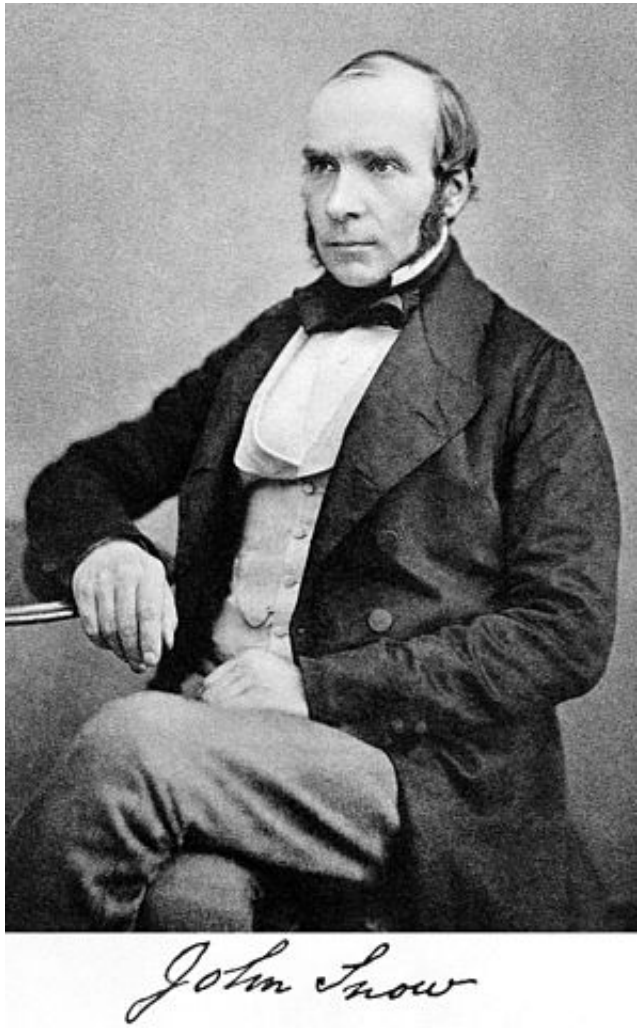
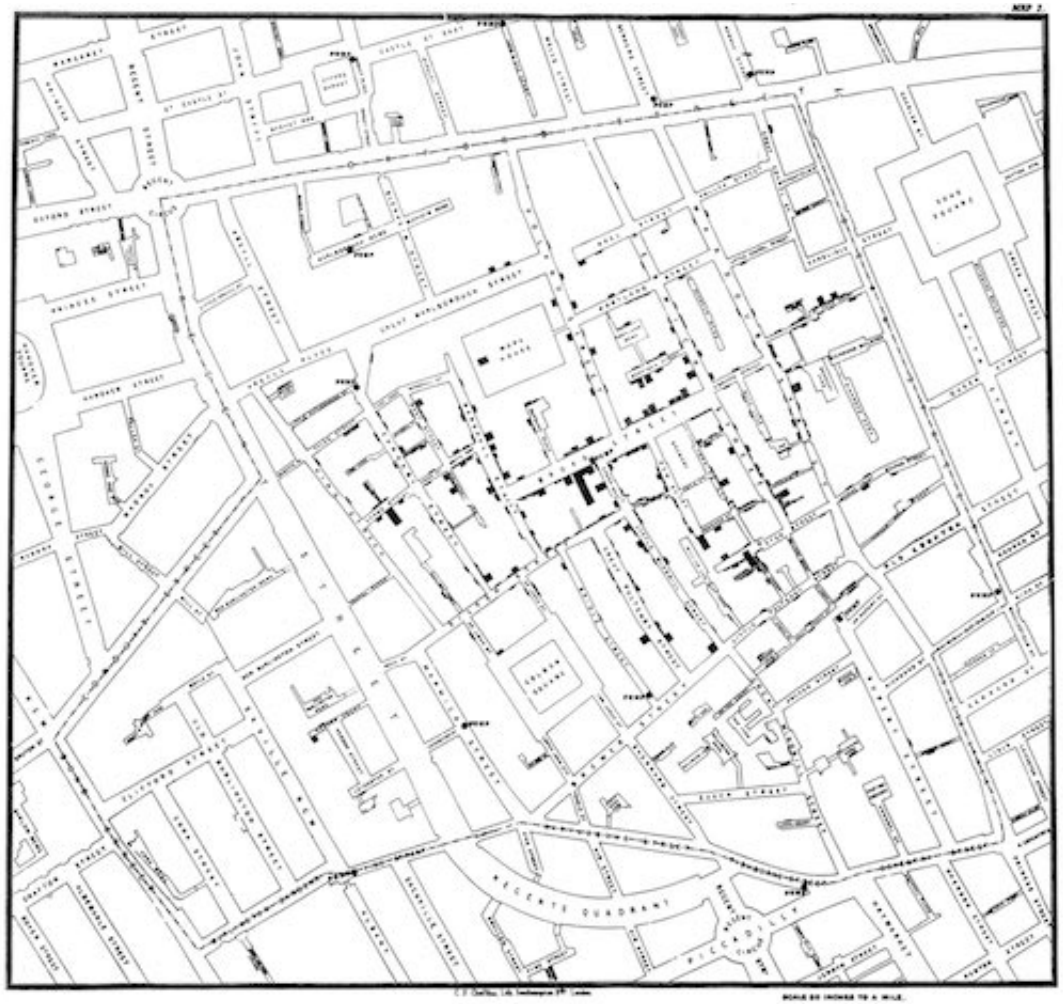
An introduction to the 'cholera' package

Bay Area R Users Group | 2018-October-09 | Peter Li

Dr. John Snow



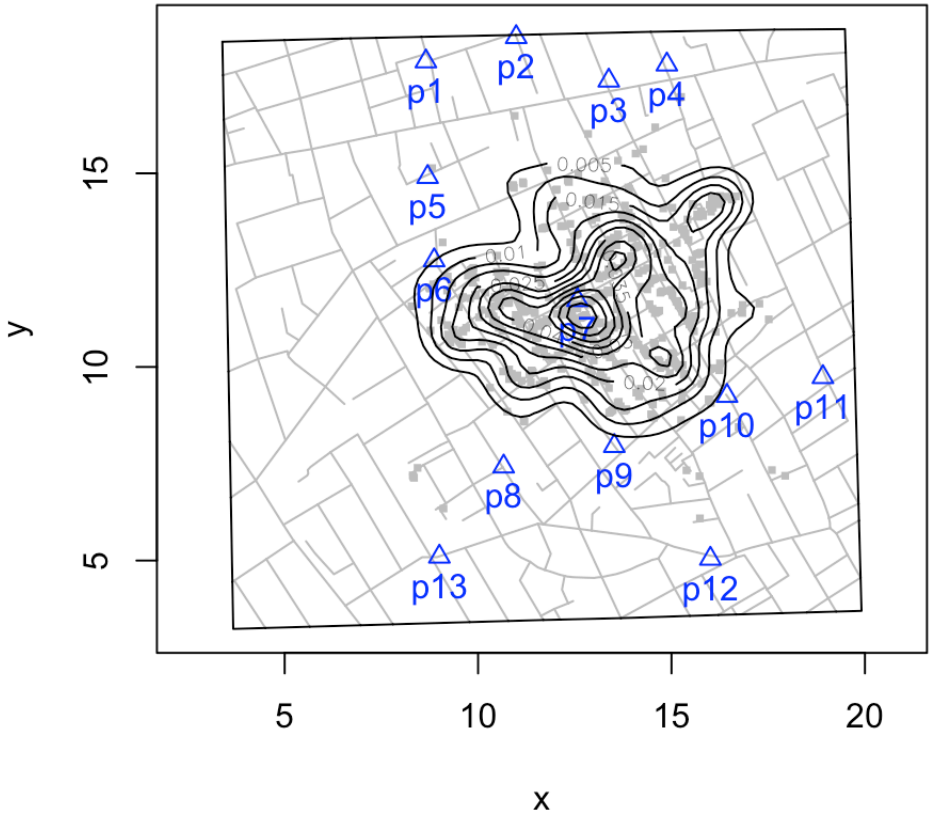
map version 1.0


REGENT QUADRANT
PICCADILLY

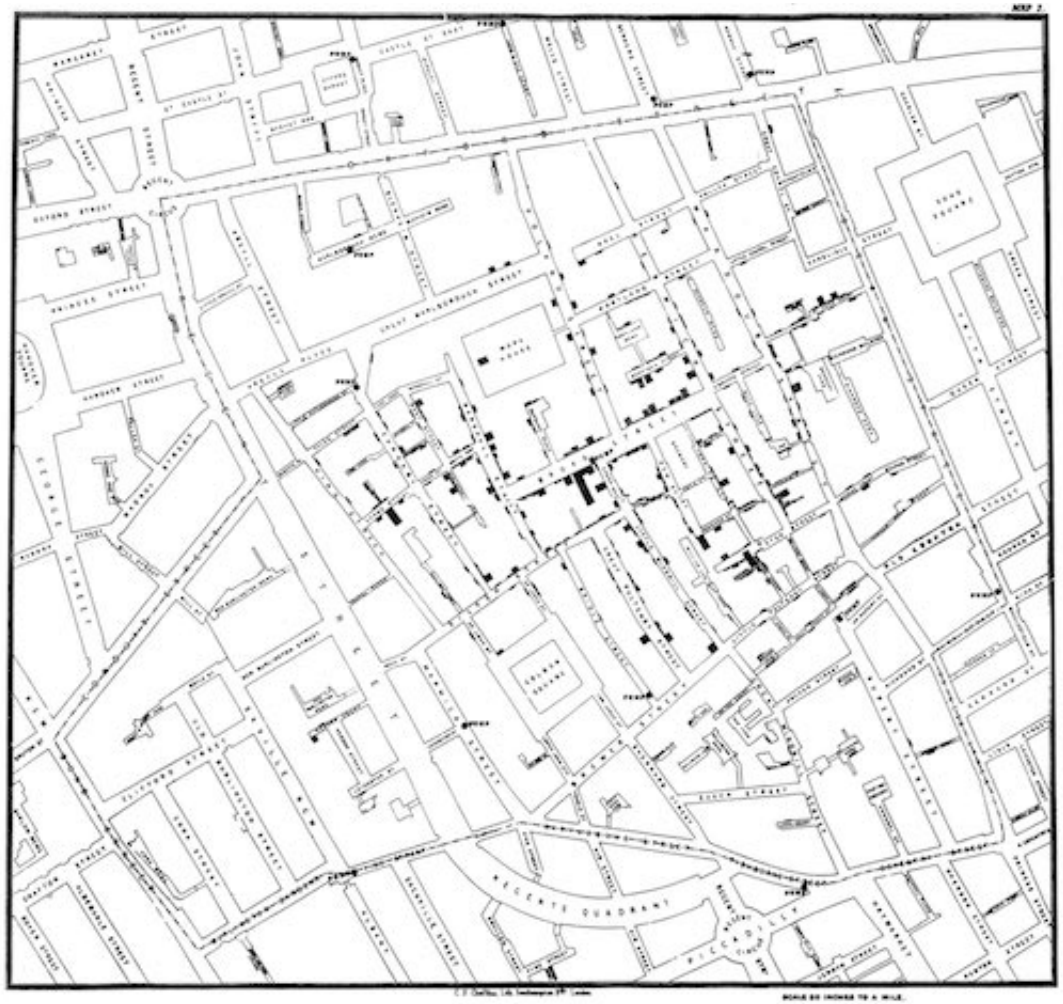
two claims

- cholera is waterborne disease
- the Broad Street pump is the source of the outbreak

```
snowMap(); addKernelDensity(obs.unit = "fatality")
```

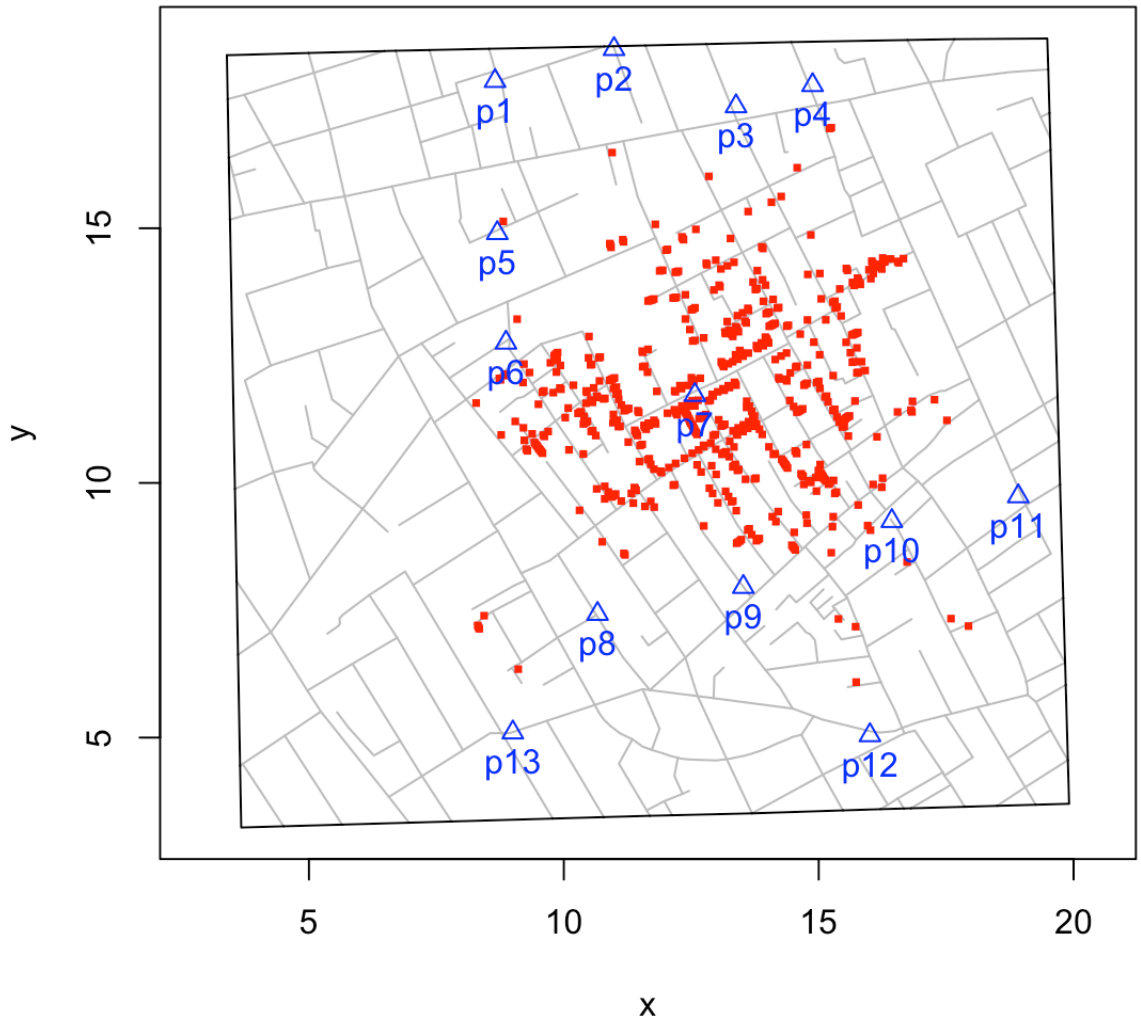


take a second look


REGENT STREET
REGENT QUADRANT
PICCADILLY

questions

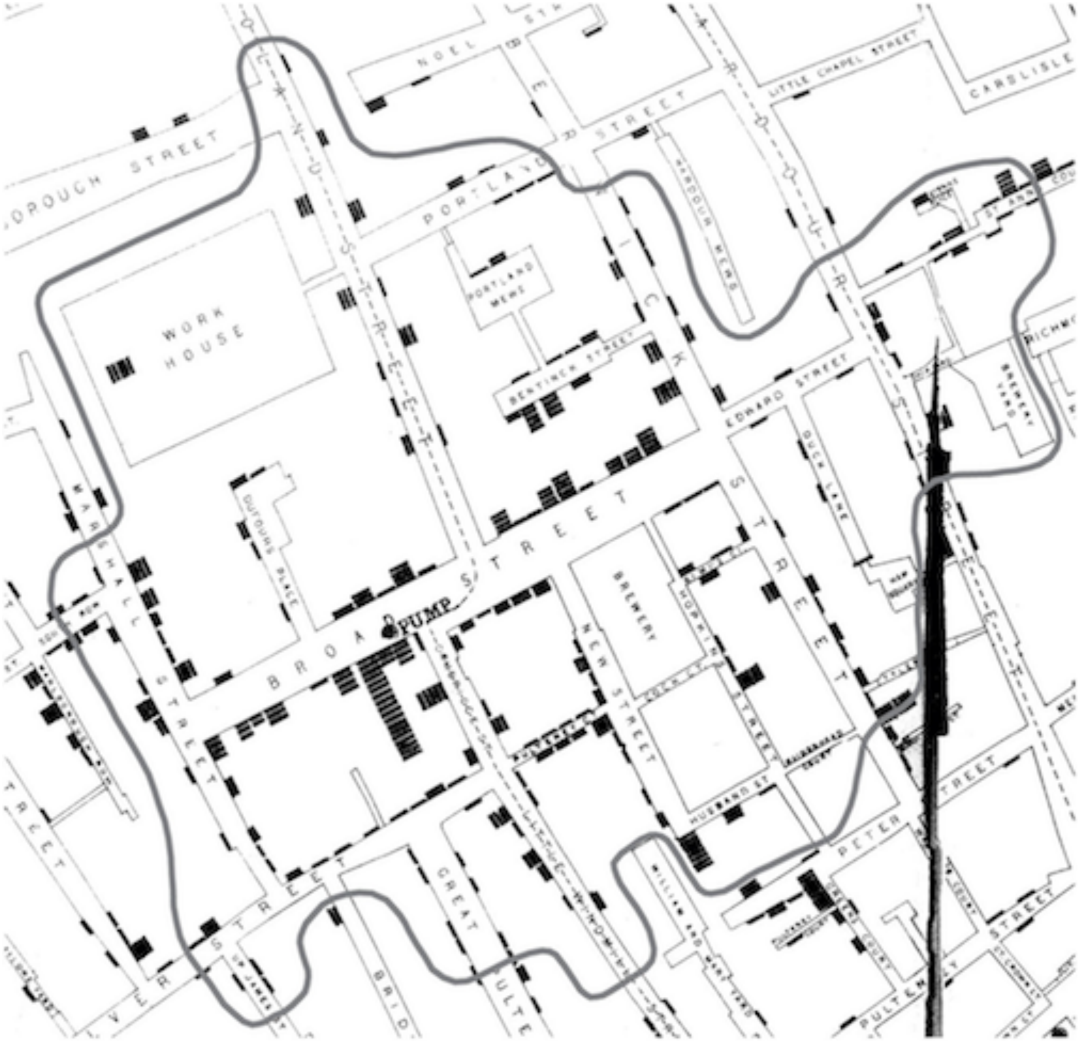
- does the map support his claims?


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

more questions

- what would waterborne transmission look like?
- what would airborne transmission (miasma) look like?
- does the map help us to distinguish the two?

map version 2.0


WORK HOUSE
PUMP
BROAD STREET
PORTLAND MEWS
BENTINCK STREET
BREWERY
LITTLE CHAPEL STREET
CARLISLE
EDWARD STREET
DUCK LANE
DUFOURS PLACE
NEW STREET
HUSBAND ST
PETER
NOEL

Snow says …

"The inner dotted line on the map shews [sic] the various points which have been found by careful measurement to be at an equal distance by the nearest road from the pump in Broad Street and the surrounding pumps …"

"… it will be observed that the deaths either very much diminish, or cease altogether, at every point where it becomes decidedly nearer to send to another pump than to the one in Broad Street".

- Report On The Cholera Outbreak In The Parish Of St. James, Westminster, During The Autumn Of 1854, p. 109.

why the annotation?

