Package ‘cholera’

March 1, 2023

Type Package

Title Amend, Augment and Aid Analysis of John Snow's Cholera Map

Version 0.8.0

Date 2023-03-01

Description Amends errors, augments data and aids analysis of John Snow's map of the 1854 London cholera outbreak.

URL https://github.com/lindbrook/cholera

BugReports https://github.com/lindbrook/cholera/issues

License GPL (>= 2)

LazyData true

Depends R (>= 3.4)

Imports deldir (>= 1.0-2), elevatr, geosphere, ggplot2, grDevices, HistData (>= 0.7-8), igraph, KernSmooth, pracma, RColorBrewer, sp, stats, tanaka, terra, tools, threejs, TSP, utils, viridisLite

Suggests knitr, rmarkdown

VignetteBuilder knitr

RoxygenNote 7.2.3

Encoding UTF-8

Language en-US

NeedsCompilation no

Author Peter Li [aut, cre]

Maintainer Peter Li <lindbrook@gmail.com>

Repository CRAN

Date/Publication 2023-03-01 12:40:02 UTC
**R topics documented:**

- cholera-package ........................................ 4
- addCase .................................................. 5
- addDelaunay .............................................. 6
- addEuclideanPath ....................................... 7
- addFrame .................................................. 8
- addIndexCase ............................................ 8
- addKernelDensity ....................................... 9
- addLandmarks .......................................... 10
- addMilePosts .......................................... 11
- addNeighborhoodCases ................................. 12
- addNeighborhoodEuclidean ......................... 13
- addNeighborhoodWalking ............................ 14
- addPlaguePit ........................................... 15
- addPump ................................................. 16
- addRoads .................................................. 16
- addSnow .................................................. 17
- addVoronoi .............................................. 17
- addWalkingPath ....................................... 18
- addWhitehead .......................................... 19
- anchor.case ............................................ 20
- border .................................................... 21
- caseDistance ........................................... 21
- caseLocator ............................................ 22
- euclideanPath ......................................... 23
- fatalities ............................................... 24
- fatalities.address ................................... 25
- fatalities.unstacked ................................ 26
- fixFatalities .......................................... 26
- frame.data .............................................. 27
- frame.sample .......................................... 27
- isoLines ............................................... 28
- landmark.squares .................................... 28
- landmarkData .......................................... 29
- landmarks .............................................. 29
- latlong.ortho.addr ................................... 30
- latlong.ortho.pump ................................... 30
- latlong.ortho.pump.vestry ......................... 31
- latlongAddress ........................................ 31
- latlongFatalities .................................... 32
- latlongLandmarks ..................................... 33
- latlongNearestPump .................................. 33
- latlongNeighborhoodData ......................... 34
- latlongNeighborhoodVoronoi ....................... 35
- latlongNeighborhoodWalking ...................... 35
- latlongPumps .......................................... 36
- latlongRoads .......................................... 36
R topics documented:

- latlongVoronoi .................................................. 37
- latlongWalkingPath .......................................... 37
- mapRange ......................................................... 38
- nearestPump .................................................... 38
- neighborhoodData ............................................. 39
- neighborhoodEuclidean ...................................... 40
- neighborhoodVoronoi ........................................ 41
- neighborhoodWalking ........................................ 42
- ortho.proj ....................................................... 43
- ortho.proj.pump ................................................. 44
- ortho.proj.pump.vestry ....................................... 45
- oxford.weather ................................................ 45
- oxfordWeather ................................................ 46
- plague.pit ....................................................... 47
- plot.euclidean ................................................. 47
- plot.euclidean_path ......................................... 48
- plot.latlongNeighborhoodVoronoi ....................... 49
- plot.latlong_neighborhood_data ......................... 49
- plot.latlong_walking ........................................ 50
- plot.latlong_walking_path .................................. 50
- plot.neighborhood_data ..................................... 51
- plot.oxfordWeather .......................................... 51
- plot.povertyLondon ......................................... 52
- plot.profile_perspective ..................................... 52
- plot.time_series .............................................. 53
- plot.voronoi ................................................... 54
- plot.walking .................................................... 55
- plot.walking_path ............................................. 56
- plot.winterTemperatures .................................... 57
- povertyLondon ................................................ 57
- print.euclidean ............................................... 58
- print.euclidean_path ....................................... 58
- print.iso ......................................................... 59
- print.latlongNeighborhoodVoronoi ..................... 59
- print.latlong_walking_path ................................ 60
- print.time_series ............................................ 60
- print.voronoi .................................................. 61
- print.walking .................................................. 61
- print.walking_path .......................................... 62
- profile2D ....................................................... 63
- profile3D ....................................................... 63
- pumpCase ....................................................... 64
- pumpData ....................................................... 65
- pumpFatalities ............................................... 66
- pumpLocator ................................................... 66
- pumps .......................................................... 67
- pumps.vestry ................................................. 68
- rd.sample ....................................................... 69
cholera-package

Description

Amend, augment and aid the analysis of John Snow's cholera map.
addCase

Details

Features:

• Fixes three apparent coding errors in Dodson and Tobler’s 1992 digitization of Snow’s map.
• "Unstacks" the data in two ways to make analysis and visualization easier and more meaningful.
• Computes and visualizes "pump neighborhoods" based on Voronoi tessellation, Euclidean distance, and walking distance.
• Ability to overlay graphical elements and features like kernel density, Voronoi diagrams, Snow’s Broad Street neighborhood, and notable landmarks (John Snow’s residence, the Lion Brewery, etc.) via add*() functions.
• Includes a variety of functions to highlight specific cases, roads, pumps and paths.
• Appends actual street names to roads data.
• Includes the revised pump data used in the second version of Snow’s map from the Vestry report, which includes the "correct" location of the Broad Street pump.
• Adds two different aggregate time series fatalities data sets, taken from the Vestry report.
• Support for parallel computation on Linux, macOS and Windows.
• With 'cholera' version >= 0.8.0, preliminary and provisional support for georeferenced (longitude and latitude) versions of data and functions.

To learn more, see the vignettes:
- vignette("duplicate.missing.cases")
- vignette("kernel.density")
- vignette("parallelization")
- vignette("pump.neighborhoods")
- vignette("roads")
- vignette("tiles.polygons")
- vignette("time.series")
- vignette("unstacking.bars")

---

addCase

Add observed case(s) to plot.

Description

Add case(s), as "address" or "fatalities" as points or IDs, to a plot.

Usage

addCase(case = 1, type = "observed", token = "both", text.size = 0.5, pch = 1, cex = 1, point.lwd = 2, col = "black", pos = 1)
Arguments

- **case** (Numeric or Character) Vector of case ID(s). "all" plots all cases. "anchor" plots anchor cases.
- **type** (Character) Type of case: "observed" or "expected".
- **token** (Character) Type of token to plot: "point", "id" or "both".
- **text.size** (Numeric) Size of case ID text.
- **pch** (Numeric) pch.
- **cex** (Numeric) cex.
- **point.lwd** (Numeric) Point lwd.
- **col** (Character) Color.
- **pos** (Numeric) Text position.

Note

type, token, text.size, pch, cex, point.lwd and pos relevant only when case is numeric.

Examples

```r
snowMap(add.cases = FALSE)
addCase(1)

snowMap(add.cases = FALSE)
addCase(100)
```

---

**addDelaunay** 

*Add Delaunay triangles.*

**Description**

Add Delaunay triangles.

**Usage**

```r
addDelaunay(pump.select = NULL, vestry = FALSE, color = "black",
            line.type = "solid")
```

**Arguments**

- **pump.select** (Numeric) Default is NULL; all pumps are used. Otherwise, selection by a vector of numeric IDs: 1 to 13 for pumps; 1 to 14 for pumps.vestry. Exclusion (negative selection) is possible (e.g., -6).
- **vestry** (Logical) FALSE for original 13 pumps. TRUE for 14 pumps in Vestry Report.
- **color** (Character) Color of triangle edges.
- **line.type** (Character) Type of line for triangle edges.
addEuclideanPath

Note

This function uses deldir::deldir().

Examples

```r
snowMap()
addDelaunay()
```

```r
collapse = TRUE

data.euclid <- addEuclideanPath(1, 2, type = "case-pump", observed = TRUE,
                                 case.location = "address")

data.euclid <- addEuclideanPath(1, 2, type = "case-pump", observed = FALSE,
                                 case.location = "nominal")
```

Description

Add the path for the Euclidean distance between cases and/or pumps.

Usage

```r
addEuclideanPath(origin, destination = NULL, type = "case-pump",
                  observed = TRUE, case.location = "address", vestry = FALSE,
                  distance.unit = "meter", time.unit = "second", walking.speed = 5,
                  unit.posts = "distance", unit.interval = NULL, alpha.level = 1)
```

Arguments

- `origin`: Numeric or Integer. Numeric ID of case or pump.
- `destination`: Numeric or Integer. Numeric ID(s) of case(s) or pump(s). Exclusion is possible via negative selection (e.g., -7). Default is `NULL`: this returns closest pump or "anchor" case.
- `type`: Character "case-pump", "cases" or "pumps".
- `observed`: Logical. Use observed or simulated expected data.
- `case.location`: Character. For observed = FALSE: "address" or "nominal". "address" is the x-y coordinate of a stack’s "anchor" case. "nominal" is the x-y coordinate of a bar.
- `vestry`: Logical. TRUE uses the 14 pumps from the Vestry Report. FALSE uses the 13 pumps from the original map.
- `distance.unit`: Character. Unit of distance: "meter", "yard" or "native". "native" returns the map’s native scale. See vignette("roads") for information on unit distances.
- `time.unit`: Character. "hour", "minute", or "second".
- `walking.speed`: Numeric. Walking speed in km/hr.
- `unit.posts`: Character. "distance" for mileposts; "time" for timeposts; NULL for no posts.
- `alpha.level`: Numeric. Alpha level transparency for path: a value in [0, 1].
Value
An R list with 3 data frames: x-y coordinates for the origin and destination, and a summary of results.

Note
Walking time is computed using `distanceTime()`.

addFrame

Description
Add map border to plot.

Usage
`addFrame(latlong = FALSE, col = "black", ...)`

Arguments
- `latlong` Logical. Use estimated longitude and latitude.
- `col` Character. Color
- `...` Additional plotting parameters.

addIndexCase

Description
Highlight index case at 40 Broad Street.

Usage
`addIndexCase(cex = 2, col = "red", pch = 1, add.label = FALSE, text.size = 0.5)`

Arguments
- `cex` Numeric. Size of point.
- `col` Character. Color of point.
- `pch` Numeric. Type of of point.
- `add.label` Logical. Add text annotation: "40 Broad Street"
- `text.size` Numeric. Size of text label.
addKernelDensity

Value
Add base R point and (optionally) text to a graphics plot.

Examples
segmentLocator("216-1")
addIndexCase()

addKernelDensity  Add 2D kernel density contours.

Description
Add 2D kernel density contours based on selected sets of observations.

Usage
addKernelDensity(pump.subset = "pooled", pump.select = NULL,
neighborhood.type = "walking", data = "unstacked", bandwidth = 0.5,
color = "black", line.type = "solid", multi.core = TRUE)

Arguments
pump.subset Character or Numeric: "pooled", "individual", or numeric vector. "pooled" treats all observations as a single set. "individual" is a shortcut for all individual pump neighborhoods. Use of vector of numeric pump IDs to subset from the neighborhoods defined by pump.select. Negative selection possible. NULL selects all pumps in pump.select.
pump.select Numeric. Vector of numeric pump IDs to define pump neighborhoods (i.e., the "population"). Negative selection possible. NULL selects all pumps.
neighborhood.type Character. "voronoi" or "walking"
data Character. Unit of observation: "unstacked" uses fatalities.unstacked; "address" uses fatalities.address; "fatality" uses fatalities.
bandwidth Numeric. Bandwidth for kernel density estimation.
color Character. Color of contour lines.
line.type Character. Line type for contour lines.
multi.core Logical or Numeric. TRUE uses parallel::detectCores(). FALSE uses one, single core. You can also specify the number logical cores. See vignette("Parallelization") for details.

Value
Add contours to a graphics plot.
addLandmarks

Add landmarks to plot.

Description

Add landmarks to plot.

Usage

addLandmarks(text.size = 0.5, highlight.perimeter = TRUE)

Arguments

text.size Numeric. cex for text labels.
highlight.perimeter Logical. Highlight Lion Brewery and Model Housing.

Value

Base R points and text.

Note

The location of 18 Sackville Street and 28 Dean Street are approximate. Falconberg Court & Mews form an isolate: they are not part of the network of roads and are technically unreachable. Adam and Eve Court and its pump also form an isolate.
addMilePosts

Examples

snowMap(add.landmarks = FALSE)
addLandmarks()

Description

Add distance or time based "mileposts" to an observed walking neighborhood plot.

