

# Package ‘cholera’

March 14, 2025

**Type** Package

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

**Version** 0.9.0

**Date** 2025-03-14

**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 (>= 3.5)

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

**Suggests** ggplot2, knitr, rmarkdown

**VignetteBuilder** knitr

**RoxygenNote** 7.3.2

**Encoding** UTF-8

**Language** en-US

**NeedsCompilation** no

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

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

**Repository** CRAN

**Date/Publication** 2025-03-14 00:40:02 UTC

## Contents

cholera-package . . . . .	4
addCase . . . . .	5
addDelaunay . . . . .	6
addEuclideanPath . . . . .	7
addIndexCase . . . . .	8
addKernelDensity . . . . .	8
addLandmarks . . . . .	10
addLandmarkSquares . . . . .	10
addNeighborhoodCases . . . . .	11
addNeighborhoodEuclidean . . . . .	12
addNeighborhoodSnow . . . . .	14
addNeighborhoodWalking . . . . .	14
addPlaguePit . . . . .	16
addPump . . . . .	16
addRoads . . . . .	17
addVoronoi . . . . .	17
addWalkingPath . . . . .	18
addWhitehead . . . . .	19
anchor.case . . . . .	20
border . . . . .	21
caseLocator . . . . .	21
euclideanPath . . . . .	22
fatalities . . . . .	23
fatalities.address . . . . .	24
fatalities.unstacked . . . . .	25
frame.data . . . . .	25
frame.sample . . . . .	26
landmark.squares . . . . .	26
landmarks . . . . .	27
latlong.ortho.addr . . . . .	28
latlong.ortho.pump . . . . .	28
latlong.ortho.pump.vestry . . . . .	29
latlong.regular.cases . . . . .	30
latlong.sim.ortho.proj . . . . .	30
mapRange . . . . .	31
meter.to.yard . . . . .	31
neighborhoodData . . . . .	32
neighborhoodEuclidean . . . . .	32
neighborhoodSnow . . . . .	33
neighborhoodVoronoi . . . . .	34
neighborhoodWalking . . . . .	35
ortho.proj . . . . .	36
ortho.proj.pump . . . . .	36
ortho.proj.pump.vestry . . . . .	37
pearsonResiduals . . . . .	38
plague.pit . . . . .	38

plot.euclidean . . . . .	39
plot.euclideanLatlong . . . . .	40
plot.euclidean_path . . . . .	40
plot.neighborhood_data . . . . .	41
plot.profile_perspective . . . . .	42
plot.snow . . . . .	42
plot.time_series . . . . .	43
plot.voronoi_latlong . . . . .	44
plot.voronoi_nominal . . . . .	44
plot.walking . . . . .	45
plot.walking_path . . . . .	46
print.euclidean . . . . .	46
print.euclideanLatlong . . . . .	47
print.euclidean_path . . . . .	48
print.time_series . . . . .	48
print.voronoi_latlong . . . . .	49
print.voronoi_nominal . . . . .	49
print.walking . . . . .	50
print.walking_path . . . . .	50
profile2D . . . . .	51
profile3D . . . . .	51
pumpCase . . . . .	52
pumpData . . . . .	53
pumpFatalities . . . . .	54
pumpLocator . . . . .	54
pumps . . . . .	55
pumps.vestry . . . . .	56
rd.sample . . . . .	57
rectangle.filter . . . . .	57
regular.cases . . . . .	58
road.segments . . . . .	58
roads . . . . .	59
roadSegments . . . . .	60
segmentHighlight . . . . .	60
segmentLocator . . . . .	61
sim.ortho.proj . . . . .	62
sim.pump.case . . . . .	63
sim.walking.distance . . . . .	63
simulateFatalities . . . . .	64
snow.neighborhood . . . . .	65
snowColors . . . . .	65
snowMap . . . . .	66
streetHighlight . . . . .	67
streetLength . . . . .	67
streetNameLocator . . . . .	68
streetNames . . . . .	69
streetNumberLocator . . . . .	69
summary.euclidean . . . . .	70

summary.voronoi_nominal . . . . .	71
tanakaContourPlot . . . . .	72
timeSeries . . . . .	72
unstackFatalities . . . . .	73
voronoi.polygons . . . . .	74
voronoi.polygons.vestry . . . . .	74
voronoiPolygons . . . . .	75
walkingPath . . . . .	76

<b>Index</b>	<b>77</b>
--------------	-----------

---

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

---

## Description

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

## Details

- Fixes two sets of errors in Dodson and Tobler's 1992 digitization of Snow's map: 1) three misplaced cases/fatalities and 2) one missing road segment (part of Clifford Street).
- "Unstacks" the data in two ways to make analysis and visualization easier and more meaningful.
- Computes and visualizes "pump neighborhoods" based on Voronoi tessellation, Euclidean distance, and walking distance.
- Overlay graphical elements and features like kernel density estimates, Voronoi diagrams, Snow's Broad Street neighborhood, and notable landmarks (John Snow's residence, the Lion Brewery, etc.) via 'add\*()' functions.
- Includes a variety of functions to highlight specific cases, roads, pumps and paths.
- Appends actual street names to roads data.
- Includes the revised pump data used in the second version of Snow's map from the Vestry report, which includes the "correct" location of the Broad Street pump.
- Adds two different aggregate time series fatalities data sets, taken from the Vestry report.
- Support for parallel computation on Linux and macOS; limited support for Windows.
- Version  $\geq 0.9.0$ , offers provisional support for (georeferenced) longitude and latitude for practically all data and functions.

To learn more, see the vignettes:

```
vignette("duplicate.missing.cases")
vignette("kernel.density")
vignette("parallelization")
vignette("pump.neighborhoods")
```

```
vignette("roads")
vignette("tiles.polygons")
vignette("time.series")
vignette("unstacking.bars")
```

### Author(s)

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

### See Also

Useful links:

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

---

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

---

### Description

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

### Usage

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

### Arguments

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

**Note**

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

**Examples**

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

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

---

addDelaunay	<i>Add Delaunay triangles.</i>
-------------	--------------------------------

---

**Description**

Add Delaunay triangles.

**Usage**

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

**Arguments**

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

**Note**

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

**Examples**

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

---

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

---

### Description

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

### Usage

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

### Arguments

origin	Numeric. Vector of origin(s) (numeric ID or character name landmark/pump).
destination	Numeric. Vector of destination(s) (numeric or landmark/pump name).
type	Character. Path case to pump. FALSE is all other combinations of cases, landmarks and pumps.
vestry	Logical. TRUE uses the 14 pumps from the map in the Vestry Report. FALSE uses the 13 pumps from the original map.
latlong	Logical.
case.set	Character. "observed" or "expected".
location	Character. For cases and pumps. "nominal", "anchor" or "orthogonal".
weighted	Logical. TRUE computes shortest path in terms of road length. FALSE computes shortest path in terms of the number of nodes.
distance.unit	Character. Unit of distance: "meter" or "yard".
time.unit	Character. "hour", "minute", or "second".
walking.speed	Numeric. Walking speed in km/hr.
include.landmarks	Logical. Include landmarks as cases.
long.title	Logical. Tile with names.
mileposts	Logical. Plot mile/time posts.
milepost.unit	Character. "distance" or "time".
milepost.interval	Numeric. Mile post interval unit of distance (yard or meter) or unit of time (seconds).
alpha.level	Numeric. Alpha level transparency for path: a value in [0, 1].

