

# Package ‘cholera’

March 8, 2019

**Type** Package

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

**Version** 0.6.0

**Date** 2019-03-08

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

**URL** <https://github.com/lindbrook/cholera>

**BugReports** <https://github.com/lindbrook/cholera/issues>

**License** GPL (>= 2)

**LazyData** true

**Depends** R (>= 3.4)

**Imports** deldir (>= 0.0-18), HistData (>= 0.7-8), ggplot2, igraph, KernSmooth, pracma, RColorBrewer, scales, sp, threejs, TSP

**Suggests** knitr, rmarkdown

**VignetteBuilder** knitr

**RoxygenNote** 6.1.1

**Encoding** UTF-8

**Language** en-US

**NeedsCompilation** no

**Author** Peter Li [aut, cre]

**Maintainer** Peter Li <lindbrook@gmail.com>

**Repository** CRAN

**Date/Publication** 2019-03-08 17:42:47 UTC

**R topics documented:**

cholera-package . . . . .	4
addCase . . . . .	5
addDelauny . . . . .	5
addEuclideanPath . . . . .	6
addFrame . . . . .	7
addIndexCase . . . . .	7
addKernelDensity . . . . .	8
addLandmarks . . . . .	9
addMilePosts . . . . .	10
addNeighborhoodCases . . . . .	11
addNeighborhoodEuclidean . . . . .	12
addNeighborhoodWalking . . . . .	13
addPlaguePit . . . . .	14
addPump . . . . .	15
addRoads . . . . .	16
addSnow . . . . .	16
addVoronoi . . . . .	17
addWalkingPath . . . . .	17
addWhitehead . . . . .	19
anchor.case . . . . .	20
border . . . . .	20
caseLocator . . . . .	21
classifierAudit . . . . .	22
deldirVertices . . . . .	22
distanceTime . . . . .	23
euclideanPath . . . . .	24
fatalities . . . . .	25
fatalities.address . . . . .	26
fatalities.unstacked . . . . .	27
fixFatalities . . . . .	28
landmark.squares . . . . .	28
landmarkData . . . . .	29
landmarks . . . . .	29
mapRange . . . . .	30
nearestPump . . . . .	30
neighborhoodData . . . . .	31
neighborhoodEuclidean . . . . .	31
neighborhoodVoronoi . . . . .	32
neighborhoodWalking . . . . .	34
ortho.proj . . . . .	35
ortho.proj.pump . . . . .	36
ortho.proj.pump.vestry . . . . .	36
orthogonalProjection . . . . .	37
pearsonResiduals . . . . .	38
plague.pit . . . . .	38
plot.classifier_audit . . . . .	39

plot.euclidean . . . . .	39
plot.euclidean_path . . . . .	40
plot.neighborhood_data . . . . .	41
plot.profile_perspective . . . . .	42
plot.time_series . . . . .	42
plot.voronoi . . . . .	43
plot.walking . . . . .	44
plot.walking_path . . . . .	45
print.classifier_audit . . . . .	46
print.euclidean . . . . .	46
print.euclidean_path . . . . .	47
print.time_series . . . . .	48
print.voronoi . . . . .	48
print.walking . . . . .	49
print.walking_path . . . . .	50
profile2D . . . . .	50
profile3D . . . . .	51
pumpCase . . . . .	52
pumpData . . . . .	53
pumpLocator . . . . .	54
pumps . . . . .	55
pumps.vestry . . . . .	55
regular.cases . . . . .	56
road.segments . . . . .	57
roads . . . . .	58
roadSegments . . . . .	58
segmentLength . . . . .	59
segmentLocator . . . . .	60
sim.ortho.proj . . . . .	61
sim.pump.case . . . . .	61
simulateFatalities . . . . .	62
snow.neighborhood . . . . .	63
snowColors . . . . .	63
snowMap . . . . .	64
snowNeighborhood . . . . .	65
streetHighlight . . . . .	65
streetLength . . . . .	66
streetNameLocator . . . . .	66
streetNumberLocator . . . . .	67
timeSeries . . . . .	68
unitMeter . . . . .	69
unstackFatalities . . . . .	70
walkingPath . . . . .	71

