

Package: cholera (via r-universe)

June 29, 2024

Type Package

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

Version 0.8.0.9359

Date 2024-06-28

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 curl, deldir (>= 1.0-2), elevatr, geosphere, HistData (>= 0.7-8), igraph, KernSmooth, pracma, RColorBrewer, rlang, sp, tanaka, terra, tools, threejs, TSP, viridisLite

Suggests ggplot2, knitr, rmarkdown

VignetteBuilder knitr

RoxygenNote 7.3.2

Roxygen list(old_usage = TRUE)

Encoding UTF-8

Language en-US

Repository <https://lindbrook.r-universe.dev>

RemoteUrl <https://github.com/lindbrook/cholera>

RemoteRef HEAD

RemoteSha 58bb512469eaf133f55afbd5f61b297e3d8a385e

Contents

cholera-package	5
addCase	6
addDelaunay	7
addEuclideanPath	7
addEuclideanPathB	8
addFrame	9
addIndexCase	10
addKernelDensity	10
addLandmarks	12
addLandmarkSquares	12
addMilePosts	13
addNeighborhoodCases	14
addNeighborhoodEuclidean	15
addNeighborhoodWalking	16
addPlaguePit	17
addPump	18
addRoads	18
addSnow	19
addVoronoi	19
addWalkingPath	20
addWalkingPathB	21
addWhitehead	22
anchor.case	23
border	24
caseDistance	24
caseLocator	25
euclideanPath	26
euclideanPathB	27
fatalities	28
fatalities.address	29
fatalities.unstacked	30
fixFatalities	30
frame.data	31
frame.sample	31
landmark.squares	32
landmark.squaresB	32
landmarkData	33
landmarks	33
landmarksB	34
latlong.ortho.addr	35
latlong.ortho.pump	35
latlong.ortho.pump.vestry	36
latlong.regular.cases	37
latlong.sim.ortho.proj	37
latlongEuclideanPath	38
latlongNearestPump	38

latlongNeighborhoodData	39
latlongNeighborhoodDataB	40
latlongNeighborhoodEuclidean	40
latlongNeighborhoodVoronoi	41
latlongNeighborhoodWalking	42
latlongStreetNameLocator	42
latlongVoronoi	43
mapRange	44
nearestPump	44
neighborhoodData	45
neighborhoodDataB	46
neighborhoodEuclidean	46
neighborhoodVoronoi	47
neighborhoodWalking	49
ortho.proj	50
ortho.proj.pump	50
ortho.proj.pump.vestry	51
oxford.weather	52
oxfordWeather	52
pearsonResiduals	53
plague.pit	53
plot.euclidean	54
plot.euclidean_path	55
plot.euclidean_path_B	55
plot.latlongEuclidean	56
plot.latlongVoronoi	57
plot.latlong_euclidean_path	57
plot.latlong_neighborhood_data	58
plot.latlong_walking	58
plot.neighborhood_data	59
plot.oxfordWeather	59
plot.povertyLondon	60
plot.profile_perspective	61
plot.time_series	61
plot.voronoi	62
plot.walking	63
plot.walking_path	64
plot.walking_path_B	65
plot.winterTemperatures	65
povertyLondon	66
print.euclidean	66
print.euclidean_path	67
print.euclidean_path_B	67
print.latlongVoronoi	68
print.time_series	68
print.voronoi	69
print.walking	69
print.walking_path	70

print.walking_path_B	71
profile2D	71
profile3D	72
pumpCase	73
pumpData	73
pumpFatalities	74
pumpLocator	75
pumps	76
pumps.vestry	76
rd.sample	77
rectangle.filter	78
regular.cases	78
road.segments	79
roads	80
roadSegments	81
segmentHighlight	81
segmentLength	82
segmentLocator	83
sim.ortho.proj	84
sim.pump.case	85
sim.walking.distance	85
simulateFatalities	86
snow.neighborhood	87
snowColors	87
snowMap	88
snowNeighborhood	89
streetHighlight	89
streetLength	90
streetNameLocator	90
streetNames	91
streetNumberLocator	92
summary.euclidean	93
summary.voronoi	93
summary.walking	94
tanakaContourPlot	95
timeSeries	95
unitMeter	96
unstackFatalities	96
voronoi.polygons	97
voronoi.polygons.vestry	98
voronoiPolygons	98
walkingPath	99
walkingPathB	101
winterTemperatures	102

cholera-package	<i>cholera</i>
-----------------	----------------

Description

Amend errors, augment data and aid analysis of John Snow's map of the 1854 London cholera outbreak.

Details

- Fixes two sets of errors in Dodson and Tobler's 1992 digitization of Snow's map: 1) three misplaced cases/fatalities and 2) one missing road segment (part of Clifford Street).
- "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.
- Overlay graphical elements and features like kernel density estimates, 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 $\geq 0.8.0$, 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")
```

Author(s)

Maintainer: lindbrook <lindbrook@gmail.com>

See Also

Useful links:

- <https://github.com/lindbrook/cholera>
- Report bugs at <https://github.com/lindbrook/cholera/issues>

addCase	<i>Add observed case(s) to plot.</i>
---------	--------------------------------------

Description

Add case(s), as "anchor", "fatality" or "orthogonal" as points or IDs, to a plot.

Usage

```
addCase(case = 1, latlong = FALSE, 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). "anchor" plots anchor cases; "fatality" plots all cases; "orthogonal" plot projected addresses.
latlong	Logical.
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

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

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

addDelaunay *Add Delaunay triangles.*

Description

Add Delaunay triangles.

Usage

```
addDelaunay(pump.select = NULL, vestry = FALSE, color = "black",
            line.type = "solid", line.width = 1, latlong = FALSE)
```

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.
line.width	Numeric. Width of cell edges: lwd.
latlong	Logical. Use estimated longitude and latitude.

Note

This function uses `deldir::deldir()`.