Usage

addMilePosts(pump.subset = NULL, pump.select = NULL, vestry = FALSE, unit = "distance", interval = NULL, walking.speed = 5, type = "arrows", multi.core = TRUE, dev.mode = FALSE)

Arguments

pump.subset Numeric. Vector of numeric pump IDs to subset from the neighborhoods defined by pump.select. Negative selection possible. NULL uses all pumps in pump.select.
pump.select Numeric. Numeric vector of pumps to define possible pump neighborhoods (i.e. the "population"). Negative selection is possible. NULL selects all "observed" pumps (i.e., pumps with at least one case).
vestry Logical. TRUE uses the 14 pumps from the Vestry Report. FALSE uses the 13 from the original map.
unit Character. Milepost unit of measurement: "distance" or "time".
interval Numeric. Interval between mileposts: 50 meters for "distance": 60 seconds for "time".
walking.speed Numeric. Walking speed in km/hr.
type Character. "arrows" or "points".
multi.core Logical or Numeric. TRUE uses parallel::detectCores(). FALSE uses one, single core. You can also specify the number logical cores. See vignette("Parallelization") for details.
dev.mode Logical. Development mode uses parallel::parLapply().

Value

R base graphics arrows or points.
addNeighborhoodCases

Add observed cases by neighborhood.

Description
Add cases to a plot as "address" or "fatalities" and as points or IDs.

Usage
addNeighborhoodCases(pump.subset = NULL, pump.select = NULL, 
metric = "walking", type = "stack.base", token = "point", 
text.size = 0.5, pch = 16, point.size = 0.5, vestry = FALSE, 
weighted = TRUE, color = NULL, case.location = "nominal", 
alpha.level = 0.5, multi.core = TRUE)

Arguments
pump.subset Numeric. Vector of numeric pump IDs to subset from the neighborhoods defined by pump.select. Negative selection possible. NULL uses all pumps in pump.select.
pump.select Numeric. Numeric vector of pump IDs that define which pump neighborhoods to consider (i.e., specify the "population"). Negative selection possible. NULL selects all pumps.
metric Character. Type of neighborhood: "euclidean" or "walking".
type Character. Type of case: "stack.base" (base of stack), or "stack" (entire stack). For observed = TRUE.
token Character. Type of token to plot: "point" or "id".
text.size Numeric. Size of case ID text.
pch Numeric.
point.size Numeric.
vestry Logical. TRUE uses the 14 pumps from the Vestry Report. FALSE uses the 13 in the original map.
weighted Logical. TRUE computes shortest walking path weighted by road length. FALSE computes shortest walking path in terms of the number of nodes.
color Character. Use a single color for all paths. NULL uses neighborhood colors defined by snowColors().
case.location Character. For metric = "euclidean": "address" uses ortho.proj; "nominal" uses fatalities.
alpha.level Numeric. Alpha level transparency for area plot: a value in [0, 1].
multi.core Logical or Numeric. TRUE uses parallel::detectCores(). FALSE uses one, single core. You can also specify the number logical cores. See vignette("Parallelization") for details.
### Examples

```r
## Not run:
snowMap(add.cases = FALSE)
addNeighborhoodCases(pump.subset = c(6, 10))

snowMap(add.cases = FALSE)
addNeighborhoodCases(pump.select = c(6, 10))

## End(Not run)
```

### Description

Add expected Euclidean pump neighborhoods.

### Usage

```r
addNeighborhoodEuclidean(pump.subset = NULL, pump.select = NULL,
vestry = FALSE, case.location = "nominal", type = "star",
alpha.level = 0.5, multi.core = TRUE, dev.mode = FALSE)
```

### Arguments

- **pump.subset**: Numeric. Vector of numeric pump IDs to subset from the neighborhoods defined by `pump.select`. Negative selection possible. `NULL` selects all pumps in `pump.select`.
- **pump.select**: Numeric. Vector of numeric pump IDs to define pump neighborhoods (i.e., the "population"). Negative selection possible. `NULL` selects all pumps.
- **vestry**: Logical. `TRUE` uses the 14 pumps from the Vestry Report. `FALSE` uses the 13 in the original map.
- **case.location**: Character. "address" or "nominal". "address" is the x-y coordinates of `sim.ortho.proj`. "nominal" is the x-y coordinates of `regular.cases`.
- **type**: Character. Type of plot: "star", "area.points" or "area.polygons".
- **alpha.level**: Numeric. Alpha level transparency for area plot: a value in [0, 1].
- **multi.core**: Logical or Numeric. `TRUE` uses `parallel::detectCores()`. `FALSE` uses one, single core. You can also specify the number logical cores. See vignette(“Parallelization”) for details.
- **dev.mode**: Logical. Development mode uses `parallel::parLapply()`.

### Value

R graphic elements.
addNeighborhoodWalking

Add expected walking neighborhoods.

Description

Add expected walking neighborhoods.

Usage

addNeighborhoodWalking(pump.subset = NULL, pump.select = NULL, vestry = FALSE, weighted = TRUE, path = NULL, path.color = NULL, path.width = 3, alpha.level = 0.25, polygon.type = "solid", polygon.col = NULL, polygon.lwd = 2, multi.core = TRUE, dev.mode = FALSE, latlong = FALSE)

Arguments

pump.subset Numeric. Vector of numeric pump IDs to subset from the neighborhoods defined by pump.select. Negative selection possible. NULL uses all pumps in pump.select.
pump.select Numeric. Numeric vector of pump IDs that define which pump neighborhoods to consider (i.e., specify the "population"). Negative selection possible. NULL selects all pumps.
vestry Logical. TRUE uses the 14 pumps from the Vestry Report. FALSE uses the 13 in the original map.
weighted Logical. TRUE computes shortest path weighted by road length. FALSE computes shortest path in terms of the number of nodes.
path Character. "expected" or "observed".
path.color Character. Use a single color for all paths. NULL uses neighborhood colors defined by snowColors().
path.width Numeric. Set width of paths.
alpha.level Numeric. Alpha level transparency for area plot: a value in [0, 1].
addPlaguePit

polygon.type  Character. "perimeter" or "solid".
polygon.col   Character.
polygon.lwd   Numeric.
multi.core    Logical or Numeric. TRUE uses parallel::detectCores(). FALSE uses one, single core. You can also specify the number logical cores. See vignette("Parallelization") for details.
dev.mode      Logical. Development mode uses parallel::parLapply().
latlong       Logical. Use estimated longitude and latitude.

Examples

## Not run:
streetNameLocator("marshall street", zoom = 0.5)
addNeighborhoodWalking()

## End(Not run)

---

addPlaguePit  Add plague pit (Marshall Street).

Description

Draws a polygon that approximates the plague pit located around Marshall Street. From Vestry Report map.

Usage

addPlaguePit(color = "black", line.type = "solid")

Arguments

color       Character. Color of polygon.
line.type   Character. Polygon line type.

Value

Adds a polygon to a graphics plot.

Note

In progress.

Examples

snowMap(add.landmarks = FALSE)
addPlaguePit()
addPump

Add selected pump(s) to plot.

Description
Add selected pump(s) to plot.

Usage
addPump(pump.select = NULL, vestry = FALSE, col = NULL, pch = 24,
label = TRUE, pos = 1, cex = 1, latlong = FALSE)

Arguments
- pump.select: Numeric or Integer. Vector of water pump numerical ID(s). With vestry = TRUE, whole number(s) between 1 and 14. With vestry = FALSE, whole number(s) between 1 and 13. See pumps.vestry and pumps for IDs and details about specific pumps. NULL plots all pumps. Negative selection allowed.
- vestry: Logical. TRUE for the 14 pumps from Vestry Report. FALSE for the original 13 pumps.
- col: Character. Color of pump points.
- pch: Numeric. Shape of point character.
- label: Logical. TRUE adds text label.
- cex: Numeric. point cex.
- latlong: Logical. Use c("lon", "lat") or c("x", "y").

addRoads

Add all streets and roads to plot.

Description
Add all streets and roads to plot.

Usage
addRoads(latlong = FALSE, col = "gray")

Arguments
- latlong: Logical. Use estimated longitude and latitude.
- col: Character. Color
addSnow

Adds Snow’s graphical annotation of the Broad Street pump walking neighborhood.

Usage

addSnow(type = "area", color = "dodgerblue", alpha.level = 0.25, line.width = 2)

Arguments

type Character. Type of annotation plot: "area", "perimeter" or "street".
color Character. Neighborhood color.
alpha.level Numeric. Alpha level transparency: a value in [0, 1].
line.width Numeric. Line width for type = "street" and type = "perimeter".

Examples

## Not run:
plot(neighborhoodVoronoi())
addSnow()
## End(Not run)

addVoronoi

Add Voronoi cells.

Description

Add Voronoi cells.

Usage

addVoronoi(pump.select = NULL, vestry = FALSE, case.location = "nominal", color = "black", line.type = "solid", line.width = 1, latlong = FALSE)
addWalkingPath

Arguments

- **pump.select**: Numeric. Default is NULL; all pumps are used. Otherwise, selection by a vector of numeric IDs: 1 to 13 for pumps; 1 to 14 for pumps.vestry. Exclusion (negative selection) is possible (e.g., -6).

- **vestry**: Logical. FALSE for original 13 pumps. TRUE for 14 pumps in Vestry Report.

- **case.location**: Character. For observed = FALSE: "address" or "nominal". "nominal" is the x-y coordinates of regular.cases.

- **color**: Character. Color of cell edges.

- **line.type**: Character. Type of line for cell edges: lty.

- **line.width**: Numeric. Width of cell edges: lwd.

- **latlong**: Logical. Use estimated longitude and latitude.

Note

This function uses deldir::deldir().

Examples

```r
snowMap()
# addVoronoi()
```

---

**addWalkingPath**  
*Add the shortest walking path between a selected cases or pumps.*

Description

Add the shortest walking path between a selected cases or pumps.

Usage

```r
addWalkingPath(origin = 1, destination = NULL, type = "case-pump", observed = TRUE, weighted = TRUE, vestry = FALSE, distance.unit = "meter", time.unit = "second", walking.speed = 5, unit.posts = "distance", unit.interval = NULL, alpha.level = 1)
```

Arguments

- **origin**: Numeric or Integer. Numeric ID of case or pump.

- **destination**: Numeric or Integer. Numeric ID(s) of case(s) or pump(s). Exclusion is possible via negative selection (e.g., -7). Default is NULL: this returns closest pump or "anchor" case. Character landmark name (case insensitive).

- **type**: Character "case-pump", "cases" or "pumps".

- **observed**: Logical. Use observed or "simulated" expected data.
**Description**

A circle (polygon), centered around a desired pump with a radius of 210 yards. The Broad Street pump is the default.

**Usage**

```r
addWhitehead(pump = "Broad Street", radius = 210, distance.unit = "yard", 
             color = "black", line.type = "solid", vestry = FALSE, 
             add.subtitle = FALSE, walking.speed = 5)
```
anchor.case

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pump</td>
<td>Character or Numeric. Name (road name) or numerical ID of selected pump. See <code>pumps</code> or <code>pumps.vestry</code>.</td>
</tr>
<tr>
<td>radius</td>
<td>Numeric. Distance from a pump.</td>
</tr>
<tr>
<td>distance.unit</td>
<td>Character. Unit of distance: &quot;meter&quot;, &quot;yard&quot; or &quot;native&quot;. &quot;native&quot; returns the map's native scale. See <code>vignette(&quot;roads&quot;)</code> for information on conversion.</td>
</tr>
<tr>
<td>color</td>
<td>Character. Color of circle.</td>
</tr>
<tr>
<td>line.type</td>
<td>Character. Circle line type.</td>
</tr>
<tr>
<td>vestry</td>
<td>Logical. TRUE uses the 14 pumps and locations from Vestry report. FALSE uses original 13 pumps.</td>
</tr>
<tr>
<td>add.subtitle</td>
<td>Logical. Add subtitle with estimated &quot;walking&quot; time in seconds.</td>
</tr>
<tr>
<td>walking.speed</td>
<td>Numeric. Walking speed in km/hr.</td>
</tr>
</tbody>
</table>

Value

Adds a circle (polygon) to a graphics plot.

Examples

```r
snowMap(add.landmarks = FALSE)
addWhitehead()
```

---

anchor.case

*Anchor or base case of each stack of fatalities.*

Description

Data frame that links a fatality to its stack, a stack’s base case. For use with `caseLocator`.

Usage

`anchor.case`

Format

```r
case  numerical case ID
anchor  numerical case ID of anchor.case
```

Note

`unstackFatalities` documents the code for these data.
border

Numeric IDs of line segments that create the map’s border frame.

