Package 'dodgr'

October 30, 2017

Title Distances on Directed Graphs

Version 0.0.3

Description Distances on dual-weighted directed graphs using priority-queue shortest paths. Weighted directed graphs have weights from A to B which may differ from those from B to A. Dual-weighted directed graphs have two sets of such weights. A canonical example is a street network to be used for routing in which routes are calculated by weighting distances according to the type of way and mode of transport, yet lengths of routes must be calculated from direct distances.

Depends R (>= 3.2.0)

License GPL-3

Imports igraph, magrittr, methods, osmdata, rbenchmark, sp, Rcpp (>= 0.12.6)

Suggests devtools, ggplot2, igraphdata, knitr, purrr, RColorBrewer, rmarkdown, roxygen2, scales, testthat

LinkingTo Rcpp

SystemRequirements C++11

VignetteBuilder knitr

NeedsCompilation yes

Encoding UTF-8

LazyData true

URL https://github.com/gmost/dodgr

BugReports https://github.com/gmost/dodgr/issues

RoxygenNote 6.0.1

Author Mark Padgham [aut, cre],
Andreas Peutschnig [aut],
Shane Saunders [cph] (Original author of included code for priority heaps)

Maintainer Mark Padgham <mark.padgham@email.com>

Repository CRAN

Date/Publication 2017-10-30 19:49:00 UTC

R topics documented:

compare_heaps . . . 2
dodgr . . . 3
dodgr_components . . . 4
dodgr_contract_graph . . . 4
dodgr_dists . . . 5
dodgr_flows . . . 6
dodgr_paths . . . 7
dodgr_sample . . . 9
dodgr_streetnet . . . 9
dodgr_vertices . . . 10
hampi . . . 11
match_pts_to_graph . . . 12
merge_directed_flows . . . 12
os_roads_bristol . . . 13
weighting_profiles . . . 14
weight_streetnet . . . 15

Index **16**

`compare_heaps` *compare_heaps*

Description

Perform timing comparison between different kinds of heaps as well as with equivalent `igraph` routine `distances`. To do this, a random sub-graph containing a defined number of vertices is first selected. Alternatively, this random sub-graph can be pre-generated with the `dodgr_sample` function and passed directly.

Usage

```
compare_heaps(graph, nverts = 100, replications = 2)
```

Arguments

`graph`	`data.frame` object representing the network graph (or a sub-sample selected with codedodgr_sample)
`nverts`	Number of vertices used to generate random sub-graph. If a non-numeric value is given, the whole graph will be used.
`replications`	Number of replications to be used in comparison

Value

Result of `rbenachmar::benchmark` comparison in `data.frame` form.

Note

igraph caches intermediate results of graph processing, so the **igraph** comparisons will be faster on subsequent runs. To obtain fair comparisons, run only once or re-start the current R session.

Examples

```
graph <- weight_streetnet (hampi)
compare_heaps (graph, nverts = 100, replications = 1)
```

---

dodgr *dodgr.*

Description

Distances on dual-weighted directed graphs using priority-queue shortest paths. Weighted directed graphs have weights from A to B which may differ from those from B to A. Dual-weighted directed graphs have two sets of such weights. A canonical example is a street network to be used for routing in which routes are calculated by weighting distances according to the type of way and mode of transport, yet lengths of routes must be calculated from direct distances.

The Main Function

- dodgr_dists: Calculate pair-wise distances between specified pairs of points in a graph.

Functions to Obtain Graphs

- dodgr_streetnet: Extract a street network in Simple Features (sf) form.
- weight_streetnet: Convert an sf-formatted street network to a dodgr graph through applying specified weights to all edges.

Functions to Modify Graphs

- dodgr_components: Number all graph edges according to their presence in distinct connected components.
- dodgr_contract_graph: Contract a graph by removing redundant edges.

Miscellaneous Functions

- dodgr_sample: Randomly sample a graph, returning a single connected component of a defined number of vertices.
- dodgr_vertices: Extract all vertices of a graph.
- compare_heaps: Compare the performance of different priority queue heap structures for a given type of graph.

dodgr_components *dodgr_components*

Description

Identify connected components of graph and add corresponding `component` column to `data.frame`.

Usage

```
dodgr_components(graph)
```

Arguments

`graph`	A `data.frame` of edges

Value

Equivalent graph with additional `component` column, sequentially numbered from 1 = largest component.

