Package ‘edgebundle’

October 13, 2022

Title  Algorithms for Bundling Edges in Networks and Visualizing Flow and Metro Maps

Version  0.4.0


URL  http://edgebundle.schochastics.net/,
     https://github.com/schochastics/edgebundle

BugReports  https://github.com/schochastics/edgebundle/issues

License  MIT + file LICENSE

Suggests  testthat (>= 2.0.0), network, tidygraph

Config/testthat/edition  2

Encoding  UTF-8

LazyData  true

RoxygenNote  7.2.0

LinkingTo  Rcpp

Imports  Rcpp, igraph, reticulate, interp

Depends  R (>= 3.5)

NeedsCompilation  yes

Author  David Schoch [aut, cre] (<https://orcid.org/0000-0003-2952-4812>)

Maintainer  David Schoch <david@schochastics.net>

Repository  CRAN

Date/Publication  2022-07-05 00:10:10 UTC
Migration from California in 2010

Description
A dataset containing the number of people who migrated from California to other US states

Usage
cali2010

Format
igraph object

Source
https://www.census.gov/data/tables/time-series/demo/geographic-mobility/state-to-state-migration.html
**convert_edges**

Convert edges

**Description**

Converts edges of an igraph/network/tidygraph object into format useable for edge bundling.

**Usage**

```r
convert_edges(object, coords)
```

## Default S3 method:
```r
convert_edges(object, coords)
```

## S3 method for class 'igraph'
```r
convert_edges(object, coords)
```

## S3 method for class 'network'
```r
convert_edges(object, coords)
```

## S3 method for class 'tbl_graph'
```r
convert_edges(object, coords)
```

**Arguments**

- **object**
  - graph object

- **coords**
  - coordinates of vertices

**Value**

Data frame of edges with coordinates.

**Author(s)**

David Schoch

---

**edge_bundle_force**

force directed edge bundling

**Description**

Implements the classic edge bundling by Holten.
edge_bundle_force

Usage

edge_bundle_force(
  object,
  xy,
  K = 1,
  C = 6,
  P = 1,
  S = 0.04,
  P_rate = 2,
  I = 50,
  I_rate = 2/3,
  compatibility_threshold = 0.6,
  eps = 1e-08
)

Arguments

object a graph object (igraph/network tbl_graph)
xy coordinates of vertices
K spring constant
C number of iteration cycles
P number of initial edge divisions
S initial step size
P_rate rate of edge divisions
I number of initial iterations
I_rate rate of iteration decrease per cycle
compatibility_threshold threshold for when edges are considered compatible
eps accuracy

Details

This is a re-implementation of https://github.com/upphiminn/d3.ForceBundle. Force directed edge bundling is slow (O(E^2)).

see online for plotting tips

Value

data.frame containing the bundled edges

Author(s)

David Schoch
edge_bundle_hammer

References


See Also

edge_bundle_hammer, edge_bundle_stub, edge_bundle_path

Examples

library(igraph)
g <- graph_from_edgelist(matrix(c(1,12,2,11,3,10,4,9,5,8,6,7), ncol = 2, byrow = TRUE), FALSE)
xy <- cbind(c(rep(0,6), rep(1,6)), c(1:6,1:6))
edge_bundle_force(g, xy)

edge_bundle_hammer  hammer edge bundling

Description

Implements the hammer edge bundling by Ian Calvert.

Usage

data.frame containing the bundled edges

edge_bundle_hammer(object, xy, bw = 0.05, decay = 0.7)

Arguments

object  a graph object (igraph/network/tbl_graph)
xy      coordinates of vertices
bw      bandwidth parameter
decay    decay parameter

Details

This function only wraps existing python code from the datashader library. Original code can be found at https://gitlab.com/ianjcalvert/edgehammer. Datashader is a huge library with a lot of dependencies, so think twice if you want to install it just for edge bundling. Check https://datashader.org/user_guide/Networks.html for help concerning parameters bw and decay. To install all dependencies, use install_bundle_py.

see online for plotting tips

Value

data.frame containing the bundled edges
edge_bundle_path

**Author(s)**
David Schoch

**See Also**
edge_bundle_force, edge_bundle_stub, edge_bundle_path

---

**edge_bundle_path**  
*Edge-Path Bundling*

**Description**
Implements edge-path bundling.

**Usage**
```
edge_bundle_path(g, xy, max_distortion = 2, weight_fac = 2, segments = 20)
```

**Arguments**
- `g` an igraph object
- `xy` coordinates of vertices
- `max_distortion` maximum distortion
- `weight_fac` edge weight factor
- `segments` number of subdivisions of edges

