Have you tessellated today?

Bay Area useR Group | 2018-December-11 | Peter Li

'deldir'

graphics::polygon()

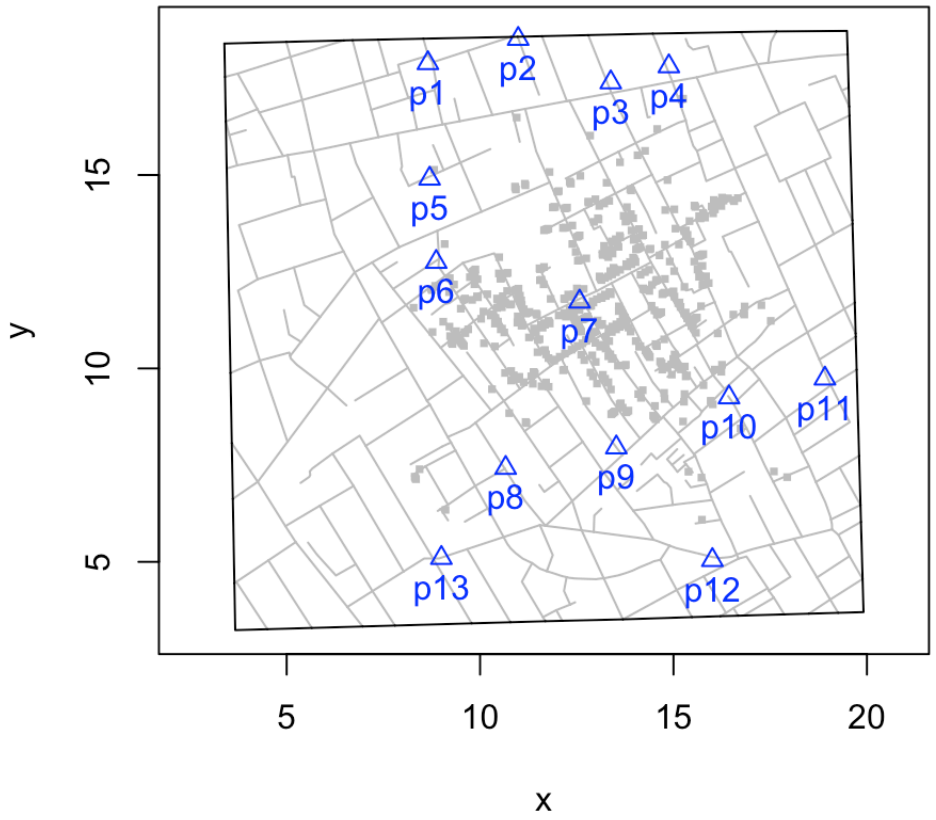
sp::point.in.polygon(()

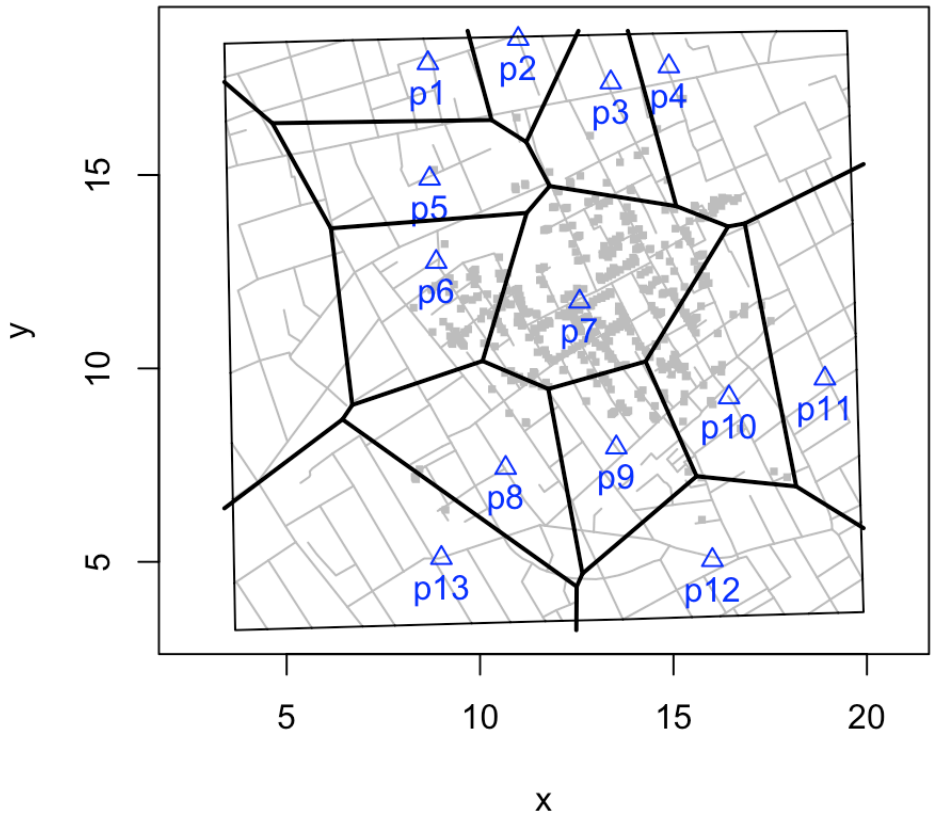
Voronoi tessellation

Definition

"The Voronoi region of a site ***s*** is the set of points in the plane for which ***s*** is the closest site among all the sites."

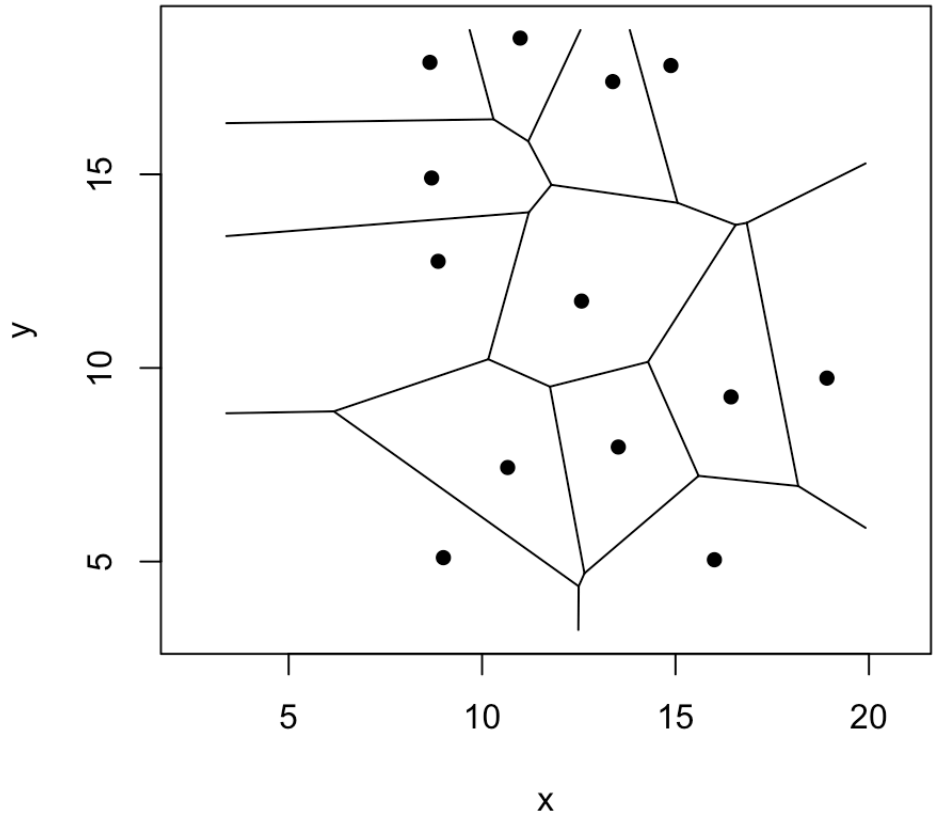
- Steven Fortune. 1987. "A Sweepline Algorithm for Voronoi Diagrams". *Algorithmica*. 2:153-174.