Examples

```
graph <- weight_streetnet (hampi)
graph <- dodgr_components (graph)
```

dodgr_contract_graph *dodgr_contract_graph*

Description

Removes redundant (straight-line) vertices from graph, leaving only junction vertices.

Usage

```
dodgr_contract_graph(graph, verts = NULL)
```

Arguments

`graph`	A flat table of graph edges. Must contain columns labelled `from` and `to`, or `start` and `stop`. May also contain similarly labelled columns of spatial coordinates (for example `from_x`) or `stop_lon`).
`verts`	Optional list of vertices to be retained as routing points. These must match the `from_id` and `to_id` columns of `graph`.

Value

A list of two items: graph containing contracted version of the original graph, converted to a standardised format, and edge_map, a two-column matrix mapping all newly contracted edges onto corresponding edges in original (uncontracted) graph.

Examples

```
graph <- weight_streetnet (hampi)
nrow (graph) # 5,742
graph <- dodgr_contract_graph (graph)
nrow (graph$graph) # 2,878
```

---

dodgr_dists *dodgr_dists*

Description

Calculate matrix of pair-wise distances between points.

Usage

```
dodgr_dists(graph, from, to, wt_profile = "bicycle", expand = 0,
  heap = "BHeap", quiet = TRUE)
```

Arguments

graph	data.frame or equivalent object representing the network graph (see Details)
from	Vector or matrix of points **from** which route distances are to be calculated (see Details)
to	Vector or matrix of points **to** which route distances are to be calculated (see Details)
wt_profile	Name of weighting profile for street networks (one of foot, horse, wheelchair, bicycle, moped, motorcycle, motorcar, goods, hgv, psv).
expand	Only when graph not given, the multiplicative factor by which to expand the street network surrounding the points defined by from and/or to.
heap	Type of heap to use in priority queue. Options include Fibonacci Heap (default; FHeap), Binary Heap (BHeap), Radix, Trinomial Heap (TriHeap), Extended Trinomial Heap (TriHeapExt, and 2-3 Heap (Heap23).
quiet	If FALSE, display progress messages on screen.

Value

square matrix of distances between nodes

Note

graph must minimally contain four columns of from, to, dist. If an additional column named weight or wt is present, shortest paths are calculated according to values specified in that column; otherwise according to dist values. Either way, final distances between from and to points are calculated according to values of dist. That is, paths between any pair of points will be calculated according to the minimal total sum of weight values (if present), while reported distances will be total sums of dist values.

The from and to columns of graph may be either single columns of numeric or character values specifying the numbers or names of graph vertices, or combinations to two columns specifying geographical (longitude and latitude) coordinates. In the latter case, almost any sensible combination of names will be accepted (for example, fromx, fromy, from_x, from_y, or fr_lat, fr_lon.)

from and to values can be either two-column matrices of equivalent of longitude and latitude coordinates, or else single columns precisely matching node numbers or names given in graph$from or graph$to. If to is missing, pairwise distances are calculated between all points specified in from. If neither from nor to are specified, pairwise distances are calculated between all nodes in graph.

Examples

```
# A simple graph
graph <- data.frame (from = c ("A", "B", "B", "B", "C", "C", "D", "D"),
                     to = c ("B", "A", "C", "D", "B", "D", "C", "A"),
                     d = c (1, 2, 1, 3, 2, 1, 2, 1))
dodgr_dists (graph)

# A larger example from the included \code{\link{hampi}} data.
graph <- weight_streetnet (hampi)
from <- sample (graph$from_id, size = 100)
to <- sample (graph$to_id, size = 50)
d <- dodgr_dists (graph, from = from, to = to)
# d is a 100-by-50 matrix of distances between \code{from} and \code{to}
```

---

dodgr_flows *dodgr_flows*

Description

Aggregate flows throughout a network based on an input matrix of flows between all pairs of from and to points.

Usage

```
dodgr_flows(graph, from, to, flows, wt_profile = "bicycle",
  contract = FALSE, heap = "BHeap", quiet = TRUE)
```

Arguments

graph	data.frame or equivalent object representing the network graph (see Details)
from	Vector or matrix of points **from** which route distances are to be calculated (see Details)
to	Vector or matrix of points **to** which route distances are to be calculated (see Details)
flows	Matrix of flows with nrow(flows)==length(from) and ncol(flows)==length(to).
wt_profile	Name of weighting profile for street networks (one of foot, horse, wheelchair, bicycle, moped, motorcycle, motorcar, goods, hgv, psv).
contract	If TRUE, calculate flows on contracted graph before mapping them back on to the original full graph (recommended as this will generally be much faster).
heap	Type of heap to use in priority queue. Options include Fibonacci Heap (default; FHeap), Binary Heap (BHeap), Radix, Trinomial Heap (TriHeap), Extended Trinomial Heap (TriHeapExt, and 2-3 Heap (Heap23).
quiet	If FALSE, display progress messages on screen.