---

addIndexCase                    *Highlight index case at 40 Broad Street.*

---

### Description

Highlight index case at 40 Broad Street.

### Usage

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

### Arguments

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

### Value

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

### Examples

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

---

addKernelDensity                *Add 2D kernel density contours.*

---

### Description

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

### Usage

```
addKernelDensity(pump.subset = "pooled", pump.select = NULL,
                 neighborhood.type = "walking", data = "unstacked", bandwidth = 0.5,
                 color = "black", line.type = "solid", multi.core = FALSE,
                 latlong = 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>data</code>	Character. Unit of observation: "unstacked" uses <code>fatalities.unstacked</code> ; "address" uses <code>fatalities.address</code> ; "fatality" uses <code>fatalities</code> .
<code>bandwidth</code>	Numeric. Bandwidth for kernel density estimation.
<code>color</code>	Character. Color of contour lines.
<code>line.type</code>	Character. Line type for contour lines.
<code>multi.core</code>	Logical or Numeric. TRUE uses <code>parallel::detectCores()</code> . FALSE uses one, single core. You can also specify the number logical cores. See <code>vignette("Parallelization")</code> for details.
<code>latlong</code>	Logical.

**Value**

Add contours to a graphics plot.

**Note**

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

**Examples**

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

snowMap()
addKernelDensity("individual")

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

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

## End(Not run)
```

addLandmarks      *Add landmarks to plot.*

---

**Description**

Add landmarks to plot.

**Usage**

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

**Arguments**

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

**Value**

Base R points and text.

**Note**

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

**Examples**

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

---

addLandmarkSquares      *Add Golden and Soho Squares to plot.*

---

**Description**

Add Golden and Soho Squares to plot.

**Usage**

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

**Arguments**

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

**Value**

Base R points and text.

**Examples**

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

---

addNeighborhoodCases *Add observed neighborhood cases.*

---

**Description**

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

**Usage**

```
addNeighborhoodCases(pump.subset = NULL, pump.select = NULL,
  metric = "walking", case.set = "observed", location = "nominal",
  token = "point", text.size = 0.5, pch = 16, point.size = 0.5,
  vestry = FALSE, weighted = TRUE, color = NULL, alpha.level = 0.5,
  latlong = FALSE, 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".
case.set	Character. "observed" or "expected".
location	Character. "nominal", "anchor" or "orthogonal".
token	Character. Type of token to plot: "point" or "id".
text.size	Numeric. Size of case ID text.
pch	Numeric.
point.size	Numeric.

vestry	Logical. TRUE uses the 14 pumps from the Vestry Report. FALSE uses the 13 in the original map.
weighted	Logical. TRUE computes shortest walking path weighted by road length. FALSE computes shortest walking path in terms of the number of nodes.
color	Character. Use a single color for all paths. NULL uses neighborhood colors defined by snowColors().
alpha.level	Numeric. Alpha level transparency for area plot: a value in [0, 1].
latlong	Logical. Longitude and latitude coordinates.
multi.core	Logical or Numeric. TRUE uses parallel::detectCores(). FALSE uses one, single core. You can also specify the number logical cores. See vignette("Parallelization") for details.

### Examples

```
## Not run:
snowMap(add.cases = FALSE)
addNeighborhoodCases()

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

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

snowMap(add.cases = FALSE, latlong = TRUE)
addNeighborhoodCases(latlong = TRUE)

snowMap(add.cases = FALSE, latlong = TRUE)
addNeighborhoodCases(pump.subset = c(6, 10), latlong = TRUE)

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

## End(Not run)
```

---

```
addNeighborhoodEuclidean
```

*Add expected Euclidean pump neighborhoods.*

---

### Description

Plots star graph from pump to its cases.

**Usage**

```
addNeighborhoodEuclidean(pump.select = NULL, vestry = FALSE,
  case.set = "observed", case.select = "address", latlong = FALSE,
  location = "nominal", brute.force = FALSE, type = "star",
  add.observed.points = TRUE, alpha.level = 0.25, polygon.type = "solid",
  multi.core = FALSE, dev.mode = FALSE)
```

**Arguments**

pump.select	Numeric. Vector of numeric pump IDs to define pump neighborhoods (i.e., the "population"). Negative selection possible. NULL selects all pumps.
vestry	Logical. TRUE uses the 14 pumps from the Vestry Report. FALSE uses the 13 in the original map.
case.set	Character. "observed" or "expected".
case.select	Character. Fatalities: "all" or "address".
latlong	Logical. Longitude and latitude coordinates
location	Character. "nominal", "anchor" or "orthogonal".
brute.force	Logical. For latlong = FALSE. TRUE computes nearest pump for each case. FALSE uses Voronoi cells as shortcut.
type	Character. "star", "area.points" or "area.polygons". "area" flavors only valid when case.set = "expected".
add.observed.points	Logical. Add observed fatality "addresses".
alpha.level	Numeric. Alpha level transparency for area plot: a value in [0, 1].
polygon.type	Character. "border" or "solid".
multi.core	Logical or Numeric. TRUE uses parallel::detectCores(). FALSE uses one, single core. You can also specify the number logical cores. See vignette("Parallelization") for details.
dev.mode	Logical. Development mode uses parallel::parLapply().

**Value**

R graphic elements.

**Examples**

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

streetNameLocator("marshall street", zoom = -50, highlight = FALSE)
addNeighborhoodEuclidean(case.set = "expected", type = "area.polygons")

## End(Not run)
```

---

addNeighborhoodSnow     *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

```
addNeighborhoodSnow(latlong = FALSE, type = "area.polygons",
  non.snow.cases = FALSE, alpha.level = 1/3, polygon.type = "solid",
  polygon.col = NULL, polygon.lwd = NULL)
```

### Arguments

latlong	Logical.
type	Character. Type of annotation plot: "area.points", "area.polygons", or "roads".
non.snow.cases	Logical. Plot anchor cases outside Snow neighborhood.
alpha.level	Numeric. Alpha level transparency for area plot: a value in [0, 1].
polygon.type	Character. "border" or "solid".
polygon.col	Character.
polygon.lwd	Numeric.