Description

Vector of ordered numbers that identify the line segments that make up the frame of the map. For use with sp::Polygon().

Usage

border

Format

border numerical ID

caseDistance

Compute distance between case fatalities.

Description

Compute distance between case fatalities.

Usage

caseDistance(a = 19, b = 263, meters = FALSE)

Arguments

a Numeric. Case ID.
b Numeric. Case ID.
meters Logical. Compute metric (meters) or nominal distance.
caseLocator

 Locate case by numerical ID.

Description

Highlight selected observed or simulated case and its home road segment.

Usage

caseLocator(case = 1, zoom = 1, observed = TRUE, add.title = TRUE,
highlight.segment = TRUE, data = FALSE, add = FALSE, col = "red")

Arguments

case Numeric or Integer. Whole number between 1 and 578.
zoom Logical or Numeric. A numeric value \( \geq 0 \) controls the degree of zoom. The
default is 1.
observed Logical. TRUE for observed. FALSE for simulated.
add.title Logical. Include title.
highlight.segment Logical. Highlight case’s segment.
data Logical. Output data.
add Logical. Add to existing plot or separate plot.
col Character. Point color.

Value

A base R graphics plot.

Examples

caseLocator(290)
caseLocator(290, zoom = TRUE)
caseLocator(290, observed = FALSE)
euclideanPath

Compute path of the Euclidean distance between cases and/or pumps.

Description

Compute path of the Euclidean distance between cases and/or pumps.

Usage

euclideanPath(origin = 1, destination = NULL, type = "case-pump",
observed = TRUE, case.location = "nominal", landmark.cases = TRUE,
vestry = FALSE, distance.unit = "meter", time.unit = "second",
walking.speed = 5)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>origin</td>
<td>Numeric or Character. Numeric ID of case or pump. Character landmark name.</td>
</tr>
</tbody>
</table>
| destination      | Numeric or Character. Numeric ID(s) of case(s) or pump(s). Exclusion is pos-
                        possible via negative selection (e.g., -7). Default is NULL, which returns the closest
                        pump, "anchor" case or landmark.                                                          |
| type             | Character "case-pump", "cases" or "pumps".                                   |
| observed         | Logical. Use observed or "simulated" expected data.                          |
| case.location    | Character. For observed = FALSE: "address" or "nominal". "nominal" is the x-y
                        coordinates of regular.cases.                                                            |
| landmark.cases   | Logical. TRUE includes landmarks as cases.                                   |
| vestry           | Logical. TRUE uses the 14 pumps from the Vestry Report. FALSE uses the 13
                        pumps from the original map.                                                              |
| distance.unit    | Character. Unit of distance: "meter", "yard" or "native". "native" returns the
                        map’s native scale. See vignette("roads") for information on unit distances.            |
| time.unit        | Character. "hour", "minute", or "second".                                   |
| walking.speed    | Numeric. Default is 5 km/hr.                                                 |

Value

An R list with 3 data frames: x-y coordinates for the origin and destination, and a summary of
results.

Note

The function uses a case’s "address" (i.e., "anchor" case of a stack) to compute distance. Time is
computed using distanceTime().
Examples

```
# path from case 1 to nearest pump.
euclideanPath(1)

# path from pump 1 to nearest case.
euclideanPath(NULL, 1)

# path from case 1 to pump 6.
euclideanPath(1, 6)

# exclude pump 7 from consideration.
euclideanPath(1, -7)

# path from case 1 to case 6.
euclideanPath(1, 6, type = "cases")

# path from pump 1 to pump 6.
euclideanPath(1, 6, type = "pumps")

# compute multiple cases.
lapply(1:3, euclideanPath)

# plot path
plot(euclideanPath(1))
```

Description

An amended version of Dodson and Tobler’s digitization of John Snow’s map of the 1854 London cholera outbreak. It removes 3 duplicate observations and imputes the location for 3 "missing" observation. This information is also available in HistData::Snow.deaths2 (>= ver. 0.7-8).

Usage

`fatalities`

Format

A data frame with 3 variable that records the position and the nearest pump for the 578 bars on Snow’s map.

- `case` numeric case ID
- `x` x-coordinate
- `y` y-coordinate
- `lon` longitude
- `lat` latitude
fatalities.address

Note

fixFatalities documents the code for these data. For details, see vignette("duplicate.missing.cases").

See Also

caseLocator
streetNameLocator
streetNumberLocator
caseLocator
streetNameLocator
streetNumberLocator

fatalities.address  "Unstacked" amended cholera data with address as unit of observation.

Description

An "unstacked" version of the fatalities dataset. It changes the unit of observation from the case (bar) to the "address", the x-y coordinates of the case at the base of a stack, and makes the number of fatalities an attribute of the "address".

Usage

fatalities.address

Format

A data frame with 4 variables for 321 addresses
anchor  numerical case ID of address
x  x-coordinate
y  y-coordinate
case.count  number of fatalities at address
lon  longitude
lat  latitude

Note

unstackFatalities documents the code for these data. For details, see vignette("unstacking.fatalities").

See Also

caseLocator
streetNameLocator
streetNumberLocator
fatalities.unstacked  "Unstacked" amended cholera fatalities data with fatality as unit of observation.

Description
An "unstacked" version of the fatalities dataset. It changes the unit of observation from the case (bar) to the "address", the x-y coordinates of the case at the base of a stack, and assigns the base case’s coordinates to all cases in the stack.

Usage
fatalities.unstacked

Format
A data frame with 3 variable that records the position of the 578 bars on Snow’s map.

<table>
<thead>
<tr>
<th>variable</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>case</td>
<td>numerical case ID</td>
</tr>
<tr>
<td>x</td>
<td>x-coordinate</td>
</tr>
<tr>
<td>y</td>
<td>y-coordinate</td>
</tr>
<tr>
<td>lon</td>
<td>longitude</td>
</tr>
<tr>
<td>lat</td>
<td>latitude</td>
</tr>
</tbody>
</table>

Note
unstackFatalities documents the code for these data. For details, see vignette("unstacking.fatalities").

See Also
caseLocator
streetNameLocator
streetNumberLocator

fixFatalities  Fix errors in Dodson and Tobler's digitization of Snow's map.

Description
Fixes two apparent coding errors using three misplaced cases.

Usage
fixFatalities()
frame.data

Value

An R data frame.

See Also

vignette("duplicate.missing.cases")

frame.data  Map frame data c("x", "y") and c("lon", "lat").

Description

Map frame data c("x", "y") and c("lon", "lat").

Usage

frame.data

Format

A data frame with 106 observations (points) and 8 variables.

street  street number
n  street street component number
x  native x-coordinate
y  native y-coordinate
id  segment numeric ID
name  street name
lon  longitude
lat  latitude

frame.sample  Partitioned map frame points (segment endpoints).

Description

Partitioned map frame points (segment endpoints).

Usage

frame.sample

Format

A list of 3 vectors length 19, 19 and 18 from cholera::roads$id.
isoLines

Plot isochrone and isodistance regions (prototype)

Usage

isoLines(post = 50, post.type = "distance", palette = "plasma", alpha.level = 1/2)

Arguments

- `post`: Numeric. Distance or time increment.
- `post.type`: Character. "distance" or "time".
- `palette`: Character.
- `alpha.level`: Numeric. Alpha level transparency

landmark.squares

Centers of city squares.

Description

Centers of city squares.

Usage

landmark.squares

Format

A data frame with 6 variables that records the position of the orthogonal projection of landmarks onto the network of roads.

- `name`: square name
- `x`: x-coordinate
- `y`: y-coordinate
- `case`: numeric case ID
landmarkData

Landmark data.

Description

Nominal and orthogonal coordinates

Usage

landmarkData(multi.core = TRUE, dev.mode = FALSE)

Arguments

multi.core Logical or Numeric. TRUE uses parallel::detectCores(). FALSE uses one, single core. You can also specify the number logical cores. See vignette(“Parallelization”) for details.
dev.mode Logical. Development mode uses parallel::parLapply().

landmarks

Orthogonal projection of landmarks onto road network.

Description

Orthogonal projection of landmarks onto road network.

Usage

landmarks

Format

A data frame with 6 variables that records the position of the orthogonal projection of landmarks onto the network of roads.

road.segment "address" road segment
x.proj orthogonal x-coordinate
y.proj orthogonal y-coordinate
ortho.dist orthogonal distance to home road segment
x nominal x-coordinate
y nominal y-coordinate
name landmark name
case numeric case ID
lon longitude
lat latitude
latlong.ortho.addr  Orthogonal projection of observed address (latlong) cases onto road network.

Description
Orthogonal projection of observed address (latlong) cases onto road network.

Usage
latlong.ortho.addr

Format
A data frame with 7 variables that records the position of the orthogonal projection of the 321 cases onto the network of roads.

road.segment  "address" road segment
x.proj  x-coordinate
y.proj  y-coordinate
ortho.dist  orthogonal distance to home road segment
case  numeric case ID
lon  longitude
lat  latitude

Note
unstackFatalities documents the code for these data.

latlong.ortho.pump  Orthogonal projection of 13 original pumps (latlong).

Description
Orthogonal projection of 13 original pumps (latlong).

Usage
latlong.ortho.pump
latlong.ortho.pump.vestry

Format

A data frame with 7 variables that records the position of the orthogonal projection of the 13 original pumps onto the network of roads.

road.segment "address" road segment
x.proj x-coordinate
y.proj y-coordinate
ortho.dist orthogonal distance to home road segment
pump.id numeric ID
lon longitude
lat latitude

Note

pumpData documents the code for these data.

Description

Orthogonal projection of the 14 pumps from the Vestry Report (latlong).

Usage

latlong.ortho.pump.vestry

Format

A data frame with 7 variables that records the position of the orthogonal projection of the 14 pumps onto the network of roads.

road.segment "address" road segment
x.proj x-coordinate
y.proj y-coordinate
ortho.dist orthogonal distance to home road segment
pump.id numeric ID
lon longitude
lat latitude

Note

pumpData documents the code for these data.
latlongAddress

Compute latitude and longitude of case "addresses" (prototype).

Description

Compute latitude and longitude of case "addresses" (prototype).

Usage

latlongAddress(path, multi.core = TRUE)

Arguments

path       Character. e.g., "~/Documents/Data/
multi.core Logical or Numeric. TRUE uses parallel::detectCores(). FALSE uses one,
            single core. You can also specify the number logical cores. See vignette("Parallelization")
            for details.

Value

An R data frame.

Note

This documents the computation of the latlong version of the fatalities.address data frame.

latlongFatalities

Compute latitude and longitude of non-address fatalities (prototype).

Description

Compute latitude and longitude of non-address fatalities (prototype).

Usage

latlongFatalities(path, multi.core = TRUE)

Arguments

path       Character. e.g., "~/Documents/Data/
multi.core Logical or Numeric. TRUE uses parallel::detectCores(). FALSE uses one,
            single core. You can also specify the number logical cores. See vignette("Parallelization")
            for details.
**latlongLandmarks**

Value

An R data frame.

Note

This documents the computation of the latlong version of the fatalities data frame.

---

**Description**

Compute Georeferenced Latitude and Longitude (prototype).

**Usage**

```r
latlongLandmarks(path, orthogonal = FALSE)
```

**Arguments**

- `path` Character. e.g., "~/Documents/Data/
- `orthogonal` Logical. Use orthogonal projection coordinates.

**Note**

This documents the computation of the latlong version of the landmarks data frame.

---

**latlongNearestPump**

Compute shortest georeferenced distances (and walking paths) to selected pumps (prototype).

**Description**

Compute shortest georeferenced distances (and walking paths) to selected pumps (prototype).

**Usage**

```r
latlongNearestPump(pump.select = NULL, metric = "walking", vestry = FALSE, weighted = TRUE, time.unit = "second", walking.speed = 5, multi.core = TRUE)
```
Arguments

pump.select Numeric. Pump candidates to consider. Default is NULL: all pumps are used. Otherwise, selection by a vector of numeric IDs: 1 to 13 for pumps; 1 to 14 for pumps.vestry. Negative selection allowed.

metric Character. "euclidean" or "walking".

vestry Logical. TRUE uses the 14 pumps from the Vestry Report. FALSE uses the 13 in the original map.

weighted Logical. TRUE computes shortest path in terms of road length. FALSE computes shortest path in terms of the number of nodes.

time.unit Character. "hour", "minute", or "second".

walking.speed Numeric. Walking speed in km/hr.

multi.core Logical or Numeric. TRUE uses parallel::detectCores(). FALSE uses one, single core. You can also specify the number logical cores. See vignette("Parallelization") for details.

Value

An R data frame or list of 'igraph' path nodes.

### latlongNeighborhoodData

*Compute network graph of roads, cases and pumps.*

Description

Assembles cases, pumps and road into a network graph.

