Package 'cholera'

August 15, 2018

Type Package

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

Version 0.5.1

Date 2018-08-15

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 (>= 2.10)

Imports deldir (>= 0.0-18), HistData (>= 0.7-8), grDevices, igraph, KernSmooth, pracma, RColorBrewer, scales, sp, stats

Suggests ggplot2, knitr

VignetteBuilder knitr

RoxygenNote 6.1.0

Encoding UTF-8

NeedsCompilation no

Author Peter Li [aut, cre]

Maintainer Peter Li <lindbrook@gmail.com>

Repository CRAN

Date/Publication 2018-08-15 17:30:03 UTC

R topics documented:

`cholera-package` *cholera: amend, augment and aid analysis of John Snow's cholera map*

Description

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

Details

Features:

- Fixes three apparent coding errors in Dodson and Tobler's 1992 digitization of Snow's map.
- "Unstacks" the data in two ways to make analysis and visualization easier and more meaningful.
- Computes and visualizes "pump neighborhoods" based on Voronoi tessellation, Euclidean distance, and walking distance.
- Ability to overlay graphical elements and features like kernel density, Voronoi diagrams, Snow's Broad Street neighborhood, and notable landmarks (John Snow's residence, the Lion Brewery, etc.) via add*() functions.

- Includes a variety of functions to highlight specific cases, roads, pumps and paths.
- Appends actual street names to roads data.
- Includes the revised pump data used in the second version of Snow's map from the Vestry report, which includes the "correct" location of the Broad Street pump.
- Adds two different aggregate time series fatalities data sets, taken from the Vestry report.
- Computes and visualizes two types of "pump neighborhoods": Voronoi, based on Euclidean distance, and walking, based on computed walking distances.

To learn more, see the vignettes:

```
vignette("duplicate.missing.cases")
vignette("unstacking.bars")
vignette("pump.neighborhoods")
vignette("kernel.density")
vignette("roads")
vignette("time.series")
```

And, if interested, see the relevant lab notes, which are available online: Pump Neighborhood, Duplicate and Missing Cases and "Unstacking" Bars

---

`addBorder` *Add map border to plot.*

Description

Add map border to plot.

Usage

```
addBorder(...)
```

Arguments

...	Additional plotting parameters.

addCases *Add observed cases by walking neighborhood.*

Description

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

Usage

```
addCases(pump.subset = NULL, pump.select = NULL, type = "address",
  token = "point", text.size = 0.5, vestry = FALSE,
  weighted = TRUE, color = NULL, multi.core = FALSE, ...)
```

Arguments

pump.subset	Numeric. Vector of numeric pump IDs to subset from the neighborhoods defined by pump.select. Negative selection possible. NULL uses all pumps in pump.select.
pump.select	Numeric. Numeric vector of pump IDs that define which pump neighborhoods to consider (i.e., specify the "population"). Negative selection possible. NULL selects all pumps.
type	Character. Type of case: "address" (base of stack) or "fatalities" (entire stack).
token	Character. Type of token to plot: "point" or "id".
text.size	Numeric. Size of case ID text.
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.
color	Character. Use a single color for all paths. NULL uses neighborhood colors defined by snowColors().
multi.core	Logical or Numeric. TRUE uses parallel::detectCores(). FALSE uses one, single core. You can also specify the number logical cores. On Windows, only multi.core = FALSE is available.
...	Additional plotting parameters.

Examples

```
## Not run:

snowMap(add.cases = FALSE)
addCases(pump.subset = c(6, 10))

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

## End(Not run)
```

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

Arguments

origin	Numeric or Integer. Numeric ID of case or pump.
destination	Numeric or Integer. Numeric ID(s) of case(s) or pump(s). Exclusion is possible via negative selection (e.g., -7). Default is NULL: this returns closest pump or "anchor" case.
type	Character "case-pump", "cases" or "pumps".
observed	Logical. Use observed or simulated expected data.
vestry	Logical. TRUE uses the 14 pumps from the Vestry Report. FALSE uses the 13 pumps from the original map.
unit	Character. Unit of distance: "meter", "yard" or "native". "native" returns the map's native scale. See vignette("roads") for information on unit distances.
time.unit	Character. "hour", "minute", or "second".
walking.speed	Numeric. Walking speed in km/hr.
unit.posts	Character. "distance" for mileposts; "time" for timeposts; NULL for no posts.
unit.interval	Numeric. Sets interval between unit.posts.
alpha.level	Numeric. Alpha level transparency for path: a value in [0, 1].