### Examples

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

## End(Not run)
```

---

addNeighborhoodWalking     *Add walking neighborhoods.*

---

### Description

Add walking neighborhoods.

**Usage**

```
addNeighborhoodWalking(pump.select = NULL, vestry = FALSE,
  weighted = TRUE, case.set = "expected", latlong = FALSE,
  type = "area.polygons", tsp.method = "repetitive_nn", path.width = 2,
  alpha.level = 0.5, polygon.type = "solid", polygon.col = NULL,
  polygon.lwd = 2, 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. Note that you can't just select the pump on Adam and Eve Court (#2) because it's technically an isolate.
<code>vestry</code>	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>case.set</code>	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.
<code>latlong</code>	Logical.
<code>type</code>	Character. "roads", "area.points" or "area.polygons". "area" flavors only valid when <code>case.set = "expected"</code> .
<code>tsp.method</code>	Character. Traveling salesperson problem algorithm.
<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. "border" 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. See <code>vignette("Parallelization")</code> for details.

**Examples**

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

## End(Not run)
```

---

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

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

Adds a polygon to a graphics plot.

**Note**

In progress.

**Examples**

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

---

addPump	<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, cex = 1, latlong = FALSE)
```



**Arguments**

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

---

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

---

**Description**

Add all streets and roads to plot.

**Usage**

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

**Arguments**

latlong	Logical. Use estimated longitude and latitude.
col	Character. Color

---

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

---

**Description**

Add Voronoi cells.

**Usage**

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

**Arguments**

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

**Note**

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

**Examples**

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

---

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

---

**Description**

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

**Usage**

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

**Arguments**

<code>origin</code>	Numeric. Vector of origin(s) (numeric or case/landmark name).
<code>destination</code>	Numeric. Vector of destination(s) (numeric or landmark/pump name).
<code>type</code>	Character. Path case to pump. FALSE is all other combinations of cases, landmarks and pumps.

vestry	Logical. TRUE uses the 14 pumps from the map in the Vestry Report. FALSE uses the 13 pumps from the original map.
latlong	Logical.
case.set	Character. "observed" or "expected".
location	Character. For cases and pumps. "anchor", "fatality" or "orthogonal".
weighted	Logical. TRUE computes shortest path in terms of road length. FALSE computes shortest path in terms of the number of nodes.
distance.unit	Character. Unit of distance: "meter" or "yard".
time.unit	Character. "hour", "minute", or "second".
walking.speed	Numeric. Walking speed in km/hr.
include.landmarks	Logical. Include landmarks as cases.
long.title	Logical. Tile with names.
mileposts	Logical. Plot mile/time posts.
milepost.unit	Character. "distance" or "time".
milepost.interval	Numeric. Mile post interval unit of distance (yard or meter) or unit of time (seconds).
alpha.level	Numeric. Alpha level transparency for path: a value in [0, 1].
...	Additional plotting parameters.

---

addWhitehead

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


---

### Description

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

### Usage

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

### Arguments

pump	Character or Numeric. Name (road name) or numerical ID of selected pump. See pumps or pumps.vestry.
radius	Numeric. Distance from a pump.
distance.unit	Character. Unit of distance: "meter", "yard" or "native". "native" returns the map's native scale. See vignette("roads") for information on conversion.

latlong	Logical. Longitude-Latitude coordinates.
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.title	Logical. Add title.
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()
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 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 = FALSE, observed = TRUE, latlong = FALSE,
  add.title = TRUE, highlight.segment = TRUE, data = FALSE,
  add = FALSE, col = "red", vestry = FALSE)
```

**Arguments**

case	Numeric or Integer. Whole number between 1 and 578.
zoom	Logical or Numeric. Positive numbers zoom in; negative numbers zoom out.
observed	Logical. TRUE for observed. FALSE for simulated.
latlong	Logical. Longitude and latitude coordinates
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.
vestry	Logical. TRUE uses the 14 pumps from the Vestry report. FALSE uses the 13 in the original map.

**Value**

A base R graphics plot.

**Examples**

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

---

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

---

**Description**

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

**Usage**

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

**Arguments**

origin	Numeric. Vector of origin(s) (numeric ID or character name landmark/pump).
destination	Numeric. Vector of destination(s) (numeric or landmark/pump name).
type	Character. Path case to pump. FALSE is all other combinations of cases, landmarks and pumps.
vestry	Logical. TRUE uses the 14 pumps from the map in the Vestry Report. FALSE uses the 13 pumps from the original map.
latlong	Logical.
case.set	Character. "observed" or "expected".
location	Character. For cases and pumps. "nominal", "anchor" or "orthogonal".
weighted	Logical. TRUE computes shortest path in terms of road length. FALSE computes shortest path in terms of the number of nodes.
distance.unit	Character. Unit of distance: "meter" or "yard".
time.unit	Character. "hour", "minute", or "second".
walking.speed	Numeric. Walking speed in km/hr.
include.landmarks	Logical. Include landmarks as cases.

---

`fatalities`*Amended Dodson and Tobler's cholera data.*

---

**Description**

An amended version of Dodson and Tobler's digitization of John Snow's map of the 1854 London cholera outbreak. It removes 3 duplicate observations and imputes the location for 3 "missing" observation. This information is also available in `HistData::Snow.deaths2` ( $\geq$  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`lon` longitude`lat` latitude**Note**

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

**See Also**[caseLocator](#)[streetNameLocator](#)[streetNumberLocator](#)[caseLocator](#)[streetNameLocator](#)[streetNumberLocator](#)

---

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

---

### Description

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

### Usage

```
fatalities.address
```

### Format

A data frame with 4 variables for 321 addresses

anchor numerical case ID of address

x x-coordinate

y y-coordinate

case.count number of fatalities at address

lon longitude

lat latitude

### Note

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

### See Also

[caseLocator](#)

[streetNameLocator](#)

[streetNumberLocator](#)



---

`fatalities.unstacked` *"Unstacked" amended cholera fatalities data with fatality as unit of observation.*

---

**Description**

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

**Usage**

```
fatalities.unstacked
```

**Format**

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

`case` numerical case ID

`x` x-coordinate

`y` y-coordinate

`lon` longitude

`lat` latitude

**Note**

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

**See Also**

[caseLocator](#)

[streetNameLocator](#)

[streetNumberLocator](#)

---

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

---

**Description**

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

**Usage**

```
frame.data
```

**Format**

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

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

---

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

---

**Description**

Partitioned map frame points (segment endpoints).

**Usage**

frame.sample

**Format**

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

frame.sample cholera::roads\$id

---

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

---

**Description**

Centers of city squares.

**Usage**

landmark.squares

**Format**

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

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

---

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

---

**Description**

Landmark coordinates.

**Usage**

landmarks

**Format**

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

case numeric case ID  
 road.segment "address" road segment  
 x nominal x-coordinate  
 y nominal y-coordinate  
 x.lab label x-coordinate  
 y.lab label y-coordinate  
 lon longitude  
 lat latitude  
 lon.lab label longitude  
 lat.lab label latitude  
 name landmark name

---

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

---

**Description**

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

**Usage**

latlong.ortho.addr

**Format**

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

road.segment "address" road segment

x.proj x-coordinate

y.proj y-coordinate

ortho.dist orthogonal distance to home road segment

case numeric case ID

lon longitude

lat latitude

**Note**

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

---

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

---

**Description**

Orthogonal projection of 13 original pumps (latlong).

**Usage**

latlong.ortho.pump

**Format**

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

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

**Note**

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

---

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

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

---

**Description**

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

**Usage**

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

**Format**

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

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

**Note**

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

---

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

---

**Description**

The result of using `sp::spsample()` and `sp::Polygon()` to generate 19,993 regularly spaced simulated Cartesian/geodesic cases within the map's borders.

**Usage**