Usage

```r
latlongNeighborhoodData(vestry = FALSE, multi.core = TRUE)
```

Arguments

vestry Logical.

multi.core Logical or Numeric. TRUE uses parallel::detectCores(). FALSE uses one, single core. You can also specify the number logical cores. See vignette("Parallelization") for details.
latlongNeighborhoodVoronoi

*Compute Voronoi pump neighborhoods (lat-long prototype).*

**Description**

Group cases into neighborhoods using Voronoi tessellation.

**Usage**

```
latlongNeighborhoodVoronoi(pump.select = NULL, vestry = FALSE)
```

**Arguments**

- **pump.select**: Numeric. Vector of numeric pump IDs to define pump neighborhoods (i.e., the "population"). Negative selection possible. NULL selects all pumps.
- **vestry**: Logical. TRUE uses the 14 pumps from the Vestry report. FALSE uses the 13 in the original map.

latlongNeighborhoodWalking

*Compute walking path pump neighborhoods.*

**Description**

Group cases into neighborhoods based on walking distance.

**Usage**

```
latlongNeighborhoodWalking(pump.select = NULL, vestry = FALSE, multi.core = TRUE)
```

**Arguments**

- **pump.select**: Numeric. Vector of numeric pump IDs to define pump neighborhoods (i.e., the "population"). Negative selection possible. NULL selects all pumps. Note that you can’t just select the pump on Adam and Eve Court (#2) because it’s technically an isolate.
- **vestry**: Logical. TRUE uses the 14 pumps from the Vestry report. FALSE uses the 13 in the original map.
- **multi.core**: Logical or Numeric. TRUE uses parallel::detectCores(). FALSE uses one, single core. You can also specify the number logical cores. See vignette("Parallelization") for details.
latlongPumps

*Compute Georeferenced Latitude and Longitude (prototype).*

**Description**

Compute Georeferenced Latitude and Longitude (prototype).

**Usage**

```r
latlongPumps(path, vestry = FALSE)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>Character. e.g., &quot;~/Documents/Data/&quot;</td>
</tr>
<tr>
<td>vestry</td>
<td>Logical.</td>
</tr>
</tbody>
</table>

**Note**

This documents the computation of the latlong version of the pumps and pumps.vestry data frames.

latlongRoads

*Compute latitude and longitude for unique road segment endpoints (prototype).*

**Description**

Compute latitude and longitude for unique road segment endpoints (prototype).

**Usage**

```r
latlongRoads(path, multi.core = TRUE)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>Character. e.g., &quot;~/Documents/Data/&quot;</td>
</tr>
<tr>
<td>multi.core</td>
<td>Logical or Numeric. TRUE uses parallel::detectCores(). FALSE uses one, single core. You can also specify the number logical cores. See vignette(&quot;Parallelization&quot;) for details.</td>
</tr>
</tbody>
</table>

**Value**

An R data frame.

**Note**

This documents the computation of the lat-long version of the roads data frame.
latlongVoronoi

*Description*

Compute Georeferenced Latitude and Longitude of vertices of Voronoi polygons.

*Usage*

```r
latlongVoronoi(pump.select = NULL, vestry = FALSE)
```

*Arguments*

- `pump.select` Numeric. Vector of numeric pump IDs to define pump neighborhoods (i.e., the "population"). Negative selection possible. NULL selects all pumps.
- `vestry` Logical. TRUE uses the 14 pumps from the Vestry report. FALSE uses the 13 in the original map.

*Examples*

```r
snowMap(latlong = TRUE)
cells <- latlongVoronoi()
invisible(lapply(cells, function(x) polygon(x[, c("lon", "lat")])))
```

latlongWalkingPath

*Description*

Plot walking path to nearest pump (prototype).

*Usage*

```r
latlongWalkingPath(case = 1, destination = NULL, vestry = FALSE, weighted = TRUE, distance.unit = "meter", time.unit = "second", walking.speed = 5, multi.core = TRUE)
```

*Arguments*

- `case` Numeric.
- `destination` Numeric. Pump ID.
- `vestry` Logical. TRUE uses the 14 pumps from the map in the Vestry Report. FALSE uses the 13 pumps from the original map.
nearestPump

weighted Logical. TRUE computes shortest path in terms of road length. FALSE computes shortest path in terms of the number of nodes.
distance.unit Character. Unit of distance: "meter" or "yard".
time.unit Character. "hour", "minute", or "second".
walking.speed Numeric. Walking speed in km/hr.
multi.core Logical or Numeric. TRUE uses parallel::detectCores(). FALSE uses one, single core. You can also specify the number logical cores. See vignette("Parallelization") for details.

mapRange Compute xlim and ylim of Snow’s map.

Description
Compute xlim and ylim of Snow’s map.

Usage
mapRange(latlong = FALSE)

Arguments
latlong Logical. Use estimated longitude and latitude.

nearestPump Compute shortest distances or paths to selected pumps.

Description
Compute shortest distances or paths to selected pumps.

Usage
nearestPump(pump.select = NULL, metric = "walking", vestry = FALSE, weighted = TRUE, case.set = "observed", distance.unit = "meter", time.unit = "second", walking.speed = 5, multi.core = TRUE, dev.mode = FALSE)
neighborhoodData

**Arguments**

- **pump.select**: Numeric. Pump candidates to consider. Default is `NULL`: all pumps are used. Otherwise, selection by a vector of numeric IDs: 1 to 13 for pumps; 1 to 14 for pumps.vestry. Negative selection allowed.

- **metric**: Character. "euclidean" or "walking".

- **vestry**: Logical. TRUE uses the 14 pumps from the Vestry Report. FALSE uses the 13 in the original map.

- **weighted**: Logical. TRUE computes shortest path in terms of road length. FALSE computes shortest path in terms of the number of nodes.

- **case.set**: Character. "observed", "expected", or "snow".

- **distance.unit**: Character. Unit of distance: "meter", "yard" or "native". "native" returns the map's native scale. Meaningful only when "weighted" is TRUE. See vignette("roads") for information on unit distances.

- **time.unit**: Character. "hour", "minute", or "second".

- **walking.speed**: Numeric. Walking speed in km/hr.

- **multi.core**: Logical or Numeric. TRUE uses parallel::detectCores(). FALSE uses one, single core. You can also specify the number logical cores. See vignette("Parallelization") for details.

- **dev.mode**: Logical. Development mode uses parallel::parLapply().

**Value**

An R data frame or list of 'igraph' path nodes.

**Note**

Time is computed using distanceTime().

---

**neighborhoodData**

*Compute network graph of roads, cases and pumps.*

**Description**

Assembles cases, pumps and road into a network graph.

**Usage**

```
neighborhoodData(vestry = FALSE, case.set = "observed", embed = TRUE, embed.landmarks = TRUE)
```
Arguments

vestry Logical. Use Vestry Report pump data.
case.set Character. "observed" or "expected", or "snow". "snow" captures John Snow's annotation of the Broad Street pump neighborhood printed in the Vestry report version of the map.
embed Logical. Embed cases and pumps into road network.
embed.landmarks Logical. Embed landmarks into road network.

describe

Value

An R list of nodes, edges and an 'igraph' network graph.

---

neighborhoodEuclidean Compute Euclidean path pump neighborhoods.

Description

Plots star graph from pump to its cases.

Usage

neighborhoodEuclidean(pump.select = NULL, vestry = FALSE, case.location = "nominal", case.set = "observed", multi.core = TRUE, dev.mode = FALSE)

Arguments

pump.select Numeric. Vector of numeric pump IDs to define pump neighborhoods (i.e., the "population"). Negative selection possible. NULL selects all pumps.
vestry Logical. TRUE uses the 14 pumps from the Vestry Report. FALSE uses the 13 in the original map.
case.location Character. "address" or "nominal". For observed = TRUE: "address" uses ortho.proj and "nominal" uses fatalities. For observed = TRUE: "address" uses sim.ortho.proj and "nominal" uses regular.cases.
case.set Character. "observed" or "expected".
multi.core Logical or Numeric. TRUE uses parallel::detectCores(). FALSE uses one, single core. You can also specify the number logical cores. See vignette("Parallelization") for details.
dev.mode Logical. Development mode uses parallel::parLapply().

Value

An R vector.
Examples

```r
## Not run:
neighborhoodEuclidean()
neighborhoodEuclidean(-6)
neighborhoodEuclidean(pump.select = 6:7)
## End(Not run)
```

### neighborhoodVoronoi

Compute Voronoi pump neighborhoods.

#### Description

Group cases into neighborhoods using Voronoi tessellation.

#### Usage

```r
neighborhoodVoronoi(pump.select = NULL, vestry = FALSE, case.location = "address", pump.location = "nominal", polygon.vertices = FALSE)
```

#### Arguments

- `pump.select`: Numeric. Vector of numeric pump IDs to define pump neighborhoods (i.e., the "population"). Negative selection possible. NULL selects all pumps.
- `vestry`: Logical. TRUE uses the 14 pumps from the Vestry report. FALSE uses the 13 in the original map.
- `case.location`: Character. "address" or "nominal". "address" uses the x-y coordinates of `ortho.proj`. "nominal" uses the x-y coordinates of fatalities.
- `pump.location`: Character. "address" or "nominal". "address" uses the x-y coordinates of `ortho.proj.pump` or `ortho.proj.pump.vestry`. "nominal" uses the x-y coordinates of pumps or `pumps.vestry`.
- `polygon.vertices`: Logical. TRUE returns a list of x-y coordinates of the vertices of Voronoi cells. Useful for `sp::point.in.polygon()` as used in `print.voronoi()` method.

#### Value

An R list with 12 objects.

- `pump.id`: vector of selected pumps
- `voronoi`: output from `deldir::deldir()`.
- `snow.colors`: neighborhood color based on `snowColors()`.
- `x.rng`: range of x for plot.
- `y.rng`: range of y for plot.
neighborhoodWalking

- select.string: description of "pump.select" for plot title.
- expected.data: expected neighborhood fatality counts, based on Voronoi cell area.
- coordinates: polygon vertices of Voronoi cells.
- statistic.data: observed neighborhood fatality counts.
- pump.select: "pump.select" from neighborhoodVoronoi().
- statistic: "statistic" from neighborhoodVoronoi().
- vestry: "vestry" from neighborhoodVoronoi().

Examples

neighborhoodVoronoi()
neighborhoodVoronoi(vestry = TRUE)
neighborhoodVoronoi(pump.select = 6:7)
neighborhoodVoronoi(pump.select = -6)
neighborhoodVoronoi(pump.select = -6, polygon.vertices = TRUE)

# coordinates for vertices also available in the returned object.
dat <- neighborhoodVoronoi(pump.select = -6)
dat$coordinates

neighborhoodWalking

*Compute walking path pump neighborhoods.*

Description

Group cases into neighborhoods based on walking distance.

Usage

neighborhoodWalking(pump.select = NULL, vestry = FALSE, weighted = TRUE,
case.set = "observed", multi.core = TRUE, dev.mode = FALSE)

Arguments

- pump.select: Numeric. Vector of numeric pump IDs to define pump neighborhoods (i.e., the "population"). Negative selection possible. NULL selects all pumps. Note that you can’t just select the pump on Adam and Eve Court (#2) because it’s technically an isolate.
- vestry: Logical. TRUE uses the 14 pumps from the Vestry report. FALSE uses the 13 in the original map.
- weighted: Logical. TRUE computes shortest path weighted by road length. FALSE computes shortest path in terms of the number of nodes.
- case.set: Character. "observed", "expected" or "snow". "snow" captures John Snow’s annotation of the Broad Street pump neighborhood printed in the Vestry report version of the map.
**ortho.proj**

**multi.core** Logical or Numeric. TRUE uses parallel::detectCores(). FALSE uses one, single core. You can also specify the number logical cores. See vignette("Parallelization") for details.

**dev.mode** Logical. Development mode uses parallel::parLapply().

**Value**

An R list with 7 objects:

- **paths**: list of paths to nearest or selected pump(s).
- **cases**: list of cases by pump.
- **vestry**: "vestry" from neighborhoodWalking().
- **observed**: "observed" from neighborhoodWalking().
- **pump.select**: "pump.select" from neighborhoodWalking().
- **cores**: number of cores to use for parallel implementation.
- **metric**: incremental metric used to find cut point on split road segments.

**Examples**

```r
## Not run:
neighborhoodWalking()
neighborhoodWalking(pump.select = -6)

## End(Not run)
```

---

**ortho.proj** *Orthogonal projection of observed cases onto road network.*

**Description**

Orthogonal projection of observed cases onto road network.

**Usage**

ortho.proj

**Format**

A data frame with 5 variables that records the position of the orthogonal projection of the 578 cases onto the network of roads.