Value

Modified version of graph with additonal flow column added.

Examples

```
graph <- weight_streetnet (hampi)
from <- sample (graph$from_id, size = 10)
to <- sample (graph$to_id, size = 5)
to <- to [!to %in% from]
flows <- matrix (10 * runif (length (from) * length (to)),
                 nrow = length (from))
graph <- dodgr_flows (graph, from = from, to = to, flows = flows)
# graph then has an additonal 'flows` column of aggregate flows along all
# edges. These flows are directed, and can be aggregated to equivalent
# undirected flows on an equivalent undirected graph with:
graph_undir <- merge_directed_flows (graph)
# This graph will only include those edges having non-zero flows, and so:
nrow (graph); nrow (graph_undir) # the latter is much smaller
```

---

dodgr_paths *dodgr_paths*

Description

Calculate lists of pair-wise shortest paths between points.

Usage

```
dodgr_paths(graph, from, to, vertices = TRUE, wt_profile = "bicycle",
  heap = "BHeap", quiet = TRUE)
```

Arguments

graph	data.frame or equivalent object representing the network graph (see Details)
from	Vector or matrix of points **from** which route distances are to be calculated (see Details)
to	Vector or matrix of points **to** which route distances are to be calculated (see Details)
vertices	If TRUE, return lists of lists of vertices for each path, otherwise return corresponding lists of edge numbers from graph.
wt_profile	Name of weighting profile for street networks (one of foot, horse, wheelchair, bicycle, moped, motorcycle, motorcar, goods, hgv, psv).
heap	Type of heap to use in priority queue. Options include Fibonacci Heap (default; FHeap), Binary Heap (BHeap), Radix, Trinomial Heap (TriHeap), Extended Trinomial Heap (TriHeapExt, and 2-3 Heap (Heap23).
quiet	If FALSE, display progress messages on screen.

Value

List of list of paths tracing all connections between nodes such that if `x <- dodgr_paths (graph, from, to)`, then the path between `from[i]` and `to[j]` is `x [[i]] [[j]]`.

Note

graph must minimally contain four columns of from, to, dist. If an additional column named weight or wt is present, shortest paths are calculated according to values specified in that column; otherwise according to dist values. Either way, final distances between from and to points are calculated according to values of dist. That is, paths between any pair of points will be calculated according to the minimal total sum of weight values (if present), while reported distances will be total sums of dist values.

The from and to columns of graph may be either single columns of numeric or character values specifying the numbers or names of graph vertices, or combinations to two columns specifying geographical (longitude and latitude) coordinates. In the latter case, almost any sensible combination of names will be accepted (for example, fromx, fromy, from_x, from_y, or fr_lat, fr_lon.)

from and to values can be either two-column matrices of equivalent of longitude and latitude coordinates, or else single columns precisely matching node numbers or names given in graph$from or graph$to. If to is missing, pairwise distances are calculated between all points specified in from. If neither from nor to are specified, pairwise distances are calculated between all nodes in graph.

Examples

```
graph <- weight_streetnet (hampi)
from <- sample (graph$from_id, size = 100)
to <- sample (graph$to_id, size = 50)
dp <- dodgr_paths (graph, from = from, to = to)
# dp is a list with 100 items, and each of those 100 items has 30 items, each
# of which is a single path listing all vertiex IDs as taken from \code{graph}.
```

dodgr_sample *dodgr_sample*

Description

Sample a random but connected sub-component of a graph

Usage

```
dodgr_sample(graph, nverts = 1000)
```

Arguments

graph	A flat table of graph edges. Must contain columns labelled from and to, or start and stop. May also contain similarly labelled columns of spatial coordinates (for example from_x) or stop_lon).
nverts	Number of vertices to sample

Value

A connected sub-component of graph

Note

Graphs may occassionally have nverts + 1 vertices, rather than the requested nverts.

Examples

```
graph <- weight_streetnet (hampi)
nrow (graph) # 5,742
graph <- dodgr_sample (graph, nverts = 200)
nrow (graph) # generally around 400 edges
nrow (dodgr_vertices (graph)) # 200
```

dodgr_streetnet *dodgr_streetnet*

Description

Use the osmdata package to extract the street network for a given location. For routing between a given set of points (passed as pts), the bbox argument may be omitted, in which case a bounding box will be constructed by expanding the range of pts by the relative amount of expand.