- if cholera is waterborne & …
- if pumps are the source of water,
- then the extent of outbreak is limited by a pump's reach

a pump neighborhood

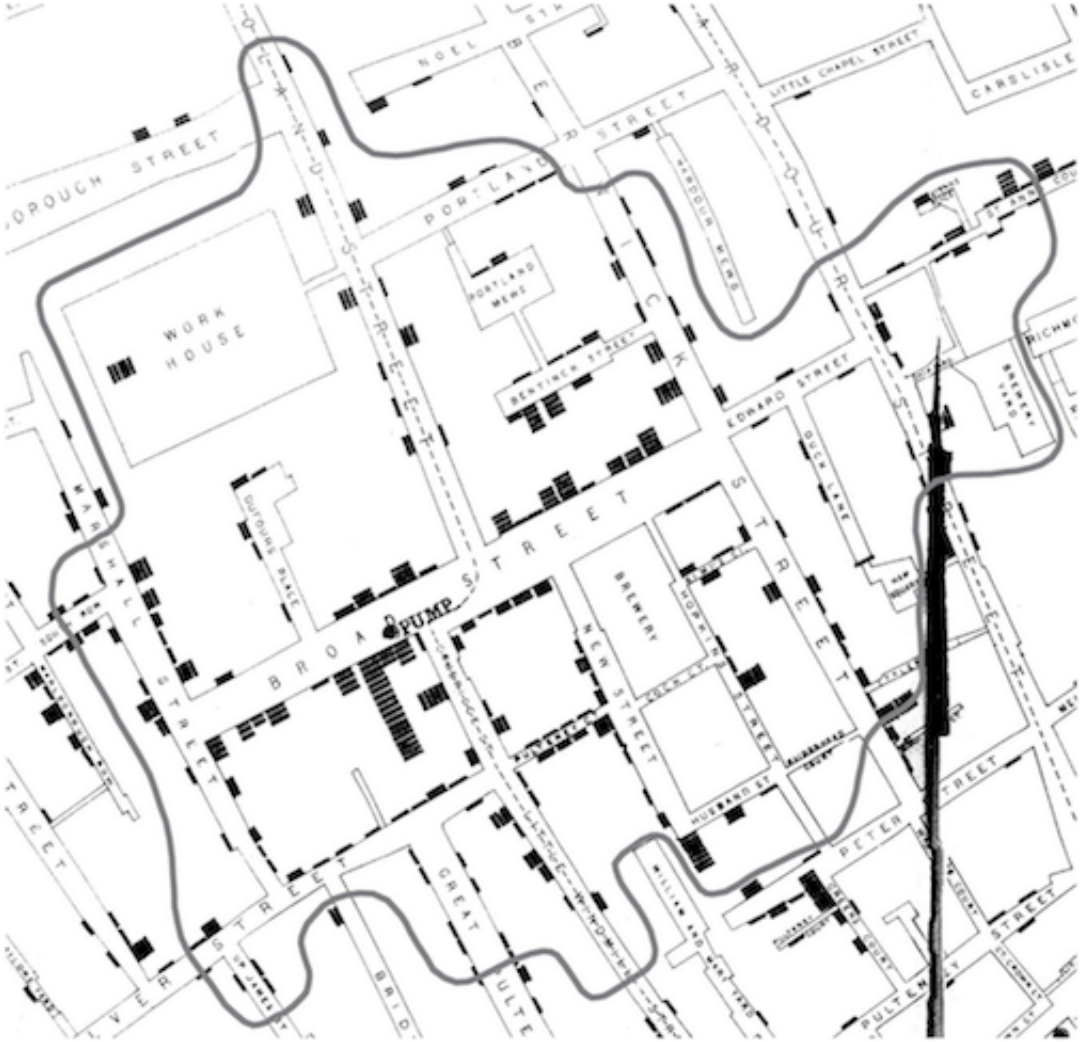
- the set of homes defined by proximity to a pump
- where *TO* find cases (inside)
- where *NOT* to find cases (outside)

intuition and justification

- proximity matters
- fetching water from a well is a chore
- so choose the nearest pump
- (rival explanations)

pump neighborhoods

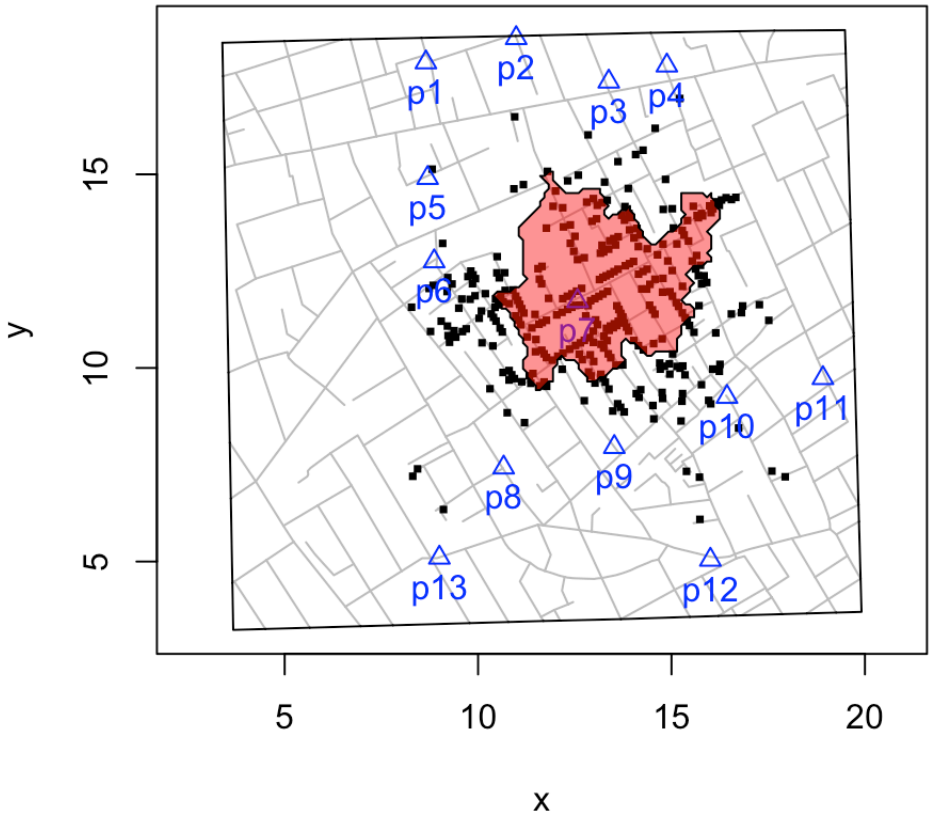
- "choose the nearest" implies choice
- other pumps shape a pump's neighborhood


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

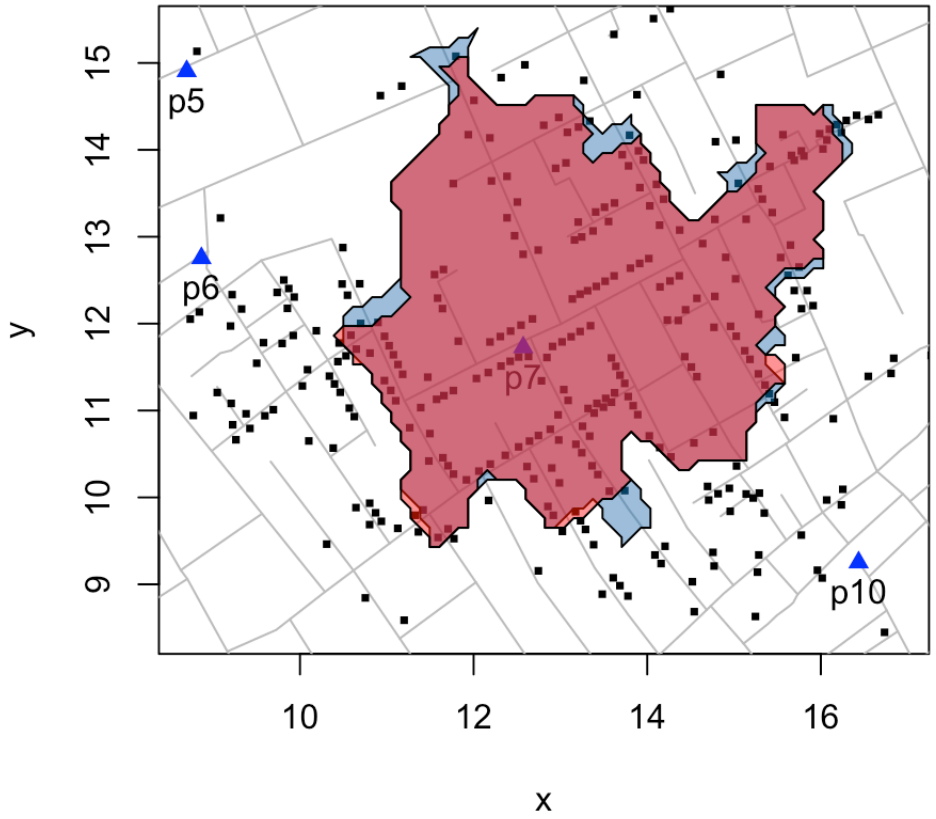
a method to the madness?

- hand drawn, eyeballed quality

explanatory fit



quality of effort



limitations

- what about other neighborhoods?
- what about hypothetical/counterfactual neighborhoods?

Cliff and Haggett (1988)

'deldir' and 'sp'

Voronoi neighborhoods

```
pumps
```

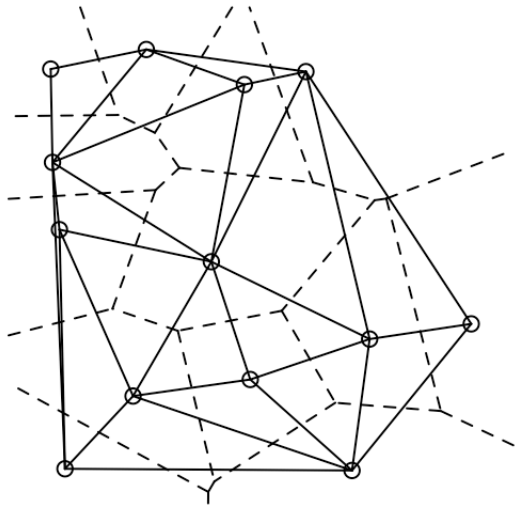
```
##    id                    street         x         y
## 1   1             Market Place  8.651201 17.891600
## 2   2       Adam and Eve Court 10.984780 18.517851
## 3   3           Berners Street 13.378190 17.394541
## 4   4            Newman Street 14.879830 17.809919
## 5   5         Marlborough Mews  8.694768 14.905470
## 6   6 Little Marlborough Street  8.864416 12.753540
## 7   7             Broad Street 12.571360 11.727170
## 8   8           Warwick Street 10.660970  7.428647
## 9   9            Bridle Street 13.521460  7.958250
## 10 10            Rupert Street 16.434891  9.252130
## 11 11              Dean Street 18.914391  9.737819
## 12 12         Tichborne Street 16.005110  5.046838
## 13 13              Vigo Street  8.999440  5.101023
```

'deldir'

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

triangulation and tessellation

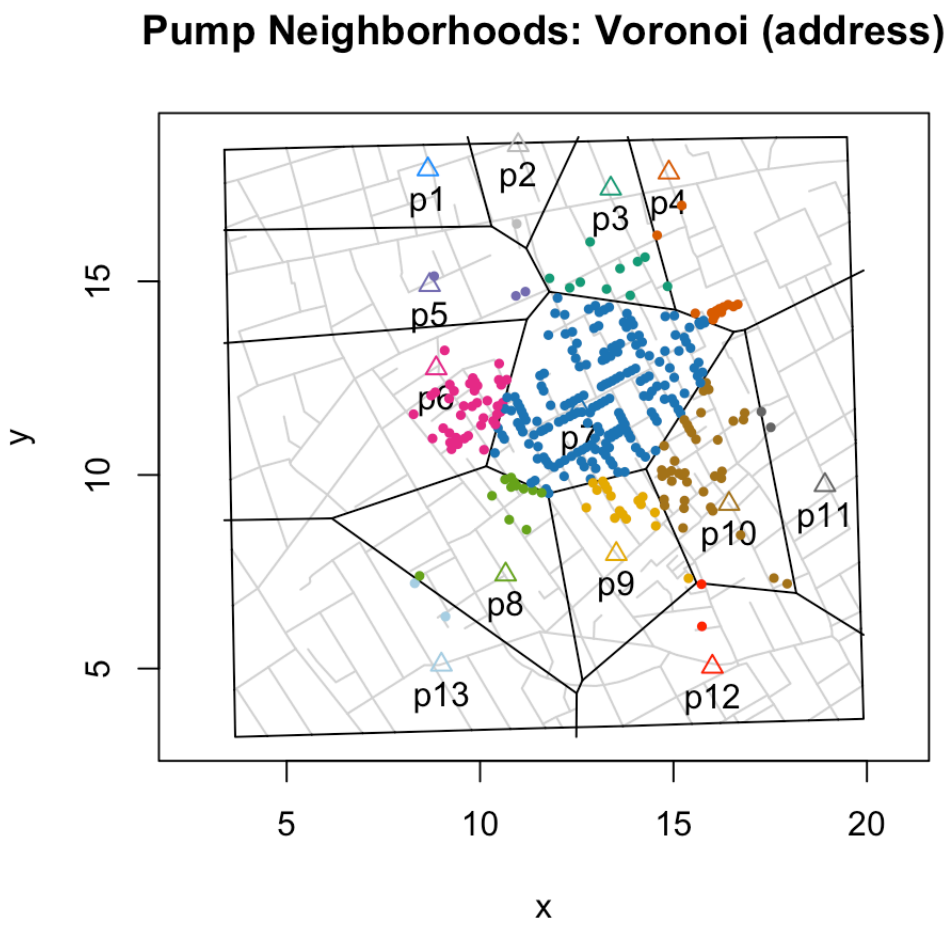
```
plot(deldir::deldir(pumps[, c("x", "y")], suppressMsge = TRUE))
```



points in polygon via 'sp'

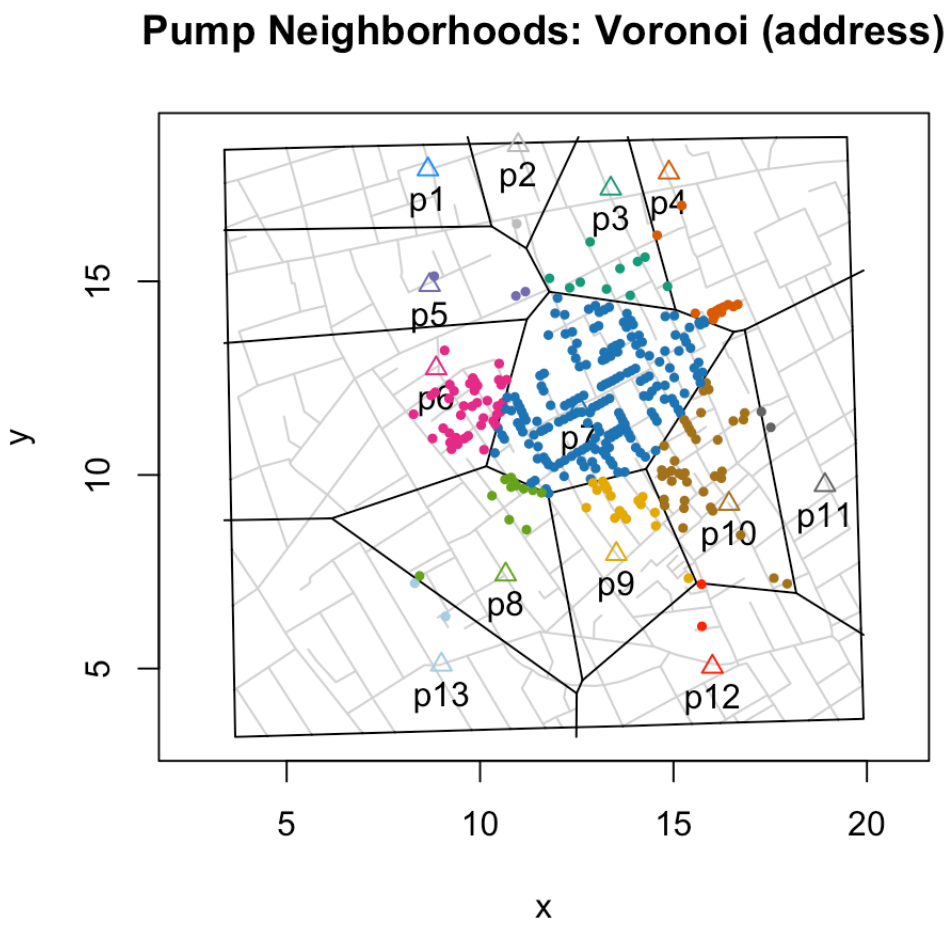
```
sp::point.in.polygon(cholera::fatalities.address$x,
  cholera::fatalities.address$y, cell$x, cell$y)
```

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

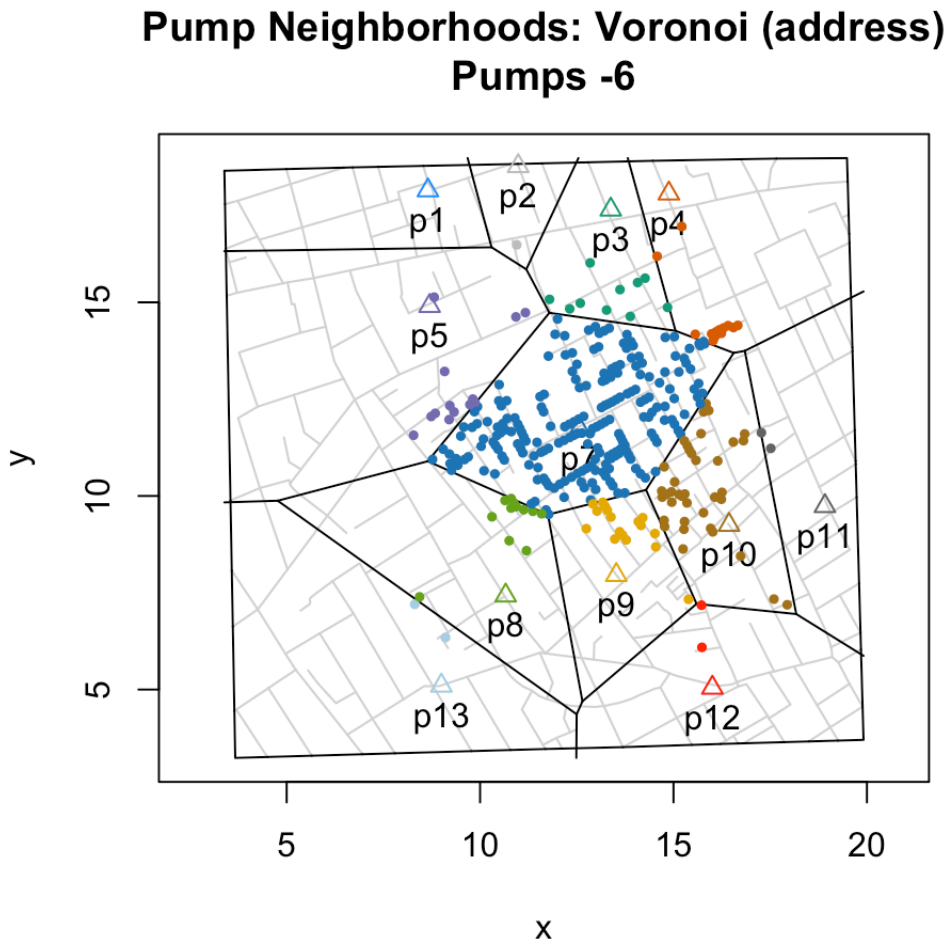


scenarios

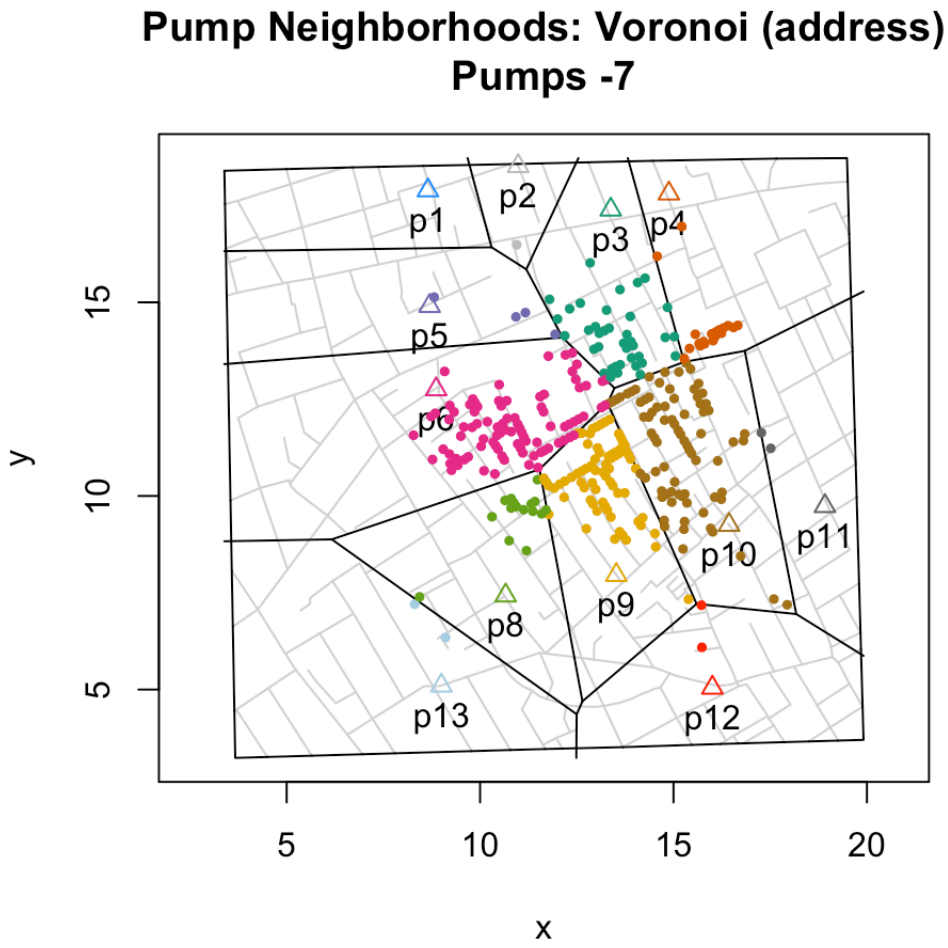
```
plot(neighborhoodVoronoi())
```



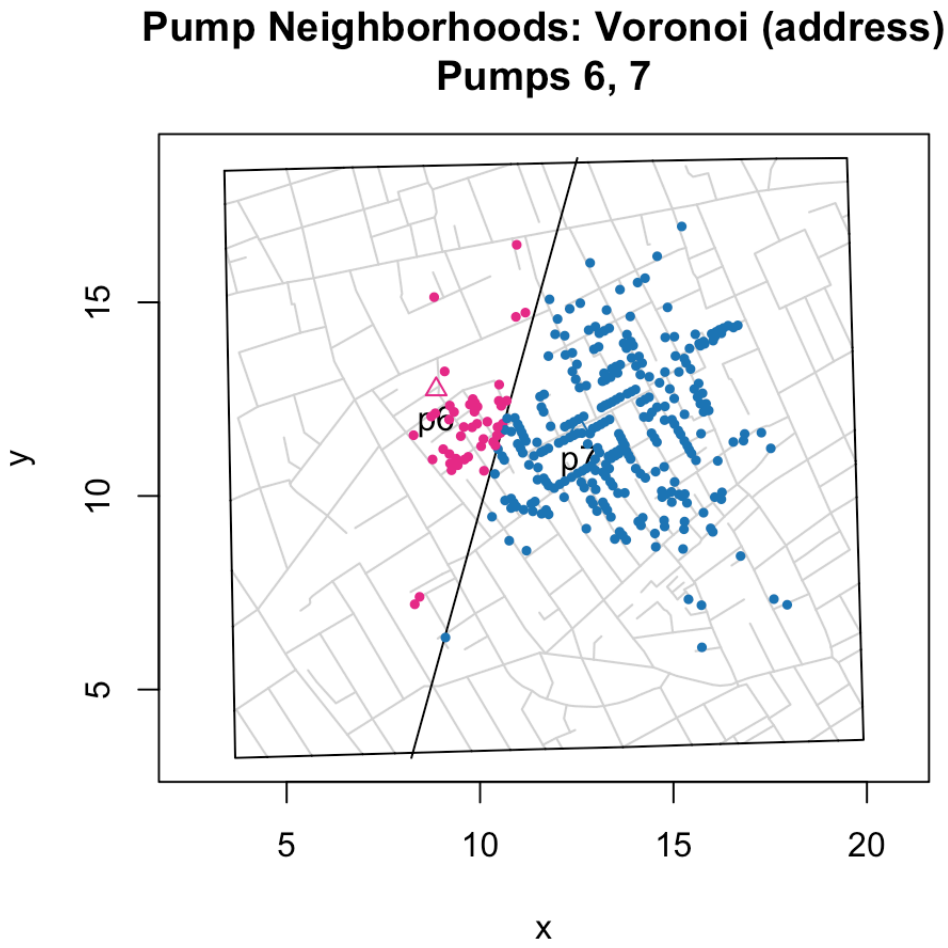
```
plot(neighborhoodVoronoi(-6))
```



```
plot(neighborhoodVoronoi(-7))
```



```
plot(neighborhoodVoronoi(6:7))
```