---

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("kernel.density")
```

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

```
vignette("roads")
```

```
vignette("tiles.polygons")
```

```
vignette("time.series")
```

```
vignette("unstacking.bars")
```

---

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

---

**Description**

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

**Usage**

```
addCase(case = 1, type = "observed", token = "both",
         text.size = 0.5, col = "red")
```

**Arguments**

case	Numeric. Vector of case ID(s).
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.
col	Character. Color.

**Examples**

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

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

---

addDelauny	<i>Add Delauny triangles.</i>
------------	-------------------------------

---

**Description**

Add Delauny triangles.

**Usage**

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

**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. 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>color</code>	Character. Color of triangle edges.
<code>line.type</code>	Character. Type of line for triangle edges.

**Note**

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

**Examples**

```
snowMap()
addDeLauny()
```

---

`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, 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.
<code>type</code>	Character "case-pump", "cases" or "pumps".
<code>observed</code>	Logical. Use observed or simulated expected data.
<code>vestry</code>	Logical. TRUE uses the 14 pumps from the Vestry Report. FALSE uses the 13 pumps from the original map.
<code>distance.unit</code>	Character. Unit of distance: "meter", "yard" or "native". "native" returns the map's native scale. See <code>vignette("roads")</code> for information on unit distances.
<code>time.unit</code>	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().

---

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

---

**Description**

Add map border to plot.

**Usage**

```
addFrame(...)
```

**Arguments**

...                Additional plotting parameters.

---

addIndexCase	<i>Highlight index case at 40 Broad Street.</i>
--------------	---

---

**Description**

Highlight index case at 40 Broad Street.

**Usage**

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

**Arguments**

<code>cex</code>	Numeric. Size of point.
<code>col</code>	Character. Color of point.
<code>pch</code>	Numeric. Type of of point.
<code>add.label</code>	Logical. Add text annotation: "40 Broad Street"
<code>text.size</code>	Numeric. Size of text label.

**Value**

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

**Examples**

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

---

<code>addKernelDensity</code>	<i>Add 2D kernel density contours.</i>
-------------------------------	--

---

**Description**

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

**Usage**

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

**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>cases</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.

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.

**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	<i>Add landmarks to plot.</i>
--------------	-------------------------------

---

**Description**

Add landmarks to plot.

**Usage**

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

**Arguments**

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

**Value**

Base R points and text.

**Note**

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

**Examples**

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

---

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

---

**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.

---

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, vestry = FALSE, weighted = TRUE, color = NULL,
  case.location = "nominal", observed = TRUE, alpha.level = 0.5,
  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.
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.
vestry	Logical. TRUE uses the 14 pumps from the Vestry Report. FALSE uses the 13 in the original map.
weighted	Logical. TRUE computes shortest walking path weighted by road length. FALSE computes shortest walking path in terms of the number of nodes.
color	Character. Use a single color for all paths. NULL uses neighborhood colors defined by snowColors().
case.location	Character. For metric = "euclidean": "address" uses ortho.proj; "nominal" uses fatalities.
observed	Logical. TRUE is observed; FALSE is expected or simulated.
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. On Windows, only multi.core = FALSE is available.

**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 = "nominal", type = "star",
  alpha.level = 0.5, polygon.method = "traveling.salesman",
  multi.core = 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 "nominal". "address" is the x-y coordinates of <code>sim.ortho.proj</code> . "nominal" 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>polygon.method</code>	Character. Method of computing polygon vertices: "pearl.string" or "traveling.salesman".
<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. On Windows, only <code>multi.core = FALSE</code> is available.

**Value**

R graphic elements.

**Note**

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

**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.subset = NULL, pump.select = NULL,
  vestry = FALSE, weighted = TRUE, polygon.method = "pearl.string",
  path = NULL, path.color = NULL, path.width = 3,
  alpha.level = 0.25, polygon.type = "solid", polygon.col = NULL,
  polygon.lwd = 2, 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.
vestry	Logical. TRUE uses the 14 pumps from the Vestry Report. FALSE uses the 13 in the original map.

<code>weighted</code>	Logical. TRUE computes shortest path weighted by road length. FALSE computes shortest path in terms of the number of nodes.
<code>polygon.method</code>	Character. Method of computing polygon vertices: "pearl.string" or "traveling.salesman".
<code>path</code>	Character. "expected" or "observed".
<code>path.color</code>	Character. Use a single color for all paths. NULL uses neighborhood colors defined by <code>snowColors()</code> .
<code>path.width</code>	Numeric. Set width of paths.
<code>alpha.level</code>	Numeric. Alpha level transparency for area plot: a value in [0, 1].
<code>polygon.type</code>	Character. "perimeter" or "solid".
<code>polygon.col</code>	Character.
<code>polygon.lwd</code>	Numeric.
<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. On Windows, only <code>multi.core = FALSE</code> is available.

### 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 30 seconds. Using the parallel implementation on 4 physical (8 logical) cores, these times fall to about 4 and 13 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:

streetNameLocator("marshall street", zoom = 0.5)
addNeighborhoodWalking()

## End(Not run)
```