Value

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

Note

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

addIndexCase *Highlight index case at 40 Broad Street.*

Description

Highlight index case at 40 Broad Street.

Usage

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

Arguments

cex	Numeric. Size of point.
col	Character. Color of point.
pch	Numeric. Type of of point.
add.label	Logical. Add text annotation: "40 Broad Street"
text.size	Numeric. Size of text label.
...	Additional plotting parameters.

Value

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

See Also

snowMap, addKernelDensity, addLandmarks, addPlaguePit, addSnow, addVoronoi, addWhitehead

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", obs.unit = "unstacked",
  bandwidth = 0.5, color = "black", line.type = "solid",
  multi.core = FALSE, ...)
```

Arguments

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

Value

Add contours to a graphics plot.

Note

This function uses KernSmooth::bkde2D().

See Also

snowMap, addIndexCase, addLandmarks, addPlaguePit, addSnow, addVoronoi, addWhitehead

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)
```

Arguments

text.size	Numeric. cex for text labels.

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.

See Also

snowMap, addIndexCase, addKernelDensity, addPlaguePit, addSnow, addVoronoi, addWhitehead

Examples

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

`addMilePosts` *Add distance or time based "mileposts" to a walking neighborhood plot.*

Description

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

Usage

```
addMilePosts(pump.subset = NULL, pump.select = NULL, vestry = FALSE,
  unit = "distance", interval = NULL, walking.speed = 5,
  type = "arrows", multi.core = 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. On Windows, only `multi.core = FALSE` is available.

Value

R base graphics arrows or points.

addNeighborhood *Add expected walking neighborhoods.*

Description

Add expected walking neighborhoods.

Usage

```
addNeighborhood(pump.subset = NULL, pump.select = NULL,
  vestry = FALSE, weighted = TRUE, multi.core = FALSE, area = TRUE,
  path = NULL, path.color = NULL, path.width = 3,
  alpha.level = 0.25, ...)
```

Arguments

pump.subset	Numeric. Vector of numeric pump IDs to subset from the neighborhoods defined by pump.select. Negative selection possible. NULL uses all pumps in pump.select.
pump.select	Numeric. Numeric vector of pump IDs that define which pump neighborhoods to consider (i.e., specify the "population"). Negative selection possible. NULL selects all pumps.
vestry	Logical. TRUE uses the 14 pumps from the Vestry Report. FALSE uses the 13 in the original map.
weighted	Logical. TRUE computes shortest path weighted by road length. FALSE computes shortest path in terms of the number of nodes.
multi.core	Logical or Numeric. TRUE uses parallel::detectCores(). FALSE uses one, single core. You can also specify the number logical cores. On Windows, only multi.core = FALSE is available.
area	Logical. Area polygons.
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].
...	Additional plotting parameters.

Note

This function is computationally intensive. On a single core of a 2.3 GHz Intel i7, plotting observed paths to PDF takes about 5 seconds while doing so for expected paths takes about 28 seconds. Using the parallel implementation on 4 physical (8 logical) cores, these times fall to about 4 and 11 seconds. Note that parallelization is currently only available on Linux and Mac, and that although some precautions are taken in R.app on macOS, the developers of the 'parallel' package, which neighborhoodWalking() uses, strongly discourage against using parallelization within a GUI or embedded environment. See vignette("parallel") for details.

See Also

snowMap, addIndexCase, addKernelDensity, addLandmarks, addPlaguePit, addVoronoi, addWhitehead

Examples

```
## Not run:

streetNameLocator("marshall street", zoom = TRUE, highlight = FALSE,
  unit = "meter", cases = NULL)
addNeighborhood(6:7)


## End(Not run)
```

---

addPlaguePit *Add plague pit (Marshall Street).*

Description

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

Usage

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

Arguments

color	Character. Color of polygon.
line.type	Character. Polygon line type.
...	Additional plotting parameters.

Value

Adds a polygon to a graphics plot.

Note

In progress.

See Also

snowMap, addIndexCase, addKernelDensity, addLandmarks, addSnow, addVoronoi, addWhitehead

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, ...)
```

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.
...	Additional plotting parameters.

addRoads *Add roads to plot.*

Description

Add roads to plot.

Usage

```
addRoads(col = "gray", ...)
```

Arguments

col	Character. Color
...	Additional plotting parameters.

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

Description

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

Usage

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

Arguments

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

See Also

snowMap, addIndexCase, addKernelDensity, addLandmarks, addPlaguePit, addVoronoi, addWhitehead

Examples

```
## Not run:

plot(neighborhoodVoronoi())
addSnow()

## End(Not run)
```

addVoronoi *Add Voronoi cells.*

Description

Add Voronoi cells.