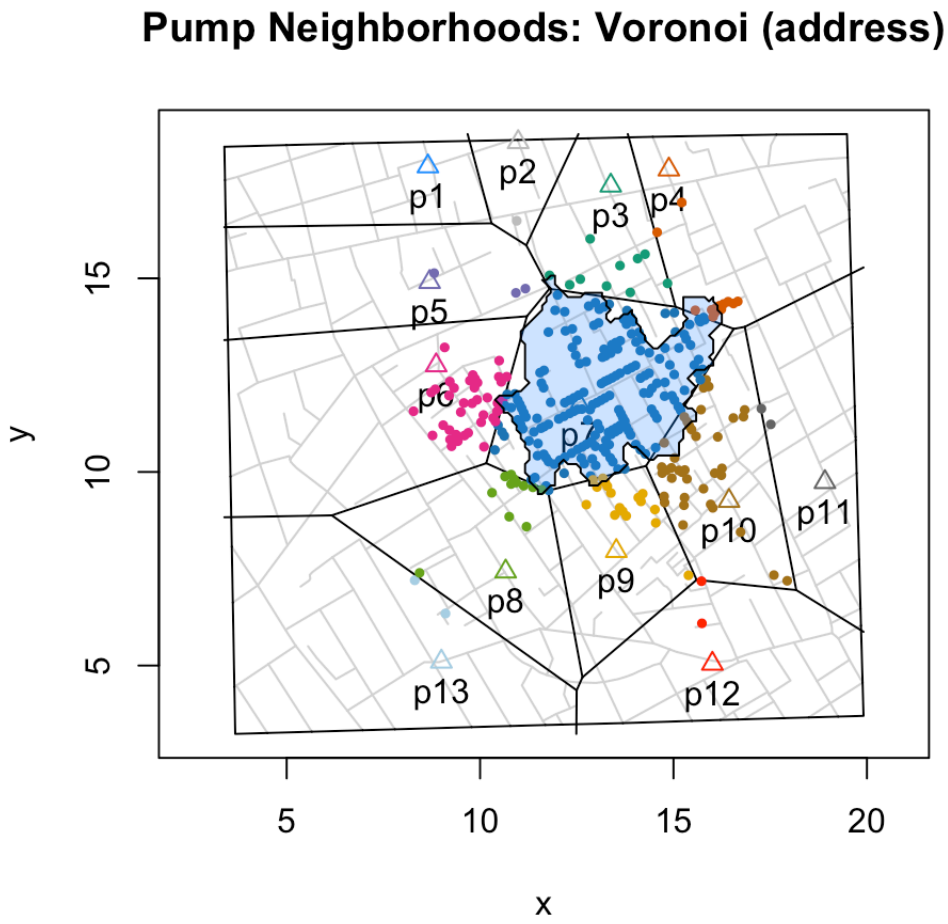
pros

- fast (uses only location of pumps)

cons

- doesn't quite match Snow's annotation

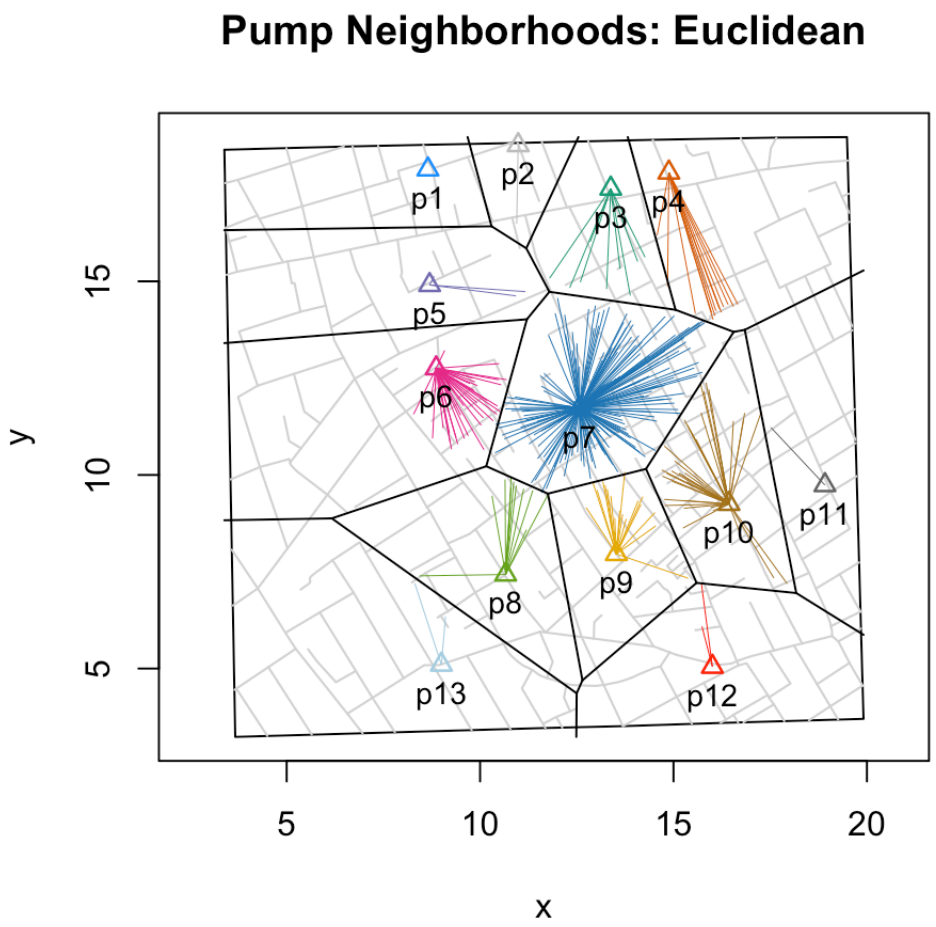
```
plot(neighborhoodVoronoi()); addSnow()
```



cons (more)

- uses only location of pumps
- Euclidean distance (Snow implies walking distance)
- network of roads and streets don't matter
- walk through walls

```
plot(neighborhoodEuclidean(case.set = "observed")); addVoronoi()
```



address

road network

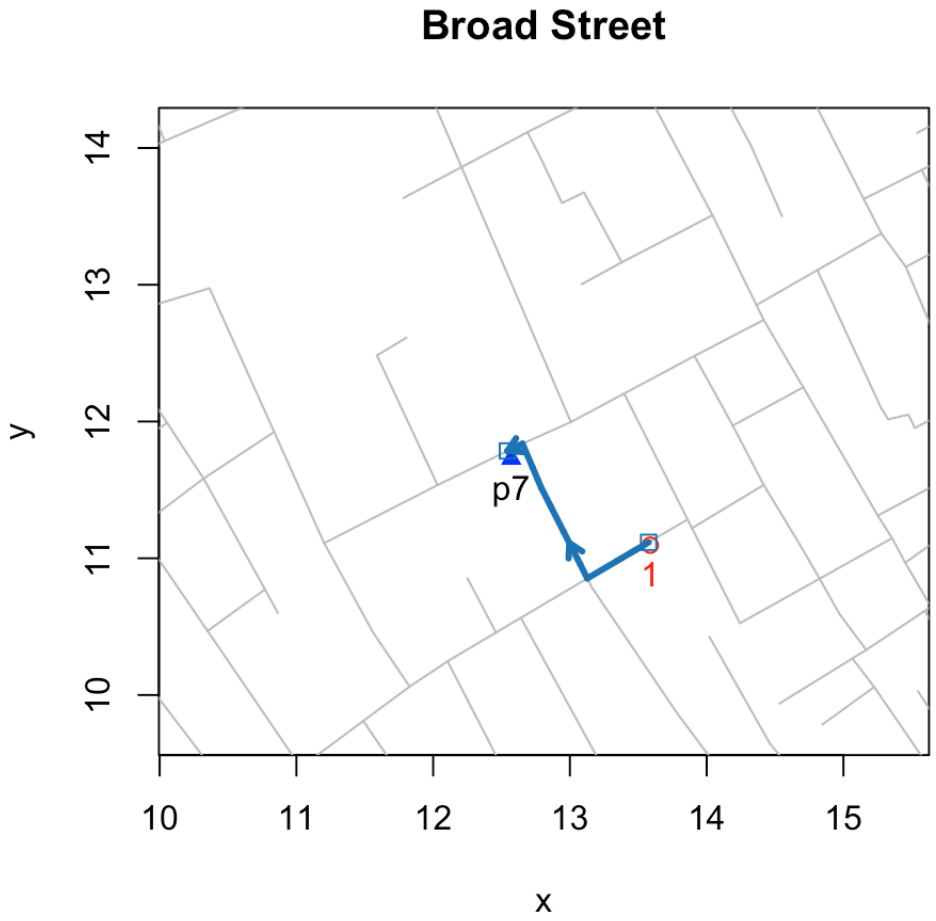
paths -> neighborhoods

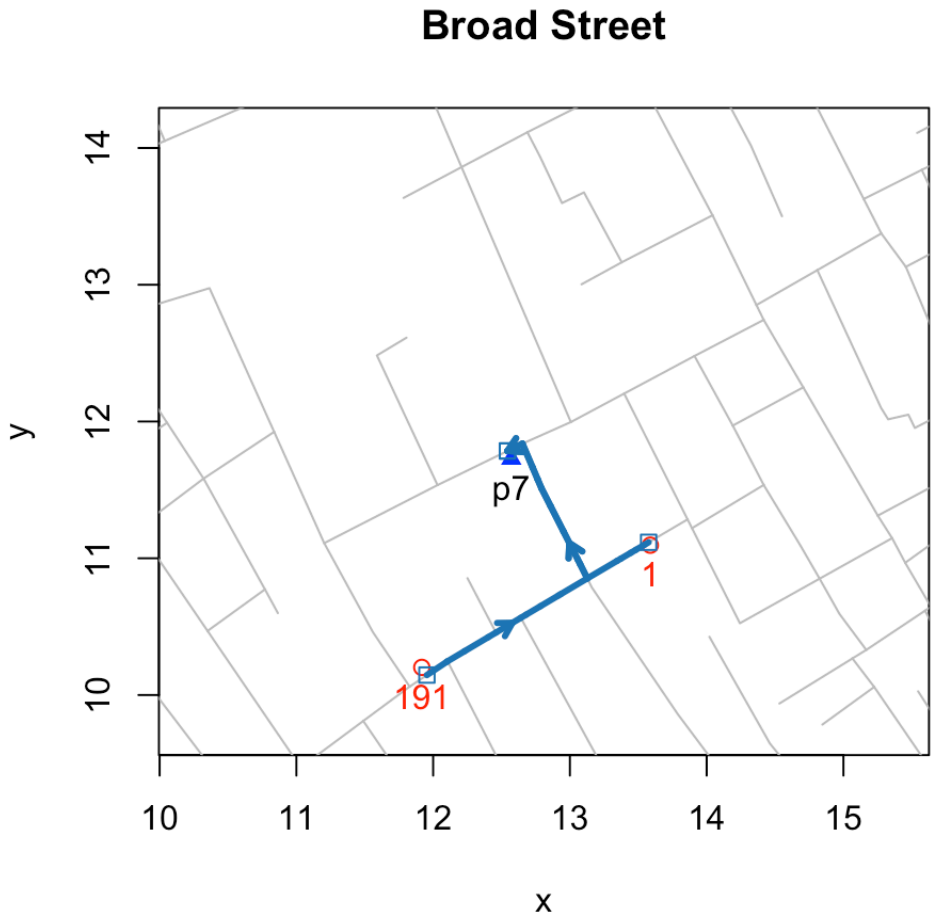
'stats', 'igraph', 'sp' and 'parallel'

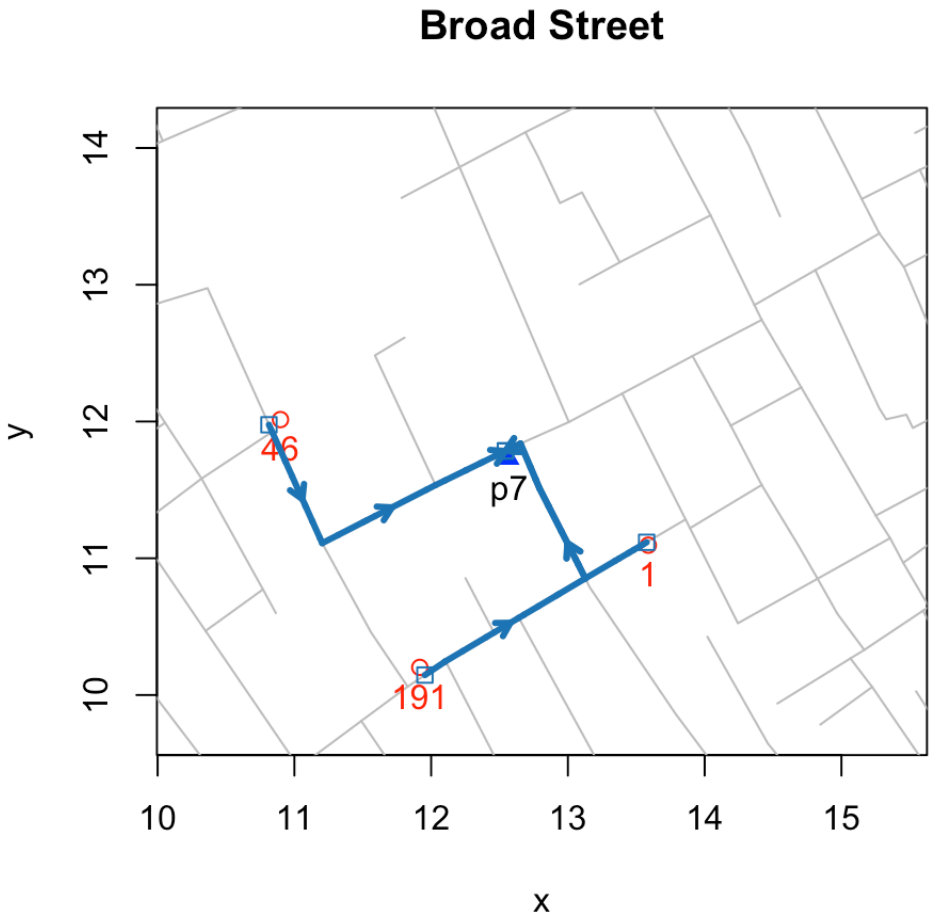
walking neighborhoods

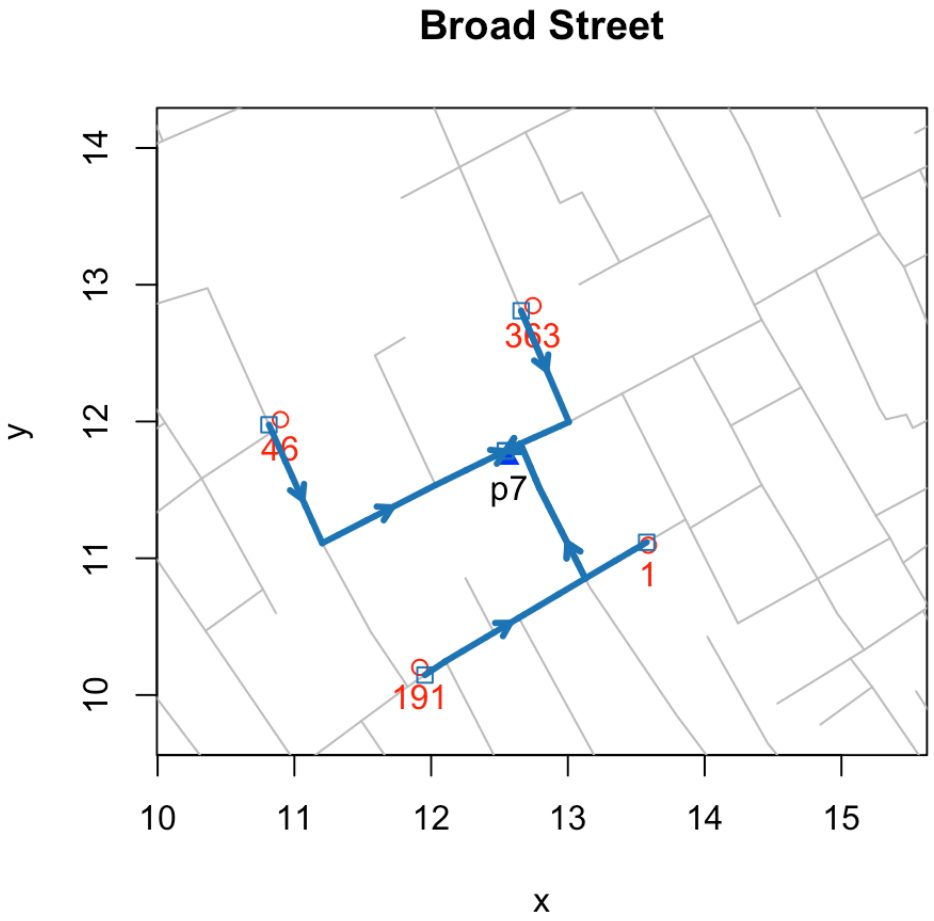
tasks: walking neighborhoods

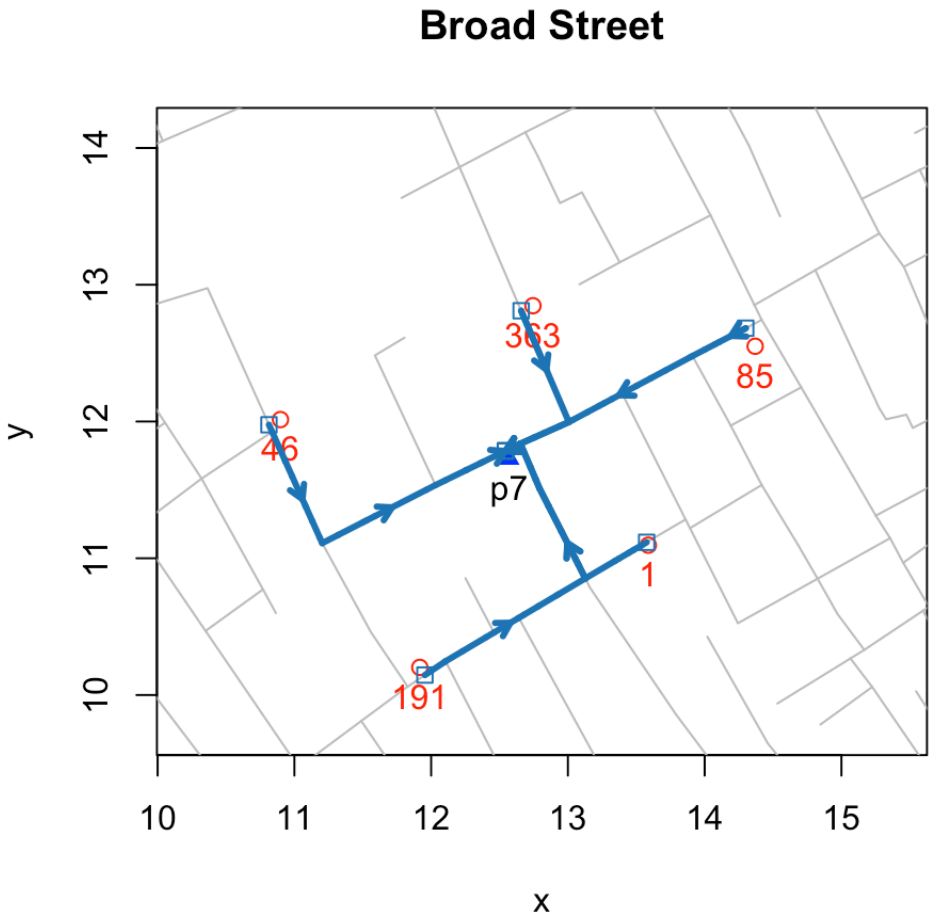
- compute and plot the path between case and pump …
- do so along roads and streets of on the map …
- rinse and repeat -> pump neighborhood