---

<code>addPlaguePit</code>	<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	<i>Add selected pump(s) to plot.</i>
---------	--------------------------------------

---

**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.

---

addRoads	<i>Add all streets and roads to plot.</i>
----------	---

---

**Description**

Add all streets and roads to plot.

**Usage**

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

**Arguments**

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

**Arguments**

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

**Examples**

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

---

addVoronoi	<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)
```

**Arguments**

pump.select	Numeric. Default is NULL; all pumps are used. Otherwise, selection by a vector of numeric IDs: 1 to 13 for pumps; 1 to 14 for pumps.vestry. Exclusion (negative selection) is possible (e.g., -6).
vestry	Logical. FALSE for original 13 pumps. TRUE for 14 pumps in Vestry Report.
case.location	Character. For observed = FALSE: "address" or "nominal". "nominal" is the x-y coordinates of regular.cases.
color	Character. Color of cell edges.
line.type	Character. Type of line for cell edges: lty.
line.width	Numeric. Width of cell edges: lwd.

**Note**

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

**Examples**

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

---

addWalkingPath	<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**

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

---

addWhitehead	<i>Add Rev. Henry Whitehead's Broad Street pump neighborhood.</i>
--------------	---

---

### 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, 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 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.
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

---

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 $\geq 0$ controls the degree of zoom. The default is 1.
observed	Logical. TRUE for observed. FALSE for simulated.
add.title	Logical. Include title.
highlight.segment	Logical. Highlight case's segment.
data	Logical. Output data.
add	Logical. Add to existing plot or separate plot.
col	Character. Point color.

## Value

A base R graphics plot.

## Examples

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

---

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"))
```

---

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

---

### Description

For use with functions that use polygons.

### Usage

```
deldirVertices(sites, rw.data = NULL, rw = NULL, type = "tiles")
```

**Arguments**

sites	Object. Data frame of sites to compute Delauny 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.

**Value**

An R list of data frames of vertices.

**Note**

This function relies on the 'deldir' package.

**Examples**

```
vertices <- deldirVertices(pumps)
snowMap()
invisible(lapply(vertices, polygon))

vertices <- deldirVertices(pumps, roads)
snowMap()
invisible(lapply(vertices, polygon))

vertices <- deldirVertices(pumps, roads, type = "triangles")
snowMap()
invisible(lapply(vertices, polygon))
```

---

distanceTime                      *Convert distance to elapsed time.*

---

**Description**

Convert distance to elapsed time.

**Usage**

```
distanceTime(x, distance.unit = "meter", time.unit = "second",
            walking.speed = 5)
```

**Arguments**

x	Numeric. Nominal map distance.
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. Unit of measurement: "hour", "minute" or "second".
walking.speed	Numeric. Walking speed in km/hr.

**Value**

An R vector.