p1
p2
p3
p4
p5
p6
p7
p8
p9
p10
p11
p12
p13
y
x
5
10
15
20
5
10
15

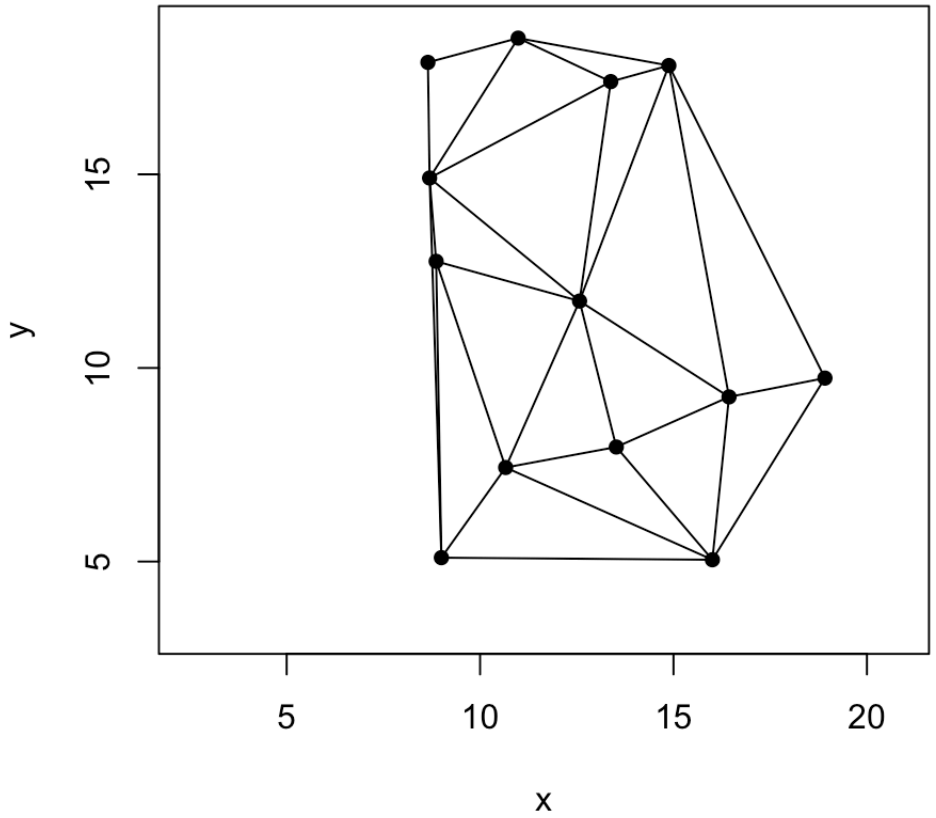

p1
p2
p3
p4
p5
p6
p7
p8
p9
p10
p11
p12
p13
y
x
5
10
15
20
5
10
15

tessellation v. triangulation

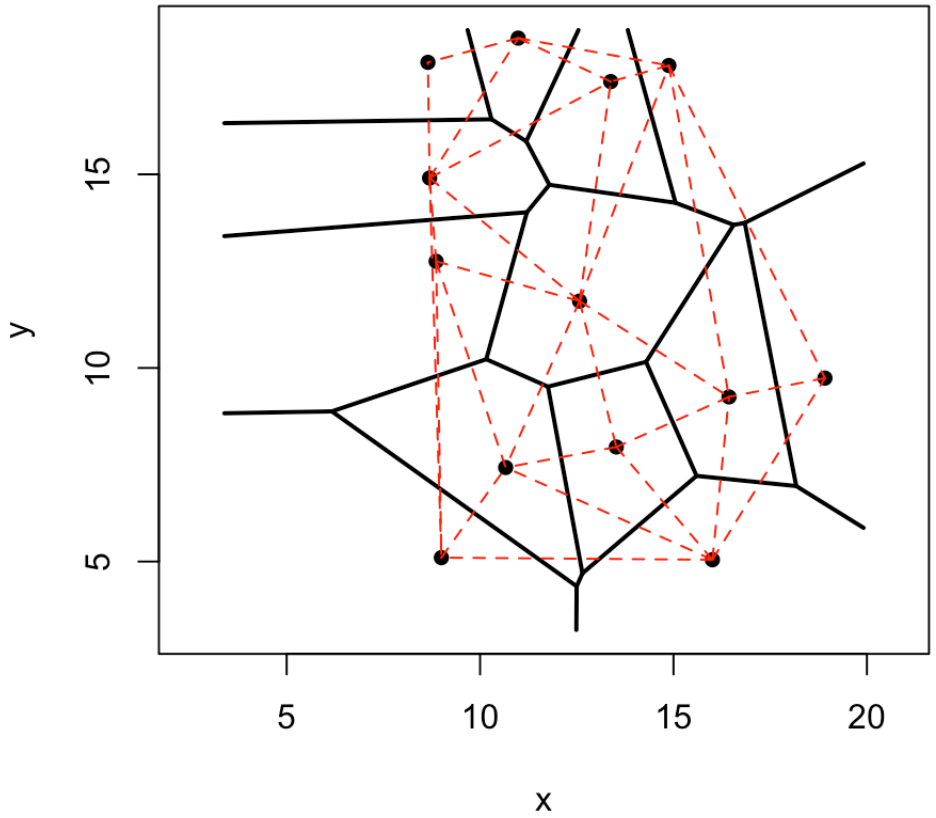
tessellation



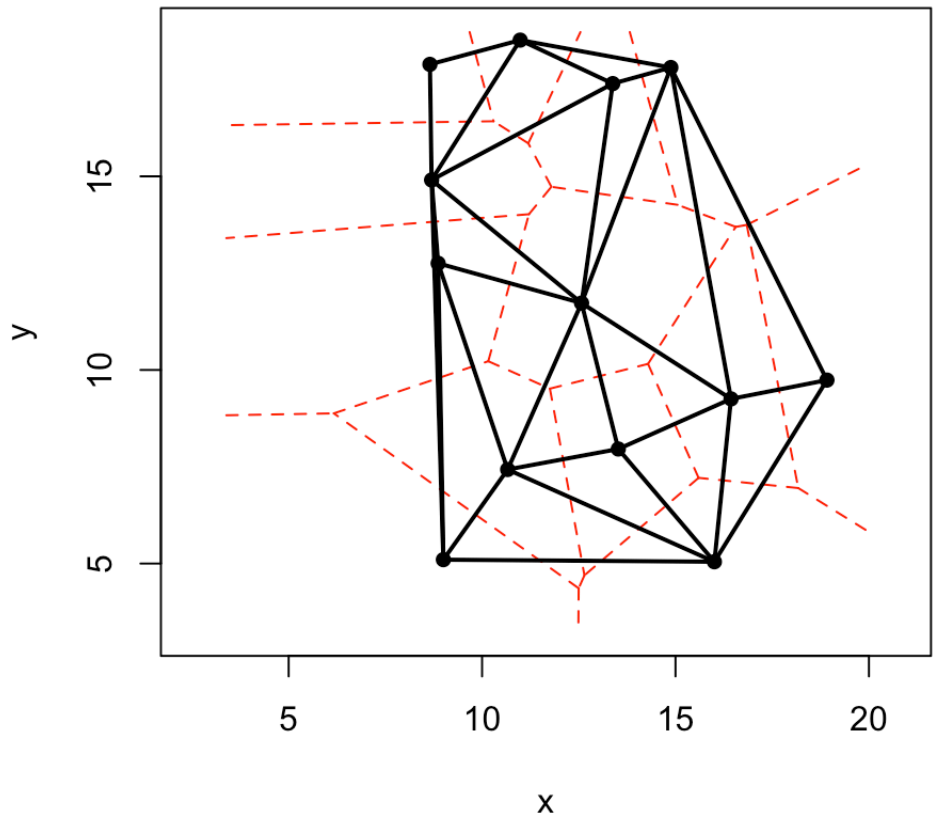
triangulation



duality: tessellation v. triangulation



duality: triangulation v. tessellation

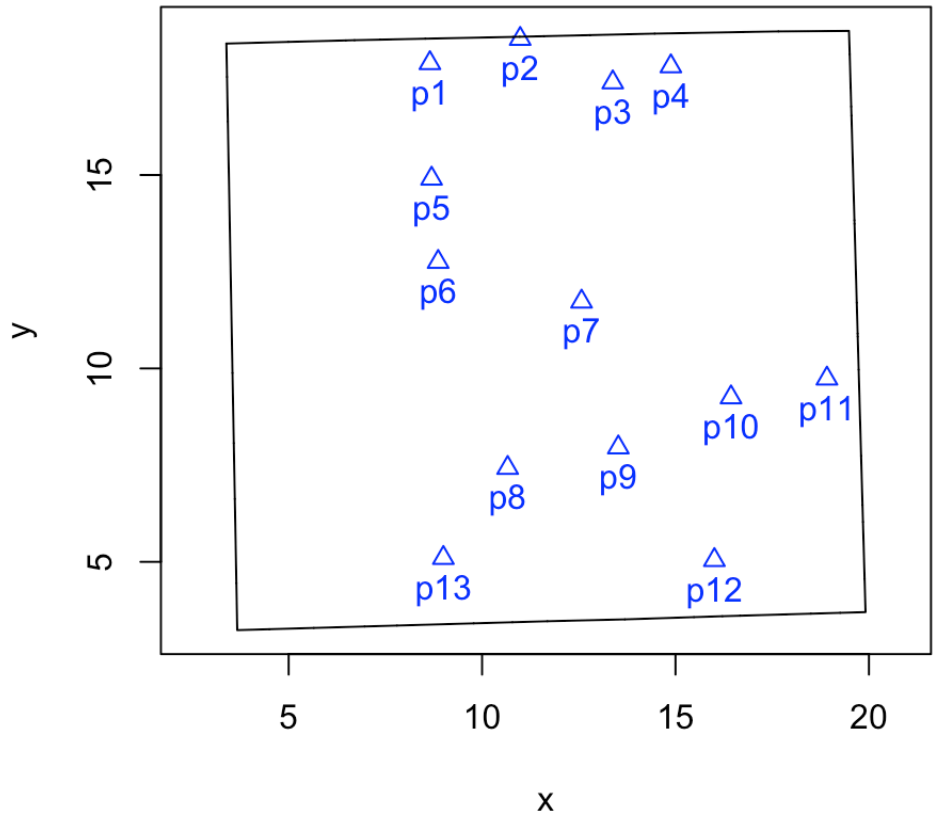


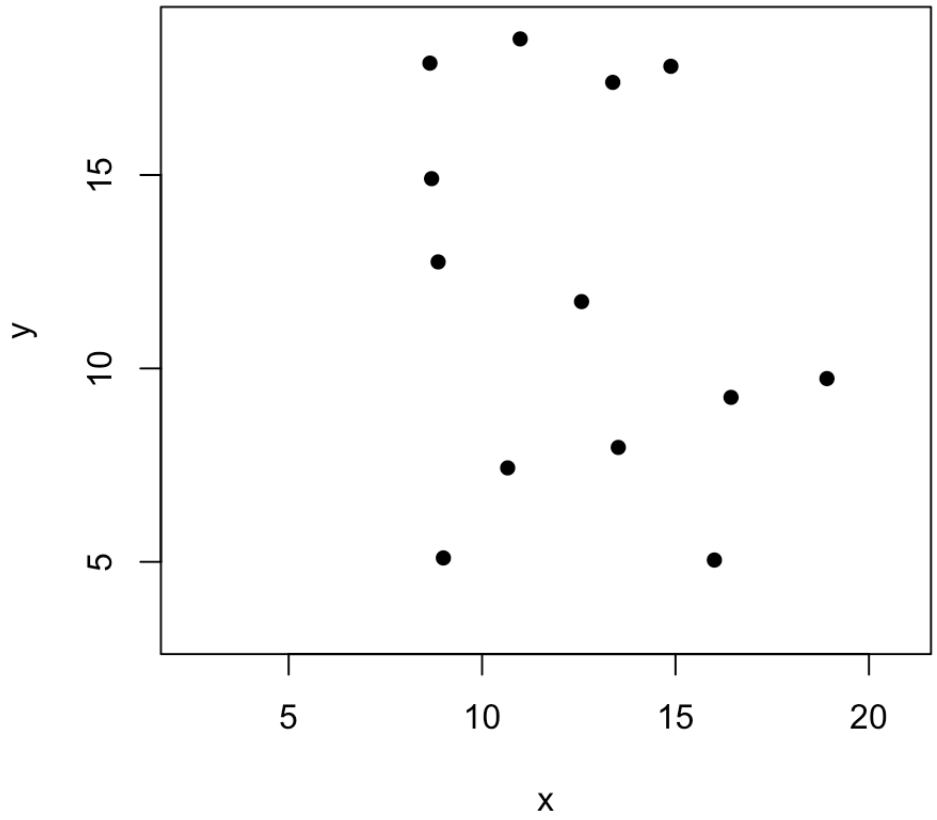
applications

- optimal facility location problems
- *k* nearest and farthest neighbor
- crystals and fungi
- grain behavior: sand, corn, rice
- algorithms homework assignment
- d3.js mouseover "touch" interface

'deldir' package

- Delauny triangulation
- Dirchelet (Voronoi) tessellation


p1
p2
p3
p4
p5
p6
p7
p8
p9
p10
p11
p12
p13
y
x
5
10
15
20
5
10
15


y
5
10
15
x
5
10
15
20

```
sites
```

```
##            x         y
## 1   8.651201 17.891600
## 2  10.984780 18.517851
## 3  13.378190 17.394541
## 4  14.879830 17.809919
## 5   8.694768 14.905470
## 6   8.864416 12.753540
## 7  12.571360 11.727170
## 8  10.660970  7.428647
## 9  13.521460  7.958250
## 10 16.434891  9.252130
## 11 18.914391  9.737819
## 12 16.005110  5.046838
## 13  8.999440  5.101023
```

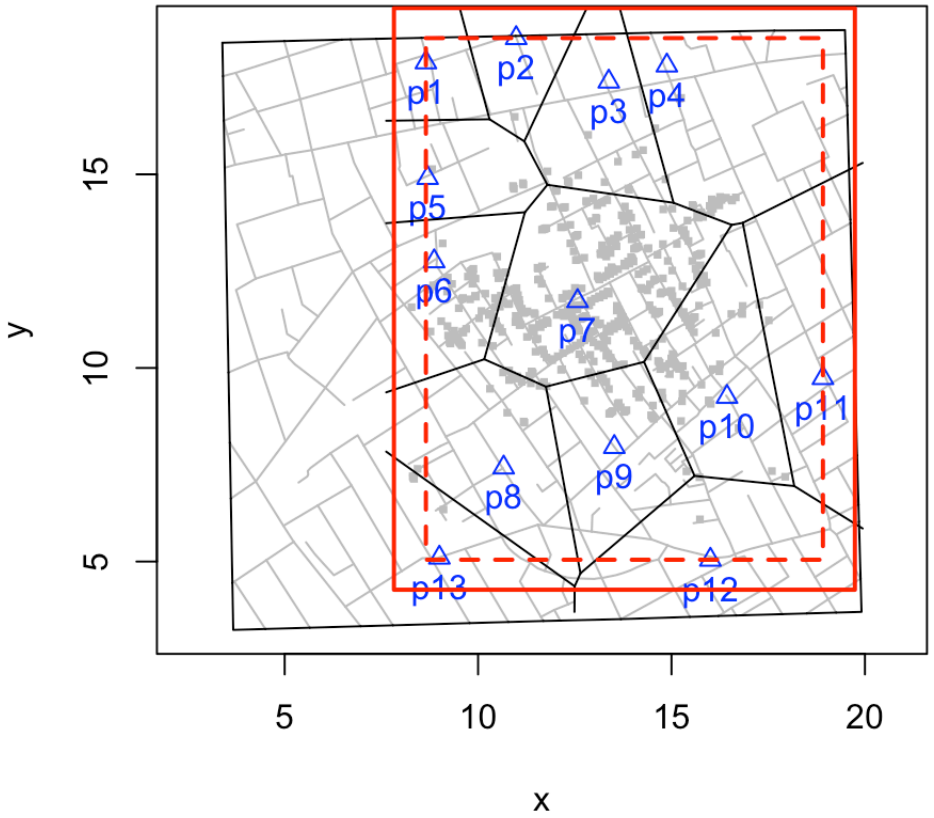
syntax

```
deldir::deldir(sites[, c("x", "y")])
```