**Details**
This is a re-implementation of https://github.com/mwallinger-tu/edge-path-bundling
see online for plotting tips

**Value**
data.frame containing the bundled edges

**Author(s)**
David Schoch

**References**
edge_bundle_stub

See Also
edge_bundle_hammer, edge_bundle_stub, edge_bundle_force

Examples

library(igraph)
g <- graph_from_edgelist(matrix(c(1,2,1,6,1,4,2,3,3,4,4,5,5,6),ncol = 2,byrow = TRUE),FALSE)
xy <- cbind(c(0,10,25,40,50,50),c(0,15,25,15,0,-10))
edge_bundle_path(g,xy)

description stub edge bundling

Description

Implements the stub edge bundling by Nocaj and Brandes

Usage

edge_bundle_stub(
  object,
  xy,
  alpha = 11,
  beta = 75,
  gamma = 40,
  t = 0.5,
  tshift = 0.5
)

Arguments

  object  a graph object (igraph/tbl_graph). Does not support network objects
  xy      coordinates of vertices
  alpha   maximal angle (in degree) between consecutive edges in a bundle
  beta    angle (in degree) at which to connect two stubs
  gamma   maximal overall angle (in degree) of an edge bundle
  t       numeric between 0 and 1. control point location
  tshift  numeric between 0 and 1. The closer to one, the longer the bigger bundle

Details

  see online for plotting tips

Value

  data.frame containing the bundled edges
install_bundle_py

**install python dependencies for hammer bundling**

---

**Description**

install datashader and scikit-image

**Usage**

install_bundle_py(method = "auto", conda = "auto")
**metro_berlin**

**Arguments**

- **method**: Installation method (by default, "auto" automatically finds a method that will work in the local environment, but note that the "virtualenv" method is not available on Windows).
- **conda**: Path to conda executable (or "auto" to find conda using the PATH and other conventional install locations).

---

**Description**

A dataset containing the subway network of Berlin

**Usage**

```r
metro_berlin
```

**Format**

igraph object

**References**


**metro_multicriteria**

**Metro Map Layout**

**Description**

Metro map layout based on multicriteria optimization

**Usage**

```r
metro_multicriteria(object, xy, l = 2, gr = 0.0025, w = rep(1, 5), bsize = 5)
```

**Arguments**

- **object**: original graph
- **xy**: initial layout of the original graph
- **l**: desired multiple of grid point spacing. \(l*gr\) determines desired edge length
- **gr**: grid spacing. \(l*gr\) determines desired edge length
- **w**: weight vector for criteria (see details)
- **bsize**: number of grid points a station can move away from its original position
Details

The function optimizes the following five criteria using a hill climbing algorithm:

- **Angular Resolution Criterion**: The angles of incident edges at each station should be maximized, because if there is only a small angle between any two adjacent edges, then it can become difficult to distinguish between them.

- **Edge Length Criterion**: The edge lengths across the whole map should be approximately equal to ensure regular spacing between stations. It is based on the preferred multiple, $l$, of the grid spacing, $g$. The purpose of the criterion is to penalize edges that are longer than or shorter than $lg$.

- **Balanced Edge Length Criterion**: The length of edges incident to a particular station should be similar.

- **Line Straightness Criterion**: (not yet implemented) Edges that form part of a line should, where possible, be co-linear either side of each station that the line passes through.

- **Octilinearity Criterion**: Each edge should be drawn horizontally, vertically, or diagonally at 45 degree, so we penalize edges that are not at a desired angle [see online](#) for more plotting tips.

Value

new coordinates for stations

Author(s)

David Schoch

References


Examples

```r
# the algorithm has problems with parallel edges
library(igraph)
g <- simplify(metro_berlin)
xy <- cbind(V(g)$lon,V(g)$lat)*100

# the algorithm is not very stable, try playing with the parameters
xy_new <- metro_multicriteria(g,xy,l = 2, gr = 0.5, w = c(100,100,1,1,100), bsize = 35)
```
Sample points for triangulated networks

tnss_dummies

Description
uses various sampling strategies to create dummy nodes for the tnss_tree

Usage

```r
tnss_dummies(
  xy,
  root,
  circ = TRUE,
  line = TRUE,
  diag = TRUE,
  grid = FALSE,
  rand = FALSE,
  ncirc = 9,
  rcirc = 2,
  nline = 10,
  ndiag = 50,
  ngrid = 50,
  nrand = 50
)
```