---

euclideanPath	<i>Compute path of the Euclidean distance between cases and/or pumps.</i>
---------------	---

---

**Description**

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

**Usage**

```
euclideanPath(origin = 1, destination = NULL, type = "case-pump",
  observed = TRUE, case.location = "nominal", landmark.cases = TRUE,
  vestry = FALSE, distance.unit = "meter", time.unit = "second",
  walking.speed = 5, multi.core = 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, which returns the closest pump, "anchor" case or landmark.
type	Character "case-pump", "cases" or "pumps".
observed	Logical. Use observed or "simulated" expected data.
case.location	Character. For observed = FALSE: "address" or "nominal". "nominal" is the x-y coordinates of regular.cases.
landmark.cases	Logical. TRUE includes landmarks as cases.
vestry	Logical. TRUE uses the 14 pumps from the Vestry Report. FALSE uses the 13 pumps from the original map.
distance.unit	Character. Unit of distance: "meter", "yard" or "native". "native" returns the map's native scale. See vignette("roads") for information on unit distances.
time.unit	Character. "hour", "minute", or "second".
walking.speed	Numeric. Default is 5 km/hr.
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 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))
```

---

`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` ( $\geq$  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	<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

**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

**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")
```

---

`landmark.squares`*Centers of city squares.*

---

**Description**

Centers of city squares.

**Usage**

```
landmark.squares
```

**Format**

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

name square name

x x-coordinate

y y-coordinate

case numeric case ID

---

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

---

**Description**

Nominal and orthogonal coordinates

**Usage**

```
landmarkData(multi.core = 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. On Windows, only <code>multi.core = FALSE</code> is available.
------------	---

---

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 6 variables that records the position of the orthogonal projection of landmarks onto the network of roads.

road.segment	"address" road segment
x.proj	orthogonal x-coordinate
y.proj	orthogonal y-coordinate
ortho.dist	orthogonal distance to home road segment
x	nominal x-coordinate
y	nominal y-coordinate
name	landmark name
case	numeric case ID

**Note**

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

---

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

---

**Description**

Compute xlim and ylim of Snow's map.

**Usage**

```
mapRange()
```

---

nearestPump	<i>Compute shortest walking distances or paths.</i>
-------------	---

---

**Description**

Compute shortest walking distances or paths.

**Usage**

```
nearestPump(pump.select = NULL, output = "distance", vestry = FALSE,
  weighted = TRUE, case.set = "observed", distance.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".
distance.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", embed = TRUE,
  embed.landmarks = TRUE)
```

**Arguments**

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

**Value**

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

---

neighborhoodEuclidean      *Compute Euclidean path pump neighborhoods.*

---

**Description**

Plots star graph from pump to its cases.

**Usage**

```
neighborhoodEuclidean(pump.select = NULL, vestry = FALSE,
  case.location = "nominal", case.set = "observed",
  multi.core = 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 "nominal". For <code>observed = TRUE</code> : "address" uses <code>ortho.proj</code> and "nominal" uses <code>fatalities</code> . For <code>observed = FALSE</code> : "address" uses <code>sim.ortho.proj</code> and "nominal" uses <code>regular.cases</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. On Windows, only <code>multi.core = FALSE</code> is available.

**Value**

An R vector.

**Note**

This function is computationally intensive when `case.set = "expected"`.

**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 = "nominal", statistic = NULL,
  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. For <code>observed = FALSE</code> : "address" or "nominal". "address" uses the x-y coordinates of <code>ortho.proj</code> . "nominal" uses the x-y coordinates of fatalities.
<code>statistic</code>	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.
<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 = 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. On Windows, only <code>multi.core = FALSE</code> 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 30 seconds. Using the parallel implementation on 4 physical (8 logical) cores, these times fall to about 4 and 13 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.

---

orthogonalProjection *Compute coordinates of orthogonal projection from case to road segment.*

---

**Description**

Compute coordinates of orthogonal projection from case to road segment.

**Usage**