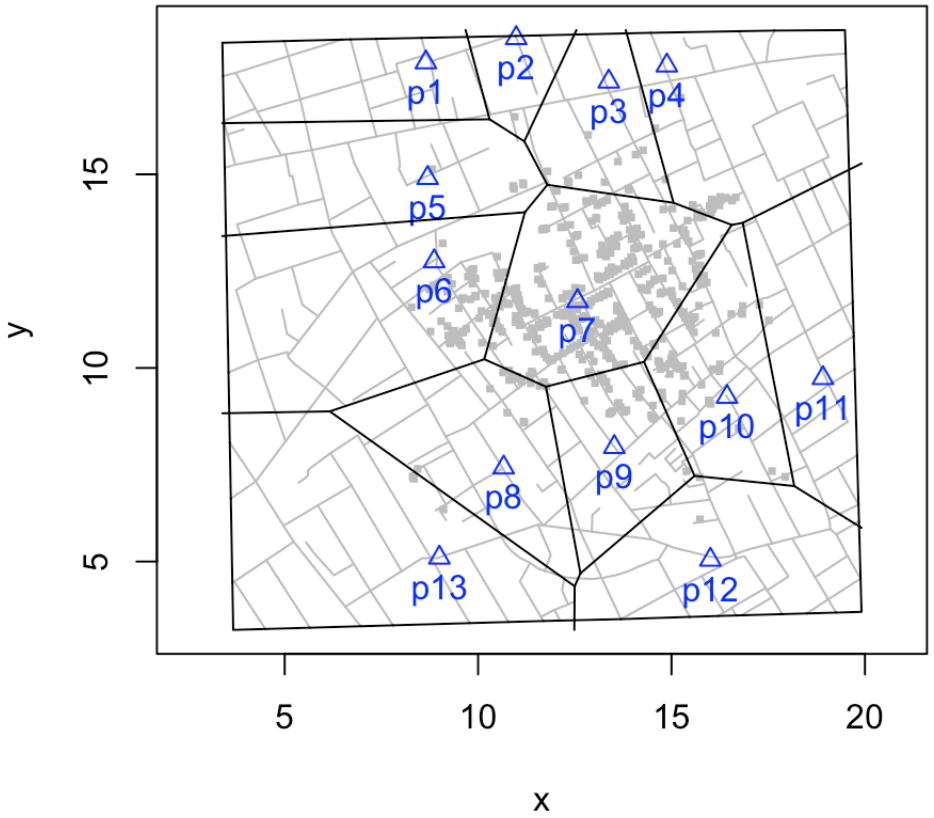
real-life syntax

```
voronoi <- deldir::deldir(sites[, c("x", "y")], rw = c(x.rng, y.rng),
  suppressMsge = TRUE)
```

rw: rectangular window (default NULL)



rw: rectangular window (manually set)



rw: rectangular window (manually set)

```
x.rng <- range(cholera::roads$x)
y.rng <- range(cholera::roads$y)

voronoi <- deldir::deldir(sites[, c("x", "y")], rw = c(x.rng, y.rng),
  suppressMsge = TRUE)
```

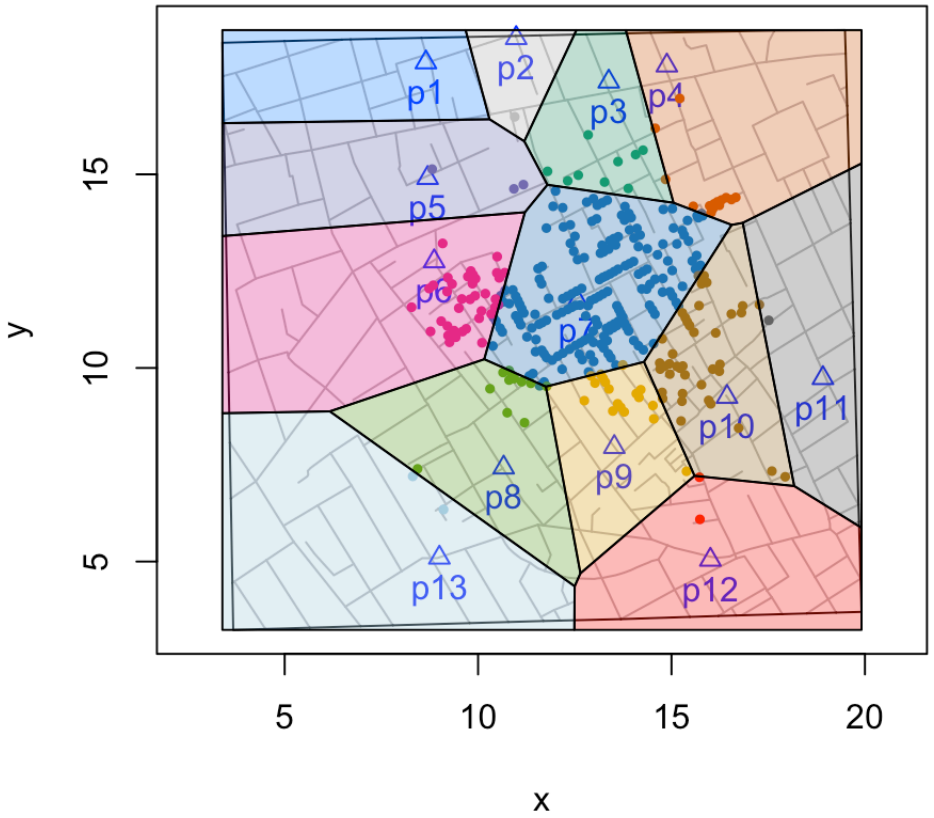
'deldir' Features

- centroids, areas and perimeters

'deldir' Limits

- graphics::segments()
  - just a visual layer
- only sites matter
  - other data (layers), by definition, do not
- but …

1 - Coloring Tiles



2 - Counting Elements in Tiles

```
##  p1  p2  p3  p4  p5  p6  p7  p8  p9 p10 p11 p12 p13
##   0   1  13  23   6  61 361  16  27  62   2   2   4
```

Tiles -> Polygons

- translate tiles to polygons …
- use graphics::polygon(), sp::point.in.polygon(), etc.

tile.list()

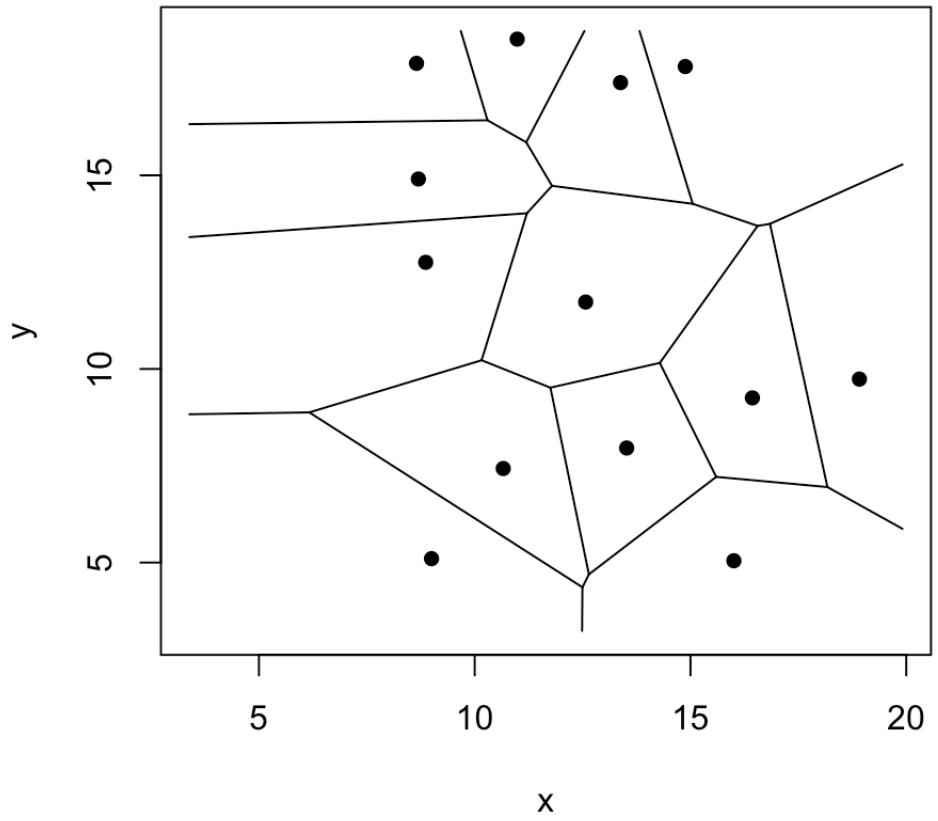
plot.deldir()

'deldir' approach

syntax

```
voronoi <- deldir::deldir(sites[, c("x", "y")], rw = c(x.rng, y.rng),
  suppressMsge = TRUE)
```

```
plot(sites, pch = 16, xlim = x.rng, ylim = y.rng)
plot(voronoi, add = TRUE, wline = "tess", wpoints = "none", lty = "solid")
```



extract tile data

```
tile.data <- deldir::tile.list(voronoi)
```

```
tile.data[[1]]
```

```
## $ptNum
## [1] 1
##
## $pt
##          x          y
##  8.651201 17.891600
##
## $x
## [1]  9.678367  3.390000  3.390000 10.296353
##
## $y
## [1] 18.72500 18.72500 16.32146 16.42222
##
## $bp
## [1]  TRUE  TRUE  TRUE FALSE
##
## $area
## [1] 15.54022
```

list of data frames of vertices

```
polygon.vertices <- lapply(tile.data, function(dat) {
  data.frame(x = dat$x, y = dat$y)
})
```

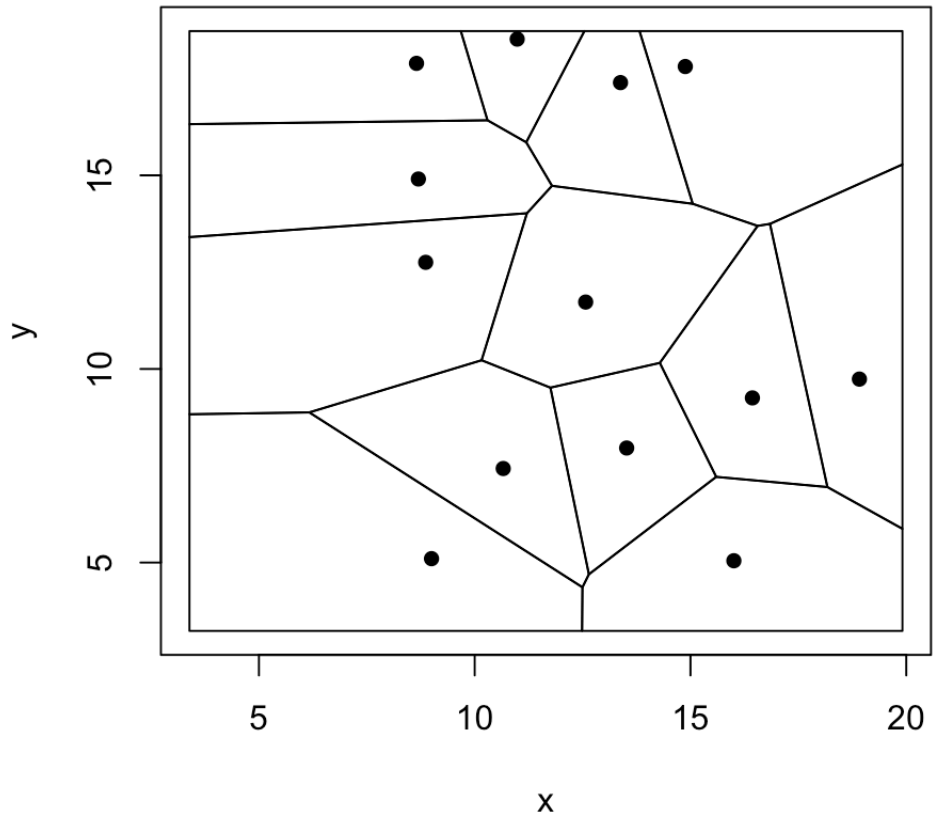
vertices data

```
polygon.vertices[1:2]
```