- **road.segment** "address" road segment
- **x.proj** x-coordinate
- **y.proj** y-coordinate
- **ortho.dist** orthogonal distance to home road segment
- **case** numeric case ID
**ortho.proj.pump**

*Orthogonal projection of 13 original pumps.*

**Description**

Orthogonal projection of 13 original pumps.

**Usage**

`ortho.proj.pump`

**Format**

A data frame with 6 variables that records the position of the orthogonal projection of the 13 original pumps onto the network of roads.

- `pump.id` numeric ID
- `road.segment` "address" road segment
- `x.proj` x-coordinate
- `y.proj` y-coordinate
- `ortho.dist` orthogonal distance to home road segment
- `node` node ID
- `lon` longitude
- `lat` latitude

**Note**

`unstackFatalities` documents the code for these data.

---

**pumpData** documents the code for these data.
Orthogonal projection of the 14 pumps from the Vestry Report.

Orthogonal projection of the 14 pumps from the Vestry Report.

Orthogonal projection of the 14 pumps from the Vestry Report.

Usage

Orthogonal projection of the 14 pumps from the Vestry Report.

Usage

Orthogonal projection of the 14 pumps from the Vestry Report.

Usage

Orthogonal projection of the 14 pumps from the Vestry Report.

Usage

Orthogonal projection of the 14 pumps from the Vestry Report.

Usage

Orthogonal projection of the 14 pumps from the Vestry Report.

Usage
Format

A data frame with 7 variables and 95 observations.

- **year**: yyyy
- **mo**: month (mm)
- **tmax**: maximum temperature degrees Celsius
- **tmin**: minimum temperature degrees Celsius
- **airfrost**: days
- **rain**: millimeters (mm)
- **sun**: sunshine hours

Note

December 1860 excluded due to missing tmin observation.

---

| oxfordWeather | Weather data recorded in Oxford (Met Office UK). |
---|---|

Description

Add and use last day of month as unit of observation to oxford.weather.

Usage

```r
oxfordWeather()
```

Value

An R data frame.

Note

December 1860 observation is dropped due to missing "tmin" value.
plague.pit

Plague pit coordinates.

Description

Coordinates for polygon() or sp::Polygon(). In progress.

Usage

plague.pit

Format

A data frame with 13 observations and 2 variables.

x  x-coordinate
y  y-coordinate

plot.euclidean  Plot method for neighborhoodEuclidean().

Description

Plot method for neighborhoodEuclidean().

Usage

## S3 method for class 'euclidean'
plot(x, type = "star", add.observed.points = TRUE,
   add.title = TRUE, msg = FALSE, ...)

Arguments

x  An object of class "euclidean" created by neighborhoodEuclidean().
type  Character. "star", "area.points" or "area.polygons". "area" flavors only valid when case.set = "expected".
add.observed.points  Logical. Add observed fatality "addresses".
add.title  Logical. Add title.
msg  Logical. Toggle in-progress messages.
...  Additional plotting parameters.

Value

A base R plot.
plot.euclidean_path

Note

This uses an approximate computation of polygons, using the 'TSP' package, that may produce non-simple and/or overlapping polygons.

Examples

```r
## Not run:
plot(neighborhoodEuclidean())
plot(neighborhoodEuclidean(-6))
plot(neighborhoodEuclidean(pump.select = 6:7))
plot(neighborhoodEuclidean(case.set = "expected"), type = "area.points")
plot(neighborhoodEuclidean(case.set = "expected"), type = "area.polygons")

## End(Not run)
```

`plot.euclidean_path`  
Plot the path of the Euclidean distance between cases and/or pumps.

Description

Plot the path of the Euclidean distance between cases and/or pumps.

Usage

```r
## S3 method for class 'euclidean_path'
plot(x, zoom = 0.5, unit.posts = "distance",
     unit.interval = NULL, ...)
```

Arguments

- `x`  
  An object of class "euclidean_path" created by `euclideanPath()`.
- `zoom`  
  Logical or Numeric. A numeric value >= 0 controls the degree of zoom. The default is 0.5.
- `unit.posts`  
  Character. "distance" for mileposts; "time" for timeposts; NULL for no posts.
- `unit.interval`  
  Numeric. Set interval between posts. When `unit.posts` is "distance", `unit.interval` automatically defaults to 50 meters. When `unit.posts` is "time", `unit.interval` automatically defaults to 60 seconds.
- `...`  
  Additional plotting parameters.

Value

A base R plot.

Examples

```r
plot(euclideanPath(15))
plot(euclideanPath(15), unit.posts = "time")
```
**plot.latlongNeighborhoodVoronoi**

*Plot method for plot.latlongNeighborhoodVoronoi()*

**Description**

Plot method for plot.latlongNeighborhoodVoronoi()

**Usage**

```r
## S3 method for class 'latlongNeighborhoodVoronoi'
plot(x, add.cases = TRUE,
     add.pumps = TRUE, euclidean.paths = FALSE, ...)
```

**Arguments**

- `x` Object. Currently separate classification check.
- `add.cases` Logical.
- `add.pumps` Logical.
- `euclidean.paths` Logical. Currently separate classification check.
- `...` Additional plotting parameters.

**Value**

An igraph base graphics plot.

---

**plot.latlong_neighborhood_data**

*Plot method for plot.latlongNeighborhoodData()*.  

**Description**

Visualize underlying road network (with or without cases and pumps).

**Usage**

```r
## S3 method for class 'latlong_neighborhood_data'
plot(x, ...)
```

**Arguments**

- `x` An `igraph` object of class "latlong_neighborhood_data" created by latlongNeighborhoodData().
- `...` Additional plotting parameters.

**Value**

An igraph base graphics plot.
### Description

Plot method for `latlongNeighborhoodWalking()`.

### Usage

```r
## S3 method for class 'latlong_walking'
plot(x, ...)  
```  
**Arguments**

- `x` An object of class "latlong_walking" created by `latlongNeighborhoodWalking()`.
- `...` Additional plotting parameters.

**Value**

A base R plot.

### Description

Plot the walking path between selected cases and/or pumps.

### Usage

```r
## S3 method for class 'latlong_walking_path'
plot(x, zoom = TRUE, mileposts = TRUE, milepost.unit = "distance", milepost.interval = NULL, alpha.level = 1, ...)  
```  
**Arguments**

- `x` An object of class "latlong_walking_path" created by `latlongWalkingPath()`.
- `zoom` Logical or Numeric. A numeric value >= 0 that controls the degree of zoom.
- `mileposts` Logical. Plot mile/time posts.
- `milepost.unit` Character. "distance" or "time".
- `milepost.interval` Numeric. Mile post interval unit of distance (yard or meter) or unit of time (seconds).
- `alpha.level` Numeric. Alpha level transparency for path: a value in [0, 1].
- `...` Additional plotting parameters.
plot.neighborhood_data

Description
Visualize underlying road network (with or without cases and pumps).

Usage
## S3 method for class 'neighborhood_data'
plot(x, ...)

Arguments
x An 'igraph' object of class "neighborhood_data" created by neighborhoodData().
... Additional plotting parameters.

Value
A base R plot.

Examples
plot(neighborhoodData())
plot(neighborhoodData(embed = FALSE))

plot.oxfordWeather

Description
Plot method for oxfordWeather().

Usage
## S3 method for class 'oxfordWeather'
plot(x, statistic = "temperature",
     month = "september", ...)


Arguments

\begin{itemize}
  \item \texttt{x} \quad \text{object.}
  \item \texttt{statistic} \quad \text{Character.}
  \item \texttt{month} \quad \text{Character. "august" or "september".}
  \item \ldots \quad \text{Additional plotting parameters.}
\end{itemize}

Value

A base R plot.

---

\texttt{plot.povertyLondon} \quad \textit{Plot method for \texttt{povertyLondon}().}

Description

Plot method for \texttt{povertyLondon}().

Usage

\begin{verbatim}
## S3 method for class 'povertyLondon'
plot(x, district = c("City", "Westminster", "Marylebone", "St. Giles"), ...)
\end{verbatim}

Arguments

\begin{itemize}
  \item \texttt{x} \quad \text{object.}
  \item \texttt{district} \quad \text{Character. Selected district(s).}
  \item \ldots \quad \text{Additional plotting parameters.}
\end{itemize}

---

\texttt{plot.profile_perspective} \quad \textit{Plot method for \texttt{profilePerspective}().}

Description

Plot method for \texttt{profilePerspective}().

Usage

\begin{verbatim}
## S3 method for class 'profile_perspective'
plot(x, ...)\end{verbatim}

Arguments

\begin{itemize}
  \item \texttt{x} \quad \text{An object of class "profile" created by \texttt{profilePerspective}().}
  \item \ldots \quad \text{Additional plotting parameters.}
\end{itemize}
**Description**

Plot aggregate fatality data and indicates the date of the removal of the handle of the Broad Street pump.

**Usage**

```r
## S3 method for class 'time_series'
plot(x, statistic = "fatal.attacks",
     pump.handle = TRUE, main = "Removal of the Broad Street Pump Handle",
     type = "o", xlab = "Date", ylab = "Fatalities", ...)
```

**Arguments**

- `x`: An object of class "time_series" from timeSeries().
- `statistic`: Character. Fatality measure: either "fatal.attacks" or "deaths".
- `pump.handle`: Logical. Indicate date of removal of Broad Street pump handle.
- `main`: Character. Title of graph.
- `type`: Character. R plot type.
- `xlab`: Character. x-axis label.
- `ylab`: Character. y-axis label.
- `...`: Additional plotting parameters.

**See Also**

timeSeries

**Examples**

```r
plot(timeSeries())
plot(timeSeries(), statistic = "deaths")
plot(timeSeries(), bty = "n", type = "h", lwd = 4)
```
Description

Plot Voronoi neighborhoods.

Usage

## S3 method for class 'voronoi'
plot(x, voronoi.cells = TRUE, delaunay.triangles = FALSE,
     euclidean.paths = FALSE, ...)

Arguments

x An object of class "voronoi" created by neighborhoodVoronoi().

voronoi.cells Logical. Plot Voronoi tessellation cells.

delaunay.triangles Logical. Plot Delaunay triangles.

euclidean.paths Logical. Plot all Euclidean paths (star graph).

... Additional plotting parameters.

Value

A base R graph.

See Also

neighborhoodVoronoi()

addVoronoi()

Examples

plot(neighborhoodVoronoi())
plot.walking

Plot method for neighborhoodWalking().

Description

Plot method for neighborhoodWalking().

Usage

## S3 method for class 'walking'
plot(x, type = "roads", msg = FALSE,
tsp.method = "repetitive_nn", ...) 

Arguments

x An object of class "walking" created by neighborhoodWalking().
type Character. "roads", "area.points" or "area.polygons". "area" flavors only valid when case.set = "expected".
msg Logical. Toggle in-progress messages.
tsp.method Character. Traveling salesperson problem algorithm.
...

Value

A base R plot.

Note

When plotting area graphs with simulated data (i.e., case.set = "expected"), there may be discrepancies between observed cases and expected neighborhoods, particularly between neighborhoods.

Examples

## Not run:
plot(neighborhoodWalking())
plot(neighborhoodWalking(case.set = "expected"))
plot(neighborhoodWalking(case.set = "expected"), type = "area.points")
plot(neighborhoodWalking(case.set = "expected"), type = "area.polygons")

## End(Not run)
plot.walking_path

Plot the walking path between selected cases and/or pumps.

Description

Plot the walking path between selected cases and/or pumps.

Usage

```r
## S3 method for class 'walking_path'
plot(x, zoom = 0.5, stacked = TRUE,
     unit.posts = "distance", unit.interval = NULL, alpha.level = 1, ...)
```

Arguments

- `x` An object of class "walking_path" created by walkingPath().
- `zoom` Logical or Numeric. A numeric value >= 0 controls the degree of zoom. The default is 0.5.
- `stacked` Logical. Use stacked fatalities.
- `unit.posts` Character. "distance" for mileposts; "time" for timeposts; NULL for no posts.
- `unit.interval` Numeric. Set interval between posts. When `unit.posts = "distance"`, `unit.interval` defaults to 50 meters. When `unit.posts = "time"`, `unit.interval` defaults to 60 seconds.
- `alpha.level` Numeric. Alpha level transparency for path: a value in [0, 1].
- `...` Additional plotting parameters.

Value

A base R plot.

Note

Arrows represent mileposts or timeposts to the destination.

Examples

```r
## Not run:
plot(walkingPath(15))
plot(walkingPath(15), unit.posts = "time")
```

## End(Not run)
plot.winterTemperatures

*Plot method for winterTemperatures().*

**Description**

Plot method for winterTemperatures().

