Package: neighbourhoods (via r-universe)

November 10, 2026

Title Efficient Identification of Neighbourhoods within Networks

Version 0.0.2.009

Description Algorithm for efficient identification of neighbourhoods within networks.

License GPL-3

URL https://github.com/ATFutures-labs/neighbourhoods

BugReports https://github.com/ATFutures-labs/neighbourhoods/issues

Depends R (>= 4.1.0)

Imports caret, cli, digest, dodgr, geodist, pbapply, raster, reproj, sf, stars

Suggests dplyr, testthat

LinkingTo cpp11

Remotes ATFutures/dodgr

Encoding UTF-8

LazyData true

Config/roxygen2/version 8.1.1

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev make libicu-dev libuv1-dev libxml2-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev

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

Date/Publication 2026-11-06 10:34:32 UTC

RemoteUrl https://github.com/UrbanAnalyst/neighbourhoods

RemoteRef HEAD

RemoteSha 8e08513a474c1bf2b94a9b48d0bffe16eba59ac1

Contents

adjacent_cycles 2
cut_nbs 2
hampi_sc 3

ltn_score 3
ltn_train 4
neighbourhoods 4
network_cycles 5
uncontract_cycles 5

Index **6**

`adjacent_cycles` *Construct adjacency matrix of neighbourhood cycles*

Description

Construct adjacency matrix of neighbourhood cycles

Usage

```
adjacent_cycles(cycles)
```

Arguments

`cycles`	List of cycles obtained from network_cycles.

Value

A 'data.frame' of three columns:

1. from - cycle from which connection is made
2. to - cycle to which connection is made
3. edges - List-column of all shared edges between (from, to) pair.

`cut_nbs` *Use result of neighbourhoods function to make and score an LTN*

Description

Use result of neighbourhoods function to make and score an LTN

Usage

```
cut_nbs(nbs, i, dmax = 10000)
```

Arguments

`nbs`	Results of neighbourhoods function.
`i`	Index of which neighbour pair is to be cut.
`dmax`	Maximal distance in metres around neighbourhood to use to generate centrality scores.

`hampi_sc` *hampi_sc*

Description

A sample street network from the primarily pedestrian township of Hampi, Karnataka, India.

Format

A **silicate** representation of the way network of Hampi.

`ltn_score` *Score an LTN formed by blocking one street segment between two adjacent neighbourhoods.*

Description

Score an LTN formed by blocking one street segment between two adjacent neighbourhoods.

Usage

```
ltn_score(nbs, index, dmax = 10000)
```

Arguments

`nbs`	Output of main neighbourhoods function.
`index`	Index into rows of ‘nbs$nbs‘ specifying which pairs of adjacent neighbourhoods are to be scored.
`dmax`	Maximal distance in metres around neighbourhood to use to generate scores.

Value

Modified version of ‘nbs$nbs‘ from input parameter, reduced to only those neighbour pairs specified in ‘index‘, and with additional column, ‘pop_decr_in‘ and ‘pop_incr_out‘ specifying absolute decreases within and increases surrounding proposed LTN.

ltn_train *Train a prediction model to score LTNs from a sample of size, 'n'.*

Description

Train a prediction model to score LTNs from a sample of size, 'n'.

Usage

```
ltn_train(nbs, n = 100, dmax = 10000)
```

Arguments

nbs	Output of main neighbourhoods function.
n	Size of sample to use in training and predicting data sets.
dmax	Maximal distance in metres around neighbourhood to use to generate scores.

Value

A trained model which can be used to predict additional LTN scores.

neighbourhoods *Find candidate low-traffic-neighbourhoods in an input street network.*

Description

Find candidate low-traffic-neighbourhoods in an input street network.

Usage

```
neighbourhoods(network, popdens)
```

Arguments

network	Street network in **silicate** 'sc' format, extracted with **dodgr** function, 'dodgr_streetnet_sc'.
popdens	Path to local population density file covering region of street network, and in 'geotiff' format.

Value

A 'data.frame' of candidate low-traffic neighbourhoods.

`network_cycles` *network_cycles*

Description

Get the minimal cycles of an undirected version of a **dodgr** street network in contracted and undirected form.

Usage

```
network_cycles(x)
```

Arguments

`x`	An **dodgr** street network processed with the ‘dodgr_contract_graph‘ and ‘merge_directed_graph‘ functions.

Value

A list of the minimal cycles of the street network, each of which has three columns of (‘.vx0‘, ‘.vx1‘, ‘.edge_‘).

`uncontract_cycles` *Convert cycles created on contracted graph back to equivalent uncontracted cycles.*

Description

Convert cycles created on contracted graph back to equivalent uncontracted cycles.

Usage

```
uncontract_cycles(paths, graph, graph_c)
```

Arguments

`paths`	List of cycle paths as a result of network_cycles.
`graph`	Full, non-contracted graph.
`graph_c`	Contracted graph resulting from call to ‘dodgr_contract_graph‘.

Value

Equivalent list of ‘paths‘, with each path expanded out to full edges in original, non-contracted graph.

Index

∗ **datasets**
hampi_sc, 3

adjacent_cycles, 2

cut_nbs, 2

hampi_sc, 3

ltn_score, 3
ltn_train, 4

neighbourhoods, *2, 3*, 4, *4*
network_cycles, *2*, 5, *5*

uncontract_cycles, 5