Examples

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

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

Description

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

Usage

```
addEuclideanPath(origin = 1, destination = NULL, type = "case-pump",
                 case.location = "address", case.set = "observed", vestry = FALSE,
                 distance.unit = "meter", time.unit = "second", walking.speed = 5,
                 mileposts = TRUE, milepost.unit = "distance", milepost.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".
case.location	Character. For case.set = "observed": "address" or "orthogonal". "address" is the x-y coordinate of a bar. "orthogonal" is the x-y coordinate of a stack's "anchor" case.
case.set	Character. "observed" or "expected".
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.
mileposts	Logical. Plot mile/time posts.
milepost.unit	Character. "distance" for mileposts; "time" for timeposts.
milepost.interval	Numeric. Sets interval between milepost.unit.
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().

addEuclideanPathB	<i>Add Euclidean path from case/landmark to nearest or selected pump. (prototype)</i>
-------------------	---------------------------------------------------------------------------------------

Description

Add Euclidean path from case/landmark to nearest or selected pump. (prototype)

Usage

```
addEuclideanPathB(origin = 1, destination = NULL, type = "case-pump",
  vestry = FALSE, latlong = FALSE, weighted = TRUE,
  distance.unit = "meter", time.unit = "second", walking.speed = 5,
  include.landmarks = TRUE, mileposts = TRUE, milepost.unit = "distance",
  milepost.interval = NULL, alpha.level = 1)
```


Arguments

origin	Numeric. Vector of origin(s) (numeric ID or character name landmark/pump).
destination	Numeric. Vector of destination(s) (numeric or landmark/pump name).
type	Character. Path case to pump. FALSE is all other combinations of cases, landmarks and pumps.
vestry	Logical. TRUE uses the 14 pumps from the map in the Vestry Report. FALSE uses the 13 pumps from the original map.
latlong	Logical.
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.
include.landmarks	Logical. Include landmarks as cases.
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].

addFrame	<i>Add map border to plot.</i>
----------	--------------------------------

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 *Highlight index case at 40 Broad Street.*

Description

Highlight index case at 40 Broad Street.

Usage

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

Arguments

latlong	Logical.
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.

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

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

Value

Add contours to a graphics plot.

Note

This function uses `KernSmooth::bkde2D()`.

Examples

```
## Not run:
snowMap()
addKernelDensity()

snowMap()
addKernelDensity("individual")

snowMap()
addKernelDensity(c(6, 8))

snowMap()
addKernelDensity(pump.select = c(6, 8))

## End(Not run)
```

addLandmarks *Add landmarks to plot.*

Description

Add landmarks to plot.

Usage

```
addLandmarks(text.size = 0.5, text.col = "black",  
             highlight.perimeter = TRUE, latlong = FALSE)
```

Arguments

text.size	Numeric. cex for text labels.
text.col	Character. col for text labels.
highlight.perimeter	Logical. Highlight Lion Brewery and Model Housing.
latlong	Logical. Use estimated longitude and latitude.

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.

Examples

```
snowMap(add.landmarks = FALSE)  
addLandmarks()
```

addLandmarkSquares *Add Golden and Soho Squares to plot.*

Description

Add Golden and Soho Squares to plot.

Usage

```
addLandmarkSquares(latlong = FALSE, text.size = 0.5, text.col = "black")
```

Arguments

latlong	Logical. Use estimated longitude and latitude.
text.size	Numeric. cex for text labels.
text.col	Character. col for text labels.

Value

Base R points and text.

Examples

```
snowMap()
addLandmarkSquares()
```

addMilePosts	<i>Add distance or time based "mileposts" to an observed walking neighborhood plot.</i>
--------------	-----------------------------------------------------------------------------------------

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 = "address",
  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. "address" uses fatalities; "orthogonal" uses ortho.proj.
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

```
## 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)
```

```
addNeighborhoodEuclidean
```

Add expected Euclidean pump neighborhoods.

Description

Add expected Euclidean pump neighborhoods.

Usage

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

Arguments

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

Value

R graphic elements.

Examples

```
## Not run:
streetNameLocator("marshall street", zoom = 0.5, highlight = FALSE,
  add.subtitle = FALSE)
addNeighborhoodEuclidean()

streetNameLocator("marshall street", zoom = 0.5, highlight = FALSE,
  add.subtitle = FALSE)
addNeighborhoodEuclidean(type = "area.points")

## End(Not run)
```

```
addNeighborhoodWalking
```

Add expected walking neighborhoods.

Description

Add expected walking neighborhoods.

Usage

```
addNeighborhoodWalking(pump.select = NULL, pump.subset = 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.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.
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.
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].

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

Examples

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

## End(Not run)
```

addPlaguePit	<i>Add plague pit (Marshall Street).</i>
--------------	------------------------------------------

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.
pos	Numeric. Position of 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	<i>Adds Snow's graphical annotation of the Broad Street pump walking neighborhood.</i>
---------	----------------------------------------------------------------------------------------

Description

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

Usage

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

Arguments

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

Examples

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

## End(Not run)
```

addVoronoi	<i>Add Voronoi cells.</i>
------------	---------------------------

Description

Add Voronoi cells.

Usage

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

Arguments

<code>pump.select</code>	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. <code>vestry</code> . Exclusion (negative selection) is possible (e.g., -6).
<code>vestry</code>	Logical. FALSE for original 13 pumps. TRUE for 14 pumps in Vestry Report.
<code>case.location</code>	Character. For observed = FALSE: "address" or "nominal". "nominal" is the x-y coordinates of regular cases.
<code>color</code>	Character. Color of cell edges.
<code>line.type</code>	Character. Type of line for cell edges: lty.
<code>line.width</code>	Numeric. Width of cell edges: lwd.
<code>latlong</code>	Logical. Use estimated longitude and latitude.

Note

This function uses `deldir::deldir()`.

Examples

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

<code>addWalkingPath</code>	<i>Add the shortest walking path between a selected cases or pumps.</i>
-----------------------------	-------------------------------------------------------------------------

Description

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

Usage

```
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

<code>origin</code>	Numeric or Integer. Numeric ID of case or pump.
<code>destination</code>	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).
<code>type</code>	Character "case-pump", "cases" or "pumps".
<code>observed</code>	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.
unit.posts	Character. "distance" for mileposts; "time" for timeposts.
unit.interval	Numeric. Sets interval between posts: for "distance", the default is 50 meters; for "time", the default is 60 seconds.
alpha.level	Numeric. Alpha level transparency for path: a value in [0, 1].

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 cholera::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. Arrow points represent mileposts or timeposts to the destination.

Examples

```
streetNameLocator("broad street", zoom = TRUE, highlight = FALSE,
  add.subtitle = FALSE)
addWalkingPath(447)
```

addWalkingPathB	<i>Add walking path from case/landmark to nearest or selected pump. (prototype)</i>
-----------------	-------------------------------------------------------------------------------------

Description

Add walking path from case/landmark to nearest or selected pump. (prototype)

Usage

```
addWalkingPathB(origin = 1, destination = NULL, type = "case-pump",
  vestry = FALSE, latlong = FALSE, weighted = TRUE,
  distance.unit = "meter", time.unit = "second", walking.speed = 5,
  include.landmarks = TRUE, mileposts = TRUE, milepost.unit = "distance",
  milepost.interval = NULL, alpha.level = 1, ...)
```

Arguments

origin	Numeric. Vector of origin(s) (numeric or case/landmark name).
destination	Numeric. Vector of destination(s) (numeric or landmark/pump name).
type	Character. Path case to pump. FALSE is all other combinations of cases, landmarks and pumps.
vestry	Logical. TRUE uses the 14 pumps from the map in the Vestry Report. FALSE uses the 13 pumps from the original map.
latlong	Logical.
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.
include.landmarks	Logical. Include landmarks as cases.
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.

 addWhitehead

Add Rev. Henry Whitehead's Broad Street pump neighborhood.

Description

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

Usage

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

Arguments

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

Value

Adds a circle (polygon) to a graphics plot.

Examples

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

anchor.case	<i>Anchor or base case of each stack of fatalities.</i>
-------------	---------------------------------------------------------

Description

Data frame that links a fatality to its stack, a stack's base case. For use with [caseLocator](#).

Usage