**Usage**

```r
## S3 method for class 'winterTemperatures'
plot(x, end.date = "1859-6-1", ...)
```

**Arguments**

- `x` object.
- `end.date` Date. "yyyy-mm-dd" or NULL.
- `...` Additional plotting parameters.

**Value**

A base R plot.

**Examples**

```r
plot(winterTemperatures())
```

---

povertyLondon

*Poverty and Born in London.*

**Description**


**Usage**

`povertyLondon()`
print.euclidean

Print method for neighborhoodEuclidean().

Description
Parameter values for neighborhoodEuclidean().

Usage
## S3 method for class 'euclidean'
print(x, ...)

Arguments
x An object of class "euclidean" created by neighborhoodEuclidean().
... Additional parameters.

Value
A list of argument values.

Examples
## Not run:
neighborhoodEuclidean()
print(neighborhoodEuclidean())
## End(Not run)

print.euclidean_path

Print method for euclideanPath().

Description
Summary output.

Usage
## S3 method for class 'euclidean_path'
print(x, ...)

Arguments
x An object of class "euclidean_path" created by euclideanPath().
... Additional parameters.
Value
An R data frame.

Examples
euclideanPath(1)
print(euclideanPath(1))
print.time_series

Description

Return summary results.

Usage

## S3 method for class 'time_series'
print(x, ...)

Arguments

x An object of class "time_series" created by timeSeries().
...

Value

An R data frame.

---

print.latlong_walking_path

Print method for latlongWalkingPath().

Description

Summary output.

Usage

## S3 method for class 'latlong_walking_path'
print(x, ...)

Arguments

x An object of class "latlong_walking_path" created by latlongWalkingPath().
...

Value

An R data frame.

---

Value

A list of argument values.
print.voronoi

Examples

    timeSeries()
    print(timeSeries())

---

print.voronoi  Print method for neighborhoodVoronoi().

Description

Parameter values for neighborhoodVoronoi().

Usage

    ## S3 method for class 'voronoi'
    print(x, ...)

Arguments

    x  An object of class "voronoi" created by neighborhoodVoronoi().
    ... Additional arguments.

Value

A list of argument values.

Examples

    neighborhoodVoronoi()
    print(neighborhoodVoronoi())

---

print.walking  Print method for neighborhoodWalking().

Description

Parameter values for neighborhoodWalking().

Usage

    ## S3 method for class 'walking'
    print(x, ...)

Arguments

    x  An object of class "walking" created by neighborhoodWalking().
    ... Additional parameters.
Value

A list of argument values.

Examples

```r
## Not run:
neighborhoodWalking()
print(neighborhoodWalking())

## End(Not run)
```

Description

Summary output.

Usage

```r
## S3 method for class 'walking_path'
print(x, ...)
```

Arguments

- `x` An object of class "walking_path" created by `walkingPath()`.
- `...` Additional parameters.

Value

An R data frame.

Examples

```r
## Not run:
walkingPath()
print(walkingPath())

## End(Not run)
```
profile2D

Description
2D Profile.

Usage
profile2D(angle = 0, pump = 7, vestry = FALSE, type = "base",
multi.core = TRUE)

Arguments
angle Numeric. Angle of perspective axis in degrees.
pump Numeric. Select pump as focal point.
vestry Logical. TRUE uses the 14 pumps from the Vestry Report. FALSE uses the 13 in
the original map.
type Character. Type of graphic: "base" or "ggplot2".
multi.core Logical or Numeric. TRUE uses parallel::detectCores(). FALSE uses one,
single core. You can also specify the number logical cores. See vignette("Parallelization")
for details.

Examples
## Not run:
profile2D(angle = 30)
profile2D(angle = 30, type = "ggplot2")
## End(Not run)

profile3D

Description
3D Profile.

Usage
profile3D(pump.select = NULL, pump.subset = NULL, vestry = FALSE,
drop.neg.subset = FALSE, multi.core = TRUE)
Arguments

**pump.select**  Numeric. Vector of numeric pump IDs to define pump neighborhoods (i.e., the "population"). Negative selection possible. NULL selects all pumps.

**pump.subset**  Numeric. Vector of numeric pump IDs to subset from the neighborhoods defined by pump.select. Negative selection possible. NULL selects all pumps in pump.select.

**vestry**  Logical. TRUE uses the 14 pumps from the Vestry Report. FALSE uses the 13 in the original map.

**drop.neg.subset**  Logical. Drop negative subset selection

**multi.core**  Logical or Numeric. TRUE uses parallel::detectCores(). FALSE uses one, single core. You can also specify the number logical cores. See vignette("Parallelization") for details.

Examples

```r
## Not run:
profile3D(pump.select = 6:7)
profile3D(pump.subset = -7)
profile3D(pump.subset = -7, drop.neg.subset = TRUE)
## End(Not run)
```

pumpCase

*Extract numeric case IDs by pump neighborhood.*

Description

Extract numeric case IDs by pump neighborhood.

Usage

```r
pumpCase(x, case)
```

Arguments

**x**  An object created by neighborhoodEuclidean(), neighborhoodVoronoi() or neighborhoodWalking().

**case**  Character. "address" or "fatality"

Value

An R list of numeric ID of cases by pump neighborhoods.
pumpData

Examples

```r
## Not run:
pumpCase(neighborhoodEuclidean())
pumpCase(neighborhoodVoronoi())
pumpCase(neighborhoodWalking())

## End(Not run)
```

---

**pumpData**

*Compute pump coordinates.*

**Description**

Returns either the set of x-y coordinates for the pumps themselves or for their orthogonally projected "addresses" on the network of roads.

**Usage**

```r
pumpData(vestry = FALSE, orthogonal = FALSE, multi.core = TRUE)
```

**Arguments**

- `vestry` Logical. TRUE uses the 14 pumps from the Vestry report. FALSE uses the 13 in the original map.
- `orthogonal` Logical. TRUE returns pump "addresses": the coordinates of the orthogonal projection from a pump’s location onto the network of roads. FALSE returns pump location coordinates.
- `multi.core` Logical or Numeric. TRUE uses `parallel::detectCores()`. FALSE uses one, single core. With Numeric, you specify the number logical cores (rounds with `as.integer()`). See vignette("Parallelization") for details.

**Value**

An R data frame.

**Note**

Note: The location of the fourteenth pump, at Hanover Square, and the "correct" location of the Broad Street pump are approximate. This function documents the code that generates `pumps`, `pumps.vestry`, `ortho.proj.pump` and `ortho.proj.pump.vestry`.

**See Also**

`pumpLocator`
pumpFatalities  
Compute fatalities by pump.

Description
Compute fatalities by pump.

Usage
pumpFatalities(pump.select = NULL, metric = "walking", vestry = FALSE, latlong = FALSE, multi.core = TRUE)

Arguments
- **pump.select** Numeric. Pump candidates to consider. Default is NULL: all pumps are used. Otherwise, selection by a vector of numeric IDs: 1 to 13 for pumps; 1 to 14 for pumps.vestry. Negative selection allowed.
- **metric** Character. "euclidean" or "walking".
- **vestry** Logical. TRUE uses the 14 pumps from the Vestry Report. FALSE uses the 13 in the original map.
- **latlong** Logical. Use estimated longitude and latitude.
- **multi.core** Logical or Numeric. TRUE uses parallel::detectCores(). FALSE uses one, single core. You can also specify the number logical cores. See vignette("Parallelization") for details.

Examples
## Not run:
pumpFatalities(pump.select = -7)
pumpFatalities(metric = "euclidean")
pumpFatalities(metric = "euclidean", vestry = TRUE)
## End(Not run)

pumpLocator  
Locate water pump by numerical ID.

Description
Highlight selected water pump.

Usage
pumpLocator(id = 7, zoom = 1, vestry = FALSE, add.title = TRUE, highlight.segment = TRUE, data = FALSE)
Arguments

- **id**: Numeric or Integer. With `vestry = TRUE`, a whole number between 1 and 14. With `vestry = FALSE`, a whole number between 1 and 13. See `cholera::pumps.vestry` and `cholera::pumps` for IDs and details about specific pumps.

- **zoom**: Logical or Numeric. A numeric value $\geq 0$ controls the degree of zoom. The default is 1.

- **vestry**: Logical. `TRUE` for the 14 pumps from Vestry Report. `FALSE` for the original 13 pumps.

- **add.title**: Logical. Include title.

- **highlight.segment**: Logical. Highlight case’s segment.

- **data**: Logical. Output data.

Value

A base R graphics plot.

See Also

- `pumpData`

Examples

```r
pumpLocator()
pumpLocator(zoom = TRUE)
pumpLocator(14, vestry = TRUE, zoom = TRUE)
```

---

**pumps**: *Dodson and Tobler’s pump data with street name.*

Description

Adds and amends road locations for water pumps from John Snow’s map to Dodson and Tobler’s street data. The latter are available at Michael Friendly’s HistData::Snow.streets.

Usage

- `pumps`

Format

A data frame with 13 observations and 4 variables that describe the pumps on Snow’s map.

- id: pump number between 1 and 13
- street: nearest street
- x: x-coordinate
Description

These data include the fourteenth pump, at Hanover Square, and the "corrected" location of the Broad Street pump that Snow includes in the second version of his map in the Vestry report.

Usage

pumps.vestry

Format

A data frame with 14 observations and 4 variables.

- id: pump number between 1 and 14
- street: nearest street
- x: x-coordinate
- y: y-coordinate
- lon: longitude
- lat: latitude

Note

pumpData documents the code for these data.

See Also

pumpLocator
**rd.sample**

Sample of road intersections (segment endpoints).

**Description**

Sample of road intersections (segment endpoints).

**Usage**

rd.sample

**Format**

A list with 2 variables that list randomly re-arranges unique road intersections (segment endpoints).

one endpoints with 1 intersection
three endpoints with 3 intersections

---

**rectangle.filter**

Rectangular filter data.

**Description**

Coordinates to filter out frame shadow using sp::point.in.polygon().

**Usage**

rectangle.filter

**Format**

A data frame with 2 variables and 4 observations.

x longitude
y latitude
regular.cases  "Expected" cases.

Description
The result of using \texttt{sp::spsample()} and \texttt{sp::Polygon()} to generate 19,993 regularly spaced simulated cases within the map’s borders.

Usage

regular.cases

Format
A data frame with 2 variable that records the position of 19,993 "expected" cases fitted by \texttt{sp::spsample()}.

\begin{verbatim}
x  x-coordinate
y  y-coordinate
\end{verbatim}

Note
\texttt{simulateFatalities} documents the code for these data.

road.segments  Dodson and Tobler’s street data transformed into road segments.

Description
This data set transforms Dodson and Tobler’s street data to give each straight line segment of a "road" a unique ID.

Usage

road.segments

Format
A data frame with 657 observations and 7 variables. The data describe the straight line segments used to recreate the roads on Snow’s map.

\begin{verbatim}
street  numeric street ID, which range between 1 and 528
id     character segment ID
name   road name
x1     x-coordinate of first endpoint
y1     y-coordinate of first endpoint
x2     x-coordinate of second endpoint
y2     y-coordinate of second endpoint
\end{verbatim}
Note

roadSegments documents the code for these data.

See Also

roads
vignette("road.names")
streetNameLocator
streetNumberLocator
segmentLocator

roads

Dodson and Tobler’s street data with appended road names.

Description

This data set adds road names from John Snow’s map to Dodson and Tobler’s street data. The latter are also available from HistData::Snow.streets.

Usage

roads

Format

A data frame with 206 observations and 5 variables. The data describe the roads on Snow’s map.

street street segment number, which range between 1 and 528
n number of points in this street line segment
x x-coordinate
y y-coordinate
id unique numeric ID
name road name
lon longitude
lat latitude

See Also

road.segments
vignette("road.names")
streetNameLocator
streetNumberLocator
segmentLocator
roadSegmentFix

*Bar orientation classification errors.*

**Description**

Bar orientation classification errors.

**Usage**

roadSegmentFix()

**Note**

Bars lie parallel to the road where that fatality is observed. This can lead to (classification) errors when using orthogonal projection to assign a street address: the closest road is not always the right road. This R list manually assigns those problematic bars to their "correct" road segment.

roadSegments

*Reshape 'roads' data frame into 'road.segments' data frame.*

**Description**

Used to integrate pumps and cases into road network when computing walking neighborhoods.

**Usage**

roadSegments(latlong = FALSE)

**Arguments**

- **latlong** Logical. Use estimated longitude and latitude.

**Value**

An R data frame.

**Note**

This function documents the code that generates road.segments.
segmentHighlight

Highlight segment by ID.

Description

Highlight segment by ID.

Usage

segmentHighlight(id, highlight = TRUE, col = "red", angled = FALSE)

Arguments

id
  Character. A concatenation of a street’s numeric ID, a whole number between 1
  and 528, and a second number to identify the segment.

highlight
  Logical. Color segment.

col
  Character. Highlight color.

angled
  Logical. Rotate segment ID label.