Usage

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

Arguments

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

Note

This function uses deldir::deldir().

See Also

snowMap, addIndexCase, addKernelDensity, addLandmarks, addPlaguePit, addSnow, addWhitehead

Examples

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

---

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

Description

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

Usage

```
addWalkingPath(origin, destination = NULL, type = "case-pump",
  observed = TRUE, weighted = TRUE, vestry = FALSE, unit = "meter",
  time.unit = "second", walking.speed = 5, zoom = TRUE,
  radius = 0.5, unit.posts = "distance", unit.interval = NULL,
  alpha.level = 1)
```

Arguments

origin	Numeric or Integer. Numeric ID of case or pump.
destination	Numeric or Integer. Numeric ID(s) of case(s) or pump(s). Exclusion is possible via negative selection (e.g., -7). Default is NULL: this returns closest pump or "anchor" case.
type	Character "case-pump", "cases" or "pumps".
observed	Logical. Use observed or "simulated" expected data.

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.
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.
zoom	Logical.
radius	Numeric. Controls the degree of zoom.
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.

See Also

fatalities, vignette("pump.neighborhoods")

---

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, color = "black",
  line.type = "solid", vestry = FALSE, 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 in yards.
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.
subtitle	Logical. Add subtitle with estimated "walking" time in seconds.
walking.speed	Numeric. Walking speed in km/hr.
...	Additional plotting parameters.

Value

Adds a circle (polygon) to a graphics plot.

See Also

snowMap, addIndexCase, addKernelDensity, addLandmarks, addPlaguePit, addSnow, addVoronoi

addLandmarks

Examples

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

---

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

Description

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

Usage

```
anchor.case
```

Format

`case` numerical case ID

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

Note

`unstackFatalities` documents the code for these data.

`border` *Numeric IDs of line segments that create the map's border frame.*

Description

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

Usage

```
border
```

Format

`border` numerical ID

`caseLocator` *Locate case by numerical ID.*

Description

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

Usage

```
caseLocator(case, zoom = FALSE, observed = TRUE, radius = 2,
  stacked = TRUE)
```

Arguments

`case`	Numeric or Integer. Whole number between 1 and 578.
`zoom`	Logical.
`observed`	Logical. `TRUE` for observed. `FALSE` for simulated.
`radius`	Numeric. Controls the degree of zoom.
`stacked`	Logical. `TRUE` uses `fatalities` ("stacked" data); `FALSE` uses `fatalities.address` ("unstacked" data).

Value

A base R graphics plot.

See Also

fatalities, fatalities.address, fatalities.unstacked

Examples

```
caseLocator(290)
caseLocator(290, zoom = TRUE)
caseLocator(290, stacked = FALSE)
caseLocator(290, zoom = TRUE, stacked = FALSE)
caseLocator(290, observed = FALSE)
```

---

classifierAudit *Test if case is orthogonal to segment.*

Description

Diagnostic to check classification of case to a road segment.

Usage

```
classifierAudit(case = 483, segment = "326-2", observed = TRUE,
  coordinates = FALSE)
```

Arguments

case	Numeric or Integer. Numeric ID of observed case.
segment	Character. Segment ID. See road.segments.
observed	Logical. FALSE observed case; TRUE simulated case (regular.cases).
coordinates	Logical. Orthogonal projection coordinates.

Value

Logical TRUE or FALSE

Note

This function is a diagnostic. It is not a guarantee of correct classification.

Examples

```
classifierAudit(case = 483, segment = "326-2")
plot(classifierAudit(case = 483, segment = "326-2"))
```

distanceTime *Convert distance to elapsed time.*

Description

Convert distance to elapsed time.

Usage

```
distanceTime(x, unit = "second", meter.unit = cholera::unitMeter(1,
  unit = "meter"), speed = 5)
```

Arguments

x	Numeric. Nominal map distance.
unit	Character. Unit of measurement: "hour", "minute" or "second".
meter.unit	Numeric. 1 meter is approximately 54 map units. Set by cholera::unitMeter().
speed	Numeric. Walking speed in km/hr.

Value

An R vector.

euclideanDistance *Compute the Euclidean distance between cases and/or pumps.*

Description

Compute the Euclidean distance between cases and/or pumps.