```
anchor.case
```

Format

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

Note

[unstackFatalities](#) documents the code for these data.

border	<i>Numeric IDs of line segments that create the map's border frame.</i>
--------	-------------------------------------------------------------------------

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	<i>Compute distance between case fatalities (meters).</i>
--------------	-----------------------------------------------------------

Description

Compute distance between case fatalities (meters).

Usage

```
caseDistance(a = 19, b = 263, latlong = FALSE)
```

Arguments

a	Numeric. Case ID.
b	Numeric. Case ID.
latlong	Logical.

caseLocator	<i>Locate case by numerical ID.</i>
-------------	-------------------------------------

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 ≥ 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",
  case.location = "address", case.set = "observed",
  landmark.cases = TRUE, vestry = FALSE, distance.unit = "meter",
  time.unit = "second", walking.speed = 5)
```

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, which returns the closest pump, "anchor" case or landmark.
type	Character "case-pump", "cases" or "pumps".
case.location	Character. "address" or "orthogonal". For case.set = "observed": "address" uses fatalities and "orthogonal" uses ortho.proj. For case.set = "expected": "address" uses regular.cases and "orthogonal" uses sim.ortho.proj.
case.set	Character. "observed" or "expected".
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))

```

euclideanPathB	<i>Compute Euclidean path coordinates from case/landmark to nearest/selected pump.</i>
----------------	----------------------------------------------------------------------------------------

Description

Compute Euclidean path coordinates from case/landmark to nearest/selected pump.

Usage

```

euclideanPathB(origin = 1, destination = NULL, type = "case-pump",
  vestry = FALSE, latlong = FALSE, weighted = TRUE,
  distance.unit = "meter", time.unit = "second", walking.speed = 5,
  include.landmarks = TRUE)

```

Arguments

origin	Numeric. Vector of origin(s) (numeric ID or character name landmark/pump).
destination	Numeric. Vector of destination(s) (numeric or landmark/pump name).
type	Character. Path case to pump. FALSE is all other combinations of cases, landmarks and pumps.
vestry	Logical. TRUE uses the 14 pumps from the map in the Vestry Report. FALSE uses the 13 pumps from the original map.

latlong	Logical.
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.
include.landmarks	Logical. Include landmarks as cases.

fatalities

Amended Dodson and Tobler's cholera data.

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
```

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	<i>"Unstacked" amended cholera data with address as unit of observation.</i>
--------------------	------------------------------------------------------------------------------

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.

`case` numerical case ID

`x` x-coordinate

`y` y-coordinate

`lon` longitude

`lat` latitude

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()
```

Value

An R data frame.

See Also

`vignette("duplicate.missing.cases")`

frame.data	<i>Map frame data c("x", "y") and c("lon", "lat").</i>
------------	--------------------------------------------------------

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	<i>Partitioned map frame points (segment endpoints).</i>
--------------	----------------------------------------------------------

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.

```
frame.sample cholera::roads$id
```

landmark.squares	<i>Centers of city squares.</i>
------------------	---------------------------------

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

landmark.squaresB	<i>Centers of city squares.</i>
-------------------	---------------------------------

Description

Centers of city squares.

Usage

landmark.squaresB

Format

A data frame with 2 observations and 6 variables that records the position of landmark square labels.

case numeric case ID
 x x-coordinate
 y y-coordinate
 lon longitude
 lat latitude
 name square name

landmarkData	<i>Landmark data.</i>
--------------	-----------------------

Description

Nominal and orthogonal coordinates

Usage

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

Arguments

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

landmarks	<i>Orthogonal projection of landmarks onto road network.</i>
-----------	--------------------------------------------------------------

Description

Orthogonal projection of landmarks onto road network.

Usage

```
landmarks
```

Format

A data frame with 12 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
lon.proj	orthogonal longitude
lat.proj	orthogonal latitude

Note

[landmarkData](#) document the code for these data.

landmarksB	<i>Landmark coordinates.</i>
------------	------------------------------

Description

Landmark coordinates.

Usage

landmarksB

Format

A data frame of landmark coordinates with 20 observations and 11 variables.

case numeric case ID

road.segment "address" road segment

x nominal x-coordinate

y nominal y-coordinate

x.lab label x-coordinate

y.lab label y-coordinate

lon longitude

lat latitude

lon.lab label longitude

lat.lab label latitude

name landmark name

Note

[landmarkData](#) document the code for these data.

latlong.ortho.addr	<i>Orthogonal projection of observed address (latlong) cases onto road network.</i>
--------------------	-------------------------------------------------------------------------------------

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	<i>Orthogonal projection of 13 original pumps (latlong).</i>
--------------------	--------------------------------------------------------------

Description

Orthogonal projection of 13 original pumps (latlong).

Usage

latlong.ortho.pump

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
id numeric ID
lon longitude
lat latitude
```

Note

[pumpData](#) documents the code for these data.

```
latlong.ortho.pump.vestry
```

Orthogonal projection of the 14 pumps from the Vestry Report (lat-long).

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
id numeric ID
lon longitude
lat latitude
```

Note

[pumpData](#) documents the code for these data.

```
latlong.regular.cases "Expected" cases (latlong).
```

Description

The result of using `sp::spsample()` and `sp::Polygon()` to generate 20,000 regularly spaced simulated Cartesian/geodesic cases within the map's borders.

Usage

```
latlong.regular.cases
```

Format

A data frame with 4 variables that records the position of 20,000 "expected" cases fitted by `sp::spsample()`.

x x-coordinate

y y-coordinate

lon longitude

lat latitude

```
latlong.sim.ortho.proj
```

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

Description

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

Usage

```
latlong.sim.ortho.proj
```

Format

A data frame with 8 variables that records the "address" of 20,000 regularly spaced simulated Cartesian/geodesic cases regularly spaced across map.

case numeric case ID

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")

lon longitude

lat latitude

latlongEuclideanPath *Plot Euclidean path to nearest pump (prototype).*

Description

Plot Euclidean path to nearest pump (prototype).

Usage

```
latlongEuclideanPath(case = 1, destination = NULL, vestry = FALSE,
  distance.unit = "meter", time.unit = "second", walking.speed = 5,
  case.location = "nominal", 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.
distance.unit	Character. Unit of distance: "meter" or "yard".
time.unit	Character. "hour", "minute", or "second".
walking.speed	Numeric. Walking speed in km/hr.
case.location	Character. For observed = FALSE: "address" or "nominal". "nominal" is the x-y coordinates of regular.cases.
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.