Usage

```
dodgr_streetnet(bbox, pts, expand = 0.05, quiet = TRUE)
```

Arguments

bbox	Bounding box as vector or matrix of coordinates, or location name. Passed to osmdata::getbb.
pts	List of points presumably containing spatial coordinates
expand	Relative factor by which street network should extend beyond limits defined by pts (only if bbox not given).
quiet	If FALSE, display progress messages

Value

A Simple Features (sf) object with coordinates of all lines in the street network.

Examples

```
## Not run:
streetnet <- dodgr_streetnet ("hampi india", expand = 0)
# convert to form needed for \code{dodgr} functions:
graph <- weight_streetnet (streetnet)
nrow (graph) # 5,742 edges
# Alternative ways of extracting street networks by using a small selection of
# graph vertices to define bounding box:
verts <- dodgr_vertices (graph)
verts <- verts [sample (nrow (verts), size = 200), ]
streetnet <- dodgr_streetnet (pts = verts, expand = 0)
graph <- weight_streetnet (streetnet)
nrow (graph)
# This will generally have many more rows because most street networks include
# streets that extend considerably beyond the specified bounding box.

## End(Not run)
```

---

dodgr_vertices *dodgr_vertices*

Description

Extract vertices of graph, including spatial coordinates if included

Usage

```
dodgr_vertices(graph)
```

Arguments

graph	A flat table of graph edges. Must contain columns labelled from and to, or start and stop. May also contain similarly labelled columns of spatial coordinates (for example from_x) or stop_lon).

Value

A `data.frame` of vertices with unique numbers (`n`).

Note

Values of `n` are 0-indexed

Examples

```
graph <- weight_streetnet (hampi)
v <- dodgr_vertices (graph)
```

`hampi` *hampi*

Description

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

Format

A Simple Features `sf` `data.frame` containing the street network of Hampi.

Note

Can be re-created with the following command, which also removes extraneous columns to reduce size:

Examples

```
## Not run:
hampi <- dodgr_streetnet("hampi india")
cols <- c ("osm_id", "highway", "oneway", "geometry")
hampi <- hampi [, which (names (hampi) %in% cols)]

## End(Not run)
# this 'sf data.frame' can be converted to a 'dodgr' network with
net <- weight_streetnet (hampi, wt_profile = 'foot')
```

match_pts_to_graph *match_pts_to_graph*

Description

Match spatial points to a spatial graph which contains vertex coordindates

Usage

```
match_pts_to_graph(verts, xy)
```

Arguments

verts	A data.frame of vertices obtained from dodgr_vertices(graph).
xy	coordinates of points to be matched to the vertices

Value

A vector index into verts

Examples

```
net <- weight_streetnet (hampi, wt_profile = "foot")
verts <- dodgr_vertices (net)
# Then generate some random points to match to graph
npts <- 10
xy <- data.frame (
                  x = min (verts$x) + runif (npts) * diff (range (verts$x)),
                  y = min (verts$y) + runif (npts) * diff (range (verts$y))
                  )
pts <- match_pts_to_graph (verts, xy)
pts # an index into verts
pts <- verts$id [pts]
pts # names of those vertices
```

merge_directed_flows *merge_directed_flows*

Description

The dodgr_flows function returns a column of aggregated flows directed along each edge of a graph, so the aggregated flow from vertex A to vertex B will not necessarily equal that from B to A, and the total flow in both directions will be the sum of flow from A to B plus that from B to A. This function converts a directed graph to undirected form through reducing all pairs of directed edges to a single edge, and aggregating flows from both directions.

Usage

```
merge_directed_flows(graph)
```

Arguments

graph	A graph containing a flow column as returned from dodgr_flows

Value

An equivalent graph in which all directed edges have been reduced to single, undirected edges, and all directed flows aggregated to undirected flows.

Examples

```
graph <- weight_streetnet (hampi)
from <- sample (graph$from_id, size = 10)
to <- sample (graph$to_id, size = 5)
to <- to [!to %in% from]
flows <- matrix (10 * runif (length (from) * length (to)),
                 nrow = length (from))
graph <- dodgr_flows (graph, from = from, to = to, flows = flows)
# graph then has an additonal 'flows` column of aggregate flows along all
# edges. These flows are directed, and can be aggregated to equivalent
# undirected flows on an equivalent undirected graph with:
graph_undir <- merge_directed_flows (graph)
# This graph will only include those edges having non-zero flows, and so:
nrow (graph); nrow (graph_undir) # the latter is much smaller
```

---

os_roads_bristol *os_roads_bristol*

Description

A sample street network for Bristol, U.K., from the Ordnance Survey.