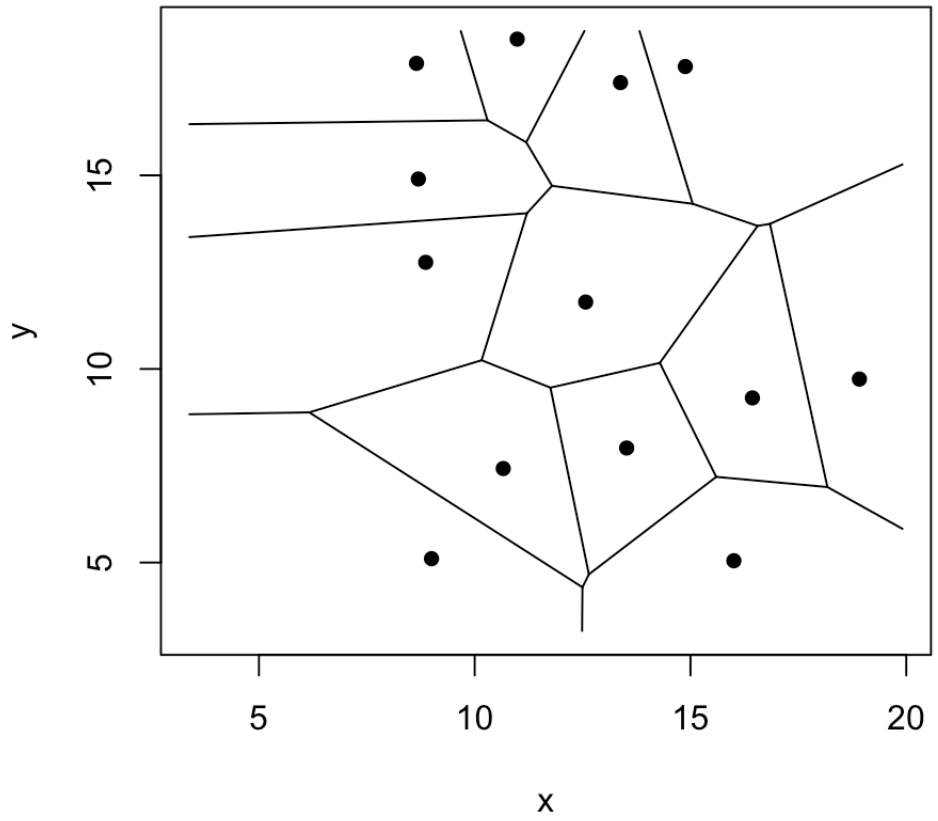
```
## [[1]]
##           x        y
## 1  9.678367 18.72500
## 2  3.390000 18.72500
## 3  3.390000 16.32146
## 4 10.296353 16.42222
##
## [[2]]
##           x        y
## 1 12.542312 18.72500
## 2  9.678367 18.72500
## 3 10.296353 16.42222
## 4 11.194356 15.85294
```

```
voronoi <- deldir::deldir(sites[, c("x", "y")], rw = c(x.rng, y.rng),
  suppressMsge = TRUE)

cell.data <- deldir::tile.list(voronoi)

polygon.vertices <- lapply(cell.data, function(dat) {
  data.frame(x = dat$x, y = dat$y)
})

plot(sites, pch = 16, xlim = x.rng, ylim = y.rng)

invisible(lapply(polygon.vertices, polygon))
```

```
plot(sites, pch = 16, xlim = x.rng, ylim = y.rng)
invisible(lapply(polygon.vertices, polygon))
```



```
plot(sites, pch = 16, xlim = x.rng, ylim = y.rng)
plot(voronoi, add = TRUE, wline = "tess", wpoints = "none", lty = "solid")
```

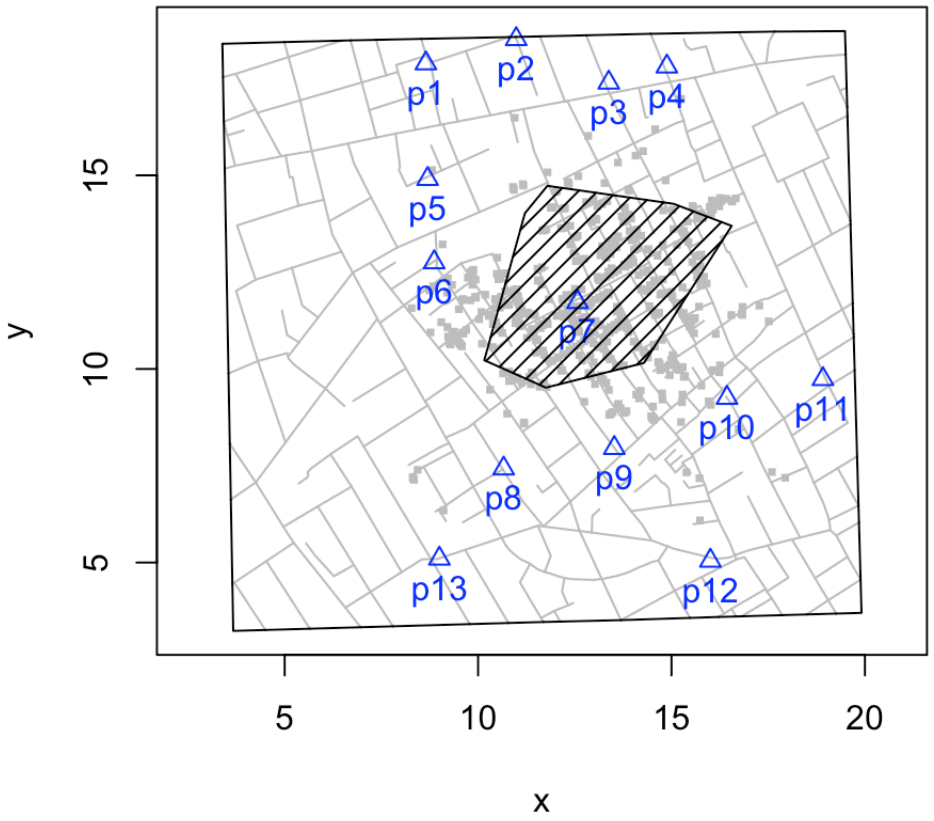


coloring polygons

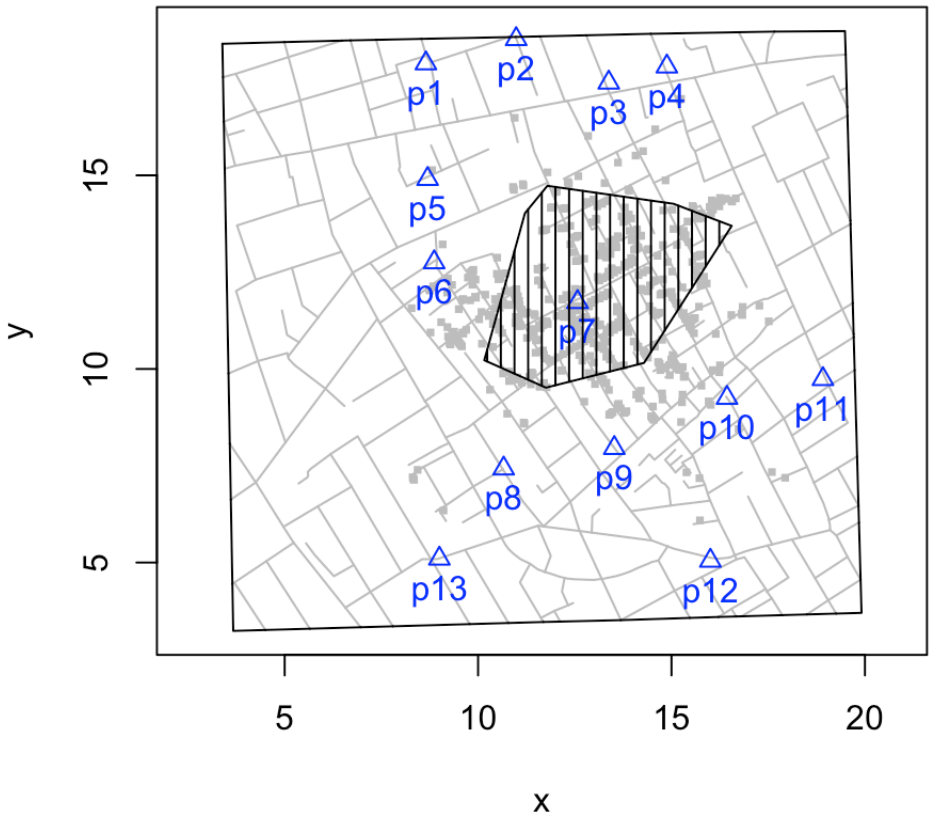
polygon data for site 7

```
tile7 <- tile.data[[7]]
```

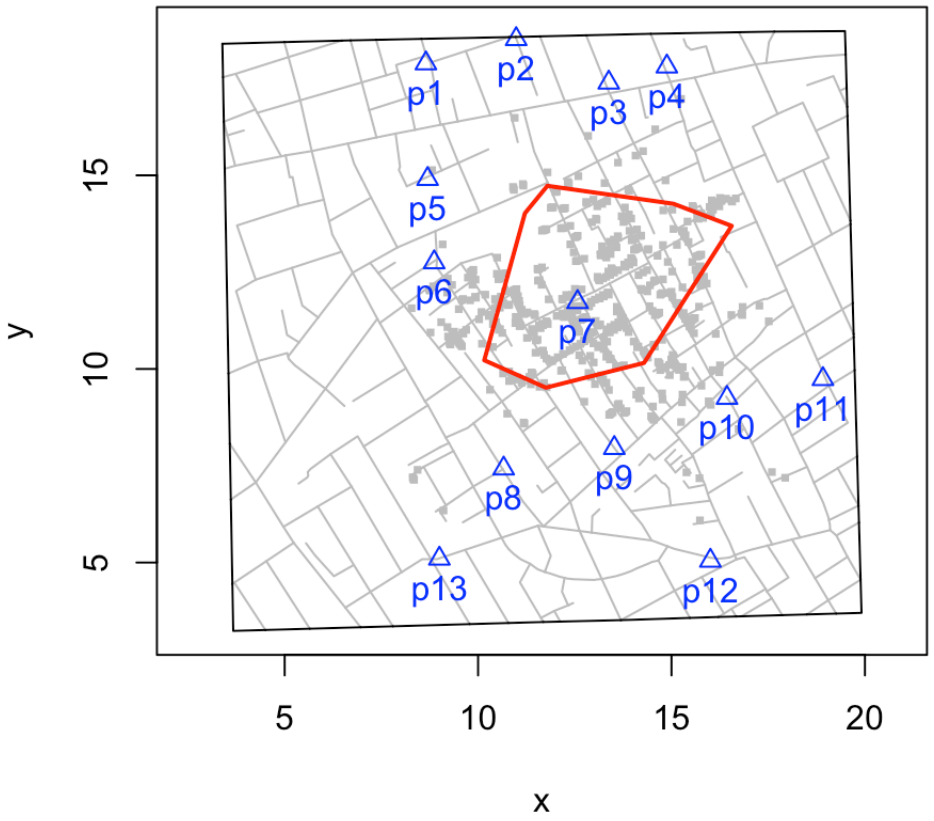
```
snowMap()
polygon(tile7, density = 15)
```