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

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

Arguments

<code>pump.select</code>	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.
<code>metric</code>	Character. "euclidean" or "walking".
<code>vestry</code>	Logical. TRUE uses the 14 pumps from the Vestry Report. FALSE uses the 13 in the original map.
<code>case.set</code>	Character. "observed", "expected" or "snow".
<code>weighted</code>	Logical. TRUE computes shortest path in terms of road length. FALSE computes shortest path in terms of the number of nodes.
<code>time.unit</code>	Character. "hour", "minute", or "second".
<code>walking.speed</code>	Numeric. Walking speed in km/hr.
<code>multi.core</code>	Logical or Numeric. TRUE uses <code>parallel::detectCores()</code> . FALSE uses one, single core. You can also specify the number logical cores. See <code>vignette("Parallelization")</code> 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

```
latlongNeighborhoodData(vestry = FALSE, case.set = "observed",
  embed.addr = TRUE, multi.core = TRUE)
```

Arguments

<code>vestry</code>	Logical.
<code>case.set</code>	Character. "observed" or "expected".
<code>embed.addr</code>	Logical. Embed case address into graph network.
<code>multi.core</code>	Logical or Numeric. TRUE uses <code>parallel::detectCores()</code> . FALSE uses one, single core. You can also specify the number logical cores. See <code>vignette("Parallelization")</code> for details.

latlongNeighborhoodDataB

Compute network graph of roads, cases and pumps (prototype).

Description

Assembles cases, pumps and road into a network graph.

Usage

```
latlongNeighborhoodDataB(vestry = FALSE, case.set = "observed",
  embed.addr = TRUE, embed.landmarks = TRUE, embed.pumps = TRUE,
  multi.core = TRUE)
```

Arguments

vestry	Logical. Use Vestry Report pump data.
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.
embed.addr	Logical. Embed cases into road network.
embed.landmarks	Logical. Embed landmarks into road network.
embed.pumps	Logical. Embed pumps into road network.
multi.core	Logical or Numeric. TRUE uses <code>parallel::detectCores()</code> . FALSE uses one, single core. You can also specify the number logical cores. See <code>vignette("Parallelization")</code> for details.

Value

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

latlongNeighborhoodEuclidean

Compute Euclidean pump neighborhoods (lat-long prototype).

Description

Plots star graph from pump to its cases.

Usage

```
latlongNeighborhoodEuclidean(pump.select = NULL, vestry = FALSE,
  case.location = "address", case.set = "observed",
  pump.location = "address")
```


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 "orthogonal". "address" uses the longitude and latitude of fatalities.address. "orthogonal" uses the longitude and latitude of latlong.ortho.address.
case.set	Character. "observed" or "expected".
pump.location	Character. "address" or "orthogonal". "address" uses the longitude and latitude coordinates of pumps or pumps.vestry. "orthogonal" uses the longitude and latitude coordinates of latlong.ortho.pump or latlong.ortho.pump.vestry.

latlongNeighborhoodVoronoi

Compute Voronoi pump neighborhoods (lat-long prototype).

Description

Group cases into neighborhoods using Voronoi tessellation.

Usage

```
latlongNeighborhoodVoronoi(pump.select = NULL, vestry = FALSE,
  case.location = "address", pump.location = "address")
```

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 "orthogonal". "address" uses the longitude and latitude of fatalities.address. "orthogonal" uses the longitude and latitude of latlong.ortho.address.
pump.location	Character. "address" or "orthogonal". "address" uses the longitude and latitude coordinates of pumps or pumps.vestry. "orthogonal" uses the longitude and latitude coordinates of latlong.ortho.pump or latlong.ortho.pump.vestry.

latlongNeighborhoodWalking

Compute walking path pump neighborhoods.

Description

Group cases into neighborhoods based on walking distance.

Usage

```
latlongNeighborhoodWalking(pump.select = NULL, vestry = FALSE,
  case.set = "observed", weighted = TRUE, 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.
case.set	Character. "observed" or "expected".
weighted	Logical. TRUE computes shortest path weighted by road length. FALSE computes shortest path in terms of the number of nodes.
multi.core	Logical or Numeric. TRUE uses <code>parallel::detectCores()</code> . FALSE uses one, single core. You can also specify the number logical cores. See <code>vignette("Parallelization")</code> for details.

latlongStreetNameLocator

Locate road by name.

Description

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

Usage

```
latlongStreetNameLocator(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.
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.

Value

A base R graphics plot.

latlongVoronoi	<i>Compute Georeferenced Latitude and Longitude of vertices of Voronoi polygons.</i>
----------------	--------------------------------------------------------------------------------------

Description

Compute Georeferenced Latitude and Longitude of vertices of Voronoi polygons.

Usage

```
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

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

mapRange	<i>Compute xlim and ylim of Snow's map.</i>
----------	---------------------------------------------

Description

Compute xlim and ylim of Snow's map.

Usage

```
mapRange(latlong = FALSE)
```

Arguments

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

nearestPump	<i>Compute shortest distances or paths to selected pumps.</i>
-------------	---------------------------------------------------------------

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)
```

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 <code>parallel::detectCores()</code> . FALSE uses one, single core. You can also specify the number logical cores. See <code>vignette("Parallelization")</code> for details.
dev.mode	Logical. Development mode uses <code>parallel::parLapply()</code> .

Value

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

Note

Time is computed using `distanceTime()`.

neighborhoodData	<i>Compute network graph of roads, cases and pumps.</i>
------------------	---------------------------------------------------------

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", "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.

Value

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

neighborhoodDataB *Compute network graph of roads, cases and pumps (prototype).*

Description

Assembles cases, pumps and road into a network graph.

Usage

```
neighborhoodDataB(vestry = FALSE, case.set = "observed",
  embed.addr = TRUE, embed.landmarks = TRUE, embed.pumps = TRUE,
  latlong = FALSE)
```

Arguments

vestry	Logical. Use Vestry Report pump data.
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.
embed.addr	Logical. Embed cases into road network.
embed.landmarks	Logical. Embed landmarks into road network.
embed.pumps	Logical. Embed pumps into road network.
latlong	Logical. Use estimated longitude and latitude.