```
orthogonalProjection(case = 12, segment.id = "216-1",
  observed = TRUE, use.pump = FALSE, vestry = FALSE,
  case.data = NULL)
```

**Arguments**

case	Numeric. case ID from fatalities.
segment.id	Character. Road segment ID.
observed	Logical. FALSE observed case; TRUE simulated case (regular.cases).
use.pump	Logical. Use pump ID as case.
vestry	Logical. Use vestry pump data.
case.data	Object. For use with simulateFatalities.

**Value**

An R data frame.

---

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.classifier\_audit *Plot result of classifierAudit().*

---

### Description

Plot case, segment and orthogonal projector.

### Usage

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

### Arguments

x	An object of class "classifier_audit" created by classifierAudit().
zoom	Logical or Numeric. A numeric value $\geq 0$ controls the degree of zoom. The default is 0.5.
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 neighborhoodEuclidean().*

---

### Description

Plot method for neighborhoodEuclidean().

### Usage

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

**Arguments**

x	An object of class "euclidean" created by neighborhoodEuclidean().
type	Character. "star", "area.points" or "area.polygons". "area" flavors only valid when case.set = "expected".
polygon.method	Character. Method of computing polygon vertices: "pearl.string" or "traveling.salesman".
add.observed.points	Logical. Add observed fatality "addresses".
msg	Logical. Toggle in-progress messages.
...	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 $\geq 0$ 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.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**

```
## Not run:

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

## End(Not run)
```

---

```
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, voronoi.cells = TRUE,
     delauny.triangles = FALSE, euclidean.paths = FALSE, ...)
```

**Arguments**

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

**Value**

A base R graph.

**See Also**

neighborhoodVoronoi()  
addVoronoi()

**Examples**

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

---

plot.walking	<i>Plot method for neighborhoodWalking().</i>
--------------	---

---

**Description**

Plot method for neighborhoodWalking().

**Usage**

```
## S3 method for class 'walking'
plot(x, type = "road",
     polygon.method = "pearl.string", msg = FALSE, ...)
```

**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".
polygon.method	Character. Method of computing polygon vertices: "pearl.string" or "traveling.salesman".
msg	Logical. Toggle in-progress messages.
...	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 = 0.5, 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 $\geq 0$ 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 = "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)
```

---

```
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 neighborhoodEuclidean().
```

---

**Description**

Print method for neighborhoodEuclidean().

**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`    *Print method for euclideanPath().*

---

**Description**

Summary output.

**Usage**

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

**Arguments**

- x                    An object of class "euclidean\_path" created by euclideanPath().
- ...                   Additional parameters.

**Value**

An R data frame.

**Examples**

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

---

```
print.time_series      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 observed counts 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 vector with observed counts.

**See Also**

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

**Examples**

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

---

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

---

**Description**

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

**Usage**

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

**Arguments**

<code>x</code>	An object of class "walking" created by <code>neighborhoodWalking()</code> .
<code>...</code>	Additional parameters.

**Value**

An R vector.

**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)
```

---

`profile2D`                      *2D Profile .*

---

### Description

2D Profile .

### Usage

```
profile2D(angle = 0, pump = 7, vestry = FALSE, type = "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.
type	Character. Type of graphic: "base" or "ggplot2".
multi.core	Logical or Numeric. TRUE uses <code>parallel::detectCores()</code> . FALSE uses one, single core. You can also specify the number logical cores. On Windows, only <code>multi.core = FALSE</code> is available.

**Examples**

```
## Not run:

profile2D(angle = 30)
profile2D(angle = 30, type = "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 = 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.
pump.subset	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> .
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. On Windows, only <code>multi.core = FALSE</code> is available.

**Examples**

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

---

pumpCase

*Extract numeric case IDs by pump neighborhood.*

---

**Description**

Extract numeric case IDs by pump neighborhood.