Usage

```
euclideanDistance(origin, destination = NULL, type = "case-pump",
  observed = TRUE, vestry = FALSE, unit = "meter",
  time.unit = "second", walking.speed = 5)
```

Arguments

origin	Numeric or Integer. Numeric ID of case or pump.
destination	Numeric or Integer. Numeric ID(s) of case(s) or pump(s). Exclusion is possible via negative selection (e.g., -7). Default is NULL: this returns closest pump or "anchor" case.
type	Character "case-pump", "cases" or "pumps".
observed	Logical. Use observed or "simulated" expected data.

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

Value

An R data frame.

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.
euclideanDistance(1)

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

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

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

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

---

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, destination = NULL, type = "case-pump",
  observed = TRUE, vestry = FALSE, unit = "meter",
  time.unit = "second", walking.speed = 5)
```

Arguments

origin	Numeric or Integer. Numeric ID of case or pump.
destination	Numeric or Integer. Numeric ID(s) of case(s) or pump(s). Exclusion is possible via negative selection (e.g., -7). Default is NULL: this returns closest pump or "anchor" case.
type	Character "case-pump", "cases" or "pumps".
observed	Logical. Use observed or "simulated" expected data.
vestry	Logical. TRUE uses the 14 pumps from the Vestry Report. FALSE uses the 13 pumps from the original map.
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 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")

# Plot result
plot(euclideanPath(1))
```

`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

Note

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

See Also

`caseLocator`

`streetNameLocator`

`streetNumberLocator`

`caseLocator`

`streetNameLocator`

`streetNumberLocator`

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

Description

An "unstacked" version of the `fatalities` dataset. It changes the unit of observation from the case (bar) to the "address", the x-y coordinate 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.case` numerical case ID of address

`x` x-coordinate

`y` y-coordinate

`case.count` number of fatalities at address

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 coordinate 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

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")`

`mapRange` *Compute xlim and ylim of Snow's map.*

Description

Compute xlim and ylim of Snow's map.

Usage

```
mapRange()
```

nearestPump *Compute shortest walking distances or paths.*

Description

Compute shortest walking distances or paths.

Usage

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

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.
output	Character. "distance" or "path".
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".
unit	Character. Unit of distance: "meter", "yard" or "native". "native" returns the map's native scale. Meaningful only when "weighted" is TRUE and "output" is "distance". See vignette("roads") for information on unit distances.
multi.core	Logical or Numeric. TRUE uses parallel::detectCores(). FALSE uses one, single core. You can also specify the number logical cores. On Windows, only multi.core = FALSE is available.
time.unit	Character. "hour", "minute", or "second".
walking.speed	Numeric. Walking speed in km/hr.

Value

An R data frame or list of 'igraph' paths.

Note

Time is computed using distanceTime().

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

Description

Assembles cases, pumps and road into a network graph.

Usage

```
neighborhoodData(vestry = FALSE, case.set = "observed")
```

Arguments

vestry	Logical. Use Vestry Report pump data.
case.set	Character. "observed" or "expected", or "snow". "snow" captures John Snow's annotation of the Broad Street pump neighborhood printed in the Vestry report version of the map.

Value

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

neighborhoodEuclidean *Plot Euclidean path pump neighborhoods.*

Description

Plots star graph from pump to its cases.

Usage

```
neighborhoodEuclidean(pump.subset = NULL, pump.select = NULL,
  vestry = FALSE, case.set = "observed", multi.core = FALSE)
```

Arguments

pump.subset	Numeric. Vector of numeric pump IDs to subset from the neighborhoods defined by pump.select. Negative selection possible. NULL selects all pumps in pump.select.
pump.select	Numeric. Vector of numeric pump IDs to define pump neighborhoods (i.e., the "population"). Negative selection possible. NULL selects all pumps.
vestry	Logical. TRUE uses the 14 pumps from the Vestry Report. FALSE uses the 13 in the original map.
case.set	Character. "observed" or "expected".
multi.core	Logical or Numeric. TRUE uses parallel::detectCores(). FALSE uses one, single core. You can also specify the number logical cores. On Windows, only multi.core = FALSE is available.

Value

A base R graph.