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 = "address", case.set = "observed", multi.core = TRUE,
  dev.mode = FALSE)
```

Arguments

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

Value

An R vector.

Examples

```
## 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

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

Arguments

<code>pump.select</code>	Numeric. Vector of numeric pump IDs to define pump neighborhoods (i.e., the "population"). Negative selection possible. NULL selects all pumps.
<code>vestry</code>	Logical. TRUE uses the 14 pumps from the Vestry report. FALSE uses the 13 in the original map.
<code>case.location</code>	Character. "address" or "anchor". "address" uses the x-y coordinates of <code>ortho.proj</code> . "anchor" uses the x-y coordinates of <code>fatalities.address</code> .
<code>pump.location</code>	Character. "address" or "nominal". "address" uses the x-y coordinates of <code>ortho.proj.pump</code> or <code>ortho.proj.pump.vestry</code> . "nominal" uses the x-y coordinates of <code>pumps</code> or <code>pumps.vestry</code> .
<code>polygon.vertices</code>	Logical. TRUE returns a list of x-y coordinates of the vertices of Voronoi cells. Useful for <code>sp::point.in.polygon()</code> as used in <code>print.voronoi()</code> 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.
- `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.
multi.core	Logical or Numeric. TRUE uses <code>parallel::detectCores()</code> . FALSE uses one, single core. You can also specify the number logical cores. See <code>vignette("Parallelization")</code> for details.
dev.mode	Logical. Development mode uses <code>parallel::parLapply()</code> .

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

```
## 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

Note

[unstackFatalities](#) documents the code for these data.

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

Note

[pumpData](#) documents the code for these data.

ortho.proj.pump.vestry

Orthogonal projection of the 14 pumps from the Vestry Report.

Description

Orthogonal projection of the 14 pumps from the Vestry Report.

Usage

ortho.proj.pump.vestry

Format

A data frame with 6 variables that records the position of the orthogonal projection of the 14 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

Note

[pumpData](#) documents the code for these data.

 oxford.weather

Oxford monthly weather data, January 1853 - February 2024.

Description

Extract from UK Met Office (<https://www.metoffice.gov.uk/pub/data/weather/uk/climate/stationdata/oxforddata.txt>):
 Lat 51.761 Lon -1.262, 63 metres amsl. Approximate 90 km (55 miles) northwest of Soho.

Usage

oxford.weather

Format

A data frame with 9 variables and 2054 observations.

year Year

month Month

tmax Mean daily maximum temperature Celsius

tmin Mean daily minimum temperature Celsius

af Days of air frost

rain Total rainfall (mm)

sun Total sunshine duration hours

prov Provisional data

date Date

 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

oxfordWeather()

Value

An R data frame.

Note

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

pearsonResiduals *Compute Pearson Residuals (prototype)*

Description

Compute Pearson Residuals (prototype)

Usage

```
pearsonResiduals(x)
```

Arguments

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

Value

An R vector.

Examples

```
## Not run:
pearsonResiduals(neighborhoodEuclidean())
pearsonResiduals(neighborhoodVoronoi())
pearsonResiduals(neighborhoodWalking())

## End(Not run)
```

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, ...)
```

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.
...	Additional plotting parameters.

Value

A base R plot.

Note

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

Examples

```
## 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

```
## 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

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

plot.euclidean_path_B *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

```
## S3 method for class 'euclidean_path_B'
plot(x, zoom = TRUE, long.title = TRUE,
     mileposts = TRUE, milepost.unit = "distance", milepost.interval = NULL,
     alpha.level = 1, ...)
```

Arguments

<code>x</code>	An object of class "euclidean_path_B" created by <code>euclideanPathB()</code> .
<code>zoom</code>	Logical or Numeric. A numeric value ≥ 0 controls the degree of zoom. The default is 0.5.
<code>long.title</code>	Logical. Tile with names.
<code>mileposts</code>	Logical. Plot mile/time posts.
<code>milepost.unit</code>	Character. "distance" or "time".
<code>milepost.interval</code>	Numeric. Mile post interval unit of distance (yard or meter) or unit of time (seconds).
<code>alpha.level</code>	Numeric. Alpha level transparency for path: a value in $[0, 1]$.
<code>...</code>	Additional plotting parameters.

Value

A base R plot.

`plot.latlongEuclidean` *Plot method for latlongNeighborhoodEuclidean()*

Description

Plot method for `latlongNeighborhoodEuclidean()`

Usage

```
## S3 method for class 'latlongEuclidean'
plot(x, type = "star", ...)
```

Arguments

<code>x</code>	Object.
<code>type</code>	Character. "star", "area.points" or "area.polygons". "area" flavors only valid when <code>case.set = "expected"</code> .
<code>...</code>	Additional plotting parameters.

plot.latlongVoronoi *Plot method for latlongNeighborhoodVoronoi()*

Description

Plot method for latlongNeighborhoodVoronoi()

Usage

```
## S3 method for class 'latlongVoronoi'
plot(x, add.pumps = TRUE,
     delaunay.voronoi = "voronoi", euclidean.paths = FALSE, ...)
```

Arguments

x	Object.
add.pumps	Logical.
delaunay.voronoi	Character "delaunay", "voronoi", or "both".
euclidean.paths	Logical.
...	Additional plotting parameters.

plot.latlong_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

```
## S3 method for class 'latlong_euclidean_path'
plot(x, zoom = TRUE, mileposts = TRUE,
     milepost.unit = "distance", milepost.interval = NULL, ...)
```

Arguments

x	An object of class "latlong_euclidean_path" created by latlongEuclideanPath().
zoom	Logical.
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).
 ... Additional plotting parameters.

plot.latlong_neighborhood_data
Plot method for latlongNeighborhoodData().

Description

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

Usage

```
## 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.

plot.latlong_walking *Plot method for latlongNeighborhoodWalking().*

Description

Plot method for latlongNeighborhoodWalking().

Usage

```
## S3 method for class 'latlong_walking'
plot(x, type = "area.points", ...)
```

Arguments

x An object of class "latlong_walking" created by latlongNeighborhoodWalking().
 type Character. "area.points", "area.polygons" or "streets". For latlongNeighborhoodWalking(case.set = "expected").
 ... Additional plotting parameters.

Value

A base R plot.

plot.neighborhood_data

Plot method for neighborhoodData().

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

Plot method for oxfordWeather().

Description

Plot method for oxfordWeather().

Usage

```
## S3 method for class 'oxfordWeather'
plot(x, statistic = "temperature",
     month = "september", end.year = NULL, unit.observation = "month", ...)
```

Arguments

x	object.
statistic	Character.
month	Character. "august" or "september".
end.year	Numeric.
unit.observation	Character. "day" or "month".
...	Additional plotting parameters.

Value

A base R plot.

plot.povertyLondon *Plot method for povertyLondon().*

Description

Plot method for povertyLondon().

Usage

```
## S3 method for class 'povertyLondon'  
plot(x, district = c("City", "Westminster",  
  "Marylebone", "St. Giles"), ...)
```

Arguments

x	object.
district	Character. Selected district(s).
...	Additional plotting parameters.