Format

A Simple Features sf data.frame representing motorways in Bristol, UK.

Note

Input data downloaded from https://www.ordnancesurvey.co.uk/opendatadownload/products.html. To download the data from that page click on the tick box next to 'OS Open Roads', scroll to the bottom, click 'Continue' and complete the form on the subsequent page. This dataset is open access and can be used under the Open Government License and must be cited as follows: Contains OS data © Crown copyright and database right (2017)

Examples

```
## Not run:
library(sf)
library(dplyr)
os_roads <- sf::read_sf("~/data/ST_RoadLink.shp") # data must be unzipped here
u <- "https://opendata.arcgis.com/datasets/686603e943f948acaa13fb5d2b0f1275_4.kml"
lads <- sf::read_sf(u)
mapview::mapview(lads)
bristol_pol <- dplyr::filter(lads, grepl("Bristol", lad16nm))
os_roads <- st_transform(os_roads, st_crs(lads))
os_roads_bristol <- os_roads[bristol_pol, ] %>%
  dplyr::filter(class == "Motorway" & roadNumber != "M32") %>%
  st_zm(drop = TRUE)
mapview::mapview(os_roads_bristol)

## End(Not run)
# Converting this 'sf data.frame' to a 'dodgr' network requires manual
# specification of weighting profile:
colnm <- "formOfWay" # name of column used to determine weights
wts <- c (0.1, 0.2, 0.8, 1)
names (wts) <- unique (os_roads_bristol [[colnm]])
net <- weight_streetnet (os_roads_bristol, wt_profile = wts,
                         type_col = colnm, id_col = "identifier")
# 'id_col' tells the function which column to use to attribute IDs of ways
```

`weighting_profiles` *weighting_profiles*

Description

Collection of weighting profiles used to adjust the routing process to different means of transport. Original data taken from the Routino project.

Format

`data.frame` with profile names, means of transport and weights.

References

https://www.routino.org/xml/routino-profiles.xml

weight_streetnet *weight_streetnet*

Description

Weight (or re-weight) an sf-formatted OSM street network according to a named routino profile, selected from (foot, horse, wheelchair, bicycle, moped, motorcycle, motorcar, goods, hgv, psv).

Usage

```
weight_streetnet(sf_lines, wt_profile = "bicycle", type_col = "highway",
  id_col = "osm_id")
```

Arguments

sf_lines	A street network represented as sf LINESTRING objects, typically extracted with get_streetnet
wt_profile	Name of weighting profile, or vector of values with names corresponding to names in type_col
type_col	Specify column of the sf data.frame object which designates different types of highways to be used for weighting (default works with osmdata objects).
id_col	Specify column of the codesf data.frame object which provides unique identifiers for each highway (default works with osmdata objects).

Value

A data.frame of edges representing the street network, along with a column of graph component numbers.

Examples

```
# hampi is included with package as an 'osmdata' sf-formatted street network
net <- weight_streetnet (hampi)
class(net) # data.frame
dim(net) # 6096  11; 6096 streets
# os_roads_bristol is also included as an sf data.frame, but in a different
# format requiring identification of columns and specification of custom
# weighting scheme.
colnm <- "formOfWay"
wts <- c (0.1, 0.2, 0.8, 1)
names (wts) <- unique (os_roads_bristol [[colnm]])
net <- weight_streetnet (os_roads_bristol, wt_profile = wts,
                         type_col = colnm, id_col = "identifier")
dim (net) # 406 11; 406 streets
```

Index

∗Topic **datasets**
    hampi, 11
    os_roads_bristol, 13
    weighting_profiles, 14

compare_heaps, 2, *3*

dodgr, 3
dodgr-package *(*dodgr*)*, 3
dodgr_components, *3*, 4
dodgr_contract_graph, *3*, 4
dodgr_dists, *3*, 5
dodgr_flows, 6
dodgr_paths, 7
dodgr_sample, *3*, 9
dodgr_streetnet, *3*, 9
dodgr_vertices, *3*, 10

hampi, 11

match_pts_to_graph, 12
merge_directed_flows, 12

os_roads_bristol, 13

weight_streetnet, *3*, 15
weighting_profiles, 14