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,
  statistic = NULL, polygon.vertices = FALSE)
```

Arguments

`pump.select`	Numeric. Vector of numeric pump IDs to define pump neighborhoods (i.e., the "population"). Negative selection possible. `NULL` selects all pumps.
`vestry`	Logical. `TRUE` uses the 14 pumps from the Vestry report. `FALSE` uses the 13 in the original map.
`statistic`	NULL or Character. `NULL`, the default, makes no summary computation. "address" computes the number of addresses in each selected pump neighborhood. "fatality" computes the number of fatalities in pump neighborhoods.
`polygon.vertices`	Logical. `TRUE` returns a list of x-y coordinates of the vertices of Voronoi cells. Useful for `sp::point.in.polygon()` as used in `print.voronoi()` method.

Value

An R list with 12 objects.

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

- y.rng: range of y for plot.
- 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().

See Also

addVoronoi, plot.voronoi, print.voronoi, vignette("pump.neighborhoods")

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 = 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 parallel::detectCores(). FALSE uses one, single core. You can also specify the number logical cores. On Windows, only multi.core = FALSE is available.

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.

Note

This function is computationally intensive. On a single core of a 2.3 GHz Intel i7, plotting observed paths to PDF takes about 5 seconds while doing so for expected paths takes about 28 seconds. Using the parallel implementation on 4 physical (8 logical) cores, these times fall to about 4 and 11 seconds. Note that parallelization is currently only available on Linux and Mac, and that although some precautions are taken in R.app on macOS, the developers of the 'parallel' package, which neighborhoodWalking() uses, strongly discourage against using parallelization within a GUI or embedded environment. See vignette("parallel") for details.

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.

`road.segment` "address" road segment

`x.proj` x-coordinate

`y.proj` y-coordinate

`ortho.dist` orthogonal distance to home road segment

`node` node ID

`pump.id` numeric 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.

`road.segment` "address" road segment

`x.proj` x-coordinate

`y.proj` y-coordinate

`ortho.dist` orthogonal distance to home road segment

`node` node ID

`pump.id` numeric ID

Note

`pumpData` documents the code for these data.

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.classifier_audit *Plot result of classifierAudit().*

Description

Plot case, segment and orthogonal projector.

Usage

```
## S3 method for class 'classifier_audit'
plot(x, zoom = TRUE, radius = 0.5,
  unit = "meter", ...)
```

Arguments

`x`	An object of class "classifier_audit" created by `classifierAudit()`.
`zoom`	Logical.
`radius`	Numeric. Controls the degree of zoom.
`unit`	Character. Unit of distance: "meter" (the default), "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.
`...`	Additional parameters.

Value

A base R graphic.

Examples

```
plot(classifierAudit(case = 483, segment = "326-2"))
```

`plot.euclidean` *Plot method for neighborhoodWalking().*

Description

Plot method for neighborhoodWalking().

Usage

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

Arguments

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

Value

A base R plot.

`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 = TRUE, radius = 0.5,
  unit.posts = "distance", unit.interval = NULL, ...)
```

Arguments

`x`	An object of class "euclidean_path" created by euclideanPath().
`zoom`	Logical.
`radius`	Numeric. Controls the degree of zoom.
`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.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 *Plot Voronoi neighborhoods.*

Description

Plot Voronoi neighborhoods.

Usage

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

Arguments

x	An object of class "voronoi" created by neighborhoodVoronoi().
voronoi.cells	Logical. Plot Voronoi tessellation cells.
euclidean.paths	Logical. Plot all Euclidean paths (star graph).
...	Additional plotting parameters.

Value

A base R graph.

See Also

neighborhoodVornoi()

addVoronoi()

Examples

```
plot(neighborhoodVoronoi())
```

plot.walking *Plot method for neighborhoodWalking().*

Description

Plot method for neighborhoodWalking().

Usage

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

Arguments

x	An object of class "walking" created by neighborhoodWalking().
type	Character. "road", "area.points" or "area.polygons". "area" flavors only valid when case.set = "expected".
...	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. The "area.points" plot takes about 28 seconds (11 using the parallel implementation). The "area.polygons" plot takes 49 seconds (17 using the parallel implementation).

Examples

```
## Not run:

plot(neighborhoodWalking())
plot(neighborhoodWalking(case.set = "expected"))
plot(neighborhoodWalking(case.set = "expected"), type = "area.points")
plot(neighborhoodWalking(case.set = "expected"), type = "area.polygons")

## End(Not run)
```

---

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

Description

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

Usage

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

Arguments

x	An object of class "walking_path" created by walkingPath().
zoom	Logical.
radius	Numeric. Control the degree of zoom.
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

Arrow represent mileposts or timeposts to the destination.

Examples

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

---

print.classifier_audit

Return result of classifierAudit().

Description

Return result of classifierAudit().

Usage

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

Arguments

x	An object of class "classifier_audit" created by classifierAudit().
...	Additional parameters.

Value

An R data frame.