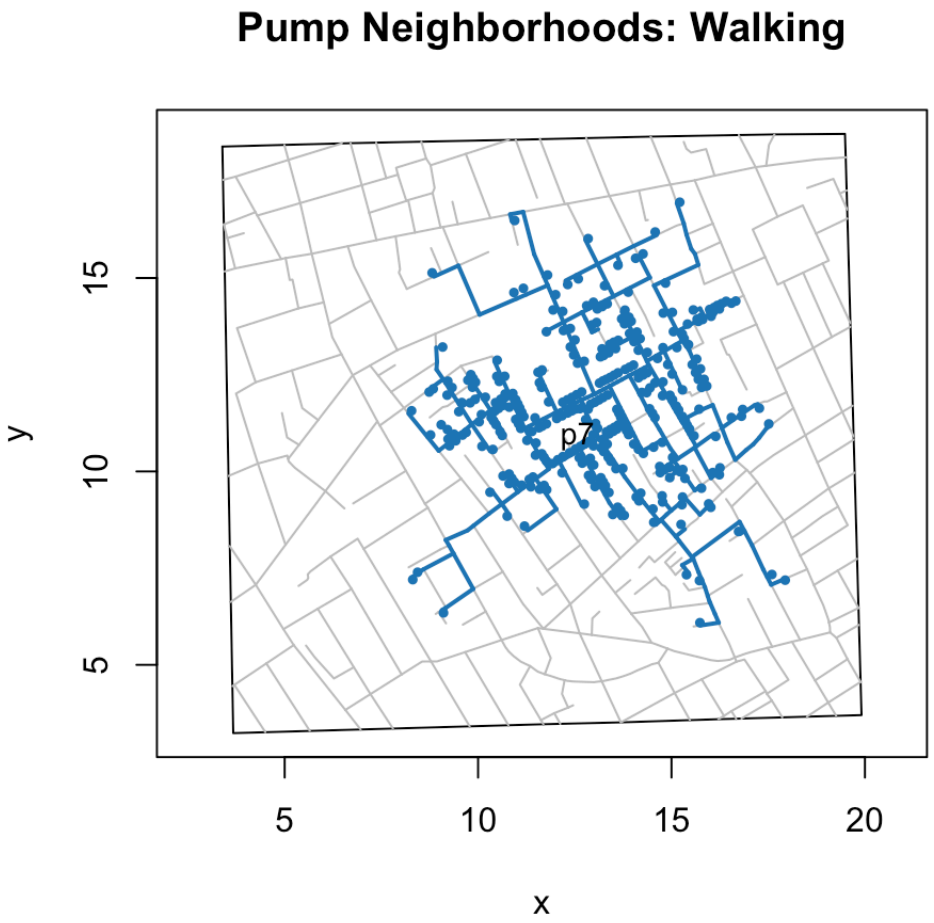

Broad Street
y
x
10
11
12
13
14
15
p7
1


Broad Street
y
x
10
11
12
13
14
15
p7
1
191


Broad Street
y
x
10
11
12
13
14
15
46
p7
1
191


Broad Street
y
x
14
13
12
11
10
10
11
12
13
14
15
363
46
p7
1
191


Broad Street
y
x
14
13
12
11
10
10
11
12
13
14
15
363
85
46
p7
1
191


Pump Neighborhoods: Walking
p7
y
x
5
10
15
20
5
10
15

orthogonal projection

hierarchical cluster analysis

address

Data

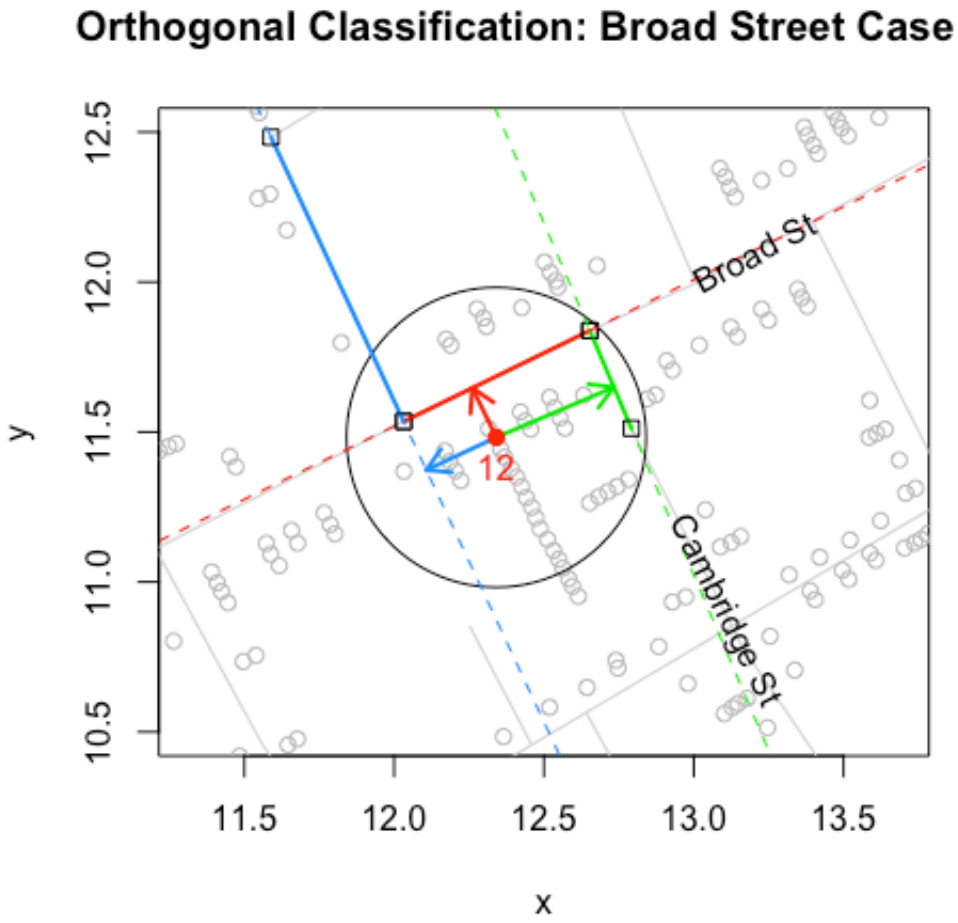
- Dodson and Tobler (1992)
  - GIS exercise
- bars (cases) and pumps are points
  - x-y coordinate
- streets are approximated by straight line segments
  - x-y coordinates of segment endpoints

lost in translation

- relationship between case and road ("address")
- relationship between cases in stack

address: horizontal bars

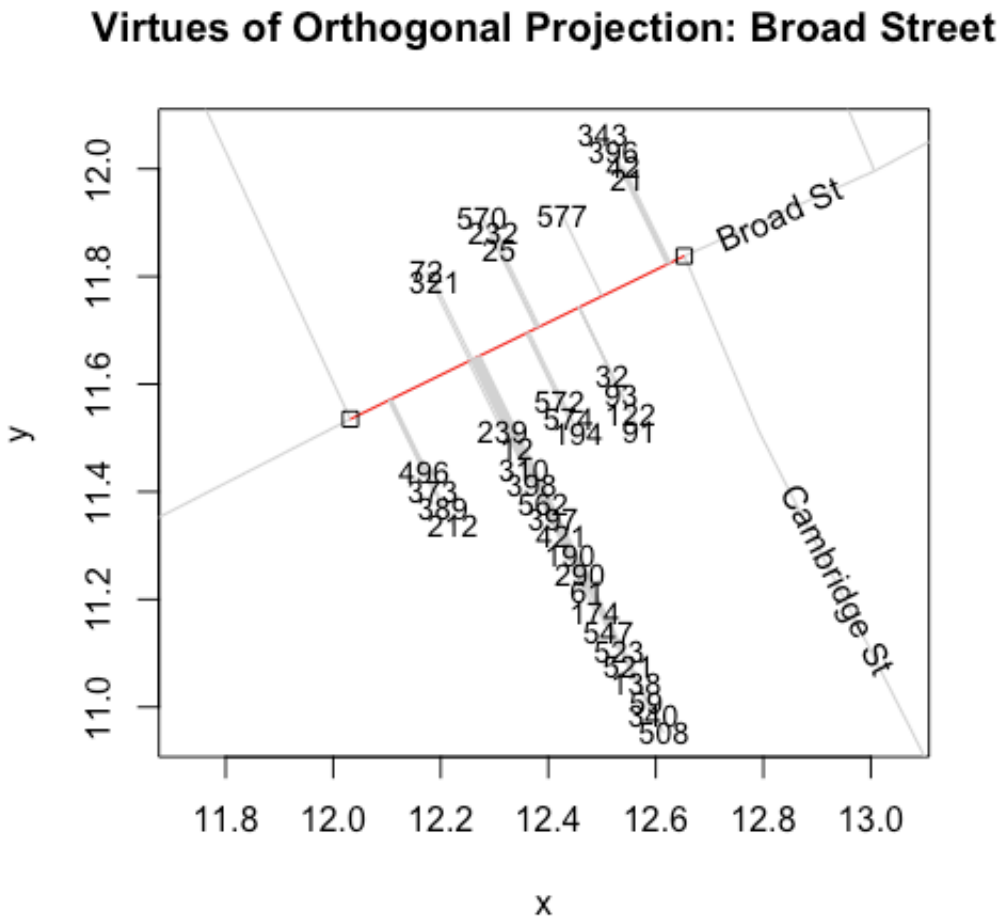
orthogonal projection & classification



orthogonal projection

```
orthogonal.slope <- -1 / segment.slope

orthogonal.intercept <- case$y - orthogonal.slope * case$x

x.proj <- (orthogonal.intercept - segment.intercept) /
          (segment.slope - orthogonal.slope)

y.proj <- segment.slope * x.proj + segment.intercept
```

"unstacking" bars

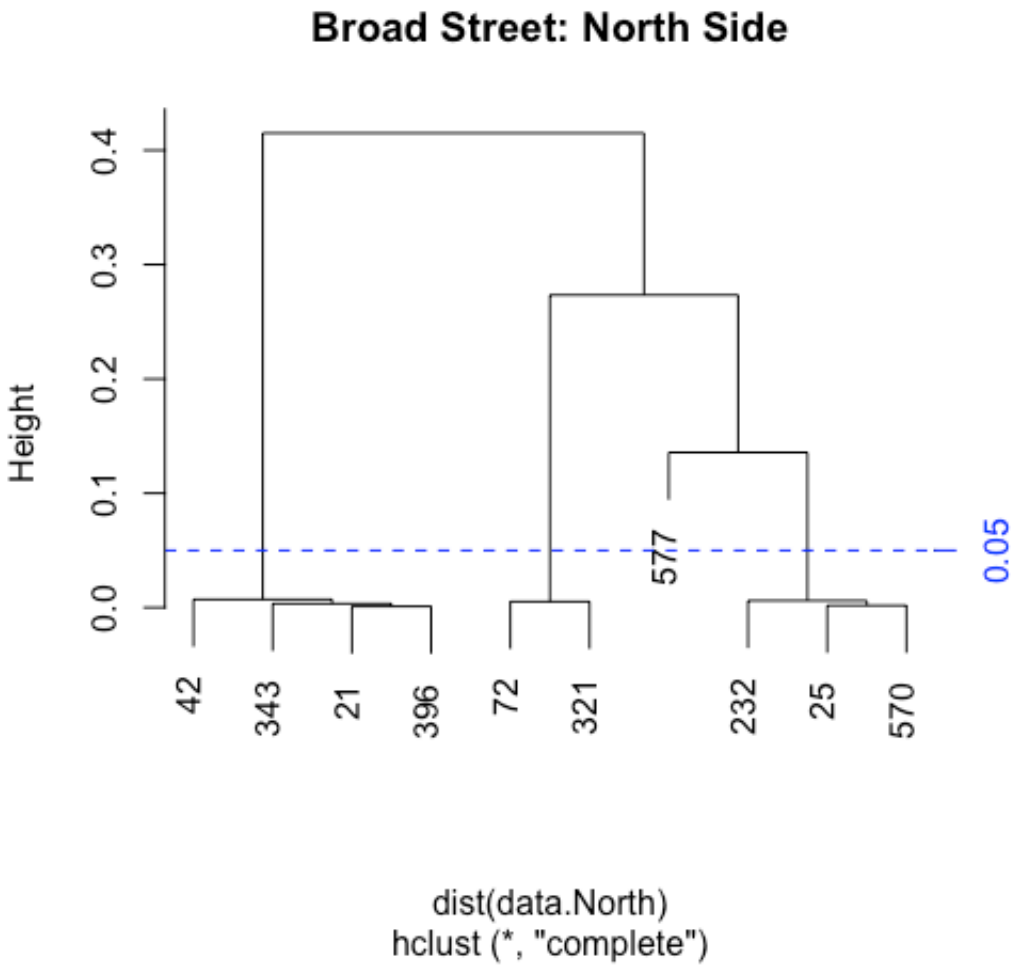


hierarchical cluster analysis via 'stats'

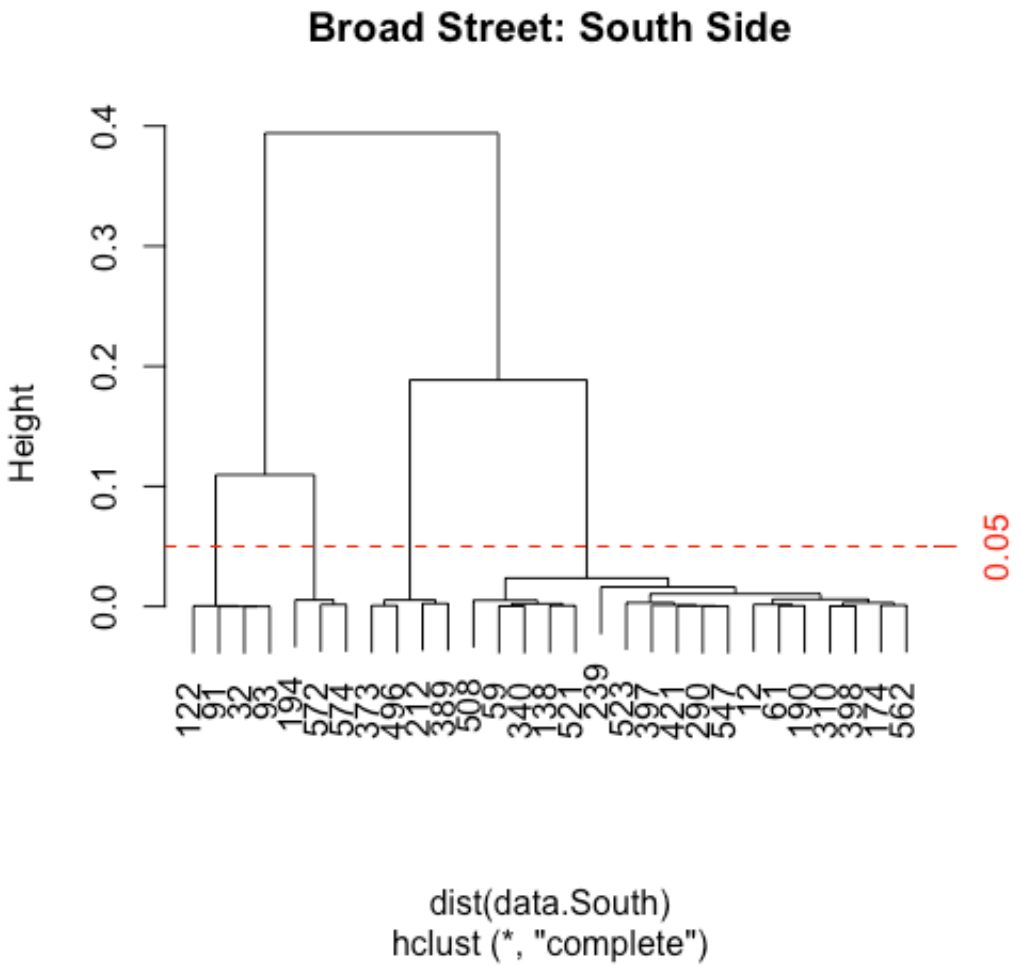
- label/group cases as a stack

```
stats::hclust()
stats::cutree()
```

North side


Broad Street: North Side
Height
0.0
0.1
0.2
0.3
0.4
0.05
577
42
343
21
396
72
321
232
25
570
dist(data.North)
hclust (*, "complete")

South side


Broad Street: South Side
Height
0.0
0.1
0.2
0.3
0.4
0.05
122
91
32
93
194
572
574
373
496
215
389
508
59
340
138
521
239
523
397
421
290
547
12
61
190
310
398
174
562
dist(data.South)
hclust (*, "complete")

"network" -> network

road network

network of roads

- roads -> edges
- street intersections -> nodes (vertices)

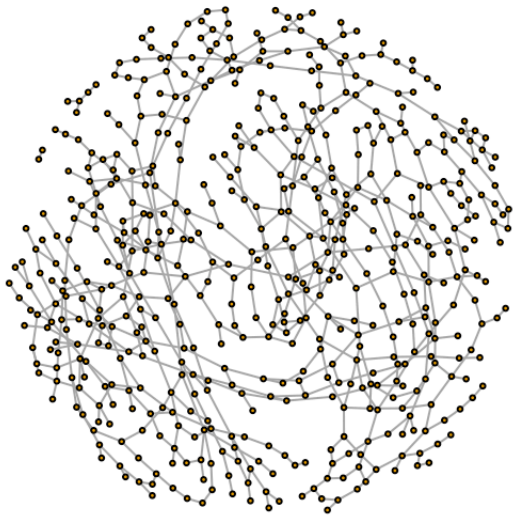
edge list (roads)

```
head(neighborhoodData(embed = FALSE)$edges)
```

```
##    street   id                name       x1       y1        x2       y2
## 26     13 13-1 Castle Street East 9.682715 18.20924 10.795917 18.56957
## 31     15 15-1       Market Place 8.657000 18.08900  8.533000 18.52400
## 33     16 16-1    Margaret Street 5.097822 18.05334  6.478946 18.47905
## 34     17 17-1      Regent Street 4.992000 18.44100  5.097822 18.05334
## 37     18 18-1 Castle Street East 8.772448 17.91459  9.682715 18.20924
## 38     19 19-1       Market Place 8.772448 17.91459  8.657000 18.08900
##                   node1                 node2   id2         d
## 26 9.6827154-18.2092381 10.7959166-18.5695744 13-1a 1.1700680
## 31 8.6569996-18.0890007      8.533-18.5240002 15-1a 0.4523278
## 33 5.0978222-18.0533352  6.4789462-18.4790497 16-1a 1.4452461
## 34     4.9920001-18.441  5.0978222-18.0533352 17-1a 0.4018486
## 37 8.7724485-17.9145889  9.6827154-18.2092381 18-1a 0.9567675
## 38 8.7724485-17.9145889  8.6569996-18.0890007 19-1a 0.2091600
```

road network

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



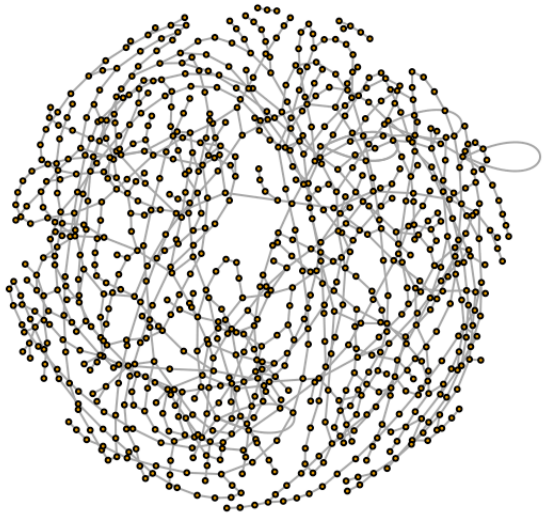
edge list (roads + addresses)

```
head(neighborhoodData(embed = TRUE)$edges)
```

```
##   street    id              name        x1       y1        x2       y2
## 1    116 116-2 Marlborough Mews  8.157000 14.71800  8.673940 14.95157
## 2    116 116-2 Marlborough Mews  8.673940 14.95157  8.857536 15.03452
## 3    116 116-2 Marlborough Mews  8.857536 15.03452  9.491089 15.32078
## 4    216 216-1     Broad Street 12.032000 11.53500 12.104121 11.57019
## 5    216 216-1     Broad Street 12.104121 11.57019 12.251776 11.64223
## 6    216 216-1     Broad Street 12.251776 11.64223 12.258823 11.64567
##                              node1                                node2
## 1              8.1569996-14.7180004 8.67393969607854-14.9515678671583
## 2 8.67393969607854-14.9515678671583 8.85753613025319-15.0345216841185
## 3 8.85753613025319-15.0345216841185              9.4910889-15.3207779
## 4             12.0319996-11.5349998 12.1041208314497-11.5701894274283
## 5 12.1041208314497-11.5701894274283  12.251775870062-11.6422337574022
## 6  12.251775870062-11.6422337574022 12.2588225788648-11.6456720106375
##      id2           d
## 1 116-2a 0.567257283
## 2 116-2b 0.201467085
## 3 116-2c 0.695220636
## 4 216-1a 0.080248252
## 5 216-1b 0.164293627
## 6 216-1c 0.007840771
```

road network + "addresses"

```
plot(neighborhoodData(embed = TRUE))
```



generic functions

S3 Object Oriented Programming

object

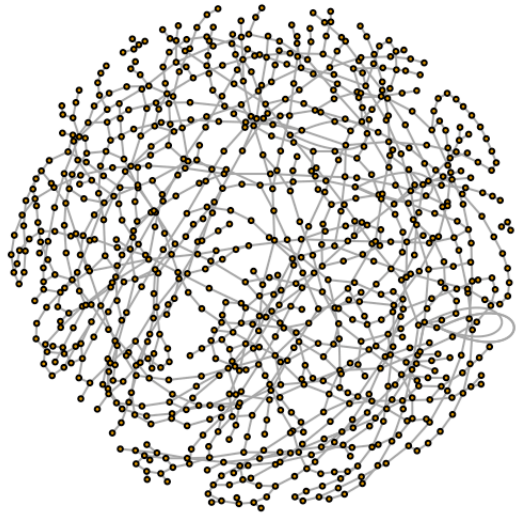
```
neighborhoodData <- function(vestry = FALSE, case.set = "observed",
  embed = TRUE, embed.landmarks = FALSE) {

  node.data <- nodeData(embed = embed, embed.landmarks = embed.landmarks)
  nodes <- node.data$nodes
  edges <- node.data$edges
  g <- node.data$g
  nodes.pump <- nodes[nodes$pump != 0, ]
  nodes.pump <- nodes.pump[order(nodes.pump$pump), c("pump", "node")]
  out <- list(g = g, nodes = nodes, edges = edges, nodes.pump = nodes.pump)
  class(out) <- "neighborhood_data"
  out
}
```

method

```
plot.neighborhood_data <- function(x, ...) {
  plot(x$g, vertex.label = NA, vertex.size = 2, ...)
}
```

road network + "addresses"

```
plot(neighborhoodData())
```



paths in network

tasks: network path

- find *nodal* path between case and pump
- many possible paths
- use the "shortest" path
- translate *nodal* path -> *walking* path