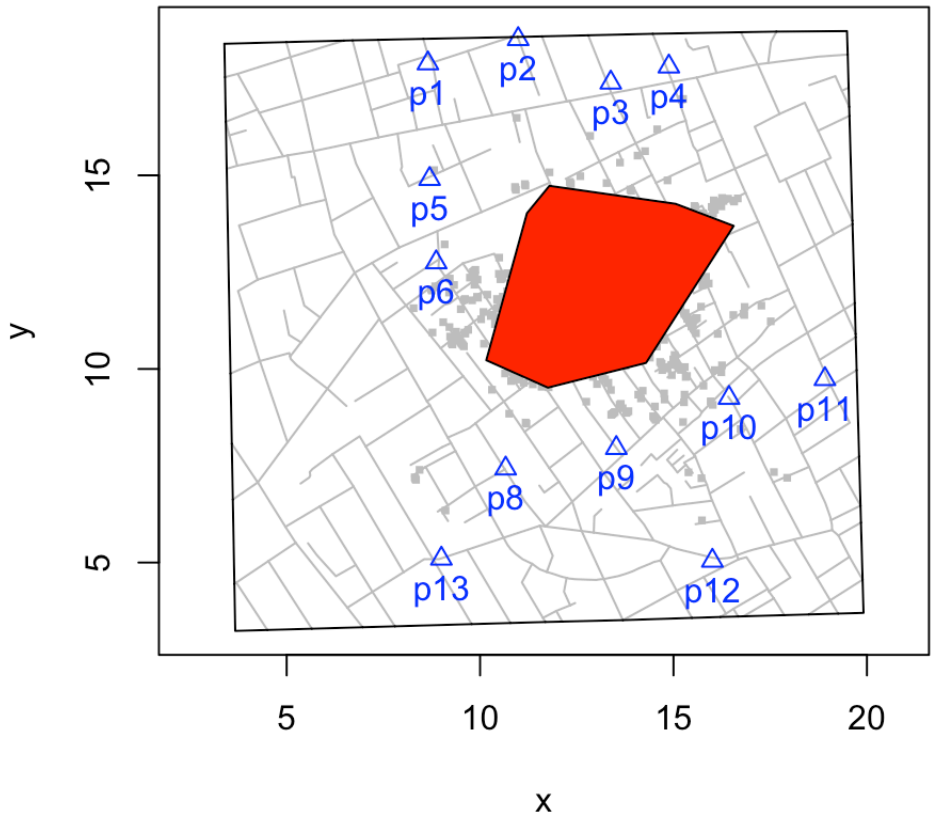
```
snowMap()
polygon(tile7, density = 15, angle = 90)
```



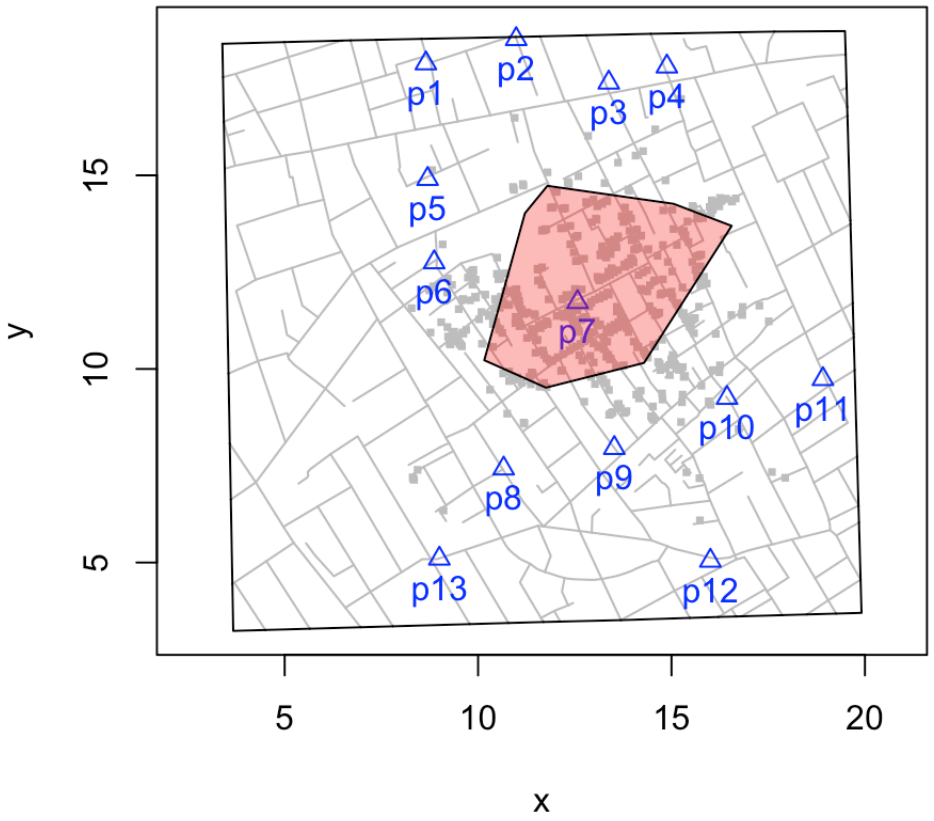
```
snowMap()
polygon(tile7, border = "red", lwd = 2)
```



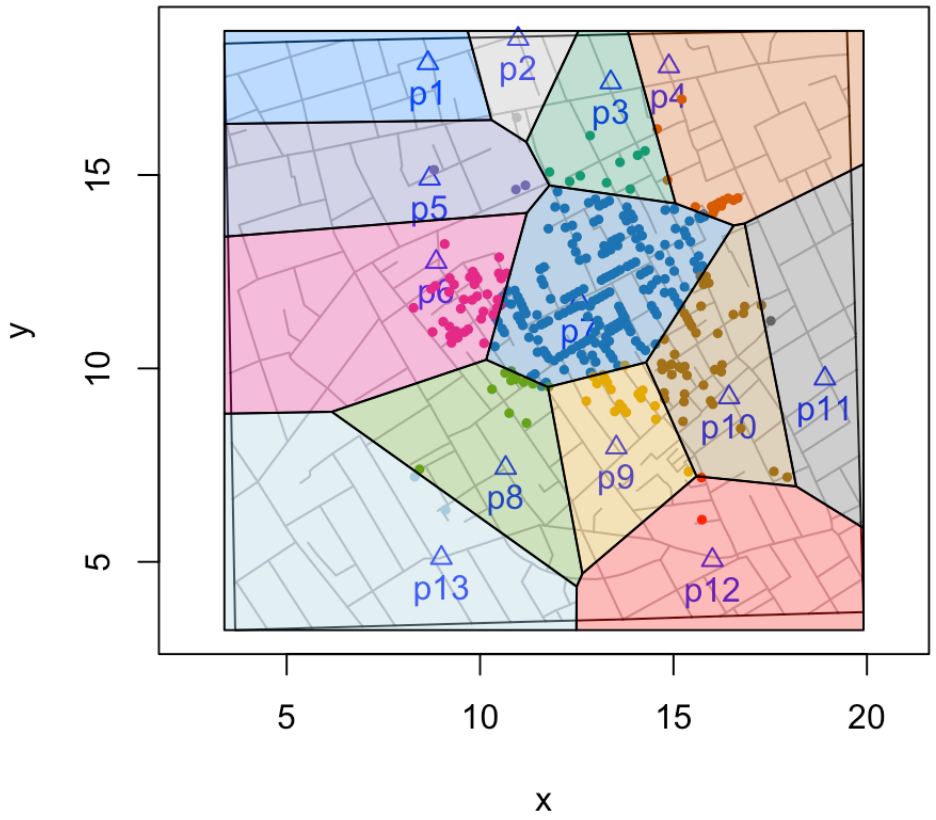
```
snowMap()
polygon(tile7, col = "red")
```



```
snowMap()
polygon(tile7, col = grDevices::adjustcolor("red", alpha.f = 1/3))
```



```
voronoi <- deldir::deldir(sites[, c("x", "y")], rw = c(x.rng, y.rng),
  suppressMsge = TRUE)

cell.data <- deldir::tile.list(voronoi)

polygon.vertices <- lapply(cell.data, function(dat) {
  data.frame(x = dat$x, y = dat$y)
})

snow.colors <- grDevices::adjustcolor(cholera::snowColors(),
  alpha.f = 1/3)
cholera::snowMap(add.cases = FALSE)
cholera::addNeighborhoodCases(metric = "euclidean")

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


p1
p2
p3
p4
p5
p6
p7
p8
p9
p10
p11
p12
p13
y
x
5
10
15
20
5
10
15

cholera::voronoiPolygons()

```
polygon.vertices <- cholera::voronoiPolygons(sites = pumps, rw.data = roads)

snow.colors <- grDevices::adjustcolor(cholera::snowColors(),
  alpha.f = 1/3)
cholera::snowMap(add.cases = FALSE)
cholera::addNeighborhoodCases(metric = "euclidean")

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

sp::point.in.polygon()

Counting elements

syntax

```
sp::point.in.polygon(obs$x, obs$y, tile$x, tile$y)
```

count fatalities by neighborhood

```
cholera::neighborhoodVoronoi(statistic = "fatality")
```

```
##   1   2   3   4   5   6   7   8   9  10  11  12  13
##   0   1  13  23   6  61 361  16  27  62   2   2   4
```

cholera::voronoiPolygons()

```
polygon.vertices <- cholera::voronoiPolygons(sites = pumps, rw.data = roads)

cases <- cholera::fatalities.address

census <- lapply(polygon.vertices, function(tile) {
  sp::point.in.polygon(cases$x, cases$y, tile$x, tile$y)
})

names(census) <- paste0("p", cholera::pumps$id)

vapply(census, sum, integer(1L))
```

```
vapply(census, sum, integer(1L))
```

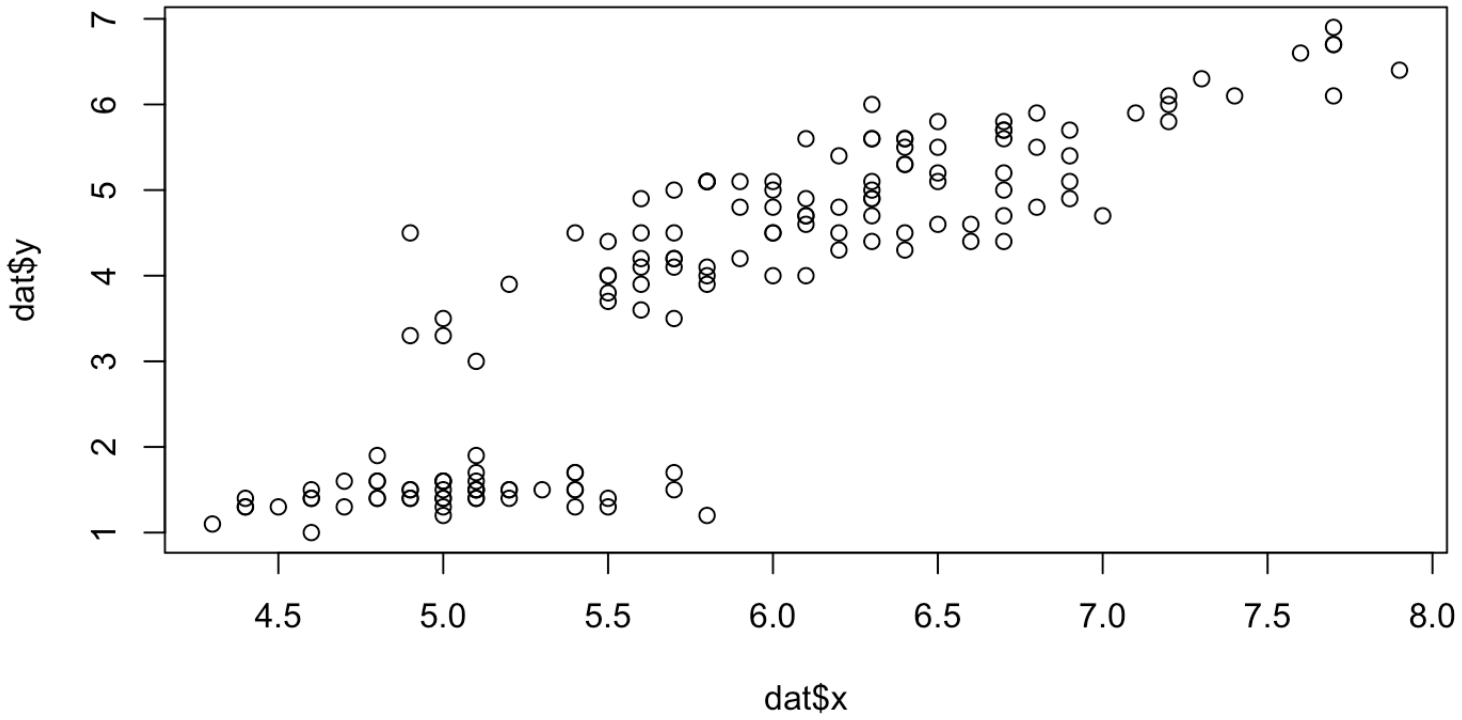
```
##  p1  p2  p3  p4  p5  p6  p7  p8  p9 p10 p11 p12 p13
##   0   1  13  23   6  61 361  16  27  62   2   2   4
```