Arguments

- `xy`: coordinates of "real" nodes
- `root`: root node id
- `circ`: logical. create circular dummy nodes around leafs.
- `line`: logical. create dummy nodes on a straight line between root and leafs.
- `diag`: logical. create dummy nodes diagonally through space.
- `grid`: logical. create dummy nodes on a grid.
- `rand`: logical. create random dummy nodes.
- `ncirc`: numeric. number of circular dummy nodes per leaf.
- `rcirc`: numeric. radius of circles around leaf nodes.
- `nline`: numeric. number of straight line nodes per leaf.
- `ndiag`: numeric. number of dummy nodes on diagonals.
- `ngrid`: numeric. number of dummy nodes per dim on grid.
- `nrand`: numeric. number of random nodes to create.

Value

coordinates of dummy nodes
tnss_smooth

Author(s)
David Schoch

Examples
# dummy nodes for tree rooted in California
xy <- cbind(state.center$x, state.center$y)
xy_dummy <- tnss_dummies(xy, 4)

---

**tnss_smooth**  
**Smooth a Steiner tree**

Description
Converts the Steiner tree to smooth paths

Usage
tnss_smooth(g, bw = 3, n = 10)

Arguments
- `g`: Steiner tree computed with `tnss_tree`
- `bw`: bandwidth of Gaussian Kernel
- `n`: number of extra nodes to include per edge

Details
see see online for tips on plotting the result

Value
data.frame containing the smoothed paths

Author(s)
David Schoch

Examples
xy <- cbind(state.center$x, state.center$y)[!state.name%in%c("Alaska","Hawaii"),]
xy_dummy <- tnss_dummies(xy, root = 4)
gtree <- tnss_tree(cali2010, xy, xy_dummy, root = 4, gamma = 0.9)
tree_smooth <- tnss_smooth(gtree, bw = 10, n = 10)
tnss_tree  

Create Steiner tree from real and dummy points

Description

creates an approximated Steiner tree for a flow map visualization

Usage

tnss_tree(
  g,
  xy,
  xydummy,
  root,
  gamma = 0.9,
  epsilon = 0.3,
  elen = Inf,
  order = "random"
)

Arguments

g     original flow network (must be a one-to-many flow network, i.e star graph). Must have a weight attribute indicating the flow
xy    coordinates of "real" nodes
xydummy coordinates of "dummy" nodes
root   root node id of the flow
gamma edge length decay parameter
epsilon percentage of points kept on a line after straightening with Visvalingam Algorithm
elen   maximal length of edges in triangulation
order  in which order shortest paths are calculated ("random","weight","near","far")

Details

Use tnss_smooth to smooth the edges of the tree

Value

approximated Steiner tree from dummy and real nodes as igraph object

Author(s)

David Schoch
References


Examples

```r
xy <- cbind(state.center$x, state.center$y)[!state.name%in%c("Alaska", "Hawaii")]
xy_dummy <- tnss_dummies(xy, root = 4)
gtree <- tnss_tree(cali2010, xy, xy_dummy, root = 4, gamma = 0.9)
```

---

### us_flights

**Flights within the US**

**Description**

A dataset containing flights between US airports as igraph object

**Usage**

```r
us_flights
```

**Format**

igraph object

**Source**

[https://gist.github.com/mbostock/7608400/raw](https://gist.github.com/mbostock/7608400/raw)

---

### us_migration

**Migration within the US 2010-2019**

**Description**

A dataset containing the number of people migrating between US states from 2010-2019

**Usage**

```r
us_migration
```

**Format**

data.frame

**Source**

[https://www.census.gov/data/tables/time-series/demo/geographic-mobility/state-to-state-migration.html](https://www.census.gov/data/tables/time-series/demo/geographic-mobility/state-to-state-migration.html)
Index

* datasets
  - cali2010, 2
  - metro_berlin, 9
  - us_flights, 14
  - us_migration, 14

cali2010, 2
convert_edges, 3

datasets
  - edge_bundle_force, 3, 6–8
  - edge_bundle_hammer, 5, 5, 7, 8
  - edge_bundle_path, 5, 6, 6, 8
  - edge_bundle_stub, 5–7, 7

install_bundle_py, 5, 8

metro_berlin, 9
metro_multicriteria, 9

tnss_dummies, 11
tnss_smooth, 12, 13
tnss_tree, 11, 12, 13

us_flights, 14
us_migration, 14