**Usage**

```
pumpCase(x)
```

**Arguments**

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

**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                      *Compute pump coordinates.*

---

### Description

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

### Usage

```
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 <code>parallel::detectCores()</code> . FALSE uses one, single core. With Numeric, you specify the number logical cores (rounds with <code>as.integer()</code> ). On Windows, only <code>multi.core = FALSE</code> 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 = 7, zoom = 1, vestry = FALSE, add.title = TRUE,  
            highlight.segment = TRUE, data = FALSE)
```

**Arguments**

id	Numeric or Integer. With vestry = TRUE, a whole number between 1 and 14. With vestry = FALSE, a whole number between 1 and 13. See cholera::pumps.vestry and cholera::pumps for IDs and details about specific pumps.
zoom	Logical or Numeric. A numeric value $\geq 0$ controls the degree of zoom. The default is 1.
vestry	Logical. TRUE for the 14 pumps from Vestry Report. FALSE for the original 13 pumps.
add.title	Logical. Include title.
highlight.segment	Logical. Highlight case's segment.
data	Logical. Output data.

**Value**

A base R graphics plot.

**See Also**

[pumpData](#)

**Examples**

```
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

### 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	<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 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	<i>Dodson and Tobler's street data with appended road names.</i>
-------	--

---

### 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	<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()
```

**Value**

An R data frame.

**Note**

This function documents the code that generates [road.segments](#).

---

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

---

**Description**

Compute length of road segment.

**Usage**

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

**Arguments**

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

**Value**

An R vector of length one.

**Examples**

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

---

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

---

### Description

Highlights the selected road segment and its cases.

### Usage

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

### Arguments

id	Character. A concatenation of a street's numeric ID, a whole number between 1 and 528, and a second number to identify the segment.
zoom	Logical or Numeric. A numeric value $\geq 0$ controls the degree of zoom. The default is 0.5.
cases	Character. Plot cases: NULL, "address" or "fatality".
distance.unit	Character. Unit of distance: "meter", "yard" or "native". "native" returns the map's native scale. See vignette("roads") for information on conversion.
time.unit	Character. "hour", "minute", or "second".
walking.speed	Numeric. Walking speed in km/hr.
add.title	Logical. Print title.
add.subtitle	Logical. Print subtitle.
highlight	Logical. Highlight selected road and its cases.

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

---

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)
```

---

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 = FALSE,
  simulated.obs = 20000L)
```

**Arguments**

<code>compute</code>	Logical. TRUE computes data. FALSE uses pre-computed data. For replication of data used in the package,
<code>multi.core</code>	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> ). On Windows, only <code>multi.core = FALSE</code> is available.
<code>simulated.obs</code>	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 105 minutes to run on a single core of a 2.3 GHz Intel Core i7 with R version 3.5.2 and 25 minutes to run on eight logical (four physical) cores. 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.cases = TRUE,  
        add.landmarks = FALSE, add.pumps = TRUE, add.roads = TRUE,  
        add.frame = TRUE, main = NA, case.col = "gray", case.pch = 15,  
        ...)
```

**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.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>...</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()
```

**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)
```

**Arguments**

road.name	Character vector. The functions tries to correct for case and to remove extra spaces.
-----------	---

**Value**

A base R graphics segment(s).

**Examples**

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

---

streetLength                    *Compute length of selected street.*

---

### Description

Compute length of selected street.

### Usage

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

### Arguments

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

### Value

An R vector of length one.

### Examples

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

---

streetNameLocator            *Locate road by name.*

---

### Description

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

### Usage

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

**Arguments**

road.name	Character vector. Note that streetNameLocator() tries to correct for case and to remove extra spaces.
zoom	Logical or Numeric. A numeric value $\geq 0$ controls the degree of zoom. The default is FALSE, which is equivalent to zero.
cases	Character. Plot cases: NULL, "address" or "fatality".
token	Character. "id" or "point".
add.title	Logical. Include title.
add.subtitle	Logical. Include subtitle with road information.
add.pump	Logical. Include nearby pumps.
vestry	Logical. TRUE uses the 14 pumps from the Vestry report. FALSE uses the 13 in the original map.
highlight	Logical. Highlight selected road and its cases.
distance.unit	Character. Unit of 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.

**Examples**

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

---

streetNumberLocator     *Locate road by numerical ID.*

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

A base R graphics plot.

**Examples**

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

---

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, distance.unit = "meter", yard.unit = 177/3,
          meter.unit = 54)
```