Examples

```
classifierAudit(case = 483, segment = "326-2")
print(classifierAudit(case = 483, segment = "326-2"))
```

`print.euclidean` *Print method for neighborhoodWalking().*

Description

Print method for neighborhoodWalking().

Usage

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

Arguments

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

Value

An R class 'table' vector.

`print.euclidean_path` *Summary of euclideanPath().*

Description

Summary of euclideanPath().

Usage

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

Arguments

`x`	An object of class "euclidean_path" created by `euclideanPath()`.
`...`	Additional parameters.

Value

An R data frame.

Examples

```
euclideanPath(1)
print(euclideanPath(1))
```

`print.time_series` *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` *Print method for neighborhoodVoronoi().*

Description

Return summary statistics for Voronoi neighborhoods.

Usage

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

Arguments

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

Value

A data frame with observed and expected counts, observed percentage, and the Pearson residual, (observed - expected) / sqrt(expected).

See Also

`addVoronoi()` `plot.voronoi()`

Examples

```
neighborhoodVoronoi()
print(neighborhoodVoronoi())
```

---

print.walking *Print method for neighborhoodWalking().*

Description

Return count of paths (anchor cases) by pump neighborhood.

Usage

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

Arguments

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

Value

An R vector.

Examples

```
## Not run:

neighborhoodWalking()
print(neighborhoodWalking())

## End(Not run)
```

print.walking_path *Print summary output for walkingPath().*

Description

Print summary output for walkingPath().

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

```
walkingPath(1)
print(walkingPath(1))
```

pumpCase *Extract numeric case IDs by pump neighborhood.*

Description

Extract numeric case IDs by pump neighborhood.

Usage

```
pumpCase(x)
```

Arguments

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

Value

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

See Also

neighborhoodVoronoi, neighborhoodVoronoi, neighborhoodEuclidean,

Examples

```
## Not run:

pumpCase(neighborhoodEuclidean())
pumpCase(neighborhoodVoronoi())
pumpCase(neighborhoodWalking())

## End(Not run)
```

---

pumpData *Compute pump coordinates.*

Description

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

Usage

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

Arguments

vestry	Logical. TRUE uses the 14 pumps from the Vestry report. FALSE uses the 13 in the original map.
orthogonal	Logical. TRUE returns pump "addresses": the coordinates of the orthogonal projection from a pump's location onto the network of roads. FALSE returns pump location coordinates.
multi.core	Logical or Numeric. TRUE uses parallel::detectCores(). FALSE uses one, single core. With Numeric, you specify the number logical cores (rounds with as.integer()). On Windows, only multi.core = FALSE is available.

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.

The Dodson and Tobler coordinates of the original thirteen pumps, appended with name of nearest road, or the fourteen pumps included in the second version of Snow's map included in the Vestry report. This function documents the code that generates pumps, pumps.vestry, ortho.proj.pump and ortho.proj.pump.vestry.

See Also

pumpLocator

---

pumpLocator *Locate water pump by numerical ID.*

Description

Highlight selected water pump.

Usage

```
pumpLocator(id, zoom = FALSE, radius = 2, vestry = 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.
radius	Numeric. Controls the degree of zoom.
vestry	Logical. `TRUE` for the 14 pumps from Vestry Report. `FALSE` for the original 13 pumps.

Value

A base R graphics plot.

See Also

pumpData

Examples

```
pumpLocator(7) # Broad Street Pump
pumpLocator(7, zoom = TRUE)
pumpLocator(7, zoom = TRUE, radius = 1)
pumpLocator(14, vestry = TRUE, zoom = TRUE, radius = 1)
```

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

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

Note

`pumpData` documents the code for these data.

See Also

`pumpLocator`

`regular.cases` *"Expected" cases.*

Description

The result of using sp::spsample() and sp::Polygon() to generate 20007 regularly spaced simulated cases within the map's borders.

Usage

```
regular.cases
```

Format

A data frame with 2 variable that records the position of 20007 "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 657 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 206 observations and 5 variables. The data describe the roads on Snow's map.

`street` street segment number, which range between 1 and 528

`n` number of points in this street line segment

`x` x-coordinate

`y` y-coordinate

`id` unique numeric ID

`name` road name

See Also

`road.segments`

`vignette("road.names")`

`streetNameLocator`

`streetNumberLocator`

`segmentLocator`

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

Description

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

Usage

```
roadSegments()
```

Value

An R data frame.

Note

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

`segmentLength` *Compute length of road segment.*

Description

Compute length of road segment.

Usage