```
plot.profile_perspective
    Plot method for profilePerspective().
```

Description

Plot method for profilePerspective().

Usage

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

Arguments

x An object of class "profile" created by profilePerspective().
 ... Additional plotting parameters.

```
plot.time_series        Plot aggregate time series data from Vestry report.
```

Description

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

Usage

```
## 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**

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

plot.voronoi	<i>Plot Voronoi neighborhoods.</i>
--------------	------------------------------------

Description

Plot Voronoi neighborhoods.

Usage

```
## S3 method for class 'voronoi'
plot(x, delaunay.voronoi = "voronoi",
     euclidean.paths = FALSE, ...)
```

Arguments

x	An object of class "voronoi" created by neighborhoodVoronoi().
delaunay.voronoi	Character "delaunay", "voronoi", or "both".
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	<i>Plot method for neighborhoodWalking().</i>
--------------	-----------------------------------------------

Description

Plot method for neighborhoodWalking().

Usage

```
## S3 method for class 'walking'  
plot(x, type = "area.points",  
     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".
tsp.method	Character. Traveling salesperson problem algorithm.
...	Additional plotting parameters.

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. type = "roads" inspired by Shiode et. al. (2015).

Examples

```
## Not run:  
plot(neighborhoodWalking())  
plot(neighborhoodWalking(case.set = "expected"))  
plot(neighborhoodWalking(case.set = "expected"), type = "roads")  
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

```
## S3 method for class 'walking_path'
plot(x, zoom = 0.5, stacked = TRUE,
     mileposts = 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.
mileposts	Logical. Plot mile/time posts.
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

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

## End(Not run)
```

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

Description

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

Usage

```
## S3 method for class 'walking_path_B'
plot(x, zoom = TRUE, long.title = TRUE,
     mileposts = TRUE, milepost.unit = "distance", milepost.interval = NULL,
     alpha.level = 1, ...)
```

Arguments

x	An object of class "walking_path_B" created by walkingPathB().
zoom	Logical or Numeric. A numeric value ≥ 0 controls the degree of zoom. A value of 1 equals zoom = TRUE.
long.title	Logical. Tile with names.
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.

Value

A base R plot.

plot.winterTemperatures
Plot method for winterTemperatures().

Description

Plot method for winterTemperatures().

Usage

```
## 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

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

povertyLondon *Poverty and Born in London.*

Description

Gareth Stedman Jones, p. 132. Census and Charles Booth Data, 1881.

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.euclidean_path_B
Print method for walkingPathB().

Description

Summary output.

Usage

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

Arguments

x An object of class "euclidean_path_B" created by euclideanPathB().
 ... Additional parameters.

Value

An R data frame.

print.latlongVoronoi *Print method for latlongNeighborhoodVoronoi().*

Description

Parameter values for latlongNeighborhoodVoronoi().

Usage

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

Arguments

x An object of class "latlongVoronoi" created by latlongNeighborhoodVoronoi().
 ... Additional arguments.

Value

A list of argument values.

print.time_series *Print summary data for timeSeries().*

Description

Return summary results.

Usage

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

Arguments

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

Value

An R data frame.

Examples

```
timeSeries()  
print(timeSeries())
```

print.voronoi	<i>Print method for neighborhoodVoronoi().</i>
---------------	------------------------------------------------

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	<i>Print method for neighborhoodWalking().</i>
---------------	------------------------------------------------

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

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

```
print.walking_path Print method for walkingPath().
```

Description

Summary output.

Usage

```
## 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

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

```
print.walking_path_B Print method for walkingPathB().
```

Description

Summary output.

Usage

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

Arguments

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

Value

An R data frame.

```
profile2D                    2D Profile .
```

Description

2D Profile .

Usage

```
profile2D(angle = 0, pump = 7, vestry = FALSE, graphics = "base",
  multi.core = FALSE)
```

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.
 graphics 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, graphics = "ggplot2")

## End(Not run)
```

profile3D

3D Profile.

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 <code>parallel::detectCores()</code> . FALSE uses one, single core. You can also specify the number logical cores. See <code>vignette("Parallelization")</code> for details.

Examples

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

## End(Not run)
```

pumpCase	<i>Extract numeric case IDs by pump neighborhood.</i>
----------	-------------------------------------------------------

Description

Extract numeric case IDs by pump neighborhood.

Usage

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

Arguments

x	An object created by <code>neighborhoodEuclidean()</code> , <code>neighborhoodVoronoi()</code> or <code>neighborhoodWalking()</code> .
case	Character. "address" or "fatality"

Value

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

Examples

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

pumpData	<i>Compute pump coordinates.</i>
----------	----------------------------------

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

```
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 <code>parallel::detectCores()</code> . FALSE uses one, single core. With Numeric, you specify the number logical cores (rounds with <code>as.integer()</code>). See <code>vignette("Parallelization")</code> 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	<i>Compute fatalities by pump.</i>
----------------	------------------------------------

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 <code>parallel::detectCores()</code> . FALSE uses one, single core. You can also specify the number logical cores. See <code>vignette("Parallelization")</code> for details.

Examples

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

## End(Not run)
```

pumpLocator	<i>Locate water pump by numerical ID.</i>
-------------	-------------------------------------------

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 ≥ 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

```
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

y y-coordinate

lon longitude

lat latitude

Note

[pumpData](#) documents the code for these data.

See Also

[pumpLocator](#)

pumps.vestry

Vestry report pump data.

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	<i>Rectangular filter data.</i>
------------------	---------------------------------

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	<i>"Expected" cases.</i>
---------------	--------------------------

Description

The result of using `sp::spsample()` and `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 `sp::spsample()`.

x x-coordinate

y y-coordinate

Note

[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 658 observations and 7 variables. The data describe the straight line segments used to recreate the roads on Snow's map.

`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

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 1243 observations and 6 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](#)

roadSegments	<i>Reshape 'roads' data frame into 'road.segments' data frame.</i>
--------------	--------------------------------------------------------------------

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	<i>Highlight segment by ID.</i>
------------------	---------------------------------

Description

Highlight segment by ID.

Usage

```
segmentHighlight(id, highlight = TRUE, col = "red", rotate.label = FALSE,
  latlong = FALSE)
```

Arguments

id Character. Segment ID: 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.

rotate.label Logical. Rotate segment ID label.

latlong Logical. Use estimated longitude and latitude.

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	<i>Compute length of road segment.</i>
---------------	----------------------------------------

Description

Compute length of road segment.

Usage

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

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. latlong = TRUE only returns meters.
latlong	Logical.

Value

An R vector of length one.

Examples

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

segmentLocator	<i>Plot/Locate road segment by ID.</i>
----------------	----------------------------------------

Description

Highlight selected road segment(s) and cases.

Usage