mouseover "touch" interface

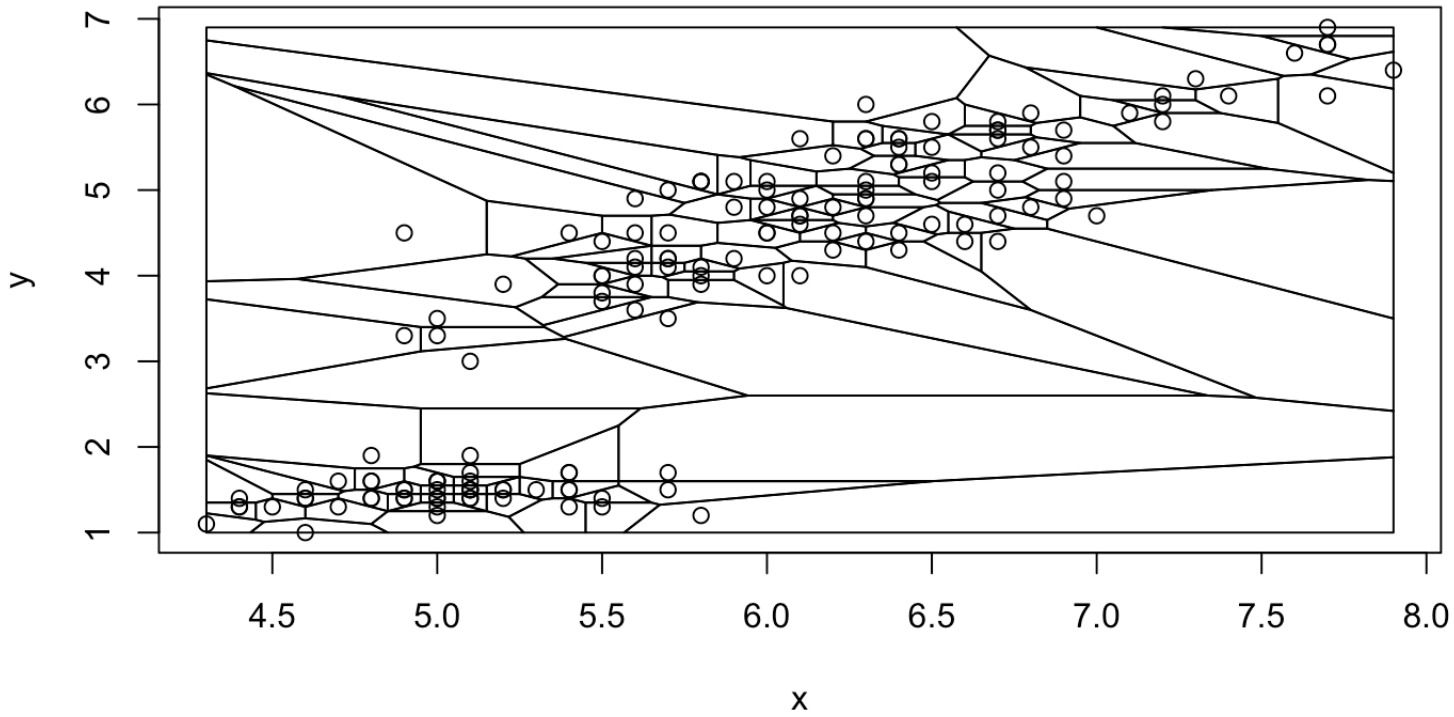
https://bl.ocks.org/mbostock/8033015

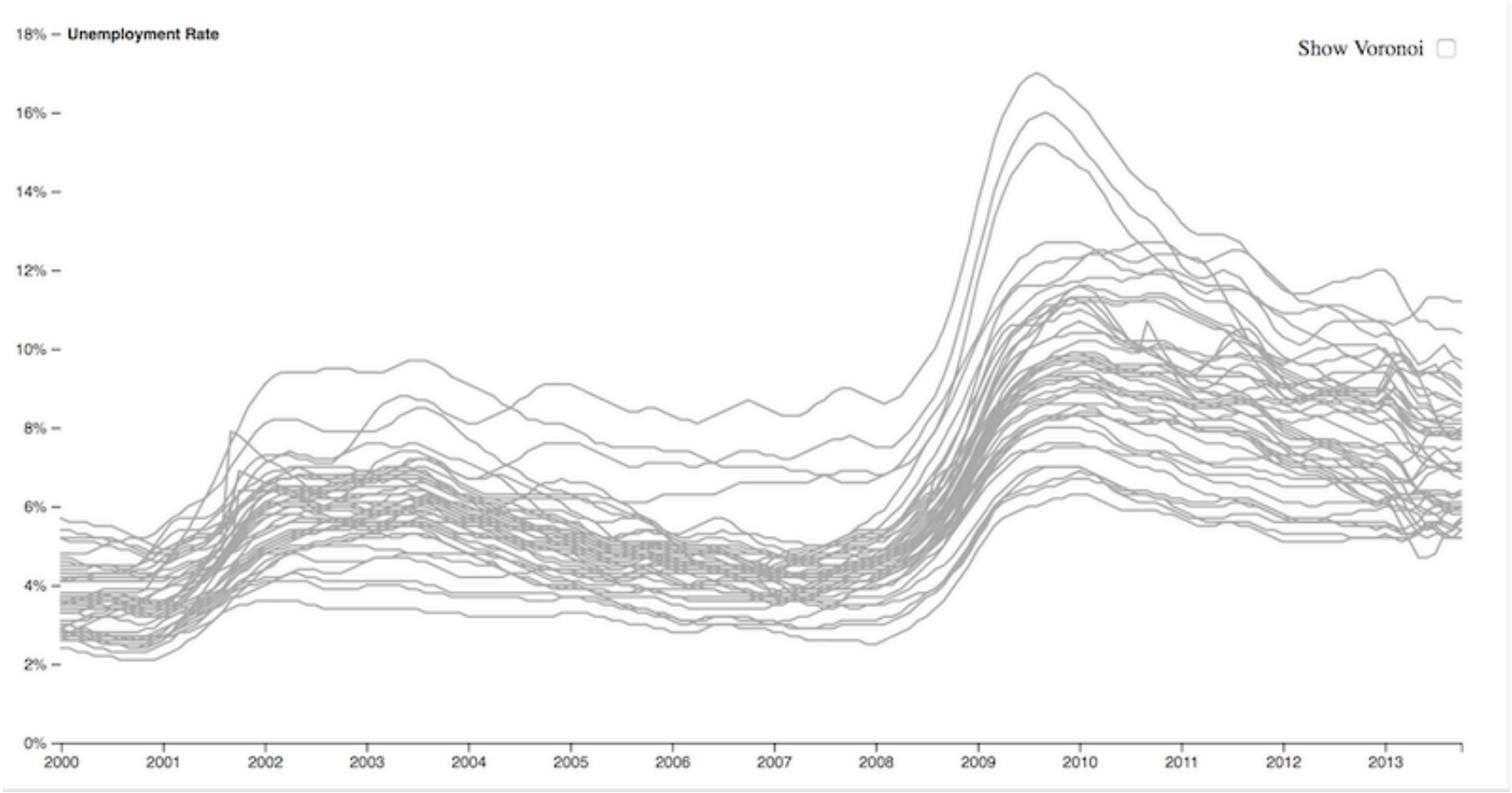
D3.js

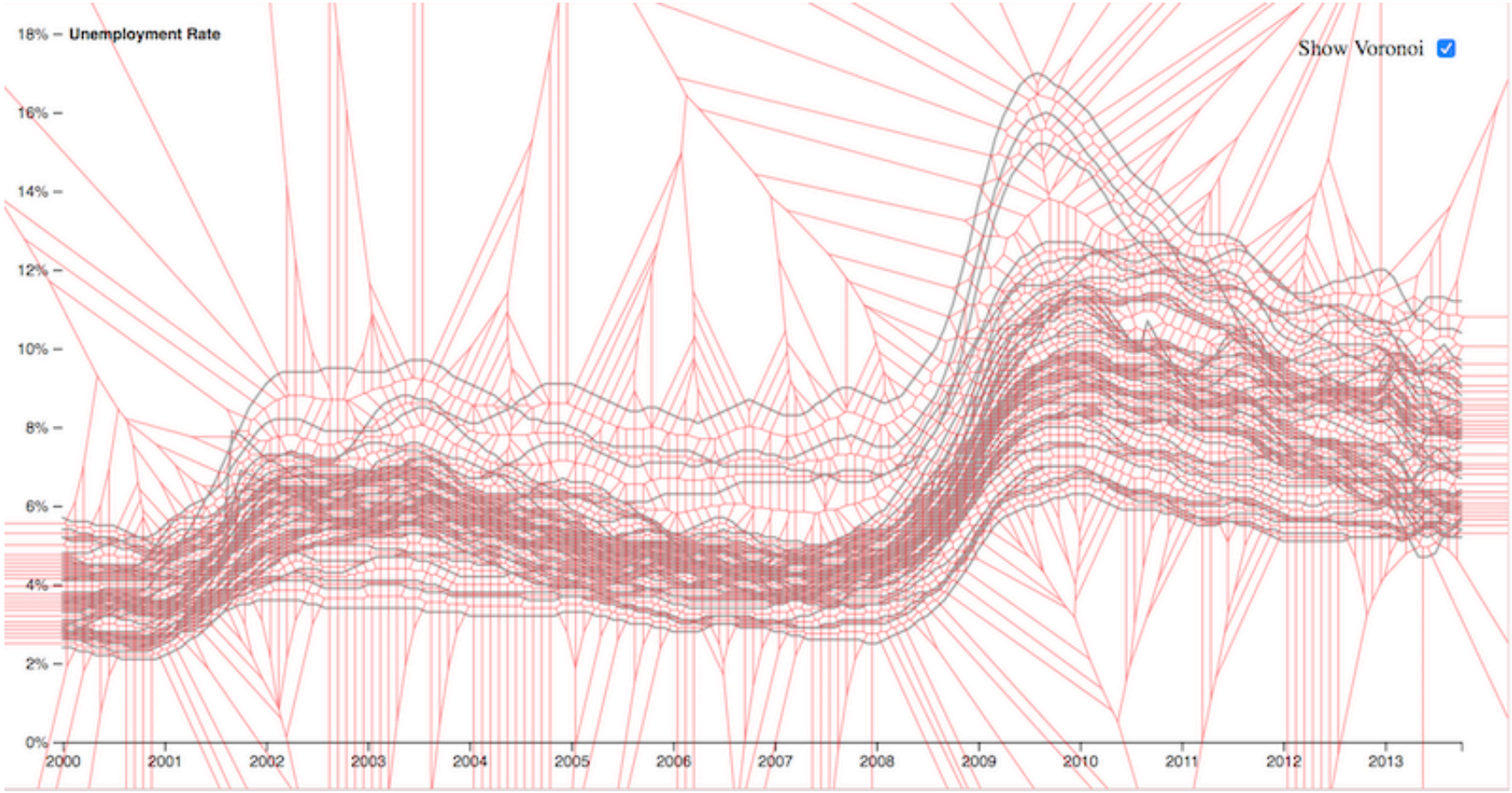
```r
dat <- iris[, c("Sepal.Length", "Petal.Length")]
names(dat) <- c("x", "y")
plot(dat$x, dat$y)
```



```
polygon.vertices <- voronoiPolygons(dat)
plot(dat)
invisible(lapply(polygon.vertices, polygon))
```