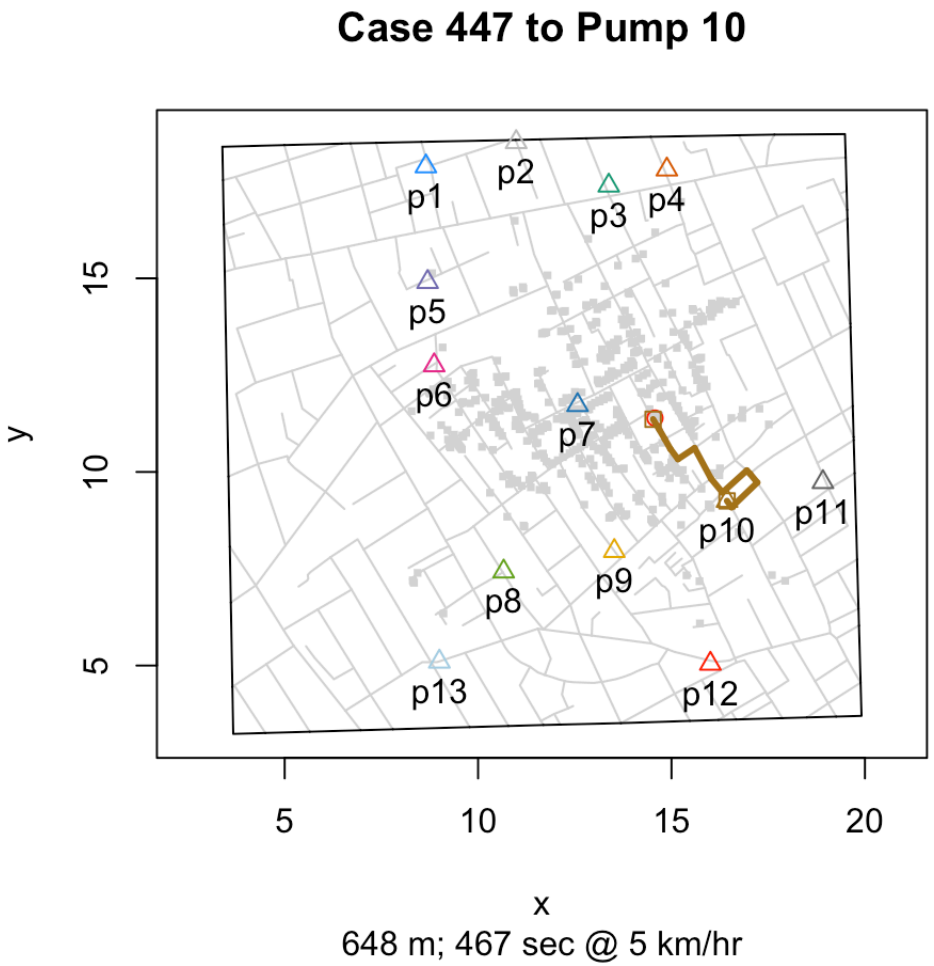
graph theory: shortest path

```
# breadth first algorithm
igraph::shortest_paths(g, ego.node, alter.node)
```

```
walkingPath(447, weighted = FALSE)$path
```

```
##  [1] "14.5236060046498-11.3601587685374"
##  [2] "14.6879997-11.0889997"
##  [3] "14.8257446-10.850667"
##  [4] "14.9709997-10.5930004"
##  [5] "15.1669998-10.3299999"
##  [6] "15.5965776-10.6166334"
##  [7] "16.0209999-9.8280001"
##  [8] "16.3169994-9.4689999"
##  [9] "16.9459991-10.0389996"
## [10] "17.0279999-9.9549999"
## [11] "17.2169991-9.7259998"
## [12] "16.552-9.0880003"
## [13] "16.4352026365887-9.25235128027873"
```

```
plot(walkingPath(447, weighted = FALSE), zoom = FALSE, unit.posts = NULL)
```



graph theory: shortest *weighted* path

```
# Dijkstra algorithm
igraph::shortest_paths(g, ego.node, alter.node, weights = edges$d)
```

edge list

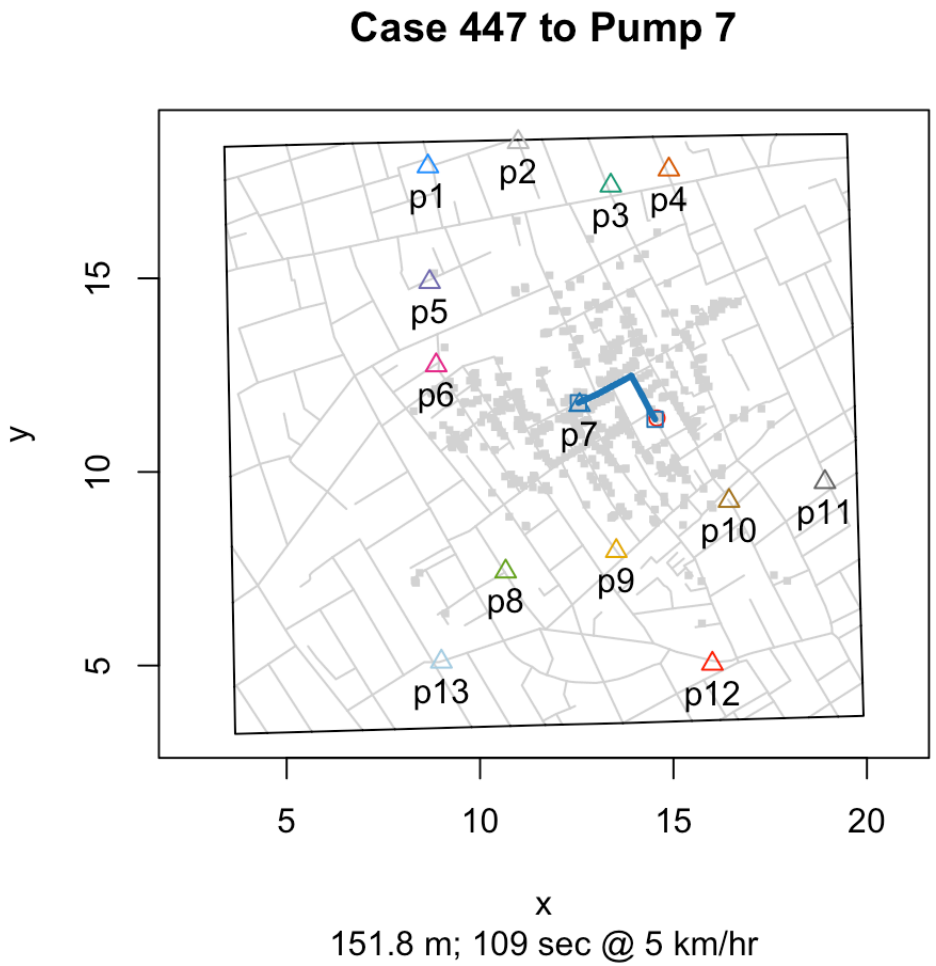
```
head(neighborhoodData(embed = TRUE)$edges)
```

```
##   street    id              name        x1       y1        x2       y2
## 1    116 116-2 Marlborough Mews  8.157000 14.71800  8.673940 14.95157
## 2    116 116-2 Marlborough Mews  8.673940 14.95157  8.857536 15.03452
## 3    116 116-2 Marlborough Mews  8.857536 15.03452  9.491089 15.32078
## 4    216 216-1     Broad Street 12.032000 11.53500 12.104121 11.57019
## 5    216 216-1     Broad Street 12.104121 11.57019 12.251776 11.64223
## 6    216 216-1     Broad Street 12.251776 11.64223 12.258823 11.64567
##                              node1                                node2
## 1              8.1569996-14.7180004 8.67393969607854-14.9515678671583
## 2 8.67393969607854-14.9515678671583 8.85753613025319-15.0345216841185
## 3 8.85753613025319-15.0345216841185              9.4910889-15.3207779
## 4             12.0319996-11.5349998 12.1041208314497-11.5701894274283
## 5 12.1041208314497-11.5701894274283  12.251775870062-11.6422337574022
## 6  12.251775870062-11.6422337574022 12.2588225788648-11.6456720106375
##      id2           d
## 1 116-2a 0.567257283
## 2 116-2b 0.201467085
## 3 116-2c 0.695220636
## 4 216-1a 0.080248252
## 5 216-1b 0.164293627
## 6 216-1c 0.007840771
```

```
walkingPath(447, weighted = TRUE)$path
```

```
##  [1] "14.5236060046498-11.3601587685374"
##  [2] "14.4573128612196-11.4695059486995"
##  [3] "14.4169998-11.5360003"
##  [4] "14.3858932445332-11.5957359505296"
##  [5] "14.191-11.9700003"
##  [6] "14.1648094983994-12.0176109227135"
##  [7] "13.9110003-12.4790001"
##  [8] "13.8955858579501-12.4707643362255"
##  [9] "13.7125927402452-12.3729931626151"
## [10] "13.6068669878267-12.3165050770277"
## [11] "13.5027696231832-12.2608870207937"
## [12] "13.4044965027844-12.208380793039"
## [13] "13.3966837-12.2042065"
## [14] "13.3205550078943-12.1635312916341"
## [15] "13.2630391506244-12.1328008349023"
## [16] "13.2282048211673-12.1141890136591"
## [17] "13.1420487887862-12.0681562437251"
## [18] "13.0367467208266-12.0118938354642"
## [19] "13.007-11.9960003"
## [20] "12.9399514368745-11.966074678584"
## [21] "12.8296233793203-11.9168322261283"
## [22] "12.7541203777477-11.8831331558499"
## [23] "12.6529999-11.8380003"
## [24] "12.6238122397928-11.8237589612493"
## [25] "12.5433800245817-11.7845142114077"
```

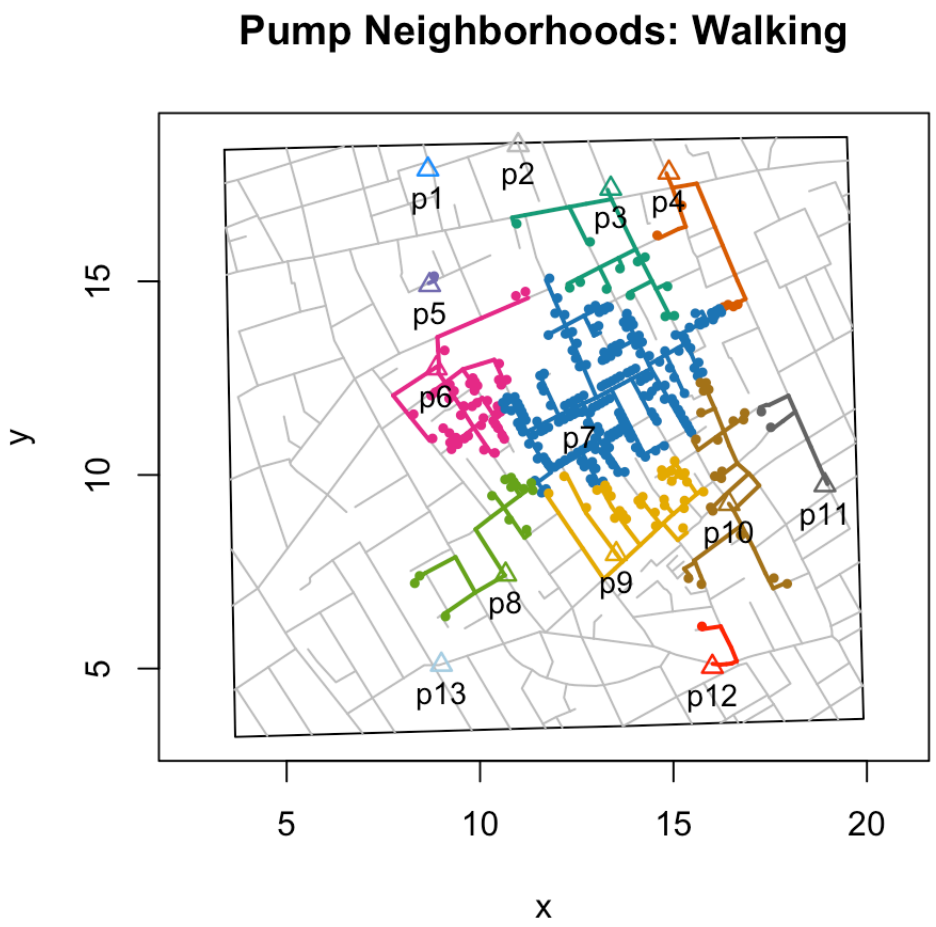
```
plot(walkingPath(447, weighted = TRUE), zoom = FALSE, unit.posts = NULL)
```



rinse & repeat

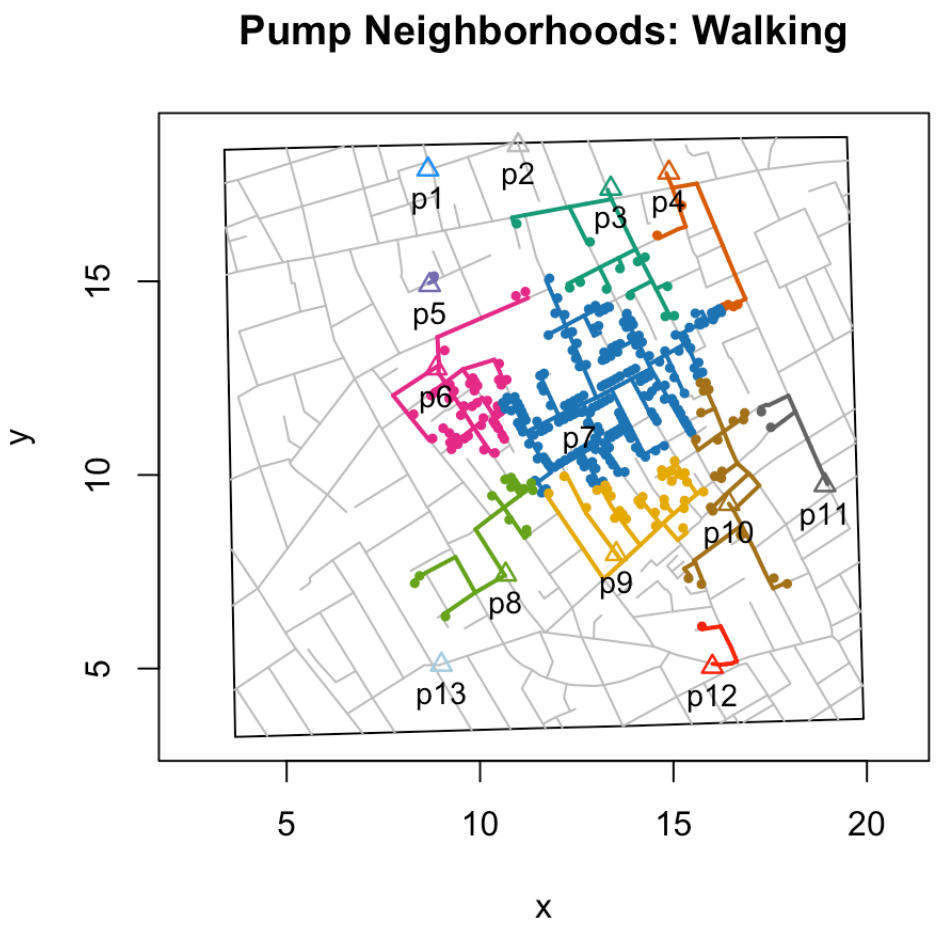
observed walking network

```
plot(neighborhoodWalking())
```