```
latlong.regular.cases
```

**Format**

A data frame with 4 variables that records the position of 19,993 "expected" cases fitted by `sp::spsample()`.

x x-coordinate

y y-coordinate

lon longitude

lat latitude

---

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

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

---

**Description**

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

**Usage**

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

**Format**

A data frame with 8 variables that records the "address" of 19,993 regularly spaced simulated Cartesian/geodesic cases regularly spaced across map.

case numeric case ID

road.segment "address" road segment

x.proj x-coordinate

y.proj y-coordinate

dist Euclidean or orthogonal distance to home road segment

type type of projection: Euclidean ("eucl") or orthogonal ("ortho")

lon longitude

lat latitude

---

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

---

**Description**

Compute xlim and ylim of Snow's map.

**Usage**

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

**Arguments**

latlong            Logical. Use estimated longitude and latitude.

---

meter.to.yard	<i>Meter to yard conversion factor.</i>
---------------	---

---

**Description**

Meter to yard conversion factor.

**Usage**

```
meter.to.yard
```

**Format**

A vector of length one.

```
meter.to.yard conversion factor = 1.093613
```

---

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

---

### Description

Assembles cases, pumps and road into a network graph.

### Usage

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

### Arguments

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

### Value

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

---

neighborhoodEuclidean      *Compute Euclidean path pump neighborhoods.*

---

### Description

Plots star graph from pump to its cases.

### Usage

```
neighborhoodEuclidean(pump.select = NULL, vestry = FALSE,
  case.set = "observed", case.select = "address", latlong = FALSE,
  location = "nominal", brute.force = FALSE, multi.core = FALSE,
  dev.mode = FALSE)
```



**Arguments**

<code>pump.select</code>	Numeric. Vector of numeric pump IDs to define pump neighborhoods (i.e., the "population"). Negative selection possible. NULL selects all pumps.
<code>vestry</code>	Logical. TRUE uses the 14 pumps from the Vestry Report. FALSE uses the 13 in the original map.
<code>case.set</code>	Character. "observed" or "expected".
<code>case.select</code>	Character. Fatalities: "all" or "address".
<code>latlong</code>	Logical. Longitude and latitude coordinates
<code>location</code>	Character. "nominal", "anchor" or "orthogonal".
<code>brute.force</code>	Logical. For <code>latlong = FALSE</code> . TRUE computes nearest pump for each case. FALSE uses Voronoi cells as shortcut.
<code>multi.core</code>	Logical or Numeric. TRUE uses <code>parallel::detectCores()</code> . FALSE uses one, single core. You can also specify the number logical cores. See <code>vignette("Parallelization")</code> for details.
<code>dev.mode</code>	Logical. Development mode uses <code>parallel::parLapply()</code> .

**Value**

An R list.

---

<code>neighborhoodSnow</code>	<i>Compute Snow's graphical annotation of Broad Street pump neighborhood.</i>
-------------------------------	---

---

**Description**

Computational approximation.

**Usage**

```
neighborhoodSnow(latlong = FALSE)
```

**Arguments**

<code>latlong</code>	Logical. Use estimated longitude and latitude.
----------------------	--

---

neighborhoodVoronoi    *Compute Voronoi pump neighborhoods.*

---

### Description

Group cases into neighborhoods using Voronoi tessellation.

### Usage

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

### Arguments

pump.select	Numeric. Vector of numeric pump IDs to define pump neighborhoods (i.e., the "population"). Negative selection possible. NULL selects all pumps.
vestry	Logical. TRUE uses the 14 pumps from the Vestry report. FALSE uses the 13 in the original map.
latlong	Logical. Longitude and latitude coordinates
location	Character. "nominal" or "orthogonal". "nominal" uses the x-y coordinates of fatalities.address. "orthogonal" uses the x-y coordinates of ortho.proj.
polygon.vertices	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**

```
## Not run:
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

## End(Not run)
```

---

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

---

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.

case numeric case ID

road.segment "address" road segment

x.proj x-coordinate

y.proj y-coordinate

dist distance to home road segment

type type of distance: orthogonal to road or Euclidean to endpoint

**Note**

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

---

ortho.proj.pump

*Orthogonal projection of 13 original pumps.*

---

**Description**

Orthogonal projection of 13 original pumps.

**Usage**

ortho.proj.pump

**Format**

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

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

**Note**

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

---

ortho.proj.pump.vestry

*Orthogonal projection of the 14 pumps from the Vestry Report.*

---

**Description**

Orthogonal projection of the 14 pumps from the Vestry Report.

**Usage**

ortho.proj.pump.vestry

**Format**

A data frame with 6 variables that records the position of the orthogonal projection of the 14 pumps onto the network of roads.

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

**Note**

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

---

pearsonResiduals      *Compute Pearson Residuals (prototype)*

---

**Description**

Compute Pearson Residuals (prototype)

**Usage**

```
pearsonResiduals(x)
```

**Arguments**

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

**Value**

An R vector.

**Examples**

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

## End(Not run)
```

---

plague.pit              *Plague pit coordinates.*

---

**Description**

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

**Usage**

```
plague.pit
```

**Format**

A data frame with 13 observations and 2 variables.

x x-coordinate

y y-coordinate

---

plot.euclidean      *Plot method for neighborhoodEuclidean().*

---

### Description

Plot method for neighborhoodEuclidean().

### Usage

```
## S3 method for class 'euclidean'
plot(x, type = "star", add = FALSE,
     add.observed.points = TRUE, alpha.level = 0.75, polygon.type = "solid",
     ...)
```

### Arguments

x	An object of class "euclidean" created by neighborhoodEuclidean().
type	Character. "star", "area.points" or "area.polygons". "area" flavors only valid when case.set = "expected".
add	Logical. Add graphic to an existing plot.
add.observed.points	Logical. Add observed fatality "addresses".
alpha.level	Numeric. Alpha level transparency for area plot: a value in [0, 1].
polygon.type	Character. "perimeter" or "solid".
...	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.euclideanLatlong *Plot method for euclideanLatlong()*

---

### Description

Plot method for euclideanLatlong()

### Usage

```
## S3 method for class 'euclideanLatlong'
plot(x, type = "star", add = FALSE,
     add.observed.points = TRUE, alpha.level = 0.75, polygon.type = "solid",
     ...)
```

### Arguments

x	Object.
type	Character. "star", "area.points" or "area.polygons". "area" flavors only valid when case.set = "expected".
add	Logical. Add graphic to an existing plot.
add.observed.points	Logical. Add observed fatality "addresses".
alpha.level	Numeric. Alpha level transparency for area plot: a value in [0, 1].
polygon.type	Character. "perimeter" or "solid".
...	Additional plotting parameters.