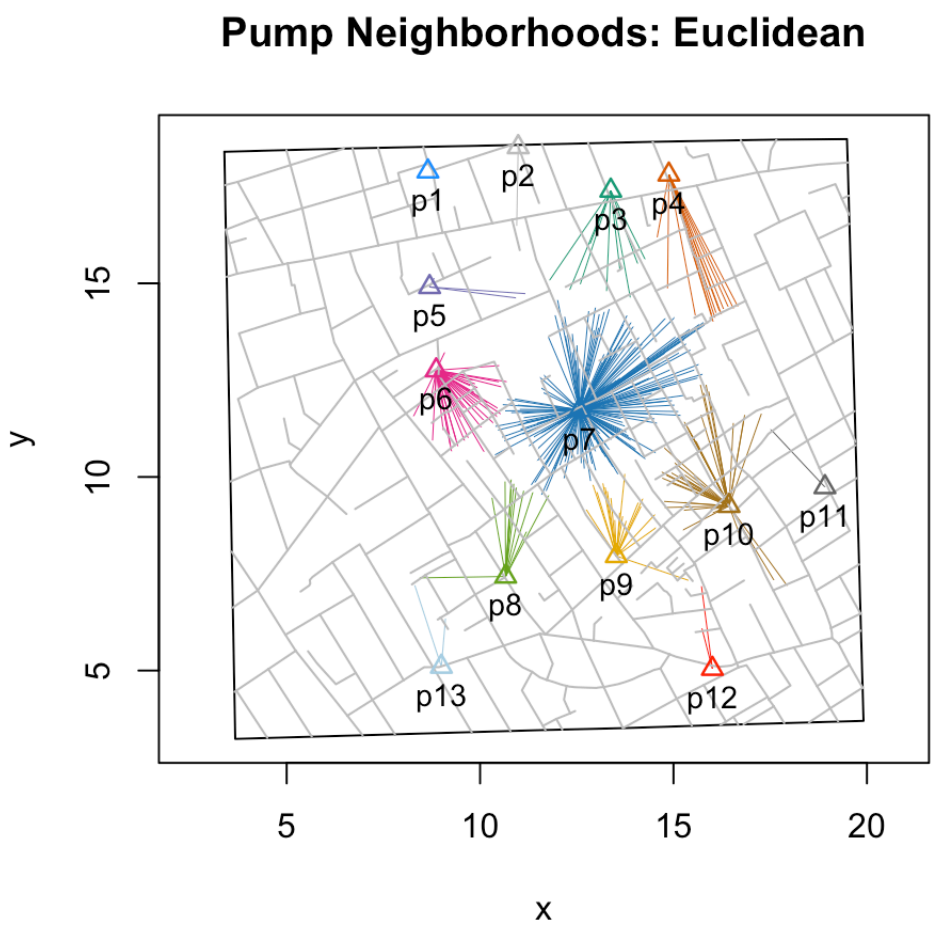



18% – Unemployment Rate
Show Voronoi
16% –
14% –
12% –
10% –
8% –
6% –
4% –
2% –
0%
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013

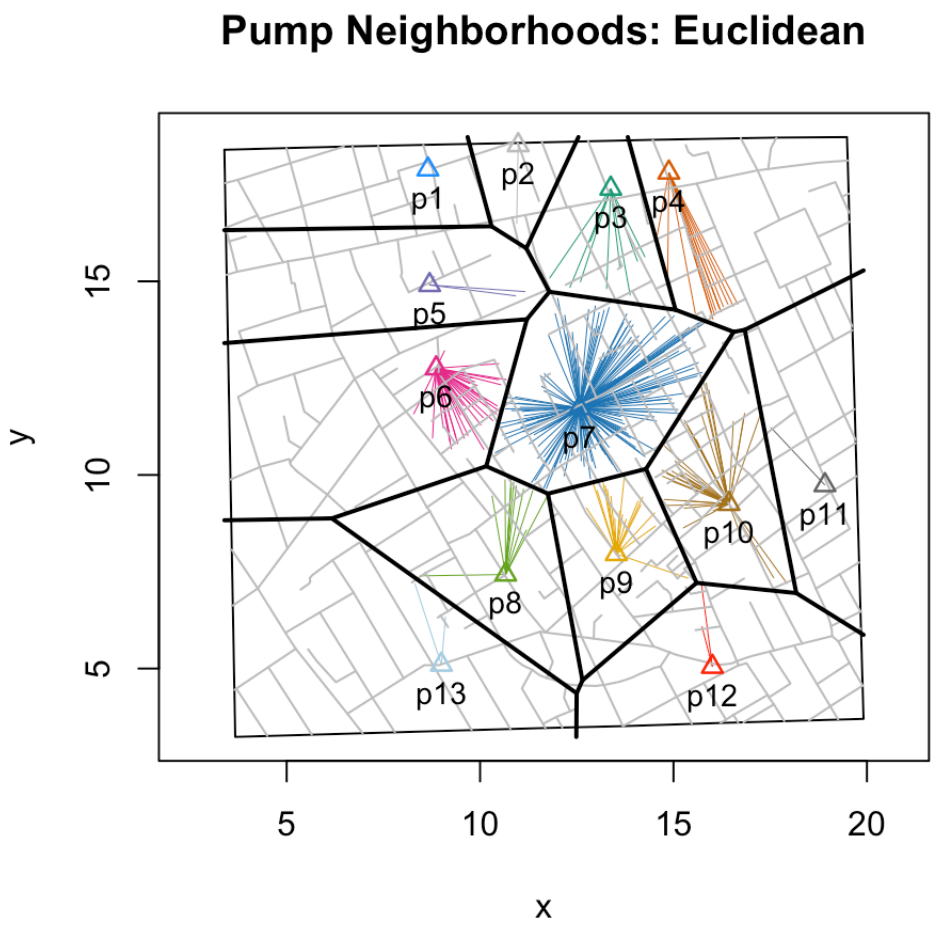

18% – Unemployment Rate
16% –
14% –
12% –
10% –
8% –
6% –
4% –
2% –
0%
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
Show Voronoi

A puzzle?

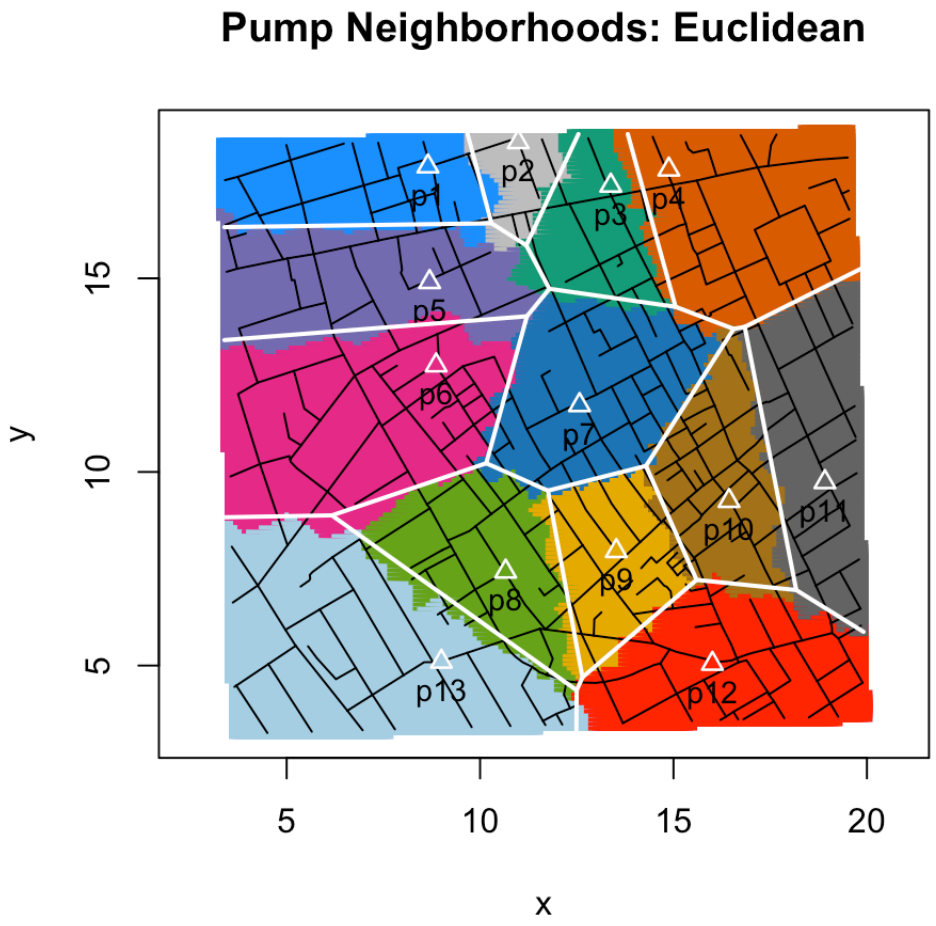
```
plot(cholera::neighborhoodEuclidean())
```



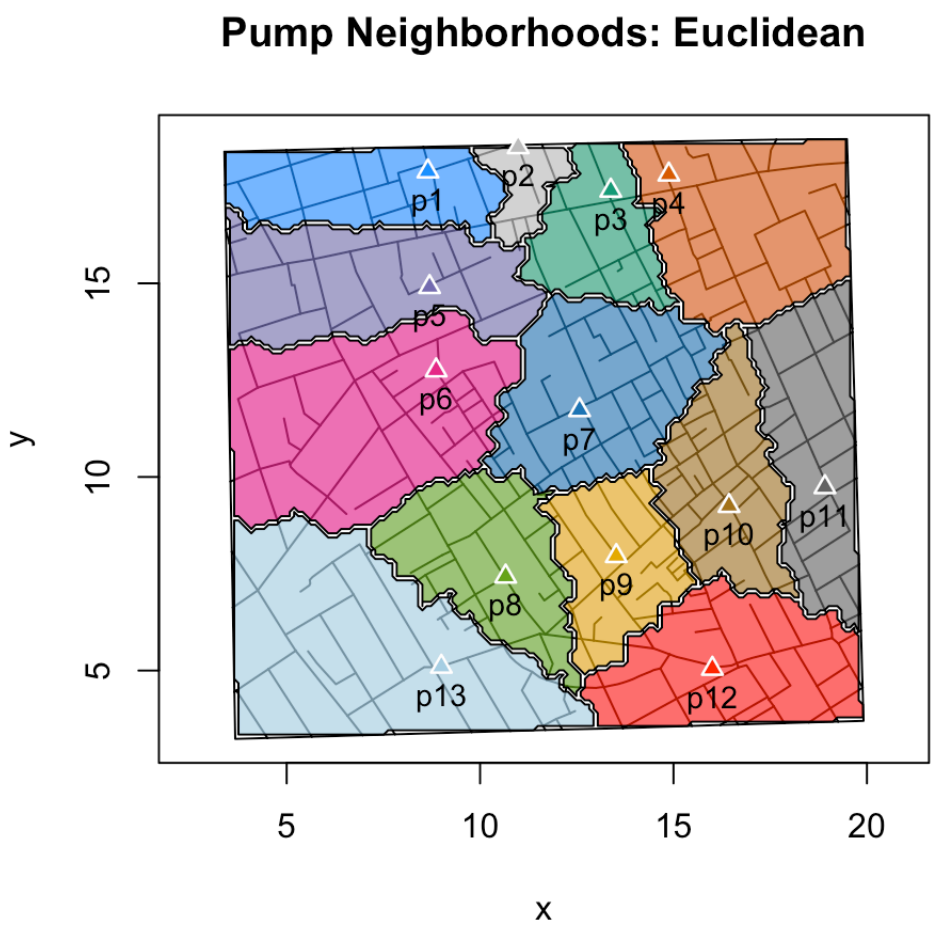
```
plot(cholera::neighborhoodEuclidean()); cholera::addVoronoi(lwd = 2)
```