```
segmentLocator(segment.id = "216-1", zoom = TRUE, latlong = FALSE,
  cases = "address", token = "id", vestry = FALSE, add.pump = TRUE,
  add.title = TRUE, add.subtitle = TRUE, highlight = TRUE,
  distance.unit = "meter", time.unit = "second", walking.speed = 5,
  cex.text = 0.67)
```

Arguments

segment.id	Character. A vector of segment IDs. See Note.
zoom	Logical or Numeric. Positive value zoom in. Negative values zoom out.
latlong	Logical. Longitude and latitude coordiantes
cases	Character. Cases to plot: NULL, "address" or "fatality".
token	Character. Cases as "id" or "point".
vestry	Logical. TRUE uses the 14 pumps from the Vestry report. FALSE uses the 13 in the original map.
add.pump	Logical. Include pumps.
add.title	Logical. Include title.
add.subtitle	Logical. Include subtitle.
highlight	Logical. Highlight selected segment(s) and 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.
cex.text	Numeric.

Value

A base R graphics plot.

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

```
segmentLocator("216-1")
segmentLocator("216-1", zoom = -10)
segmentLocator("216-1", latlong = TRUE, zoom = -10)
segmentLocator("216-1", distance.unit = "yard")
segmentLocator("216-1", zoom = FALSE)
```

sim.ortho.proj	<i>Road "address" of simulated (i.e., "expected") cases.</i>
----------------	--------------------------------------------------------------

Description

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

Usage

```
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.

sim.pump.case	<i>List of "simulated" fatalities grouped by walking-distance pump neighborhood.</i>
---------------	--------------------------------------------------------------------------------------

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

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

sim.walking.distance	<i>Walking distance to Broad Street Pump (#7).</i>
----------------------	----------------------------------------------------

Description

Walking distance to Broad Street Pump (#7).

Usage

```
sim.walking.distance
```

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

simulateFatalities *Generate simulated fatalities.*

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

```
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 <code>parallel::detectCores()</code> . FALSE uses one, single core. With Numeric, you specify the number logical cores (rounds with <code>as.integer()</code>). See <code>vignette("Parallelization")</code> for details.
simulated.obs	Numeric. Number of sample cases.
dev.mode	Logical. Development mode uses <code>parallel::parLapply()</code> .

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.

snow.neighborhood	<i>Snow neighborhood fatalities.</i>
-------------------	--------------------------------------

Description

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

Usage

```
snow.neighborhood
```

Format

A vector with 384 observations.

```
snow.neighborhood numeric case ID
```

snowColors	<i>Create a set of colors for pump neighborhoods.</i>
------------	-------------------------------------------------------

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.axes_box = 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

<code>vestry</code>	Logical. TRUE uses the 14 pumps from the map in the Vestry Report. FALSE uses the 13 pumps from the original map.
<code>stacked</code>	Logical. Use stacked fatalities.
<code>add.axes_box</code>	Logical. Add plot axes and plot box.
<code>add.cases</code>	Logical. Add observed cases.
<code>add.landmarks</code>	Logical. Add landmarks.
<code>add.pumps</code>	Logical. Add pumps.
<code>add.roads</code>	Logical. Add roads.
<code>add.frame</code>	Logical. Add map frame.
<code>main</code>	Character. Title of graph.
<code>case.col</code>	Character. Color of fatalities.
<code>case.pch</code>	Character. Color of fatalities.
<code>latlong</code>	Logical. Use estimated longitude and latitude.
<code>...</code>	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	<i>Plotting data for Snow's graphical annotation of the Broad Street pump neighborhood.</i>
------------------	---------------------------------------------------------------------------------------------

Description

Computes "missing" and split road segments data, and area plot data.

Usage

```
snowNeighborhood(latlong = FALSE)
```

Arguments

latlong Logical. Use estimated longitude and latitude.

Value

An R list of edge IDs and simulated case IDs.

streetHighlight	<i>Highlight road by name.</i>
-----------------	--------------------------------

Description

Highlight road by name.

Usage

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

Arguments

road.name Character vector. The function tries to correct for case and remove extra spaces (includes "Map Frame").

col Character. Highlight color.

lwd Numeric. Line width.

latlong Logical. Use estimated longitude and latitude.

Value

A base R graphics segment(s).

Examples

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

streetLength	<i>Compute length of selected street.</i>
--------------	-------------------------------------------

Description

Compute length of selected street.

Usage

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

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.
latlong	Logical. Use estimated longitude and latitude.

Value

An R vector of length one.

Examples

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

streetNameLocator	<i>Locate road by name.</i>
-------------------	-----------------------------

Description

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

Usage

```
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 <code>streetNameLocator()</code> 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 <code>vignette("roads")</code> for information on conversion.
time.unit	Character. "hour", "minute", or "second".
walking.speed	Numeric. Walking speed in km/hr.

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	<i>Street names (alphabetized).</i>
-------------	-------------------------------------

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

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

Value

A base R graphics plot.

Examples

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

summary.euclidean *Summary method for neighborhoodEuclidean().*

Description

Return computed counts for Euclidean neighborhoods.

Usage

```
## 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

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

summary.voronoi *Summary method for neighborhoodVoronoi().*

Description

Return computed counts for Voronoi neighborhoods.

Usage

```
## 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.voronoi()

Examples

```
summary(neighborhoodVoronoi())
```

summary.walking	<i>Summary method for neighborhoodWalking().</i>
-----------------	--------------------------------------------------

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)
```

tanakaContourPlot	<i>Tanaka contour plot.</i>
-------------------	-----------------------------

Description

Soho elevation data.

Usage

```
tanakaContourPlot(add = FALSE)
```

Arguments

add	Logical. Add to existing plot.
-----	--------------------------------

timeSeries	<i>Aggregate time series fatality data from the Vestry report.</i>
------------	--------------------------------------------------------------------

Description

Aggregate time series fatality data from the Vestry report.

Usage

```
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

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

unitMeter	<i>Convert nominal map distance to meters or yards.</i>
-----------	---------------------------------------------------------

Description

A best guess estimate.

Usage

```
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	<i>Unstack "stacks" in Snow's cholera map.</i>
-------------------	------------------------------------------------

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

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

Arguments

multi.core	Logical or Numeric. TRUE uses <code>parallel::detectCores()</code> . 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 <code>parallel::parLapply()</code> .

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")`

<code>voronoi.polygons</code>	<i>Coordinates of Voronoi polygon vertices for original map.</i>
-------------------------------	------------------------------------------------------------------

Description

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

```
voronoi.polygons.vestry
```

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

```

walkingPath

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

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

```
## 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)

```

walkingPathB	<i>Compute walking path from case/landmark to nearest or selected pump.</i>
--------------	-----------------------------------------------------------------------------

Description

Compute walking path from case/landmark to nearest or selected pump.