---

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, add = FALSE,
     long.title = TRUE, mileposts = TRUE, milepost.unit = "distance",
     milepost.interval = NULL, alpha.level = 1, ...)
```



**Arguments**

x	An object of class "euclidean_path" created by euclideanPath().
zoom	Logical or Numeric. Positive numbers zoom in; negative numbers zoom out.
add	Logical. Add graphic to plot.
long.title	Logical. Tile with names.
mileposts	Logical. Plot mile/time posts.
milepost.unit	Character. "distance" or "time".
milepost.interval	Numeric. Mile post interval unit of distance (yard or meter) or unit of time (seconds).
alpha.level	Numeric. Alpha level transparency for path: a value in [0, 1].
...	Additional plotting parameters.

**Value**

A base R plot.

---

plot.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.

---

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.snow            *Plot method for neighborhoodSnow().*

---

**Description**

Plot method for neighborhoodSnow().

**Usage**

```
## S3 method for class 'snow'
plot(x, type = "area.polygons", non.snow.cases = TRUE,
     alpha.level = 1/3, polygon.type = "solid", polygon.col = NULL,
     polygon.lwd = NULL, add = FALSE, ...)
```

**Arguments**

x                    An object of class "neighborhood\_snow" created by neighborhoodSnow().  
 type                Character. "roads", "area.points" or "area.polygons".  
 non.snow.cases    Logical. Plot anchor cases outside Snow neighborhood.  
 alpha.level        Numeric. Alpha level transparency for area plot: a value in [0, 1].  
 polygon.type      Character. "border" or "solid".  
 polygon.col        Character.  
 polygon.lwd        Numeric.  
 add                Logical. Add graphic to plot.  
 ...                Additional plotting parameters.

---

plot.time_series	<i>Plot aggregate time series data from Vestry report.</i>
------------------	--

---

### 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_latlong` *Plot method for voronoiLatlong()*

---

### Description

Plot method for voronoiLatlong()

### Usage

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

### Arguments

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

---

`plot.voronoi_nominal` *Plot Voronoi neighborhoods.*

---

### Description

Plot Voronoi neighborhoods.

### Usage

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

### Arguments

<code>x</code>	An object of class "voronoi" created by voronoiNominal().
<code>delaunay.voronoi</code>	Character "delaunay", "voronoi", or "both".
<code>euclidean.paths</code>	Logical. Plot all Euclidean paths (star graph).
<code>...</code>	Additional plotting parameters.

**Value**

A base R graph.

**See Also**

voronoiNominal()  
addVoronoi()

---

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

---

**Description**

Plot method for neighborhoodWalking().

**Usage**

```
## S3 method for class 'walking'
plot(x, type = "area.points", add = FALSE,
     tsp.method = "repetitive_nn", path.width = 2, alpha.level = 0.75,
     polygon.type = "solid", polygon.col = NULL, polygon.lwd = 2, ...)
```

**Arguments**

x	An object of class "walking" created by neighborhoodWalking().
type	Character. Type of expected plot: "roads", "area.points" or "area.polygons". Valid only when case.set = "expected".
add	Logical. Add graphic to plot.
tsp.method	Character. Traveling salesperson problem algorithm.
path.width	Numeric. Set width of paths.
alpha.level	Numeric. Alpha level transparency for area plot: a value in [0, 1].
polygon.type	Character. "border" or "solid".
polygon.col	Character.
polygon.lwd	Numeric.
...	Additional plotting parameters.

**Value**

A base R plot.

**Note**

When plotting area graphs with simulated data (i.e., case.set = "expected"), there may be discrepancies between observed cases and expected neighborhoods, particularly between neighborhoods. type = "roads" inspired by Shiode et. al. (2015).

---

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, add = FALSE,
     long.title = TRUE, mileposts = TRUE, milepost.unit = "distance",
     milepost.interval = NULL, alpha.level = 1, ...)
```

### Arguments

x	An object of class "walking_path" created by walkingPath().
zoom	Logical or Numeric. Positive numbers zoom in; negative numbers zoom out.
add	Logical. Add graphic to plot.
long.title	Logical. Tile with names.
mileposts	Logical. Plot mile/time posts.
milepost.unit	Character. "distance" or "time".
milepost.interval	Numeric. Mile post interval unit of distance (yard or meter) or unit of time (seconds).
alpha.level	Numeric. Alpha level transparency for path: a value in [0, 1].
...	Additional plotting parameters.

### Value

A base R plot.

---

print.euclidean      *Print method for neighborhoodEuclidean().*

---

### Description

Parameter values for neighborhoodEuclidean().

### Usage

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

**Arguments**

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

**Value**

A list of argument values.

**Examples**

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

---

*print.euclideanLatlong*  
*Print method for neighborhoodEuclidean(latlong = TRUE).*

---

**Description**

Print method for neighborhoodEuclidean(latlong = TRUE).

**Usage**

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

**Arguments**

x                    An object of class "euclideanLatlong" created by neighborhoodEuclidean(latlong = TRUE).  
...                   Additional parameters.

**Value**

A list of argument values.

**Examples**

```
## Not run:  
neighborhoodEuclidean(latlong = TRUE)  
print(neighborhoodEuclidean(latlong = TRUE))  
  
## End(Not run)
```

print.euclidean\_path *Print method for euclideanPath().*

---

**Description**

Summary output.

**Usage**

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

**Arguments**

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

**Value**

An R data frame.

---

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_latlong` *Print method for voronoiLatlong().*

---

**Description**

Parameter values for `voronoiLatlong()`.

**Usage**

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

**Arguments**

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

**Value**

A list of argument values.

---

`print.voronoi_nominal` *Print method for voronoiNominal().*

---

**Description**

Parameter values for `voronoiNominal()`.

**Usage**

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

**Arguments**

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

**Value**

A list of argument values.

---

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

---

### Description

Parameter values for neighborhoodWalking().

### Usage

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

### Arguments

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

### Value

A list of argument values.

### Examples

```
## 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 "latlong\_walking\_path" created by latlongWalkingPath().  
 ...                  Additional parameters.

**Value**

An R data frame.

---

profile2D	<i>2D Profile</i> .
-----------	---------------------

---

**Description**

2D Profile .

**Usage**

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

**Arguments**

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

**Examples**

```
## Not run:
profile2D(angle = 30)
profile2D(angle = 30, graphics = "ggplot2")

## End(Not run)
```

---

profile3D	<i>3D Profile</i> .
-----------	---------------------

---

**Description**

3D Profile.

**Usage**

```
profile3D(pump.select = NULL, pump.subset = NULL, vestry = FALSE,
drop.neg.subset = FALSE, 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>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>vestry</code>	Logical. TRUE uses the 14 pumps from the Vestry Report. FALSE uses the 13 in the original map.
<code>drop.neg.subset</code>	Logical. Drop negative subset selection
<code>multi.core</code>	Logical or Numeric. TRUE uses <code>parallel::detectCores()</code> . FALSE uses one, single core. You can also specify the number logical cores. See <code>vignette("Parallelization")</code> for details.