```
segmentLength(id, unit = "meter")
```

Arguments

`id`	Character. A concatenation of a street's numeric ID, a whole number between 1 and 528, and a second number used to identify the sub-segments.
`unit`	Character. Unit of distance: "meter", "yard" or "native". "native" returns the map's native scale. See `vignette("roads")` for information on conversion.

Value

An R vector of length one.

See Also

`roads`, `road.segments`, `streetNameLocator`, `streetNumberLocator`, `vignette("roads")`

Examples

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

`segmentLocator` *Locate road segment by ID.*

Description

Highlights the selected road segment and its cases.

Usage

```
segmentLocator(id, zoom = FALSE, radius = 0.5, cases = "anchors",
  unit = "meter", time.unit = "minute", walking.speed = 5,
  title = TRUE, subtitle = TRUE)
```

Arguments

`id`	Character. A concatenation of a street's numeric ID, a whole number between 1 and 528, and a second number to identify the segment.
`zoom`	Logical. Default is `FALSE`.
`radius`	Numeric. Controls the degree of zoom. For values <= 5, the numeric ID of all cases or just the anchor case is plotted.
`cases`	Character. Plot cases: `NULL`, "anchors" or "all".
`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.
`title`	Logical. Print title.
`subtitle`	Logical. Print subtitle.

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.

See Also

`road.segments`

Examples

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

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

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 20007 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 *List of "simulated" fatalities grouped by walking-distance pump neighborhood.*

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)
```

---

simulateFatalities *Generate simulated fatalities and their orthogonal projection on road network.*

Description

Places regularly spaced "simulated" or "expected" cases across the face of the map. The function finds the "addresses" of cases on the road network 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 = FALSE,
  simulated.obs = 20000L)
```

Arguments

compute	Logical. TRUE computes data. FALSE uses pre-computed data. For replication of data used in the package,
multi.core	Logical or Numeric. TRUE uses parallel::detectCores(). FALSE uses one, single core. With Numeric, you specify the number logical cores (rounds with as.integer()). On Windows, only multi.core = FALSE is available.
simulated.obs	Numeric. Number of sample cases.

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), the function takes about 94 minutes to run on a single core of a 2.3 GHz Intel Core i7 with R version 3.5.1 and 22 minutes to run on eight logical (four physical) cores. This function documents the code that generates sim.ortho.proj and regular.cases. The distance between of these simulated cases is approximately 4.2 meters.

snow.neighborhood *Snow neighborhood fatalities.*

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 *Create a set of colors for pump neighborhoods.*

Description

Uses RColorBrewer::brewer.pal().

Usage

```
snowColors(vestry = FALSE)
```

Arguments

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

Value

A character vector of colors.

snowMap *Plot John Snow's cholera map.*

Description

Plot John Snow's cholera map.

Usage

```
snowMap(vestry = FALSE, stacked = TRUE, add.cases = TRUE,
  add.landmarks = FALSE, add.pumps = TRUE, add.roads = TRUE,
  add.title = TRUE, ...)
```

Arguments

vestry	Logical. TRUE uses the 14 pumps from the map in the Vestry Report. FALSE uses the 13 pumps from the original map.
stacked	Logical. Use stacked fatalities.
add.cases	Logical. Add observed cases.
add.landmarks	Logical. Add landmarks.
add.pumps	Logical. Add pumps.
add.roads	Logical. Add roads.
add.title	Logical. Add title.
...	Additional plotting parameters.

Value

A base R graphics plot.

Note

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

See Also

addLandmarks, addKernelDensity, addLandmarks, addPlaguePit, addVoronoi. addWhitehead

Examples

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

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

Description

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

Usage

```
snowNeighborhood()
```

Value

An R list of edge IDs and simulated case IDs.

streetLength *Compute length of selected street.*

Description

Compute length of selected street.

Usage

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

Arguments

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

Value

An R vector of length one.

See Also

roads, coderoad.segments, streetNameLocator, streetNumberLocator, vignette("roads")

Examples

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

`streetNameLocator` *Locate road by name.*

Description

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

Usage

```
streetNameLocator(road.name, zoom = FALSE, radius = 0.1,
  cases = "anchors", add.title = TRUE, add.pump = TRUE,
  vestry = FALSE, highlight = TRUE, 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.
`radius`	Numeric. Controls the degree of zoom.
`cases`	Character. Plot cases: `NULL`, "anchors" or "all".
`add.title`	Logical. Include title.
`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.
`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.