Usage

```

walkingPathB(origin = 1, destination = NULL, type = "case-pump",
  vestry = FALSE, latlong = FALSE, weighted = TRUE,
  distance.unit = "meter", time.unit = "second", walking.speed = 5,
  include.landmarks = TRUE)

```

Arguments

origin	Numeric. Vector of origin(s) (numeric or case/landmark name).
destination	Numeric. Vector of destination(s) (numeric or landmark/pump name).
type	Character. Path case to pump. FALSE is all other combinations of cases, landmarks and pumps.
vestry	Logical. TRUE uses the 14 pumps from the map in the Vestry Report. FALSE uses the 13 pumps from the original map.
latlong	Logical.
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.
include.landmarks	Logical. Include landmarks as cases.

winterTemperatures *Average Winter Temperatures.*

Description

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

Usage

```
winterTemperatures()
```

Examples

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

Index

* datasets

- anchor.case, 23
- border, 24
- fatalities, 28
- fatalities.address, 29
- fatalities.unstacked, 30
- frame.data, 31
- frame.sample, 31
- landmark.squares, 32
- landmark.squaresB, 32
- landmarks, 33
- landmarksB, 34
- latlong.ortho.addr, 35
- latlong.ortho.pump, 35
- latlong.ortho.pump.vestry, 36
- latlong.regular.cases, 37
- latlong.sim.ortho.proj, 37
- ortho.proj, 50
- ortho.proj.pump, 50
- ortho.proj.pump.vestry, 51
- oxford.weather, 52
- plague.pit, 53
- pumps, 76
- pumps.vestry, 76
- rd.sample, 77
- rectangle.filter, 78
- regular.cases, 78
- road.segments, 79
- roads, 80
- sim.ortho.proj, 84
- sim.pump.case, 85
- sim.walking.distance, 85
- snow.neighborhood, 87
- voronoi.polygons, 97
- voronoi.polygons.vestry, 98

addCase, 6

addDelaunay, 7

addEuclideanPath, 7

addEuclideanPathB, 8

addFrame, 9

addIndexCase, 10

addKernelDensity, 10

addLandmarks, 12

addLandmarkSquares, 12

addMilePosts, 13

addNeighborhoodCases, 14

addNeighborhoodEuclidean, 15

addNeighborhoodWalking, 16

addPlaguePit, 17

addPump, 18

addRoads, 18

addSnow, 19

addVoronoi, 19

addWalkingPath, 20

addWalkingPathB, 21

addWhitehead, 22

anchor.case, 23, 97

border, 24

caseDistance, 24

caseLocator, 23, 25, 28–30

cholera (cholera-package), 5

cholera-package, 5

euclideanPath, 26

euclideanPathB, 27

fatalities, 28

fatalities.address, 29, 97

fatalities.unstacked, 30, 97

fixFatalities, 28, 30

frame.data, 31

frame.sample, 31

landmark.squares, 32

landmark.squaresB, 32

landmarkData, 33, 34

landmarks, 33

landmarksB, 34

- latlong.ortho.addr, 35
- latlong.ortho.pump, 35
- latlong.ortho.pump.vestry, 36
- latlong.regular.cases, 37
- latlong.sim.ortho.proj, 37
- latlongEuclideanPath, 38
- latlongNearestPump, 38
- latlongNeighborhoodData, 39
- latlongNeighborhoodDataB, 40
- latlongNeighborhoodEuclidean, 40
- latlongNeighborhoodVoronoi, 41
- latlongNeighborhoodWalking, 42
- latlongStreetNameLocator, 42
- latlongVoronoi, 43

- mapRange, 44

- nearestPump, 44
- neighborhoodData, 45
- neighborhoodDataB, 46
- neighborhoodEuclidean, 46
- neighborhoodVoronoi, 47
- neighborhoodWalking, 49, 85

- ortho.proj, 50, 97
- ortho.proj.pump, 50, 74
- ortho.proj.pump.vestry, 51, 74
- oxford.weather, 52
- oxfordWeather, 52

- pearsonResiduals, 53
- plague.pit, 53
- plot.euclidean, 54
- plot.euclidean_path, 55
- plot.euclidean_path_B, 55
- plot.latlong_euclidean_path, 57
- plot.latlong_neighborhood_data, 58
- plot.latlong_walking, 58
- plot.latlongEuclidean, 56
- plot.latlongVoronoi, 57
- plot.neighborhood_data, 59
- plot.oxfordWeather, 59
- plot.povertyLondon, 60
- plot.profile_perspective, 61
- plot.time_series, 61, 96
- plot.voronoi, 62
- plot.walking, 63
- plot.walking_path, 64
- plot.walking_path_B, 65

- plot.winterTemperatures, 65
- povertyLondon, 66
- print.euclidean, 66
- print.euclidean_path, 67
- print.euclidean_path_B, 67
- print.latlongVoronoi, 68
- print.time_series, 68, 96
- print.voronoi, 69
- print.walking, 69
- print.walking_path, 70
- print.walking_path_B, 71
- profile2D, 71
- profile3D, 72
- pumpCase, 73
- pumpData, 36, 51, 73, 75–77
- pumpFatalities, 74
- pumpLocator, 74, 75, 76, 77
- pumps, 74, 76
- pumps.vestry, 74, 76

- rd.sample, 77
- rectangle.filter, 78
- regular.cases, 78, 86
- road.segments, 79, 80, 81
- roads, 79, 80
- roadSegments, 79, 81

- segmentHighlight, 81
- segmentLength, 82
- segmentLocator, 79, 80, 83
- sim.ortho.proj, 84, 86
- sim.pump.case, 85
- sim.walking.distance, 85
- simulateFatalities, 78, 84, 86
- snow.neighborhood, 87
- snowColors, 87
- snowMap, 88
- snowNeighborhood, 89
- streetHighlight, 89
- streetLength, 90
- streetNameLocator, 28–30, 79, 80, 90
- streetNames, 91
- streetNumberLocator, 28–30, 79, 80, 92
- summary.euclidean, 93
- summary.voronoi, 93
- summary.walking, 94

- tanakaContourPlot, 95
- timeSeries, 62, 95

unitMeter, [96](#)
unstackFatalities, [23](#), [29](#), [30](#), [35](#), [50](#), [96](#)

voronoi.polygons, [97](#)
voronoi.polygons.vestry, [98](#)
voronoiPolygons, [98](#)

walkingPath, [99](#)
walkingPathB, [101](#)
winterTemperatures, [102](#)