**Arguments**

x	Numeric. Nominal map distance.
distance.unit	Character. Unit of distance: "meter", "yard" or "native". "native" returns the map's native scale. See <a href="#">vignette("roads")</a> 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 = fixFatalities())
```

### Arguments

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

### 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 with R version 3.5.2, it takes approximately 195 seconds to run on one core and approximately 47 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")`

---

walkingPath	<i>Compute the shortest walking path between cases and/or pumps.</i>
-------------	--

---

**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)
```

**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.

**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)
```

# Index

## \*Topic **datasets**

- anchor.case, 20
- border, 20
- fatalities, 25
- fatalities.address, 26
- fatalities.unstacked, 27
- landmark.squares, 28
- landmarks, 29
- ortho.proj, 35
- ortho.proj.pump, 36
- ortho.proj.pump.vestry, 36
- plague.pit, 38
- pumps, 55
- pumps.vestry, 55
- regular.cases, 56
- road.segments, 57
- roads, 58
- sim.ortho.proj, 61
- sim.pump.case, 61
- snow.neighborhood, 63

- addCase, 5
- addDelauny, 5
- addEuclideanPath, 6
- addFrame, 7
- addIndexCase, 7
- addKernelDensity, 8
- addLandmarks, 9
- addMilePosts, 10
- addNeighborhoodCases, 11
- addNeighborhoodEuclidean, 12
- addNeighborhoodWalking, 13
- addPlaguePit, 14
- addPump, 15
- addRoads, 16
- addSnow, 16
- addVoronoi, 17
- addWalkingPath, 17
- addWhitehead, 19
- anchor.case, 20, 70

- border, 20
- caseLocator, 20, 21, 26, 27
- cholera-package, 4
- classifierAudit, 22
- deldirVertices, 22
- distanceTime, 23
- euclideanPath, 24
- fatalities, 25
- fatalities.address, 26, 70
- fatalities.unstacked, 27, 70
- fixFatalities, 26, 28
- landmark.squares, 28
- landmarkData, 29, 29
- landmarks, 29
- mapRange, 30
- nearestPump, 30
- neighborhoodData, 31
- neighborhoodEuclidean, 31
- neighborhoodVoronoi, 32
- neighborhoodWalking, 34, 62
- ortho.proj, 35, 70
- ortho.proj.pump, 36, 53
- ortho.proj.pump.vestry, 36, 53
- orthogonalProjection, 37
- pearsonResiduals, 38
- plague.pit, 38
- plot.classifier\_audit, 39
- plot.euclidean, 39
- plot.euclidean\_path, 40
- plot.neighborhood\_data, 41
- plot.profile\_perspective, 42
- plot.time\_series, 42, 69

plot.voronoi, 43  
plot.walking, 44  
plot.walking\_path, 45  
print.classifier\_audit, 46  
print.euclidean, 46  
print.euclidean\_path, 47  
print.time\_series, 48, 69  
print.voronoi, 48  
print.walking, 49  
print.walking\_path, 50  
profile2D, 50  
profile3D, 51  
pumpCase, 52  
pumpData, 36, 37, 53, 54–56  
pumpLocator, 53, 54, 55, 56  
pumps, 53, 55  
pumps.vestry, 53, 55  
  
regular.cases, 56, 62  
road.segments, 57, 58, 59  
roads, 57, 58  
roadSegments, 57, 58  
  
segmentLength, 59  
segmentLocator, 57, 58, 60  
sim.ortho.proj, 61, 62  
sim.pump.case, 61  
simulateFatalities, 56, 61, 62  
snow.neighborhood, 63  
snowColors, 63  
snowMap, 64  
snowNeighborhood, 65  
streetHighlight, 65  
streetLength, 66  
streetNameLocator, 26, 27, 57, 58, 66  
streetNumberLocator, 26, 27, 57, 58, 67  
  
timeSeries, 43, 68  
  
unitMeter, 69  
unstackFatalities, 20, 27, 35, 70  
  
walkingPath, 71