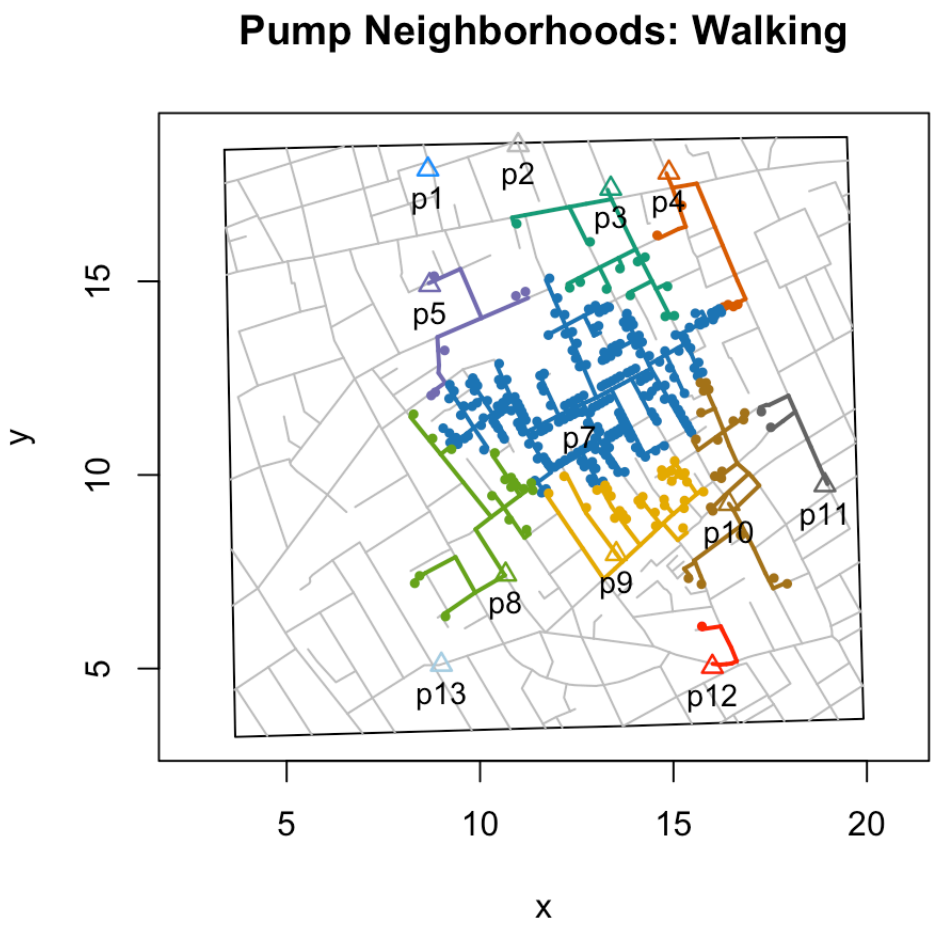


scenarios

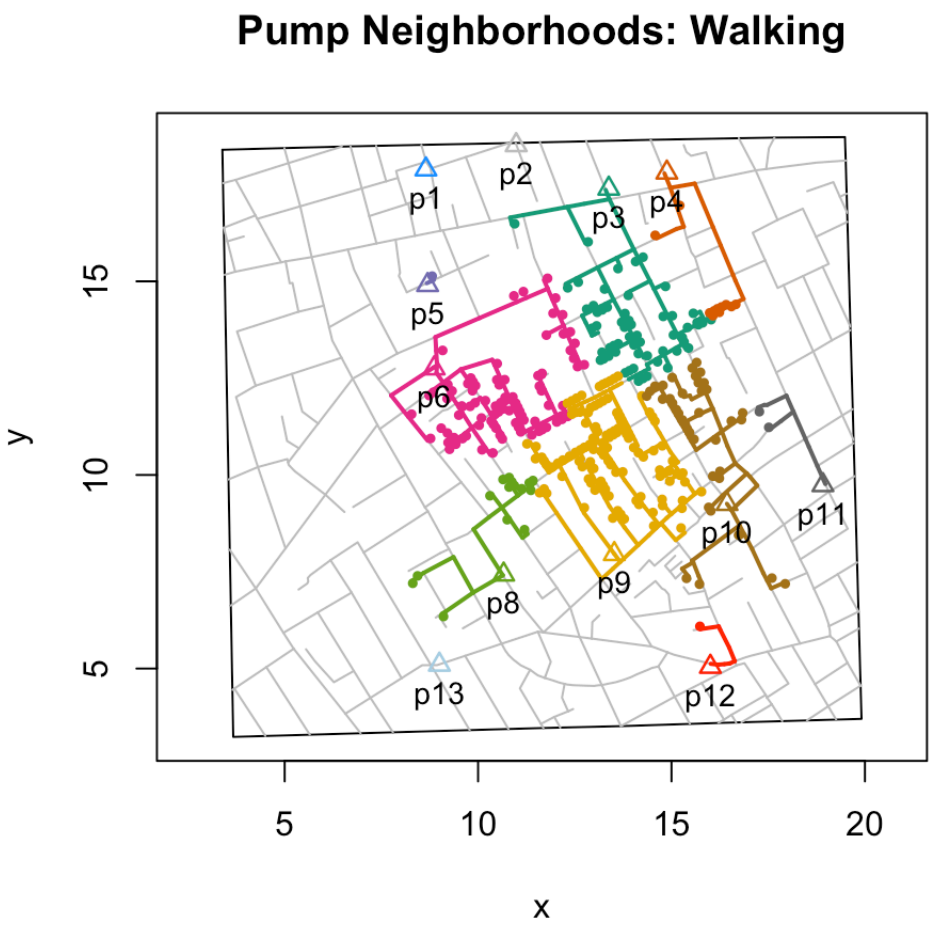
```
plot(neighborhoodWalking())
```



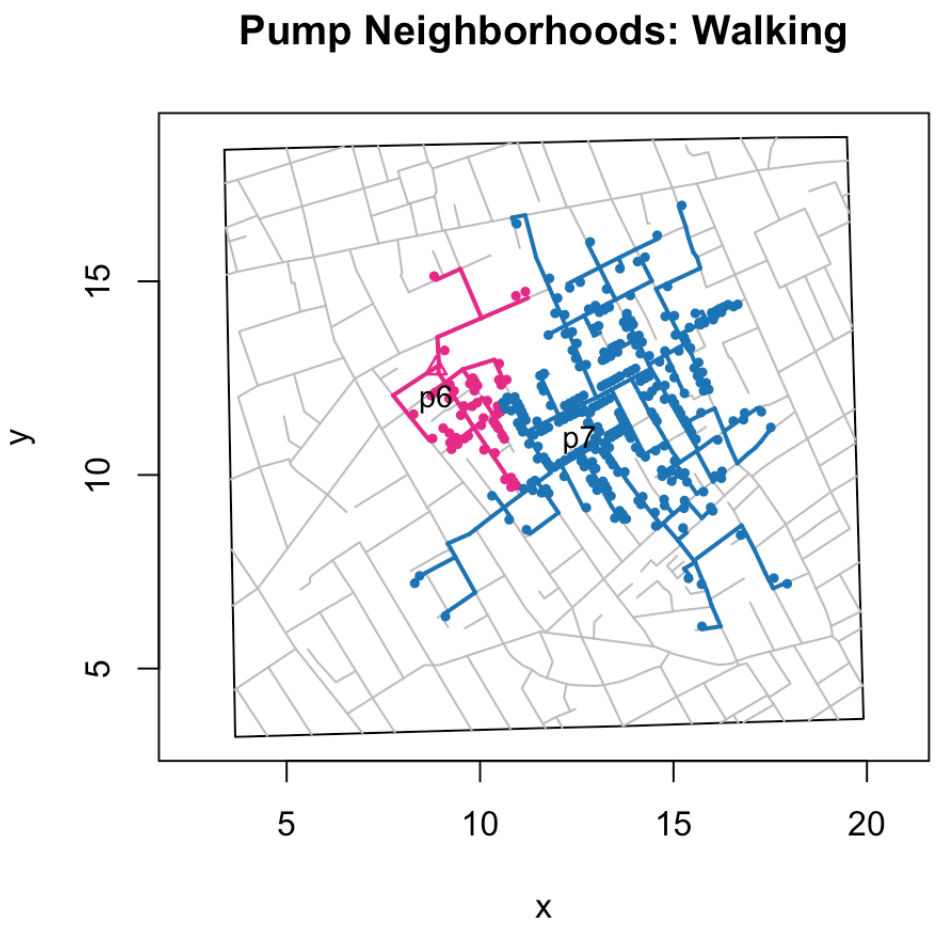
```
plot(neighborhoodWalking(-6))
```



```
plot(neighborhoodWalking(-7))
```



```
plot(neighborhoodWalking(6:7))
```



observed paths -> expected paths

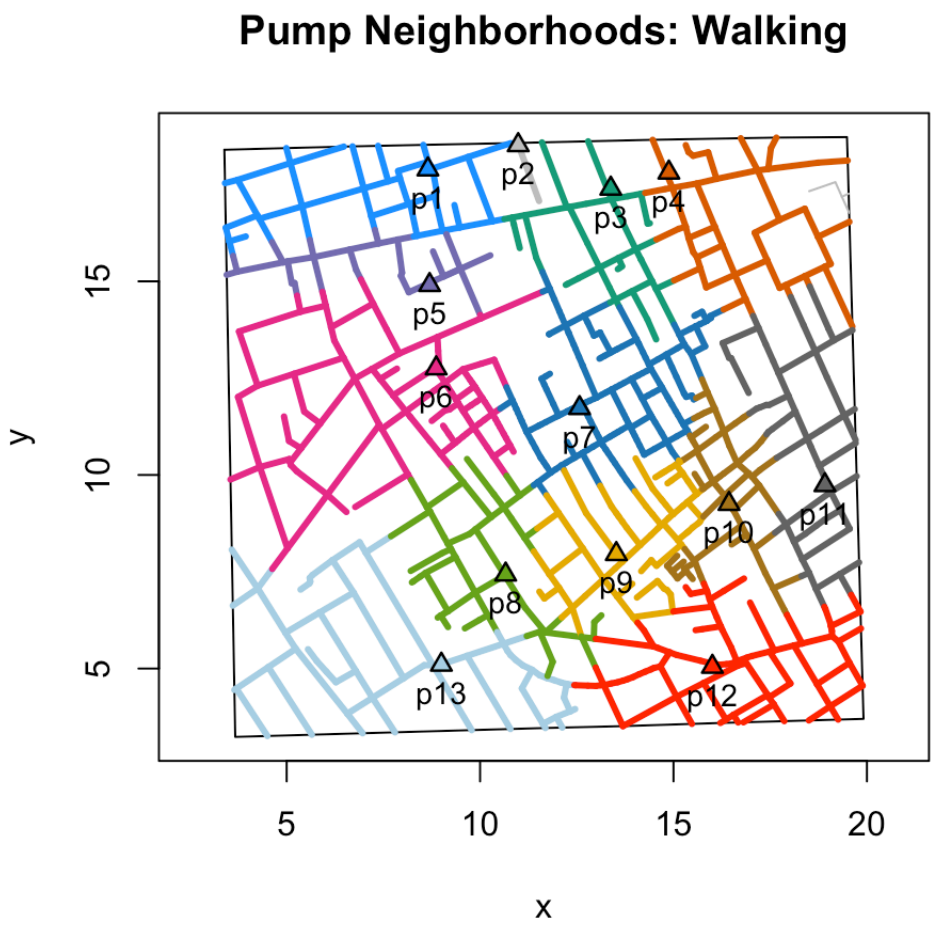
roads, points & polygons

expected walking neighborhoods

roads

- segment endpoints & …
- endpoints of split segments
- cut points via base::rle()

```
plot(neighborhoodWalking(case.set = "expected"), type = "road")
```

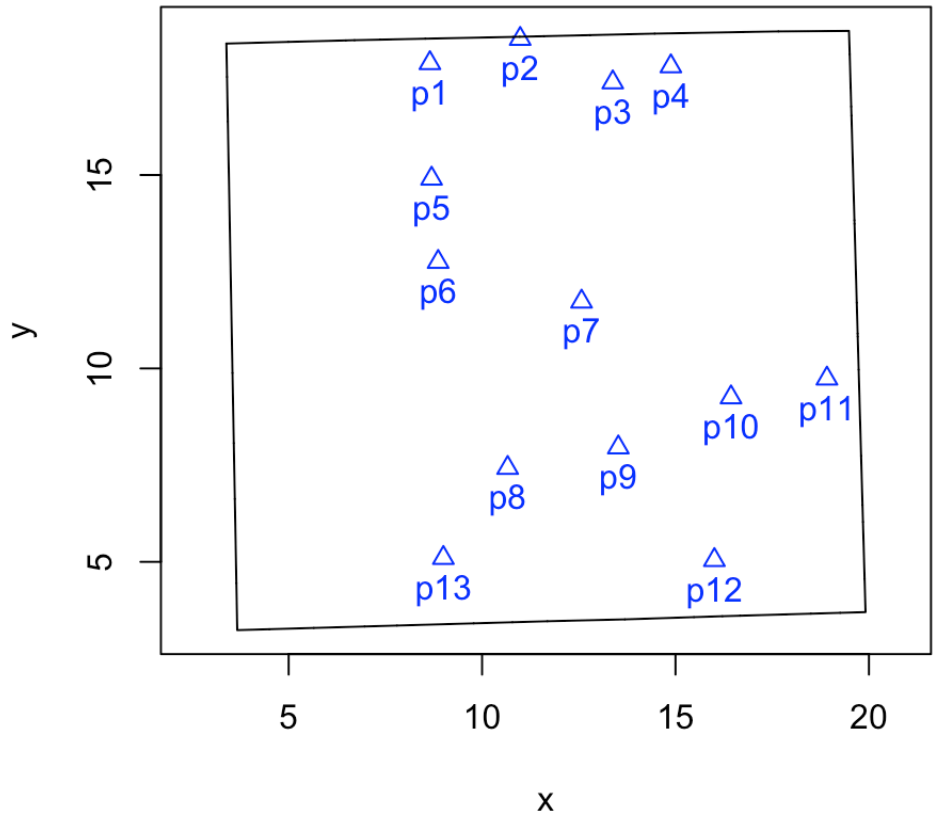


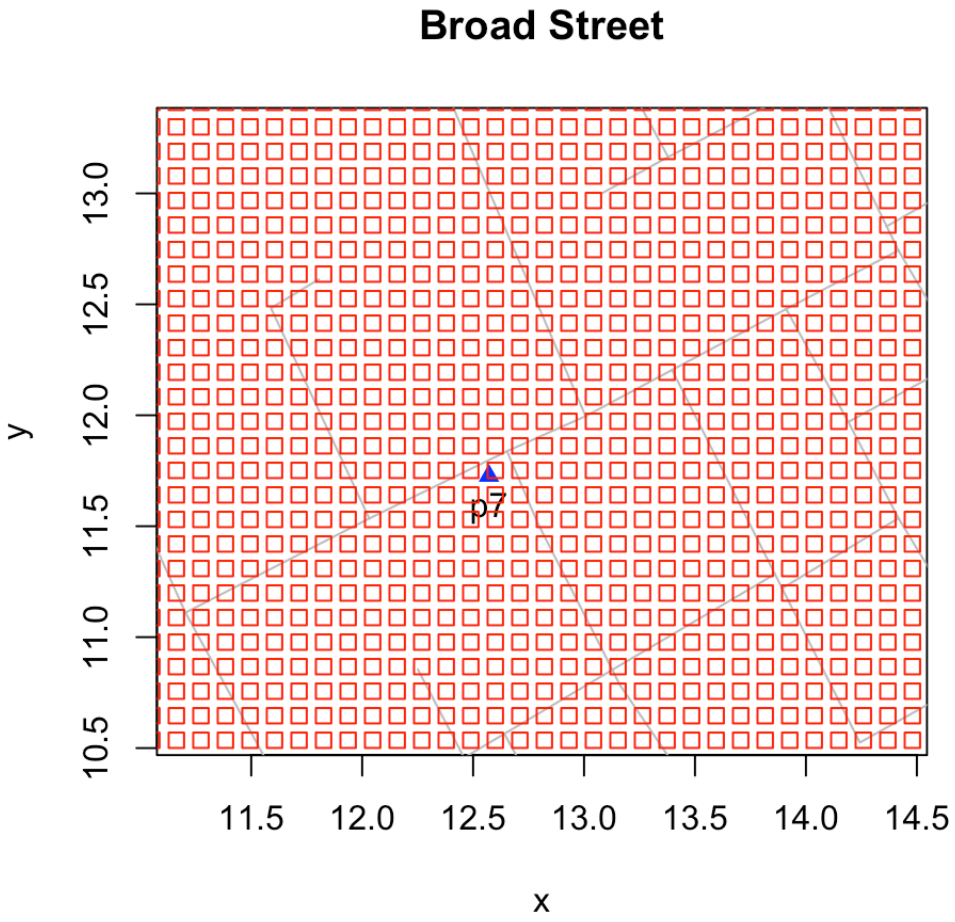
area points & area polygons

- observed paths -> expected areas
- 20K simulated cases (points)

simulated cases via 'sp'

```
sp::spsample(sp::Polygon(map.frame[, c("x", "y")]), n = 20000,
  type = "regular")
```


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


Broad Street
y
x
10.5
11.0
11.5
12.0
12.5
13.0
11.5
12.0
12.5
13.0
13.5
14.0
14.5
pump

'parallel' package

embarrassingly parallel problems

lapply() -> mclapply()

```
lapply(list, function)
```

```
mclapply(list, function)
```

roads data

```
head(roads[roads$name != "Map Frame", ])
```

```
##    street n         x        y id               name
## 26     13 2  9.682715 18.20924 25 Castle Street East
## 25     13 2 10.795917 18.56957 26 Castle Street East
## 31     15 2  8.657000 18.08900 30       Market Place
## 30     15 2  8.533000 18.52400 31       Market Place
## 33     16 2  5.097822 18.05334 32    Margaret Street
## 32     16 2  6.478946 18.47905 33    Margaret Street
```

road.segments data

```
head(road.segments)
```

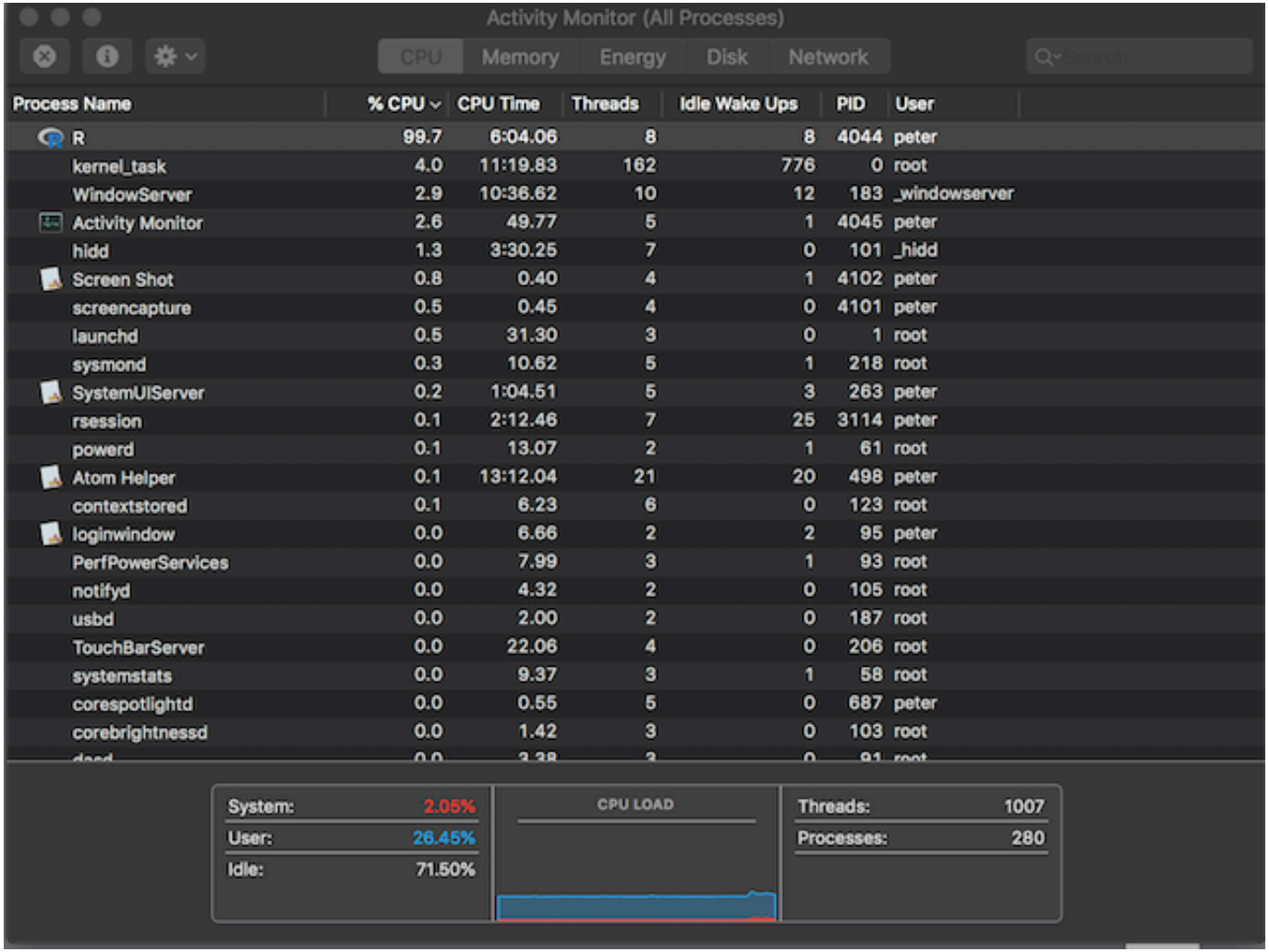
```
##    street   id                name       x1       y1        x2       y2
## 26     13 13-1 Castle Street East 9.682715 18.20924 10.795917 18.56957
## 31     15 15-1       Market Place 8.657000 18.08900  8.533000 18.52400
## 33     16 16-1    Margaret Street 5.097822 18.05334  6.478946 18.47905
## 34     17 17-1      Regent Street 4.992000 18.44100  5.097822 18.05334
## 37     18 18-1 Castle Street East 8.772448 17.91459  9.682715 18.20924
## 38     19 19-1       Market Place 8.772448 17.91459  8.657000 18.08900
```

```
road.segments <- parallel::mclapply(unique(rd$street), function(i) {
  dat <- rd[rd$street == i, ]
  names(dat)[names(dat) %in% c("x", "y")] <- c("x1", "y1")
  seg.data <- dat[-1, c("x1", "y1")]
  names(seg.data) <- c("x2", "y2")
  dat <- cbind(dat[-nrow(dat), ], seg.data)
  dat$id <- paste0(dat$street, "-", seq_len(nrow(dat)))
  dat
}, mc.cores = cores)

road.segments <- do.call(rbind, road.segments)
```