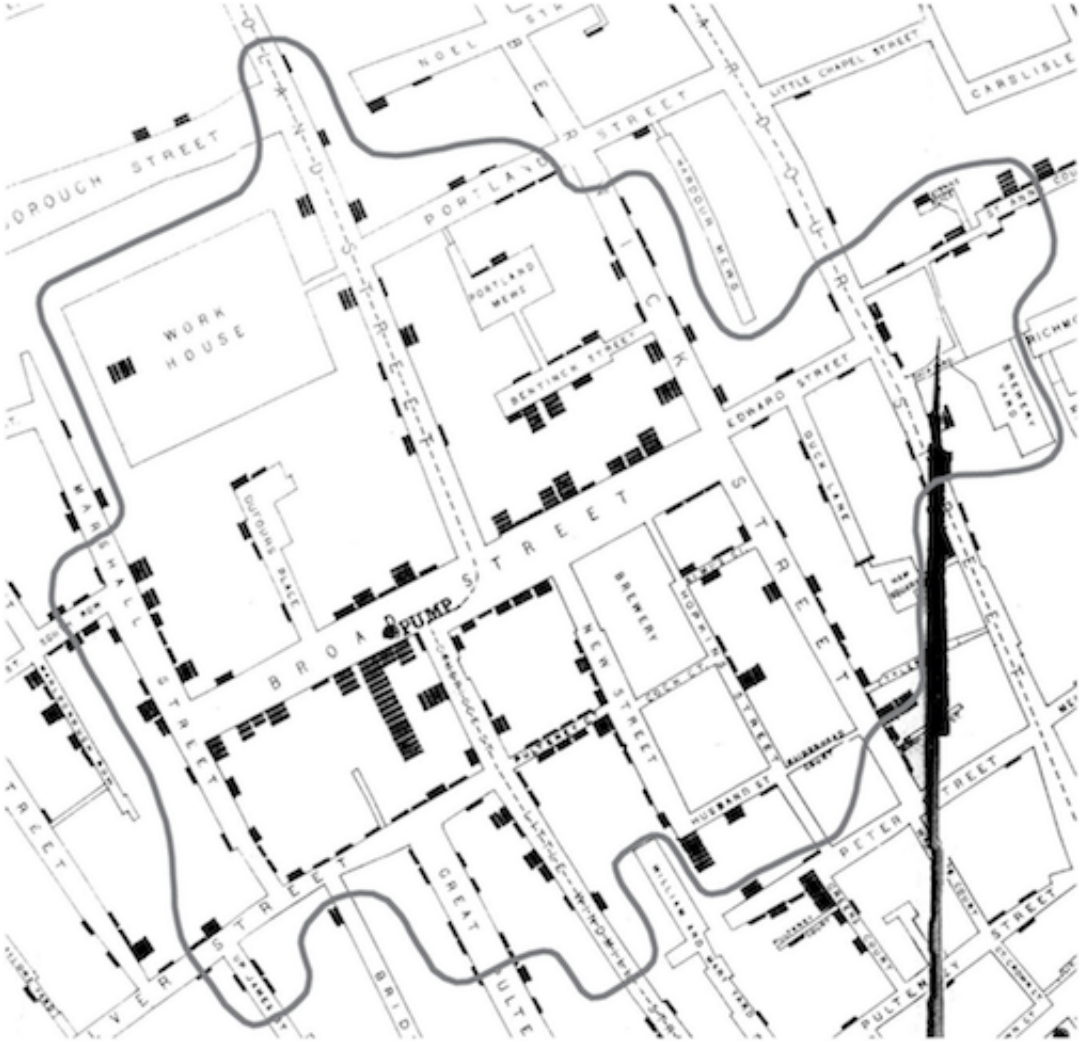


```
plot(cholera::neighborhoodEuclidean(case.set = "expected"), "area.points")
cholera::addVoronoi(color = "white", lwd = 2)
```

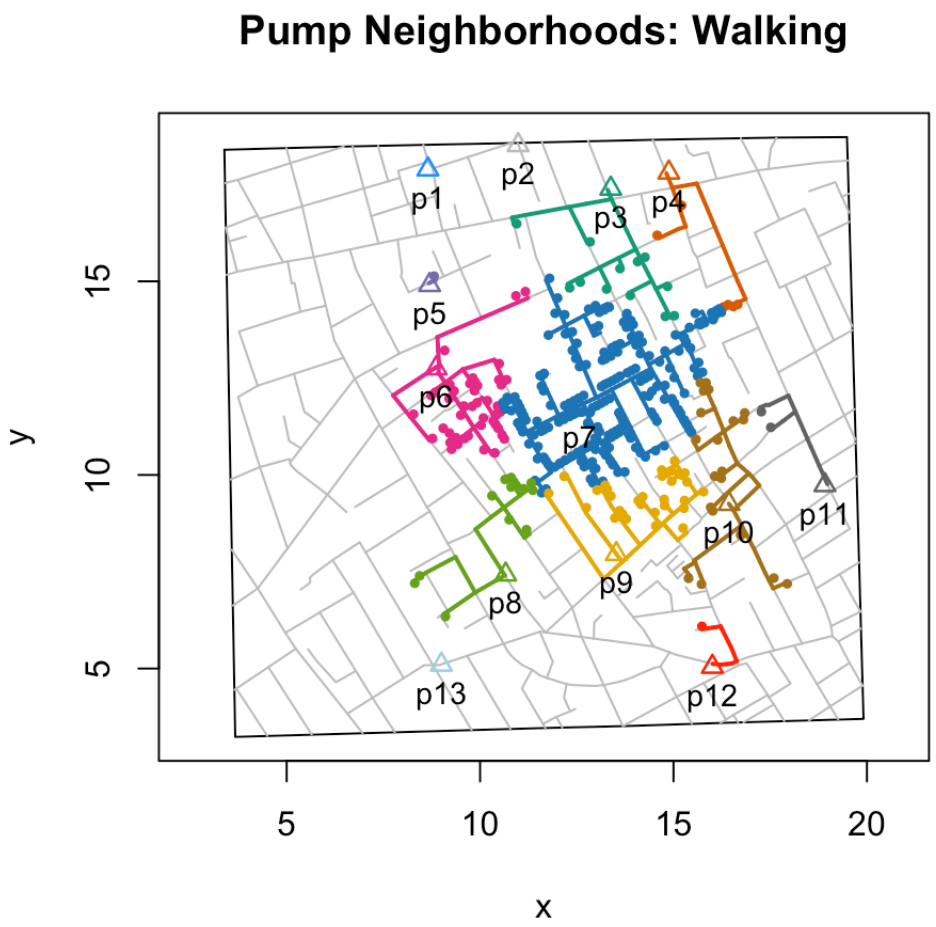


```
plot(cholera::neighborhoodEuclidean(case.set = "expected"), "area.polygons")
```

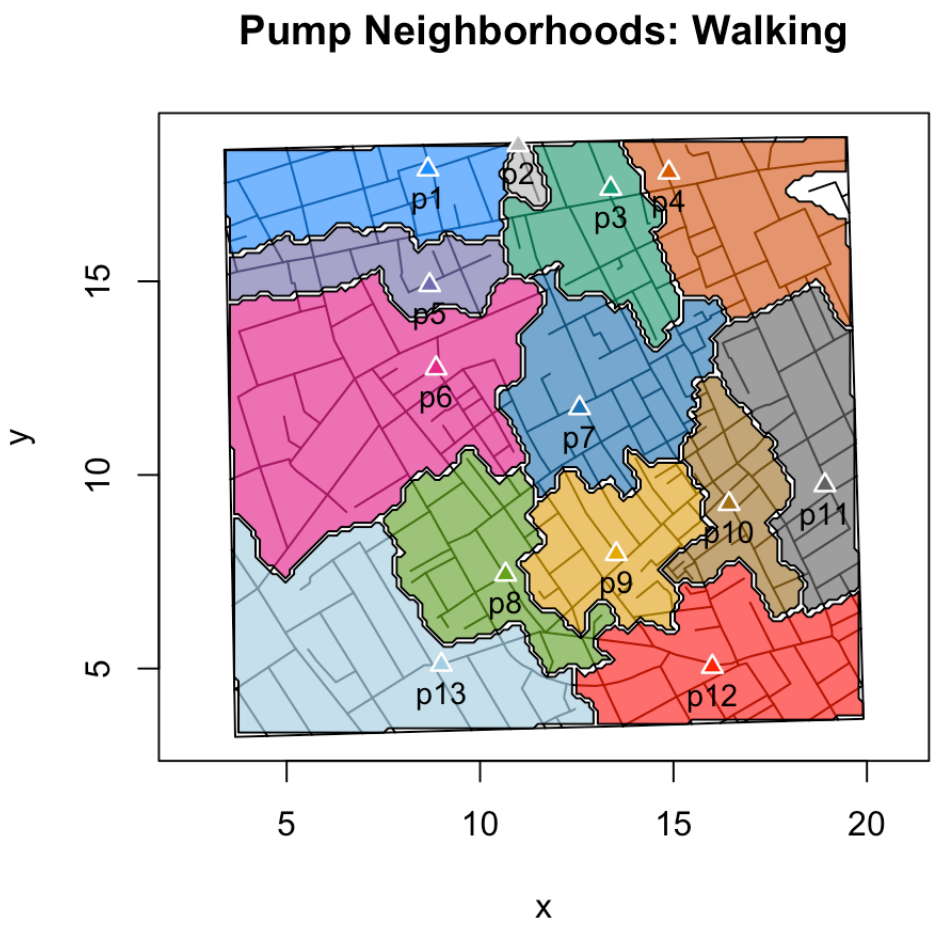



WORK HOUSE
PUMP
BROAD STREET
BREWERY
NEW STREET
EDWARD STREET
PETER STREET
GREAT PULTENEY STREET
PORTLAND MEWS
BENJAMIN STREET
HUSBAND ST
LITTLE CHAPEL STREET

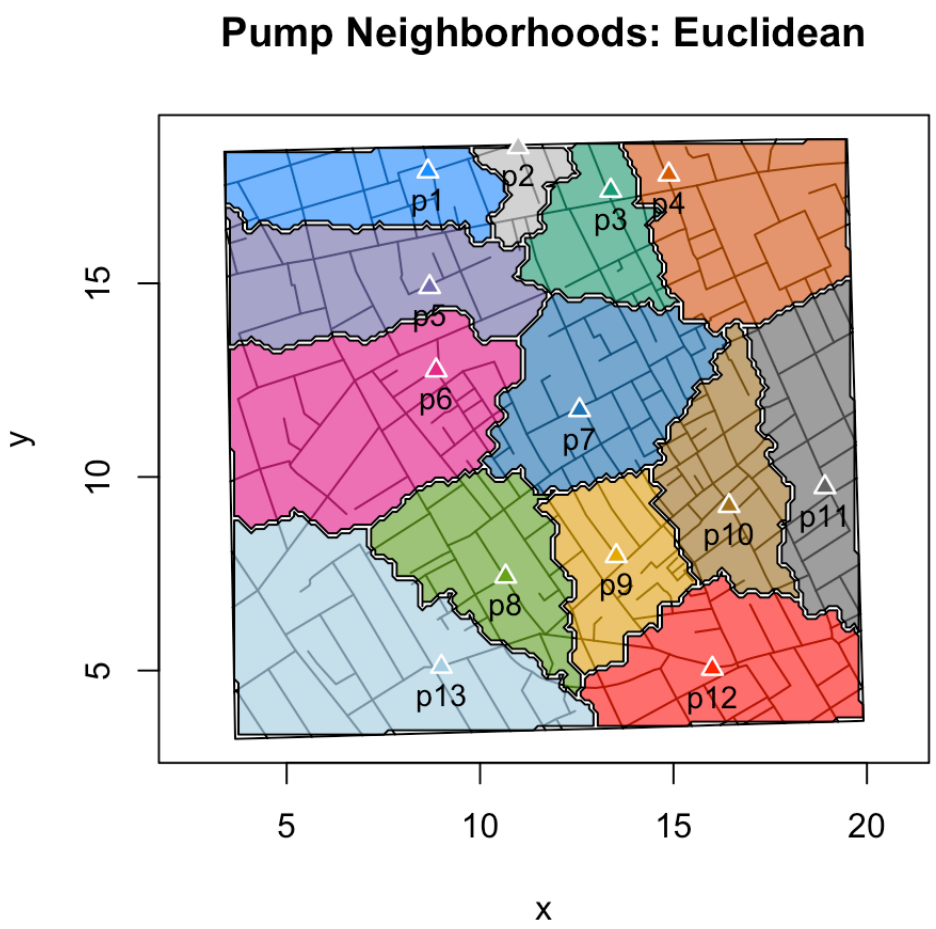
```
plot(cholera::neighborhoodWalking())
```



```
plot(cholera::neighborhoodWalking(case.set = "expected"), "area.polygons")
```



```
plot(cholera::neighborhoodEuclidean(case.set = "expected"), "area.polygons")
```



CRAN: https://CRAN.R-project.org/package=cholera/

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