See Also

`roads`, `road.segments`, `streetNumberLocator`, `vignette("roads")`

Examples

```
streetNameLocator("Oxford Street")
streetNameLocator("oxford street")
streetNameLocator("Cambridge Street", zoom = TRUE)
streetNameLocator("Cambridge Street", zoom = TRUE, radius = 0)
```

`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, zoom = FALSE, radius = 1,
  cases = "anchors", add.title = TRUE, add.pump = TRUE,
  vestry = FALSE, highlight = TRUE, unit = "meter",
  time.unit = "second", walking.speed = 5)
```

Arguments

`road.number`	Numeric or integer. A whole number between 1 and 528.
`zoom`	Logical.
`radius`	Numeric. Controls the degree of zoom.
`cases`	Character. Plot cases: `NULL`, "anchors" or "all".
`add.title`	Logical. Include title.
`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.
`unit`	Character. Unit of measurement: "meter" or "yard". Default is `NULL`, which returns the map's native scale.
`time.unit`	Character. "hour", "minute", or "second".
`walking.speed`	Numeric. Walking speed in km/hr.

Value

A base R graphics plot.

See Also

`roads`, `road.segments`, `streetNameLocator`, `vignette("roads")`

Examples

```
streetNumberLocator(243)
streetNumberLocator(243, zoom = TRUE)
streetNumberLocator(243, zoom = TRUE, radius = 0)
```

`timeSeries` *Aggregate time series fatality data from the Vestry report.*

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 *Convert nominal map distance to meters or yards.*

Description

A best guess estimate.

Usage

```
unitMeter(x, unit = "meter", yard.unit = 177/3, meter.unit = 54)
```

Arguments

x	Numeric. Nominal map distance.
unit	Character. Unit of distance: "meter", "yard" or "native". "native" returns the map's native scale. See `vignette("roads")` for information on conversion.
yard.unit	Numeric. Estimate of yards per map unit.
meter.unit	Numeric. Estimate of meters per map unit:.

unstackFatalities *Unstack "stacks" in Snow's cholera map.*

Description

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

Usage

```
unstackFatalities(multi.core = FALSE, compute = FALSE,
  fatalities = cholera::fixFatalities())
```

Arguments

multi.core	Logical or Numeric. `TRUE` uses `parallel::detectCores()`. `FALSE` uses one, single core. With Numeric, you specify the number logical cores. On Windows, only `multi.core = FALSE` is available.
compute	Logical. `TRUE` computes data. `FALSE` uses pre-computed data.
fatalities	Corrected fatalities data from `cholera::fixFatalities()`. For original data, use `HistData::Snow.deaths`.

Value

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

Notes

This function is computationally intensive. On a 2.3 GHz Intel Core i7, it takes approximately 5 minutes to run on one core and approximately 70 seconds to run on eight logical (four physical) cores. These functions document the code that generates `anchor.case`, `fatalities.address`, `fatalities.unstacked` and `ortho.proj`.

See Also

```
vignette("unstacking.fatalities")
```

`walkingDistance` *Compute the shortest walking distance between cases and/or pumps.*

Description

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

Usage

```
walkingDistance(origin, destination = NULL, type = "case-pump",
  observed = TRUE, weighted = TRUE, vestry = FALSE, unit = "meter",
  time.unit = "second", walking.speed = 5)
```

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`, which returns closest pump or "anchor" case.
`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.
`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.

Value

An R data frame.

Note

The function uses a case's "address" (i.e., "anchor" case of a stack) 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 and 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".

See Also

`fatalities`, `vignette("pump.neighborhoods")`

Examples

```
## Not run:

# distance from case 1 to nearest pump.
walkingDistance(1)

# distance from case 1 to pump 6.
walkingDistance(1, 6)

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

# distance from case 1 to case 6.
walkingDistance(1, 6, type = "cases")

# distance from pump 1 to pump 6.
walkingDistance(1, 6, type = "pumps")

## End(Not run)
```

---

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, destination = NULL, type = "case-pump",
  observed = TRUE, weighted = TRUE, vestry = FALSE, unit = "meter",
  time.unit = "second", walking.speed = 5)
```

Arguments

origin	Numeric or Integer. Numeric ID of case or pump.
destination	Numeric or Integer. Numeric ID(s) of case(s) or pump(s). Exclusion is possible via negative selection (e.g., -7). Default is NULL: this returns closest pump or "anchor" case.
type	Character "case-pump", "cases" or "pumps".
observed	Logical. Use observed or "simulated" expected data.
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.
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.

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".

See Also

fatalities, vignette("pump.neighborhoods")

Examples

```
## Not run:

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

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

## End(Not run)
```

Index