**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, case)
```

**Arguments**

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

**Value**

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

## Examples

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

---

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

---

## Description

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

## Usage

```
pumpData(vestry = FALSE, orthogonal = FALSE, multi.core = 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> ). See <code>vignette("Parallelization")</code> for details.

## Value

An R data frame.

## Note

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

## See Also

[pumpLocator](#)

---

pumpFatalities      *Compute fatalities by pump.*

---

### Description

Compute fatalities by pump.

### Usage

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

### Arguments

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

### Examples

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

## End(Not run)
```

---

pumpLocator      *Locate water pump by numerical ID.*

---

### Description

Highlight selected water pump.

### Usage

```
pumpLocator(id = 7, zoom = FALSE, vestry = FALSE, add.title = TRUE,
            highlight.segment = TRUE, latlong = FALSE)
```

**Arguments**

id	Numeric or Integer. With <code>vestry = TRUE</code> , a whole number between 1 and 14. With <code>vestry = FALSE</code> , a whole number between 1 and 13. See <code>cholera::pumps.vestry</code> and <code>cholera::pumps</code> for IDs and details about specific pumps.
zoom	Logical or Numeric. Positive numbers zoom in; negative numbers zoom out.
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.
latlong	Logical. Use longitude and latitude.

**Value**

A base R graphics plot.

**See Also**

[pumpData](#)

**Examples**

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

---

pumps

*Dodson and Tobler's pump data with street name.*

---

**Description**

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

**Usage**

```
pumps
```

**Format**

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

```
id pump number between 1 and 13
street nearest street
x x-coordinate
y y-coordinate
lon longitude
lat latitude
```

**Note**

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

**See Also**

[pumpLocator](#)

---

pumps.vestry	<i>Vestry report pump data.</i>
--------------	---------------------------------

---

**Description**

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

**Usage**

```
pumps.vestry
```

**Format**

A data frame with 14 observations and 4 variables.

id pump number between 1 and 14

street nearest street

x x-coordinate

y y-coordinate

lon longitude

lat latitude

**Note**

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

**See Also**

[pumpLocator](#)



---

rd.sample	<i>Sample of road intersections (segment endpoints).</i>
-----------	--

---

**Description**

Sample of road intersections (segment endpoints).

**Usage**

```
rd.sample
```

**Format**

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

one endpoints with 1 intersection

three endpoints with 3 intersections

---

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

---

**Description**

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

**Usage**

```
rectangle.filter
```

**Format**

A data frame with 2 variables and 4 observations.

x longitude

y latitude

---

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

---

**Description**

The result of using `sp::spsample()` and `sp::Polygon()` to generate 19,993 regularly spaced simulated cases within the map's borders.

**Usage**

```
regular.cases
```

**Format**

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

x x-coordinate

y y-coordinate

**Note**

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

---

road.segments	<i>Dodson and Tobler's street data transformed into road segments.</i>
---------------	--

---

**Description**

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

**Usage**

```
road.segments
```

**Format**

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

street numeric street ID, which range between 1 and 528

id character segment ID

name road name

x1 x-coordinate of first endpoint

y1 y-coordinate of first endpoint

x2 x-coordinate of second endpoint

y2 y-coordinate of second endpoint

**Note**

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

**See Also**

[roads](#)  
[vignette\("road.names"\)](#)  
[streetNameLocator](#)  
[streetNumberLocator](#)  
[segmentLocator](#)

---

roads

*Dodson and Tobler's street data with appended road names.*

---

**Description**

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

**Usage**

```
roads
```

**Format**

A data frame with 1243 observations and 6 variables. The data describe the roads on Snow's map.

street street segment number, which range between 1 and 528

n number of points in this street line segment

x x-coordinate

y y-coordinate

id unique numeric ID

name road name

lon longitude

lat latitude

**See Also**

[road.segments](#)  
[vignette\("road.names"\)](#)  
[streetNameLocator](#)  
[streetNumberLocator](#)  
[segmentLocator](#)

---

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

---

**Description**

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

**Usage**

```
roadSegments(latlong = FALSE)
```

**Arguments**

latlong            Logical. Use estimated longitude and latitude.

**Value**

An R data frame.

**Note**

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

---

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

---

**Description**

Highlight segment by ID.

**Usage**

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

**Arguments**

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

highlight            Logical. Color segment.

col                    Character. Highlight color.

rotate.label        Logical. Rotate segment ID label.

latlong               Logical. Use estimated longitude and latitude.

**Value**

A base R graphics segment(s).

**Examples**

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

---

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

---

**Description**

Highlight selected road segment(s) and cases.

**Usage**

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

**Arguments**

segment.id	Character. A vector of segment IDs. See Note.
zoom	Logical or Numeric. Positive value zoom in. Negative values zoom out.
latlong	Logical. Longitude and latitude coordinates
cases	Character. Cases to plot: NULL, "address" or "fatality".
token	Character. Cases as "id" or "point".
vestry	Logical. TRUE uses the 14 pumps from the Vestry report. FALSE uses the 13 in the original map.
add.pump	Logical. Include pumps.
add.title	Logical. Include title.
add.subtitle	Logical. Include subtitle.
highlight	Logical. Highlight selected segment(s) and cases.
distance.unit	Character. Unit of distance: "meter", "yard" or "native". "native" returns the map's native scale. See vignette("roads") for information on conversion.
time.unit	Character. "hour", "minute", or "second".
walking.speed	Numeric. Walking speed in km/hr.
cex.text	Numeric.

**Value**

A base R graphics plot.

**Note**

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

**Examples**

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

---

sim.ortho.proj

*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 19,993 simulate cases along the network of roads.

case numeric case ID

road.segment "address" road segment

x.proj x-coordinate

y.proj y-coordinate

dist Euclidean or orthogonal distance to home road segment

type type of projection: Euclidean ("eucl") or orthogonal ("ortho")

**Note**

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

---

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

---

**Description**

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

**Usage**

```
sim.pump.case
```

**Format**

A list 4972 IDs spread over 13 vectors.