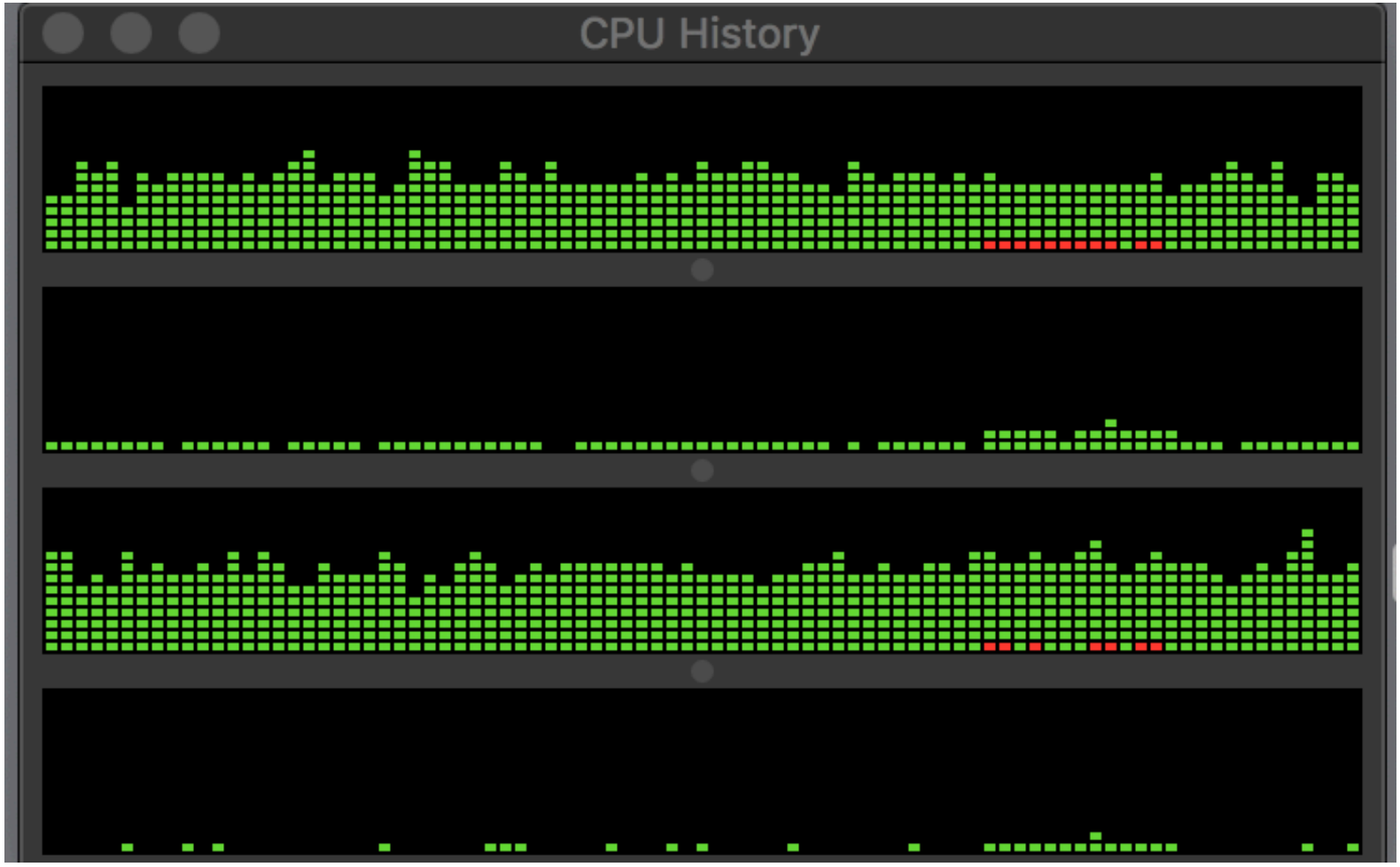
simulated cases: single thread

```
simulateFatalities(compute = TRUE, multi.core = FALSE)
```

single thread - Activity Monitor



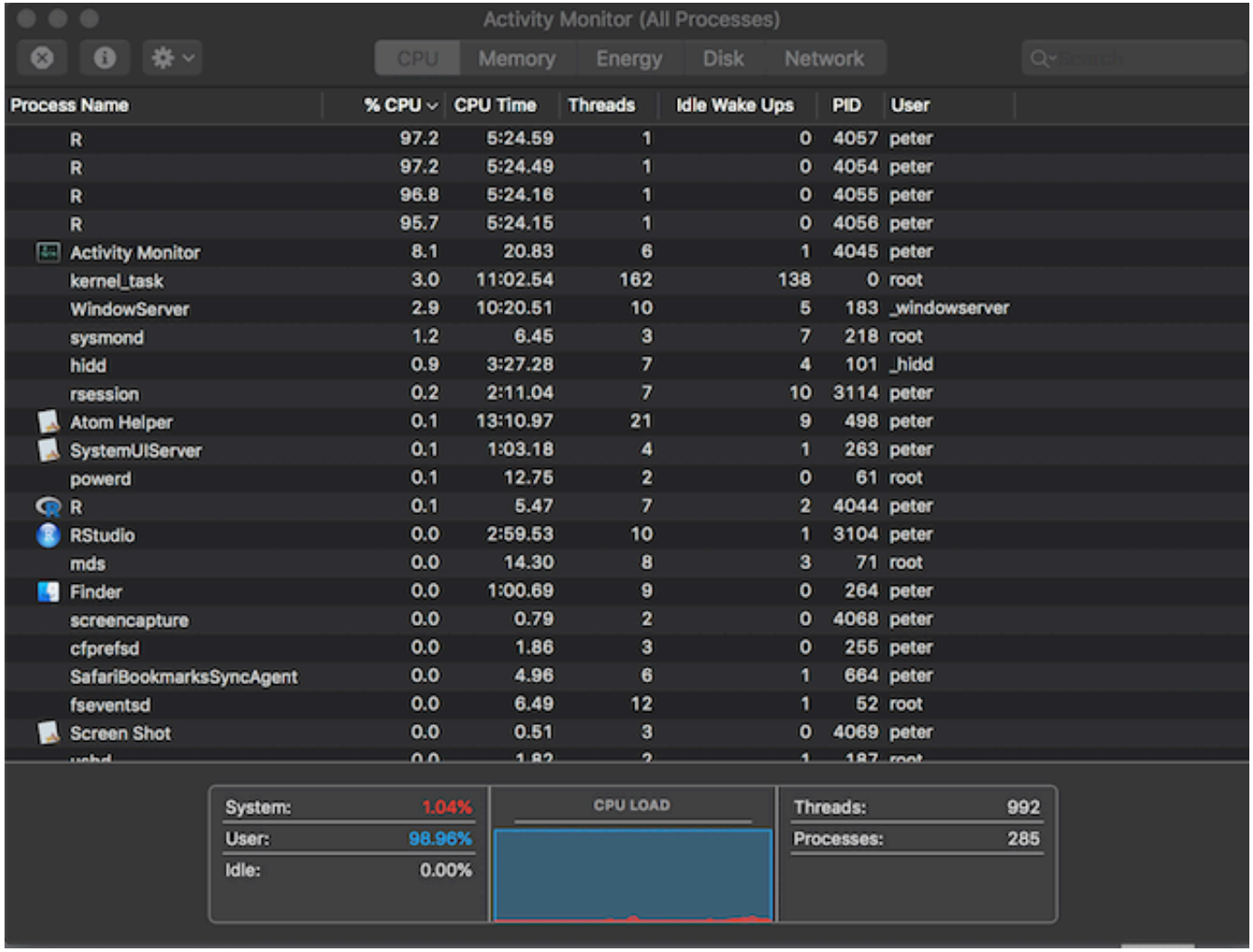
single thread - CPU History



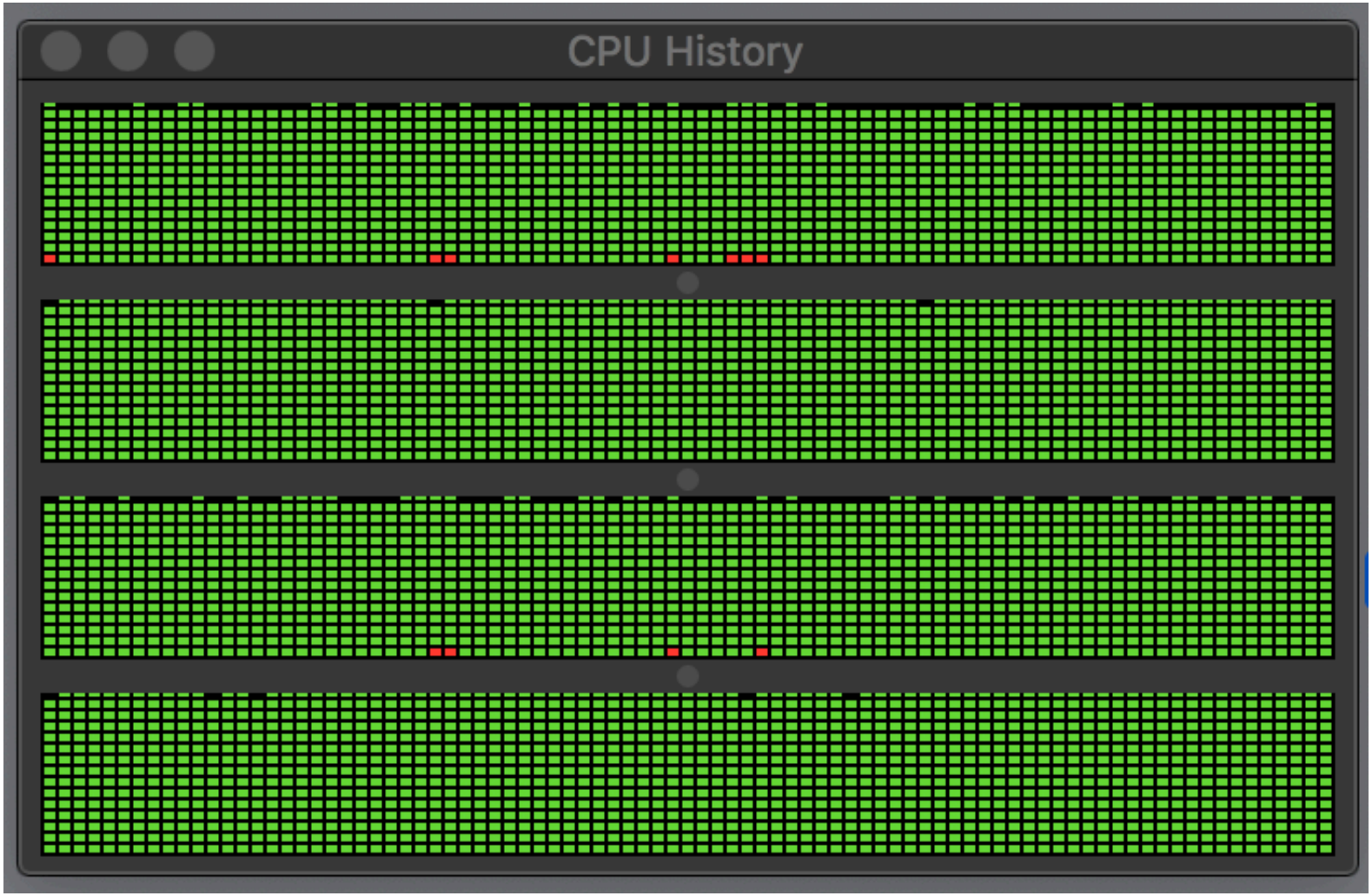
simulated cases: parallel threads

```
simulateFatalities(compute = TRUE, multi.core = TRUE)
```

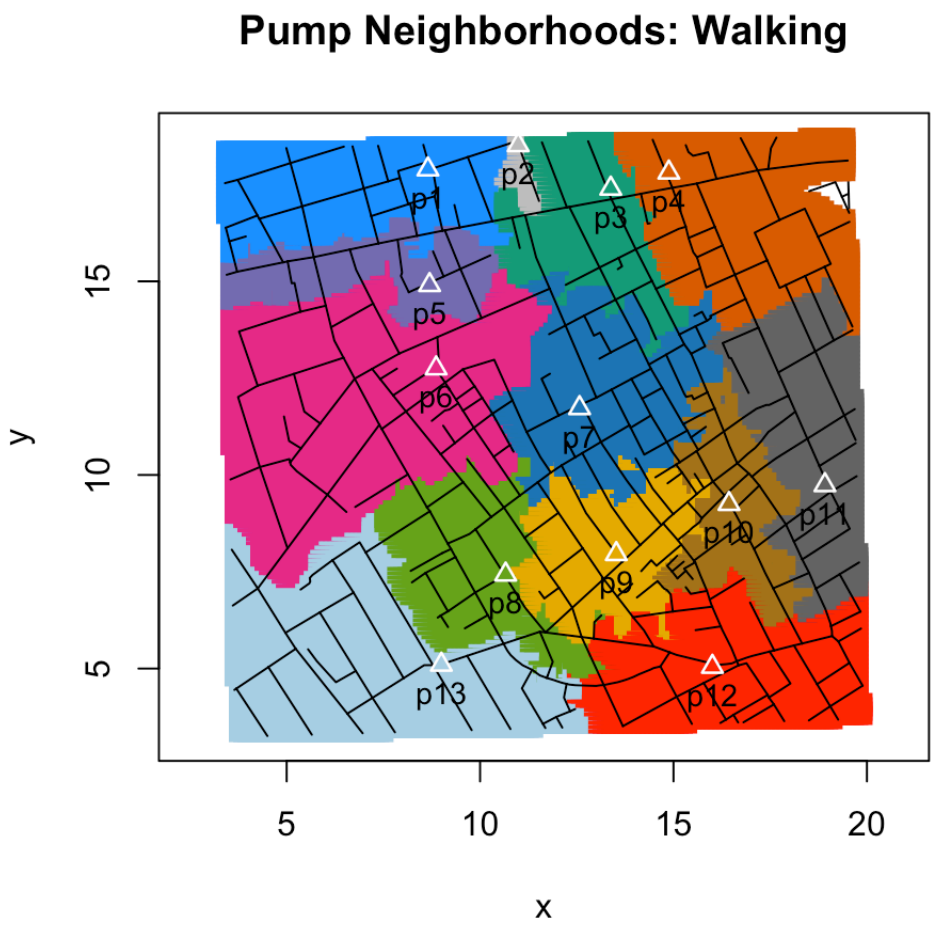
parallel threads - Activity Monitor



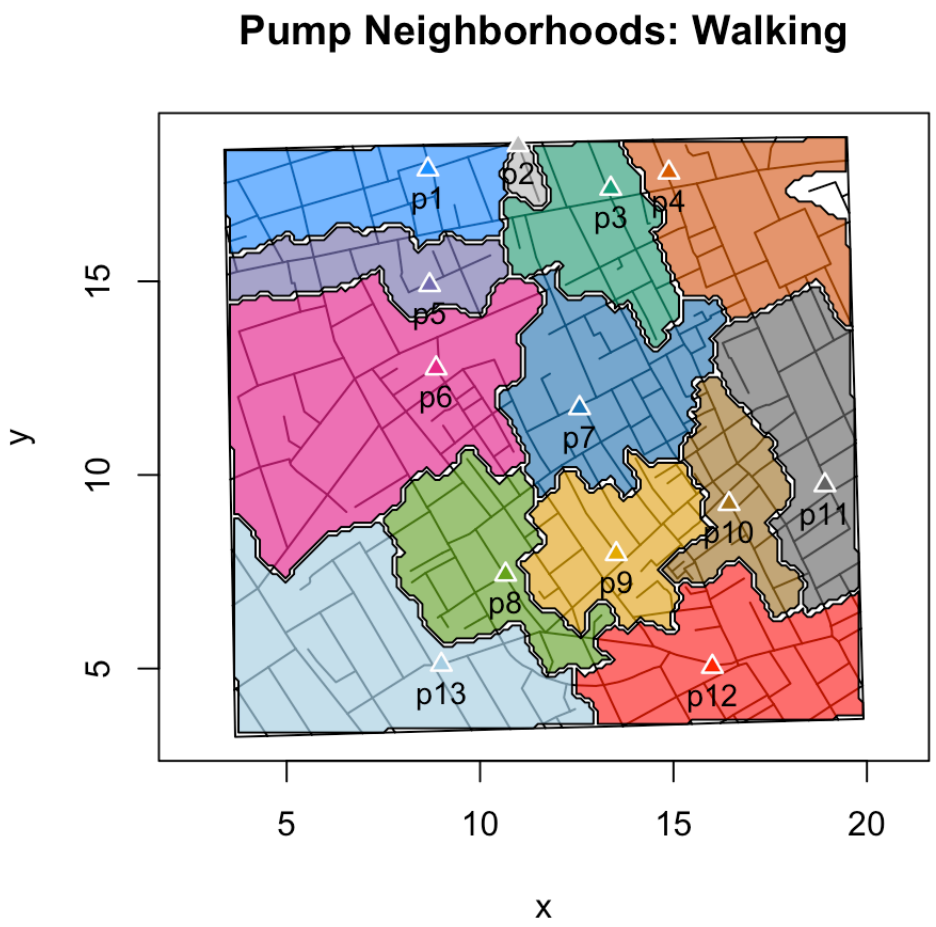
Parallel Threads - CPU History



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



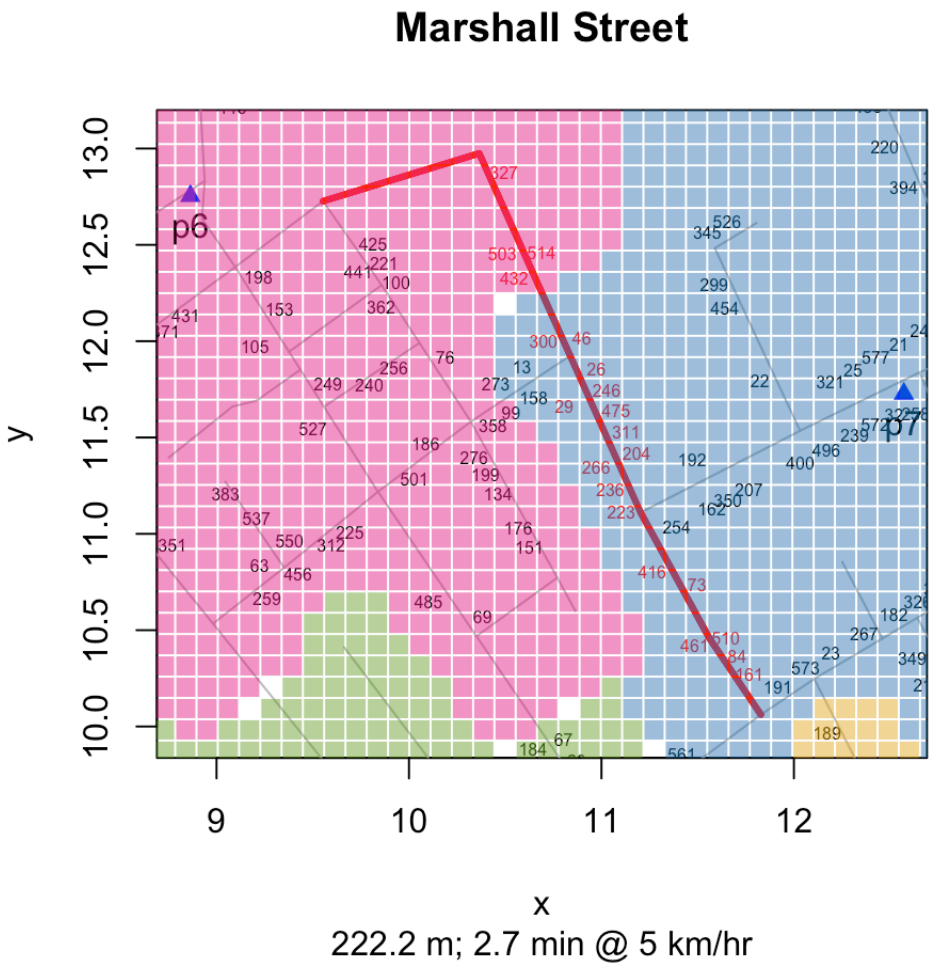
```
plot(neighborhoodWalking(case.set = "expected"), type = "area.polygons")
```



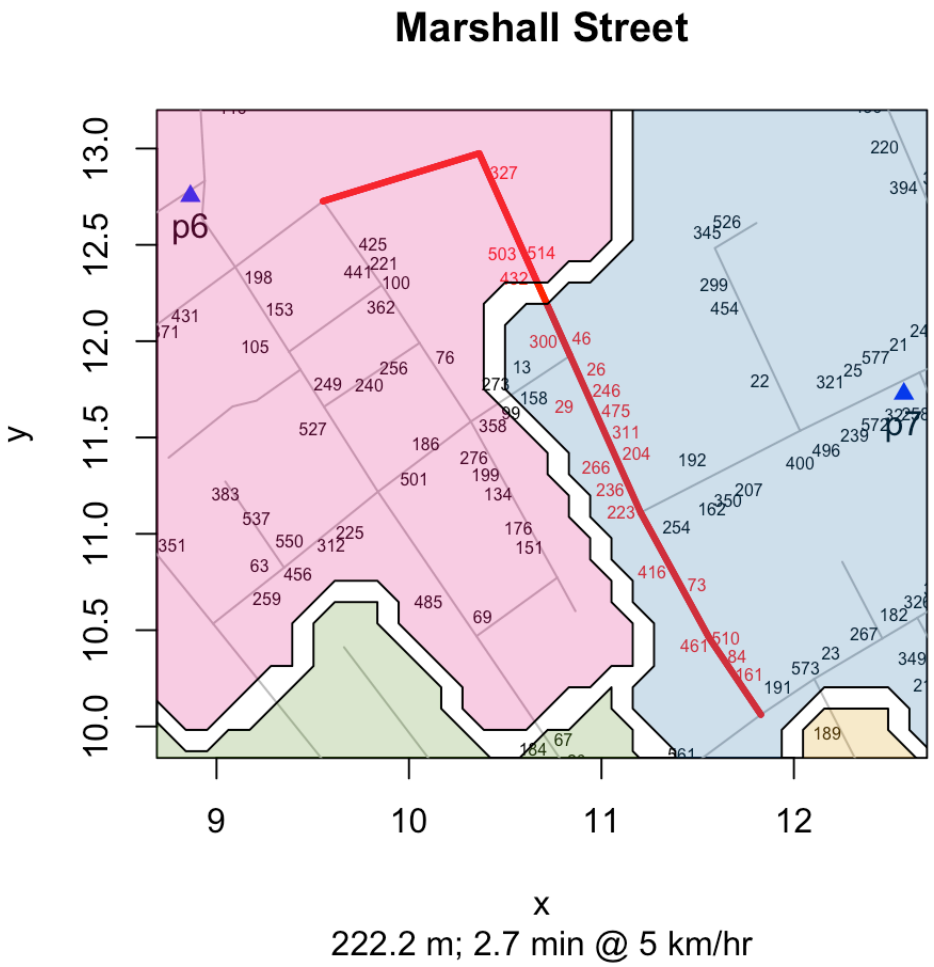
why polygons?