Value

A base R graphics segment(s).

Examples

streetNameLocator("Soho Square", zoom = TRUE, highlight = FALSE)
ids <- road.segments[road.segments$name == "Soho Square", "id"]
invisible(lapply(ids, function(x) segmentHighlight(x, highlight = FALSE)))

segmentLength

Compute length of road segment.

Description

Compute length of road segment.

Usage

segmentLength(id = "216-1", distance.unit = "meter")

Arguments

id
  Character. A concatenation of a street’s numeric ID, a whole number between 1
  and 528, and a second number used to identify the sub-segments.

distance.unit
  Character. Unit of distance: "meter", "yard" or "native". "native" returns the
  map’s native scale. See vignette("roads") for information on conversion.
segmentLocator

Value
An R vector of length one.

Examples

segmentLength("242-1")
segmentLength("242-1", distance.unit = "yard")

segmentLocator(id = "216-1", zoom = 0.5, cases = "address",
distance.unit = "meter", time.unit = "second", walking.speed = 5,
add.title = TRUE, add.subtitle = TRUE, highlight = TRUE,
cex.text = 0.67)

Arguments

id Character. A concatenation of a street’s numeric ID, a whole number between 1
and 528, and a second number to identify the segment.

zoom Logical or Numeric. A numeric value >= 0 controls the degree of zoom. The
default is 0.5.

cases Character. Plot cases: NULL, "address" or "fatality".

distance.unit Character. Unit of distance: "meter", "yard" or "native". "native" returns the
map’s native scale. See vignette("roads") for information on conversion.

time.unit Character. "hour", "minute", or "second".

walking.speed Numeric. Walking speed in km/hr.

add.title Logical. Print title.

add.subtitle Logical. Print subtitle.

highlight Logical. Highlight selected road and its cases.

cex.text Numeric.

Value
A base R graphics plot.
sim.ortho.proj

**Note**

With Dodson and Tobler’s data, a street (e.g., Broad Street) is often comprised of multiple straight line segments. To identify each segment individually, an additional number is appended to form a text string ID (e.g., “116-2”). See cholera::road.segments.

**Examples**

```r
segmentLocator("190-1")
segmentLocator("216-1")
segmentLocator("216-1", distance.unit = "yard")
```

---

**sim.ortho.proj**  
*Road “address” of simulated (i.e., “expected”) cases.*

**Description**

Road “address” of simulated (i.e., "expected") cases.

**Usage**

```r
sim.ortho.proj
```

**Format**

A data frame with 6 variables that records the "address" of 19,993 simulate cases along the network of roads.

- `road.segment` "address" road segment
- `x.proj` x-coordinate
- `y.proj` y-coordinate
- `dist` Euclidean or orthogonal distance to home road segment
- `type` type of projection: Euclidean ("eucl") or orthogonal ("ortho")
- `case` numeric case ID

**Note**

`simulateFatalities` documents the code for these data.
Data Description

List of "simulated" fatalities grouped by walking-distance pump neighborhood.

Usage

sim.pump.case

Format

A list 4972 IDs spread over 13 vectors.

sim.pump.case numerical ID

Note

neighborhoodWalking documents the code for these data. For details, see vignette("pump.neighborhoods").

Examples

```r
## Not run:
pumpCase(neighborhoodWalking(case.set = "expected"))
## End(Not run)
```

Data Description

Walking distance to Broad Street Pump (#7).

Usage

sim.walking.distance
simulateFatalities

**Format**

A data frames with 5 variables.

- **case** case ID
- **pump** pump ID
- **pump.name** pump name
- **distance** walking distance in meters
- **time** walking time in seconds based on 5 km/hr walking speed

**Description**

Places regularly spaced "simulated" or "expected" cases across the face of the map. The function finds the "addresses" of cases via orthogonal projection or simple proximity. These data are used to generate "expected" pump neighborhoods. The function relies on sp::spsample() and sp::Polygon().

**Usage**

```r
simulateFatalities(compute = FALSE, multi.core = TRUE, 
  simulated.obs = 20000L, dev.mode = FALSE)
```

**Arguments**

- **compute** Logical. TRUE computes data. FALSE uses pre-computed data. For replication of data used in the package,
- **multi.core** Logical or Numeric. TRUE uses parallel::detectCores(). FALSE uses one, single core. With Numeric, you specify the number logical cores (rounds with as.integer()). See vignette("Parallelization") for details.
- **simulated.obs** Numeric. Number of sample cases.
- **dev.mode** Logical. Development mode uses parallel::parLapply().

**Value**

An R list with two elements: **sim.ortho.proj** and **regular.cases**

**Note**

This function is computationally intensive. With "simulated.obs" set to 20,000 simulated cases (actually generating 19,993 cases). This function documents the code that generates **sim.ortho.proj** and **regular.cases**. In real world terms, the distance between of these simulated cases is approximately 6 meters.
simulateWalkingDistance

*Compute walking distance for simulated cases.*

**Description**

Compute walking distance for simulated cases.

**Usage**

```r
simulateWalkingDistance(pump.select = 7, multi.core = TRUE, 
dev.mode = FALSE, compute = FALSE)
```

**Arguments**

- `pump.select` Numeric.
- `multi.core` Logical or Numeric. TRUE uses `parallel::detectCores()`. FALSE uses one, single core. You can also specify the number logical cores.
- `dev.mode` Logical. Development mode uses `parallel::parLapply()`.
- `compute` Logical.

**Note**

This function is computationally intensive. See vignette("Parallelization") for details. This functions document the code that generates `sim.walking.distance`.

---

snow.neighborhood

*Snow neighborhood fatalities.*

**Description**

Numeric IDs of fatalities from Dodson and Tobler that fall within Snow’s Broad Street pump neighborhood.

**Usage**

```r
snow.neighborhood
```

**Format**

A vector with 384 observations.

```r
snow.neighborhood numeric case ID
```
snowColors

Create a set of colors for pump neighborhoods.

Description

Uses RColorBrewer::brewer.pal().

Usage

snowColors(vestry = FALSE)

Arguments

vestry Logical. TRUE uses the 14 pumps in the Vestry Report. FALSE uses the original 13.

Value

A character vector of colors.

Note

Built with 'RColorBrewer' package.

snowMap

Plot John Snow's cholera map.

Description

Plot John Snow's cholera map.

Usage

snowMap(vestry = FALSE, stacked = TRUE, add.cases = TRUE, add.landmarks = FALSE, add.pumps = TRUE, add.roads = TRUE, add.frame = TRUE, main = NA, case.col = "gray", case.pch = 15, latlong = FALSE, ...)

Arguments

vestry Logical. TRUE uses the 14 pumps from the map in the Vestry Report. FALSE uses the 13 pumps from the original map.

stacked Logical. Use stacked fatalities.

add.cases Logical. Add observed cases.

add.landmarks Logical. Add landmarks.

add.pumps Logical. Add pumps.

add.roads Logical. Add roads.

add.frame Logical. Add map frame.

main Character. Title of graph.

case.col Character. Color of fatalities.

case.pch Character. Color of fatalities.

latlong Logical. Use estimated longitude and latitude.

... Additional plotting parameters.

Value

A base R graphics plot.

Note

Uses amended version of Dodson and Tobler’s data included in this package.

Examples

snowMap()
snowMap(vestry = TRUE, stacked = FALSE)

snowNeighborhood Plotting data for Snow’s graphical annotation of the Broad Street pump neighborhood.

Description

Computes “missing” and split road segments data, and area plot data.

Usage

snowNeighborhood()

Value

An R list of edge IDs and simulated case IDs.
**streetHighlight**

*Highlight road by name.*

**Description**

Highlight road by name.

**Usage**

```
streetHighlight(road.name, col = "red", lwd = 3)
```

**Arguments**

- **road.name**  
  Character vector. The functions tries to correct for case and to remove extra spaces.
- **col**  
  Character. Highlight color.
- **lwd**  
  Numeric. Line width.

**Value**

A base R graphics segment(s).

**Examples**

```
snowMap()
streetHighlight("Broad Street")
```

---

**streetLength**

*Compute length of selected street.*

**Description**

Compute length of selected street.

**Usage**

```
streetLength(road = "Oxford Street", distance.unit = "meter")
```

**Arguments**

- **road**  
  Character or Numeric. Road name or number. For names, the function tries to correct for case and to remove extra spaces.
- **distance.unit**  
  Character. Unit of distance: "meter", "yard" or "native". "native" returns the map's native scale. See vignette("roads") for information on conversion.
Value

An R vector of length one.

Examples

```r
streetLength("Oxford Street")
streetLength("oxford street")
streetLength("oxford street", distance.unit = "yard")
```

---

**streetNameLocator**

*Locate road by name.*

Description

Highlight a road and its cases. See the list of road names in vignette("road.names").

Usage

```r
streetNameLocator(road.name = "Broad Street", zoom = FALSE, 
cases = "address", token = "id", add.title = TRUE, 
add.subtitle = TRUE, add.pump = TRUE, vestry = FALSE, 
highlight = TRUE, distance.unit = "meter", time.unit = "minute", 
walking.speed = 5)
```

Arguments

- `road.name` Character vector. Note that `streetNameLocator()` tries to correct for case and to remove extra spaces.
- `zoom` Logical or Numeric. A numeric value >= 0 controls the degree of zoom. The default is FALSE, which is equivalent to zero.
- `cases` Character. Plot cases: NULL, "address" or "fatality".
- `token` Character. "id" or "point".
- `add.title` Logical. Include title.
- `add.subtitle` Logical. Include subtitle with road information.
- `add.pump` Logical. Include nearby pumps.
- `vestry` Logical. TRUE uses the 14 pumps from the Vestry report. FALSE uses the 13 in the original map.
- `highlight` Logical. Highlight selected road and its cases.
- `distance.unit` Character. Unit of distance: "meter", "yard" or "native". "native" returns the map's native scale. See vignette("roads") for information on conversion.
- `time.unit` Character. "hour", "minute", or "second".
- `walking.speed` Numeric. Walking speed in km/hr.
streetNames

Value

A base R graphics plot.

Examples

streetNameLocator("Oxford Street")
streetNameLocator("oxford street")
streetNameLocator("Cambridge Street", zoom = TRUE)
streetNameLocator("Cambridge Street", zoom = 0.5)

streetNames

Street names (alphabetized).

Description

Unique road names from Snow’s cholera map.

Usage

streetNames()

Value

An R character vector.

Note

See vignette("roads"), and roads and road.segment data frames.

streetNumberLocator

Locate road by numerical ID.

Description

Highlight a road and its cases. See cholera::roads for numerical IDs and vignette("road.names") for details.

Usage

streetNumberLocator(road.number = 216, zoom = FALSE, cases = "address",
                   token = "id", add.title = TRUE, add.subtitle = TRUE, add.pump = TRUE,
                   vestry = FALSE, highlight = TRUE, distance.unit = "meter",
                   time.unit = "second", walking.speed = 5)
Arguments

road.number  Numeric or integer. A whole number between 1 and 528.
zoom  Logical or Numeric. A numeric value \( \geq 0 \) controls the degree of zoom. The default is FALSE, which is equivalent to zero.
cases  Character. Plot cases: NULL, "address" or "fatality".
token  Character. "id" or "point".
add.title  Logical. Include title.
add.subtitle  Logical. Include subtitle with road information.
add.pump  Logical. Include nearby pumps.
vestry  Logical. TRUE uses the 14 pumps from the Vestry report. FALSE uses the 13 in the original map.
highlight  Logical. Highlight selected road and its cases.
distance.unit  Character. Unit of measurement: "meter" or "yard". Default is NULL, which returns the map's native scale.
time.unit  Character. "hour", "minute", or "second".
walking.speed  Numeric. Walking speed in km/hr.

Value

A base R graphics plot.

Examples

streetNumberLocator(243)
streetNumberLocator(243, zoom = TRUE)
streetNumberLocator(243, zoom = 0.5)

subsetRoadsSamples  Sample for road segment endpoints.

Description

For endpoints with 1 or 3 intersections.

Usage

subsetRoadsSamples()
### summary.euclidean

**Summary method for neighborhoodEuclidean().**

**Description**

Return computed counts for Euclidean neighborhoods.

**Usage**

```r
## S3 method for class 'euclidean'
summary(object, ...)
```

**Arguments**

- `object` Object. An object of class "euclidean" created by `neighborhoodEuclidean()`.
- `...` Additional parameters.

**Value**

A vector of counts by neighborhood.

**Examples**

```r
## Not run:
summary(neighborhoodEuclidean())
## End(Not run)
```

### summary.voronoi

**Summary method for neighborhoodVoronoi().**

**Description**

Return computed counts for Voronoi neighborhoods.