```
sim.pump.case numerical ID
```

**Note**

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

**Examples**

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

---

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

---

**Description**

Walking distance to Broad Street Pump (#7).

**Usage**

```
sim.walking.distance
```

**Format**

A data frames with 5 variables.

case case ID

pump pump ID

pump.name pump name

distance walking distance in meters

time walking time in seconds based on 5 km/hr walking speed

---

simulateFatalities      *Project simulated fatalities onto road network*

---

**Description**

Places regularly spaced "simulated" or "expected" cases across the face of the map and then finds the "addresses" of those cases via orthogonal projection or simple proximity to road graph network. These data are used to generate "expected" pump neighborhoods.

**Usage**

```
simulateFatalities(recompute.regular.cases = FALSE, simulated.obs = 20000L,
  multi.core = FALSE)
```

**Arguments**

recompute.regular.cases	Logical. TRUE re-computes regular data. FALSE uses pre-computed data. For replication of data used in the package.
simulated.obs	Numeric. Number of regular cases. For use with recompute.regular.cases = TRUE.
multi.core	Logical or Numeric. TRUE uses <code>parallel::detectCores()</code> . FALSE uses one, single core. With Numeric, you specify the number logical cores (rounds with <code>as.integer()</code> ). See <code>vignette("Parallelization")</code> for details.

**Value**

An R data frame: [sim.ortho.proj](#).

**Note**

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



---

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

---

**Description**

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

**Usage**

```
snow.neighborhood
```

**Format**

A vector with 384 observations.

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

---

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

---

**Description**

Uses `RColorBrewer::brewer.pal()`.

**Usage**

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

**Arguments**

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

**Value**

A character vector of colors.

**Note**

Built with 'RColorBrewer' package.

---

`snowMap`*Plot John Snow's cholera map.*

---

**Description**

Plot John Snow's cholera map.

**Usage**

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

**Arguments**

<code>vestry</code>	Logical. TRUE uses the 14 pumps from the map in the Vestry Report. FALSE uses the 13 pumps from the original map.
<code>stacked</code>	Logical. Use stacked fatalities.
<code>add.axes_box</code>	Logical. Add plot axes and plot box.
<code>add.cases</code>	Logical. Add observed cases.
<code>add.landmarks</code>	Logical. Add landmarks.
<code>add.pumps</code>	Logical. Add pumps.
<code>add.roads</code>	Logical. Add roads.
<code>add.frame</code>	Logical. Add map frame.
<code>main</code>	Character. Title of graph.
<code>case.col</code>	Character. Color of fatalities.
<code>case.pch</code>	Character. Color of fatalities.
<code>latlong</code>	Logical. Use estimated longitude and latitude.
<code>...</code>	Additional plotting parameters.

**Value**

A base R graphics plot.

**Note**

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

**Examples**

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

---

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

---

**Description**

Highlight road by name.

**Usage**

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

**Arguments**

road.name	Character vector. The function tries to correct for case and remove extra spaces (includes "Map Frame").
col	Character. Highlight color.
lwd	Numeric. Line width.
latlong	Logical. Use estimated longitude and latitude.

**Value**

A base R graphics segment(s).

**Examples**

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

---

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

---

**Description**

Compute length of selected street.

**Usage**

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

**Arguments**

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

**Value**

An R vector of length one.

**Examples**

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

---

streetNameLocator      *Locate street(s) by name(s).*

---

**Description**

Highlight selected road(s) and cases.

**Usage**

```
streetNameLocator(street.name = "Broad Street", zoom = TRUE,
  latlong = FALSE, cases = "address", token = "id", vestry = FALSE,
  add.pump = TRUE, add.title = TRUE, add.subtitle = TRUE,
  highlight = TRUE, distance.unit = "meter", time.unit = "second",
  walking.speed = 5, cex.text = 0.67)
```

**Arguments**

street.name	Character. A street name or vector of street names (e.g., "Broad Street", "Poland Street").
zoom	Logical or Numeric. Positive numbers zoom in; negative numbers zoom out.
latlong	Logical. Longitude and latitude coordinates
cases	Character. Cases to plot: NULL, "address" or "fatality".
token	Character. Cases as "id" or "point".
vestry	Logical. TRUE uses the 14 pumps from the Vestry report. FALSE uses the 13 in the original map.
add.pump	Logical. Include pumps.
add.title	Logical. Include title.
add.subtitle	Logical. Include subtitle.
highlight	Logical. Highlight selected segment(s) and cases.
distance.unit	Character. Unit of distance: "meter", "yard" or "native". "native" returns the map's native scale. See vignette("roads") for information on conversion.
time.unit	Character. "hour", "minute", or "second".
walking.speed	Numeric. Walking speed in km/hr.
cex.text	Numeric.

**Value**

A base R graphics plot.

**Note**

See `streetNames()`.

**Examples**

```
streetNameLocator("broad street")
streetNameLocator("Broad Street", zoom = -10)
streetNameLocator("Broad Street", latlong = TRUE, zoom = -10)
streetNameLocator("Broad Street", distance.unit = "yard")
streetNameLocator("Broad Street", zoom = FALSE)
```

---

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

---

**Description**

Unique road names from Snow's cholera map.

**Usage**

```
streetNames()
```

**Value**

An R character vector.

**Note**

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

---

streetNumberLocator	<i>Locate street by its numerical ID.</i>
---------------------	---

---

**Description**

Highlight selected road segment(s) and cases.

**Usage**

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

**Arguments**

street.number	Character. A vector of segment IDs. See Note.
zoom	Logical or Numeric. Positive numbers zoom in; negative numbers zoom out.
latlong	Logical. Longitude and latitude coordinates
cases	Character. Cases to plot: NULL, "address" or "fatality".
token	Character. Cases as "id" or "point".
vestry	Logical. TRUE uses the 14 pumps from the Vestry report. FALSE uses the 13 in the original map.
add.pump	Logical. Include pumps.
add.title	Logical. Include title.
add.subtitle	Logical. Include subtitle.
highlight	Logical. Highlight selected segment(s) and cases.
distance.unit	Character. Unit of distance: "meter", "yard" or "native". "native" returns the map's native scale. See vignette("roads") for information on conversion.
time.unit	Character. "hour", "minute", or "second".
walking.speed	Numeric. Walking speed in km/hr.
cex.text	Numeric.

**Value**

A base R graphics plot.

**Note**

See cholera::roads.

**Examples**

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

---

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

---

**Description**

Return computed counts for Euclidean neighborhoods.

**Usage**

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

**Arguments**

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

**Value**

A vector of counts by neighborhood.

**Examples**

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

---

summary.voronoi\_nominal

*Summary method for voronoiNominal().*

---

**Description**

Return computed counts for Voronoi neighborhoods.

**Usage**

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

**Arguments**

object            Object. An object of class "voronoi" created by voronoiNominal().  
...                Additional arguments.

**Value**

A vector of counts by neighborhood.

**See Also**

addVoronoi() plot.voronoi()

---

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

---

**Description**

Soho elevation data.

**Usage**

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

**Arguments**

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

---

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

---

**Description**

Aggregate time series fatality data from the Vestry report.

**Usage**

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

**Arguments**

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

**Value**

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

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

**Note**

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



**See Also**

[plot.time\\_series](#), [print.time\\_series](#), [vignette\("time.series"\)](#)

**Examples**

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

---

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, dev.mode = FALSE)
```

**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. See <a href="#">vignette("Parallelization")</a> for details.
<code>dev.mode</code>	Logical. Development mode uses <code>parallel::parLapply()</code> .

