sf) # for subsetting sf objects:
points_to_od(od_data_centroids[1:2, ], od_data_centroids[3, ])
l = points_to_odl(od_data_centroids[1:2, ], od_data_centroids[3, ])
plot(l)
(od = points_to_od(p, interzone_only = TRUE))
l2 = od_to_sf(od, od_data_centroids)
l2$v = 1
(l2_oneway = od_oneway(l2))
plot(l2)
```

---

sfc_point_to_matrix *Extract coordinates from sfc objects with point geometry*

Description

This functions takes point geometries with class sfc from the sf package and returns a matrix representing x and y (typically lon/lat) coordinates.

Usage

```
sfc_point_to_matrix(x)
```

Arguments

x	An sfc object

Details

See https://github.com/dcooley/sfheaders/issues/52 for details

Author(s)

Dave Cooley

Examples

```
sfc_point_to_matrix(od_data_centroids$geometry[1:6])
```

Index