**Usage**

```r
## S3 method for class 'voronoi'
summary(object, ...)
```

**Arguments**

- `object` Object. An object of class "voronoi" created by `neighborhoodVoronoi()`.
- `...` Additional arguments.
Value

A vector of counts by neighborhood.

See Also

addVoronoi() plot.vorono()i

Examples

summary(neighborhoodVoronoi())

summary.walking  Summary method for neighborhoodWalking().

Description

Return computed counts for walking neighborhoods.

Usage

## S3 method for class 'walking'
summary(object, ...)  

Arguments

object  Object. An object of class "walking" created by neighborhoodWalking().

...  Additional parameters.

Value

An R vector.

Examples

## Not run:
summary(neighborhoodWalking())

## End(Not run)
timeSeries Aggregate time series fatality data from the Vestry report.

**Description**

Aggregate time series fatality data from the Vestry report.

**Usage**

```r
timeSeries(vestry = FALSE)
```

**Arguments**

vestry Logical. `TRUE` returns the data from the Vestry committee (Appendix B, p. 175). `FALSE` returns John Snow’s contribution to the report (p.117).

**Value**

A R list with two objects: "data" and "source" ("snow" or "vestry").

- date: Calendar date.
- day: Day of the week.
- deaths: Measure of fatality.
- fatal.attacks: Measure of fatality.

**Note**

The "snow" data appears on p. 117 of the report; the "vestry" data appear in Appendix B on p.175.

**See Also**

- `plot.time_series`
- `print.time_series`
- `vignette("time.series")`

**Examples**

```r
timeSeries(vestry = TRUE)
plot(timeSeries())
```
unitMeter  

*Convert nominal map distance to meters or yards.*

**Description**

A best guess estimate.

**Usage**

```r
unitMeter(x, distance.unit = "meter")
```

**Arguments**

- `x` Numeric. Nominal map distance.
- `distance.unit` Character. Unit of distance: "meter", "yard" or "native". "native" uses the map’s nominal scale. See vignette("roads") for information on conversion.

unstackFatalities  

*Unstack "stacks" in Snow’s cholera map.*

**Description**

Unstacks fatalities data by 1) assigning the coordinates of the base case to all cases in a stack and 2) setting the base case as an "address" and making the number of fatalities an attribute.

**Usage**

```r
unstackFatalities(multi.core = TRUE, compute = FALSE, dev.mode = FALSE)
```

**Arguments**

- `multi.core` Logical or Numeric. TRUE uses parallel::detectCores(). FALSE uses one, single core. With Numeric, you specify the number logical cores. See vignette("Parallelization") for details.
- `compute` Logical. TRUE computes data. FALSE uses pre-computed data.
- `dev.mode` Logical. Development mode uses parallel::parLapply().

**Value**

An R list that includes anchor.case, fatalities.address, fatalities.unstacked and ortho.proj.

**Note**

This function is computationally intensive. This function documents the code that generates anchor.case, fatalities.address, fatalities.unstacked and ortho.proj.
See Also

vignette("unstacking.fatalities")

Coordinates of Voronoi polygon vertices for original map.

Usage

voronoi.polygons

Format

A list of 13 data frames frames with 5 variables.

vertex vertex ID
x x-coordinate
y y-coordinate
lon longitude
lat latitude

Coordinates of Voronoi polygon vertices for Vestry Report map.

Description

Coordinates of Voronoi polygon vertices for Vestry Report map.

Usage

voronoi.polygons.vestry

Format

A list of 14 data frames frames with 5 variables.

vertex vertex ID
x x-coordinate
y y-coordinate
lon longitude
lat latitude
voronoiPolygons

Extract vertices of Delaunay triangles and Dirichelet (Voronoi) tiles.

Description

For construction and plotting of Delaunay and Voronoi polygons.

Usage

voronoiPolygons(sites, rw.data = NULL, rw = NULL, type = "tiles",
output = "vertices", latlong = FALSE)

Arguments

sites          Object. Data frame of sites to compute Delaunay triangulation and Dirichelet (Voronoi) tessellation with variables "x" and "y".
rw.data       Object. Data frame of secondary source of data to set the rectangular window or bounding box: observations, cases, etc. with variables "x" and "y".
rw            Numeric. Alternative to rw.data: vector of corners to define the rectangular window or bounding box: xmin, xmax, ymin, ymax.
type          Character. "tiles" (tessellation) or "triangles" (triangulation) vertices.
output         Character. "vertices" or "polygons". "vertices" re "polygons" will draw base R polygons() to an existing plot.
latlong        Logical. Use estimated longitude and latitude.

Value

An R list of data frames or base R graphics polygon()’s’.

Note

This function relies on the 'deldir' package.

Examples

snowMap()
voronoiPolygons(pumps, output = "polygons")

snowMap()
voronoiPolygons(pumps, roads, output = "polygons")

snowMap()
voronoiPolygons(pumps, roads, type = "triangles", output = "polygons")

vertices <- voronoiPolygons(pumps, roads)
snow.colors <- grDevices::adjustcolor(snowColors(), alpha.f = 1/3)
snowMap(add.cases = FALSE)
walkingPath

invisible(lapply(seq_along(vertices), function(i) {
    polygon(vertices[[i]], col = snow.colors[[i]])
}))

walkingPath

Description

Compute the shortest walking path between cases and/or pumps.

Usage

walkingPath(origin = 1, destination = NULL, type = "case-pump",
            observed = TRUE, weighted = TRUE, vestry = FALSE,
            distance.unit = "meter", time.unit = "second", walking.speed = 5,
            null.origin.landmark = FALSE)

Arguments

origin
Numeric or Character. Numeric ID of case or pump. Character landmark name.
destination
Numeric or Character. Numeric ID(s) of case(s) or pump(s). Exclusion is possible via negative selection (e.g., -7). Default is NULL: this returns closest pump or "anchor" case. Character landmark name (case insensitive).
type
Character "case-pump", "cases" or "pumps".
observed
Logical. Use observed or "simulated" expected data.
weighted
Logical. TRUE computes shortest path in terms of road length. FALSE computes shortest path in terms of nodes.
vestry
Logical. TRUE uses the 14 pumps from the Vestry report. FALSE uses the 13 in the original map.
distance.unit
Character. Unit of distance: "meter", "yard" or "native". "native" returns the map's native scale. "unit" is meaningful only when "weighted" is TRUE. See vignette("roads") for information on unit distances.
time.unit
Character. "hour", "minute", or "second".
walking.speed
Numeric. Walking speed in km/hr.
null.origin.landmark
Logical. Consider landmarks when origin = NULL and type = "case-pump".

Value

An R list with two elements: a character vector of path nodes and a data frame summary.
Note

The function uses a case's "address" (i.e., a stack's "anchor" case) to compute distance. Time is computed using distanceTime(). Adam and Eve Court, and Falconberg Court and Falconberg Mews, are disconnected from the larger road network; they form two isolated subgraphs. This has two consequences: first, only cases on Adam and Eve Court can reach pump 2 and those cases cannot reach any other pump; second, cases on Falconberg Court and Mews cannot reach any pump. Unreachable pumps will return distances of "Inf".

Examples

```r
## Not run:
# path from case 1 to nearest pump.
walkingPath(1)

# path from pump 1 to nearest case.
walkingPath(NULL, 1)

# path from case 1 to pump 6.
walkingPath(1, 6)

# exclude pump 7 from consideration.
walkingPath(1, -7)

# path from case 1 to case 6.
walkingPath(1, 6, type = "cases")

# path from pump 1 to pump 6.
walkingPath(1, 6, type = "pumps")

# for multiple cases.
lapply(1:3, walkingPath)

# path from case 1 to nearest pump.
plot(walkingPath(1))

# path from John Snow's residence to Broad Street pump.
plot(walkingPath("John Snow", 7))

## End(Not run)
```

winterTemperatures  

Description

Gareth Stedman Jones Appendix 2, Table 12, p.384.

Usage

winterTemperatures()
winterTemperatures

Examples

    plot(winterTemperatures(), "1859-6-1")
Index

* datasets
  anchor.case, 20
  border, 21
  fatalities, 24
  fatalities.address, 25
  fatalities.unstacked, 26
  frame.data, 27
  frame.sample, 27
  landmark.squares, 28
  landmarks, 29
 latlong.ortho.addr, 30
  latlong.ortho.pump, 30
  latlong.ortho.pump.vestry, 31
  ortho.proj, 43
  ortho.proj.pump, 44
  ortho.proj.pump.vestry, 45
  oxford.weather, 45
  plague.pit, 47
  pumps, 67
  pumps.vestry, 68
  rd.sample, 69
  rectangle.filter, 69
  regular.cases, 70
  road.segments, 70
  roads, 71
  sim.ortho.proj, 75
  sim.pump.case, 76
  sim.walking.distance, 76
  snow.neighborhood, 78
  voronoi.polygons, 89
  voronoi.polygons.vestry, 89

addCase, 5
addDelanay, 6
addEuclideanPath, 7
addFrame, 8
addIndexCase, 8
addKernelDensity, 9
addLandmarks, 10
addMilePosts, 11
addNeighborhoodCases, 12
addNeighborhoodEuclidean, 13
addNeighborhoodWalking, 14
addPlaguePit, 15
addPump, 16
addRoads, 16
addSnow, 17
addVoronoi, 17
addWalkingPath, 18
addWhitehead, 19
anchor.case, 20, 88
border, 21
caseDistance, 21
caseLocator, 20, 22, 25, 26
cholera.package, 4
euclideanPath, 23
fatalities, 24
fatalities.address, 25, 88
fatalities.unstacked, 25, 88
fixFatalities, 25, 26
frame.data, 27
frame.sample, 27
isolines, 28
landmark.squares, 28
landmarkData, 29, 30
landmarks, 29
latlong.ortho.addr, 30
latlong.ortho.pump, 30
latlong.ortho.pump.vestry, 31
latlongAddress, 32
latlongFatalities, 32
latlongLandmarks, 30, 33
latlongNearestPump, 33
latlongNeighborhoodData, 34
latlongNeighborhoodVoronoi, 35
INDEX

latlongNeighborhoodWalking, 35
latlongPumps, 36
latlongRoads, 36
latlongVoronoi, 37
latlongWalkingPath, 37
mapRange, 38
nearestPump, 38
neighborhoodData, 39
neighborhoodEuclidean, 40
neighborhoodVoronoi, 41
neighborhoodWalking, 42, 76
ortho.proj, 43, 88
ortho.proj.pump, 44, 65
ortho.proj.pump.vestry, 45, 65
oxford.weather, 45
oxfordWeather, 46
plague.pit, 47
plot.euclidean, 47
plot.euclidean_path, 48
plotlatlong_neighborhood_data, 49
plotlatlong_walking, 50
plotlatlong_walking_path, 50
plotlatlongNeighborhoodVoronoi, 49
plotlatitude_data, 51
plotoxfordWeather, 51
plotpovertyLondon, 52
plotprofile_perspective, 52
plottime_series, 53, 87
plotvoronoi, 54
plotwalking, 55
plotwalking_path, 56
plotwinterTemperatures, 57
povertyLondon, 57
print.euclidean, 58
print.euclidean_path, 58
print.iso, 59
printlatlong_walking_path, 60
printlatlongNeighborhoodVoronoi, 59
printtime_series, 60, 87
printvoronoi, 61
printwalking, 61
printwalking_path, 62
profile2D, 63
profile3D, 63
pumpCase, 64
pumpData, 31, 44, 45, 65, 67, 68
pumpFatalities, 66
pumpLocator, 65, 66, 68
pumps, 65, 67
pumps.vestry, 65, 68
rd.sample, 69
rectangle.filter, 69
regular.cases, 70, 77
road.segments, 70, 71, 72
roads, 71, 71
roadSegmentFix, 72
roadSegments, 71, 72
segmentHighlight, 73
segmentLength, 73
segmentLocator, 71, 74
sim.ortho.proj, 75, 77
sim.pump.case, 76
sim.walking.distance, 76, 78
simulateFatalities, 70, 75, 77
simulateWalkingDistance, 78
snow.neighborhood, 78
snowColors, 79
snowMap, 79
snowNeighborhood, 80
streetHighlight, 81
streetLength, 81
streetNameLocator, 25, 26, 71, 82
streetNames, 83
streetNumberLocator, 25, 26, 71, 83
subsetRoadsSamples, 84
summary.euclidean, 85
summary.voronoi, 85
summary.walking, 86
timeSeries, 53, 87
unitMeter, 88
unstackFatalities, 20, 25, 26, 30, 44, 88
voronoi.polygons, 89
voronoi.polygons.vestry, 89
voronoiPolygons, 90
walkingPath, 91
winterTemperatures, 92