**Value**

An R list.

**Note**

This function documents the code that generates [anchor.case](#), [fatalities.address](#), [fatalities.unstacked](#) and [ortho.proj](#).

**See Also**

[vignette\("unstacking.fatalities"\)](#)

---

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

---

**Description**

Coordinates of Voronoi polygon vertices for original map.

**Usage**

```
voronoi.polygons
```

**Format**

A list of 13 data frames frames with 5 variables.

vertex vertex ID

x x-coordinate

y y-coordinate

lon longitude

lat latitude

---

voronoi.polygons.vestry	<i>Coordinates of Voronoi polygon vertices for Vestry Report map.</i>
-------------------------	---

---

**Description**

Coordinates of Voronoi polygon vertices for Vestry Report map.

**Usage**

```
voronoi.polygons.vestry
```

**Format**

A list of 14 data frames frames with 5 variables.

vertex vertex ID

x x-coordinate

y y-coordinate

lon longitude

lat latitude

---

voronoiPolygons	<i>Extract vertices of Delaunay triangles and Dirichelet (Voronoi) tiles.</i>
-----------------	---

---

**Description**

For construction and plotting of Delaunay and Voronoi polygons.

**Usage**

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

**Arguments**

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

**Value**

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

**Note**

This function relies on the 'deldir' package.

**Examples**

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

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

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

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

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

---

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

---

### Description

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

### Usage

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

### Arguments

origin	Numeric. Vector of origin(s) (numeric or case/landmark name).
destination	Numeric. Vector of destination(s) (numeric or landmark/pump name).
type	Character. Path case to pump. FALSE is all other combinations of cases, landmarks and pumps.
vestry	Logical. TRUE uses the 14 pumps from the map in the Vestry Report. FALSE uses the 13 pumps from the original map.
latlong	Logical.
case.set	Character. "observed" or "expected".
location	Character. For cases and pumps. "anchor", "fatality" or "orthogonal".
weighted	Logical. TRUE computes shortest path in terms of road length. FALSE computes shortest path in terms of the number of nodes.
distance.unit	Character. Unit of distance: "meter" or "yard".
time.unit	Character. "hour", "minute", or "second".
walking.speed	Numeric. Walking speed in km/hr.
include.landmarks	Logical. Include landmarks as cases.

# Index

## \* datasets

- anchor.case, 20
- border, 21
- fatalities, 23
- fatalities.address, 24
- fatalities.unstacked, 25
- frame.data, 25
- frame.sample, 26
- landmark.squares, 26
- landmarks, 27
- latlong.ortho.addr, 28
- latlong.ortho.pump, 28
- latlong.ortho.pump.vestry, 29
- latlong.regular.cases, 30
- latlong.sim.ortho.proj, 30
- meter.to.yard, 31
- ortho.proj, 36
- ortho.proj.pump, 36
- ortho.proj.pump.vestry, 37
- plague.pit, 38
- pumps, 55
- pumps.vestry, 56
- rd.sample, 57
- rectangle.filter, 57
- regular.cases, 58
- road.segments, 58
- roads, 59
- sim.ortho.proj, 62
- sim.pump.case, 63
- sim.walking.distance, 63
- snow.neighborhood, 65
- voronoi.polygons, 74
- voronoi.polygons.vestry, 74

  

- addCase, 5
- addDelaunay, 6
- addEuclideanPath, 7
- addIndexCase, 8
- addKernelDensity, 8
- addLandmarks, 10

- addLandmarkSquares, 10
- addNeighborhoodCases, 11
- addNeighborhoodEuclidean, 12
- addNeighborhoodSnow, 14
- addNeighborhoodWalking, 14
- addPlaguePit, 16
- addPump, 16
- addRoads, 17
- addVoronoi, 17
- addWalkingPath, 18
- addWhitehead, 19
- anchor.case, 20, 73

  

- border, 21

  

- caseLocator, 20, 21, 23–25
- cholera (cholera-package), 4
- cholera-package, 4

  

- euclideanPath, 22

  

- fatalities, 23
- fatalities.address, 24, 73
- fatalities.unstacked, 25, 73
- frame.data, 25
- frame.sample, 26

  

- landmark.squares, 26
- landmarks, 27
- latlong.ortho.addr, 28
- latlong.ortho.pump, 28
- latlong.ortho.pump.vestry, 29
- latlong.regular.cases, 30
- latlong.sim.ortho.proj, 30

  

- mapRange, 31
- meter.to.yard, 31

  

- neighborhoodData, 32
- neighborhoodEuclidean, 32
- neighborhoodSnow, 33

- neighborhoodVoronoi, 34
- neighborhoodWalking, 35, 63
  
- ortho.proj, 36, 73
- ortho.proj.pump, 36, 53
- ortho.proj.pump.vestry, 37, 53
  
- pearsonResiduals, 38
- plague.pit, 38
- plot.euclidean, 39
- plot.euclidean\_path, 40
- plot.euclideanLatlong, 40
- plot.neighborhood\_data, 41
- plot.profile\_perspective, 42
- plot.snow, 42
- plot.time\_series, 43, 73
- plot.voronoi\_latlong, 44
- plot.voronoi\_nominal, 44
- plot.walking, 45
- plot.walking\_path, 46
- print.euclidean, 46
- print.euclidean\_path, 48
- print.euclideanLatlong, 47
- print.time\_series, 48, 73
- print.voronoi\_latlong, 49
- print.voronoi\_nominal, 49
- print.walking, 50
- print.walking\_path, 50
- profile2D, 51
- profile3D, 51
- pumpCase, 52
- pumpData, 29, 37, 53, 55, 56
- pumpFatalities, 54
- pumpLocator, 53, 54, 56
- pumps, 53, 55
- pumps.vestry, 53, 56
  
- rd.sample, 57
- rectangle.filter, 57
- regular.cases, 58, 64
- road.segments, 58, 59, 60
- roads, 59, 59
- roadSegments, 59, 60
  
- segmentHighlight, 60
- segmentLocator, 59, 61
- sim.ortho.proj, 62, 64
- sim.pump.case, 63
- sim.walking.distance, 63
  
- simulateFatalities, 58, 62, 64
- snow.neighborhood, 65
- snowColors, 65
- snowMap, 66
- streetHighlight, 67
- streetLength, 67
- streetNameLocator, 23–25, 59, 68
- streetNames, 69
- streetNumberLocator, 23–25, 59, 69
- summary.euclidean, 70
- summary.voronoi\_nominal, 71
  
- tanakaContourPlot, 72
- timeSeries, 43, 72
  
- unstackFatalities, 20, 24, 25, 28, 36, 73
  
- voronoi.polygons, 74
- voronoi.polygons.vestry, 74
- voronoiPolygons, 75
  
- walkingPath, 76