- vector graphics

```
streetNameLocator("marshall street", zoom = TRUE)
addNeighborhoodCases(type = "expected")
```

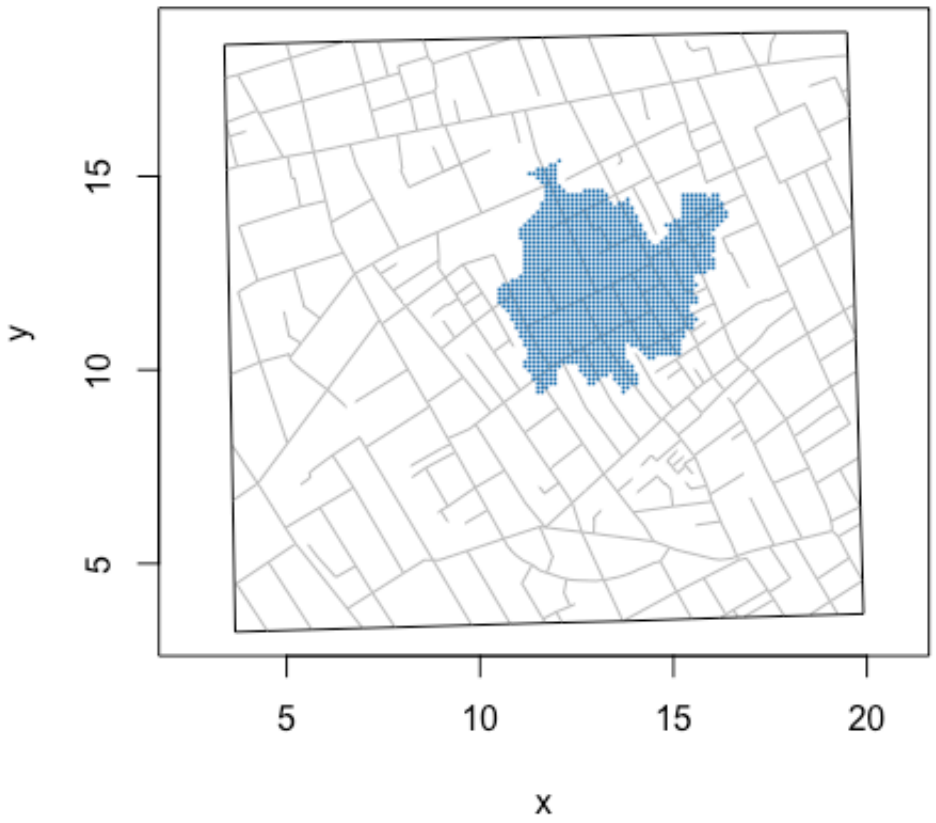


```
streetNameLocator("marshall street", zoom = TRUE)
addNeighborhoodWalking()
```

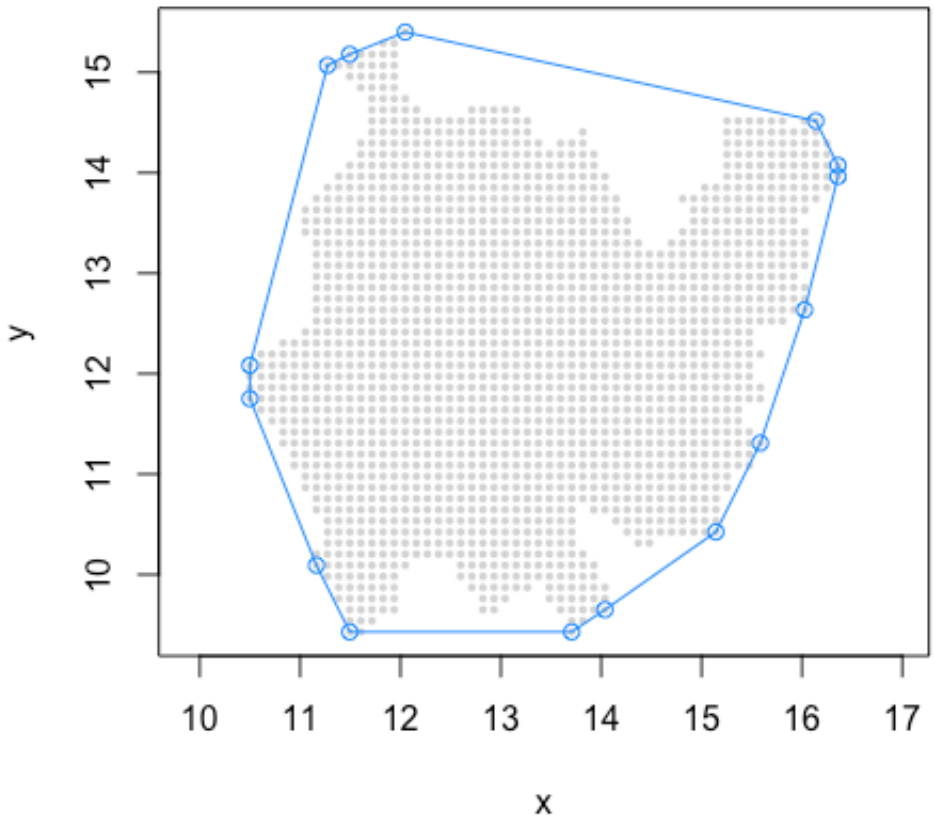


computing polygons

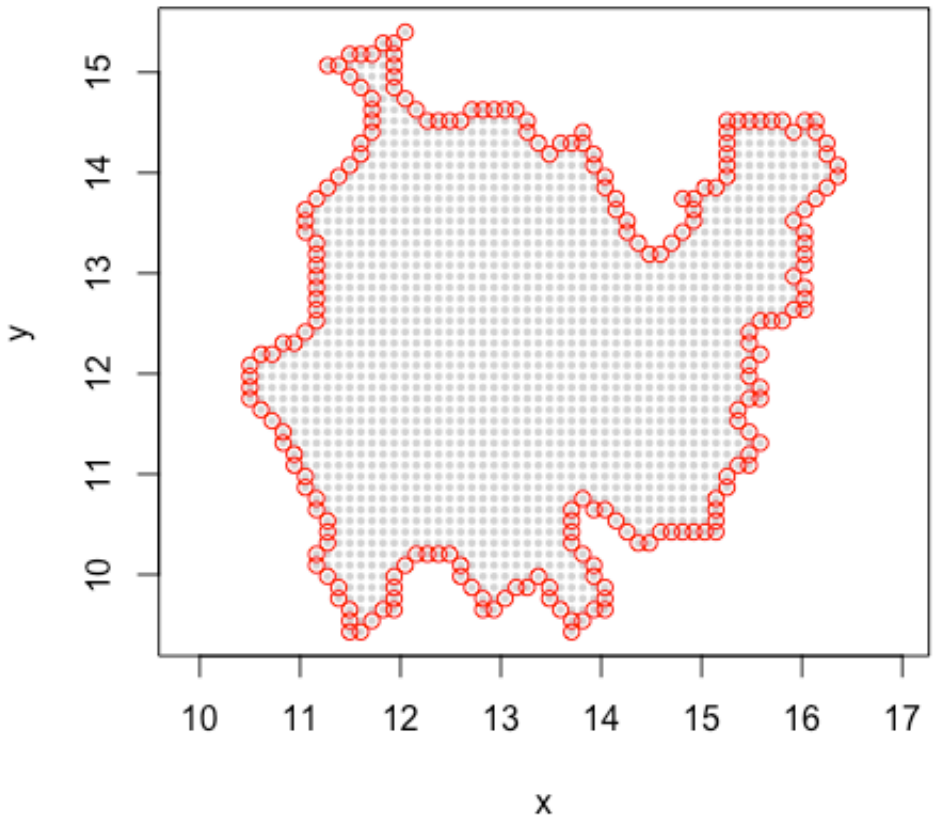
Broad Street simulated cases



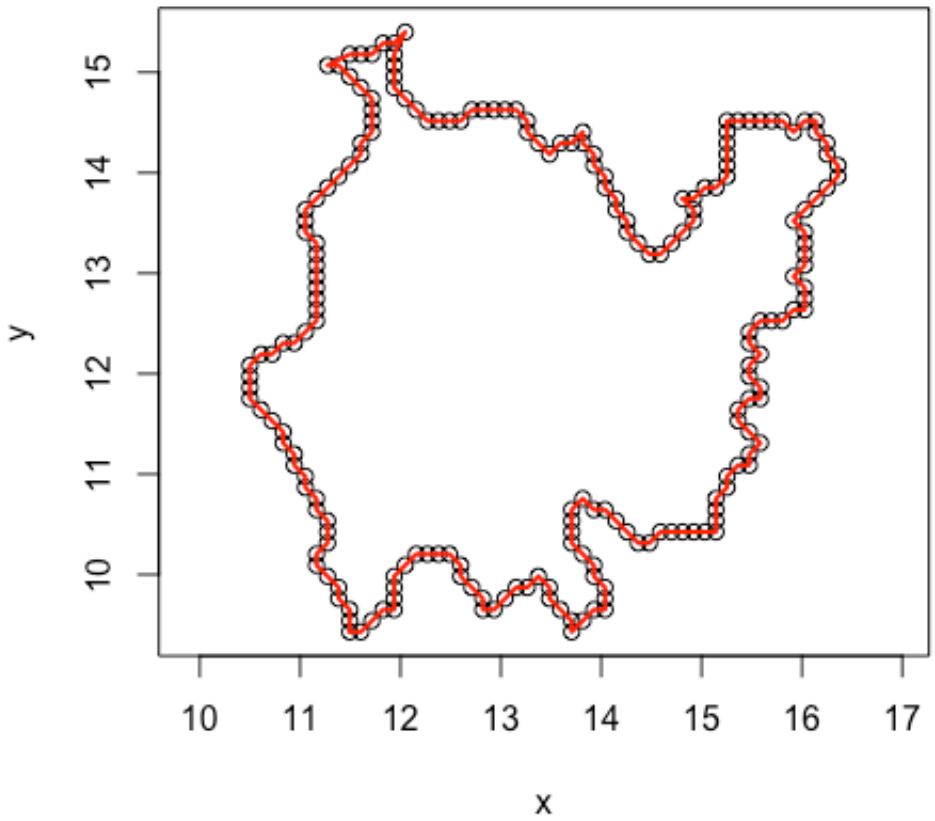
convex hull



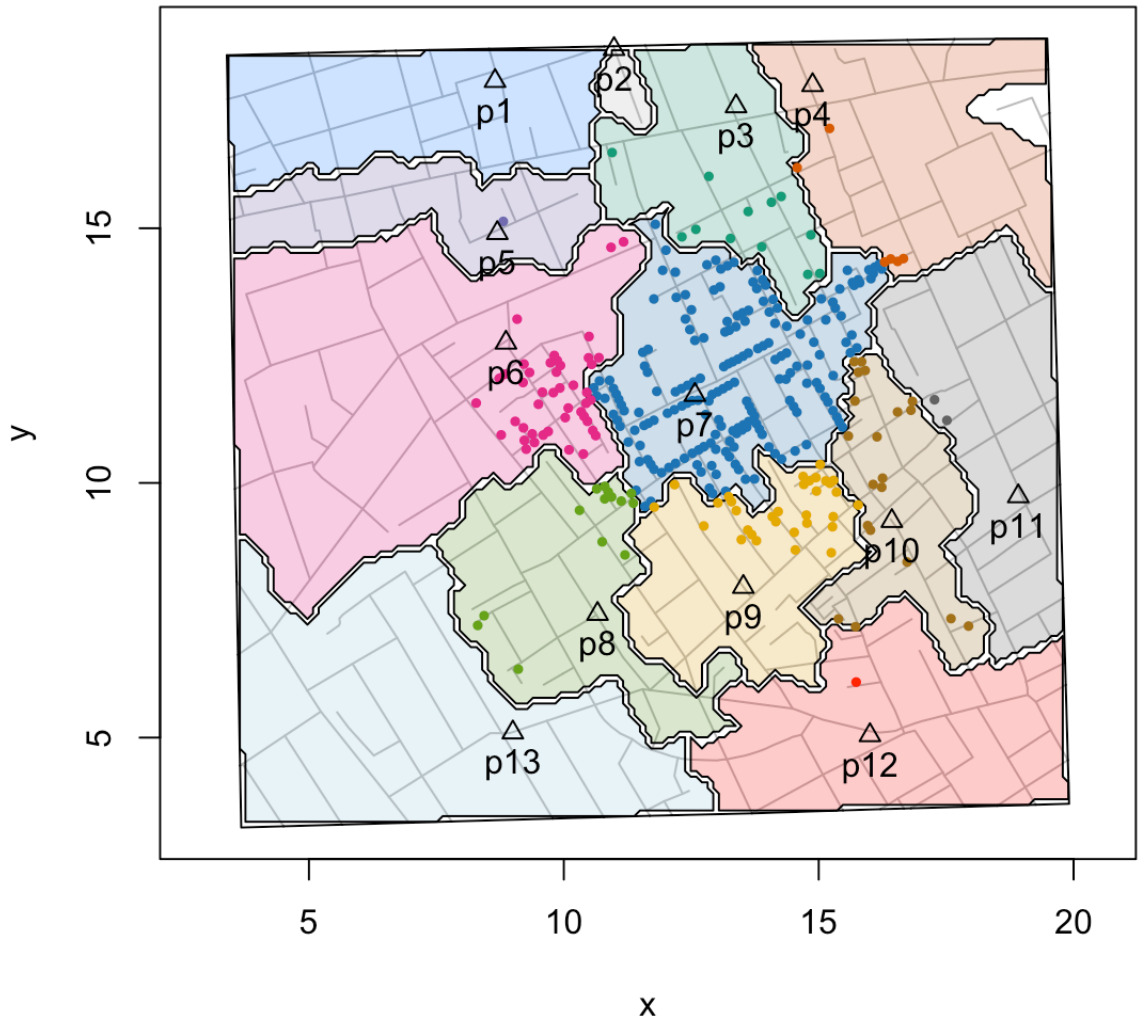
identify points on perimeter

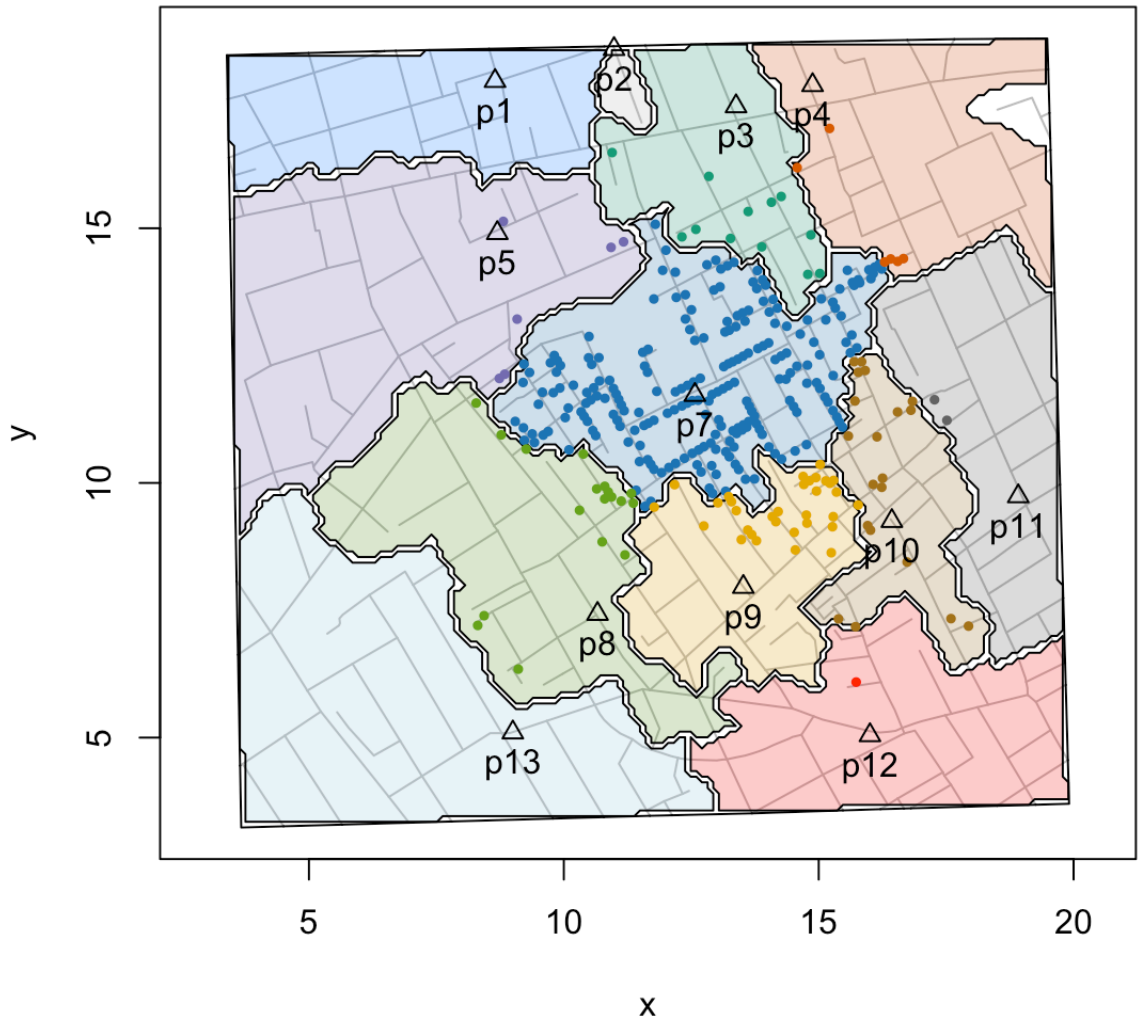


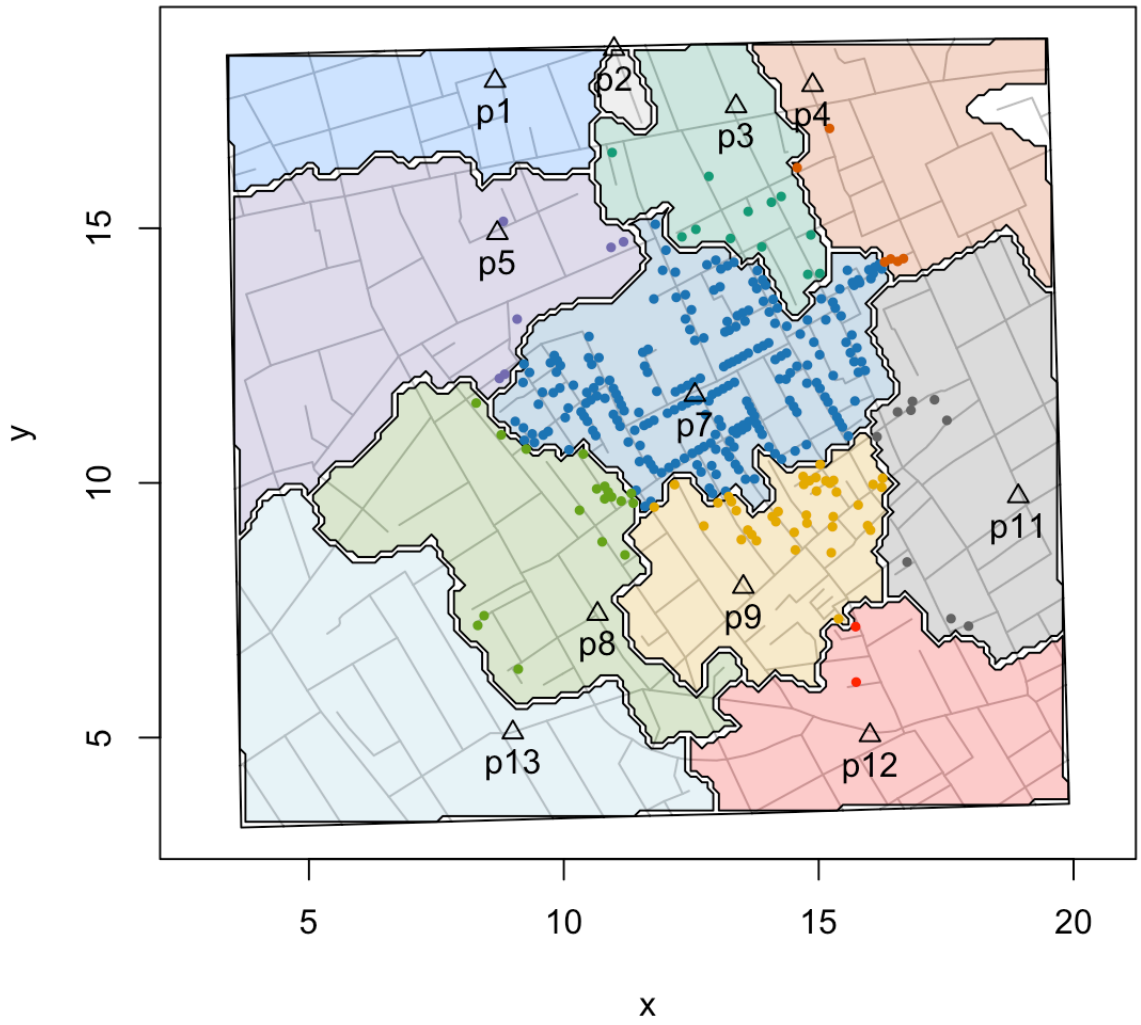
connect the dots to form polygon

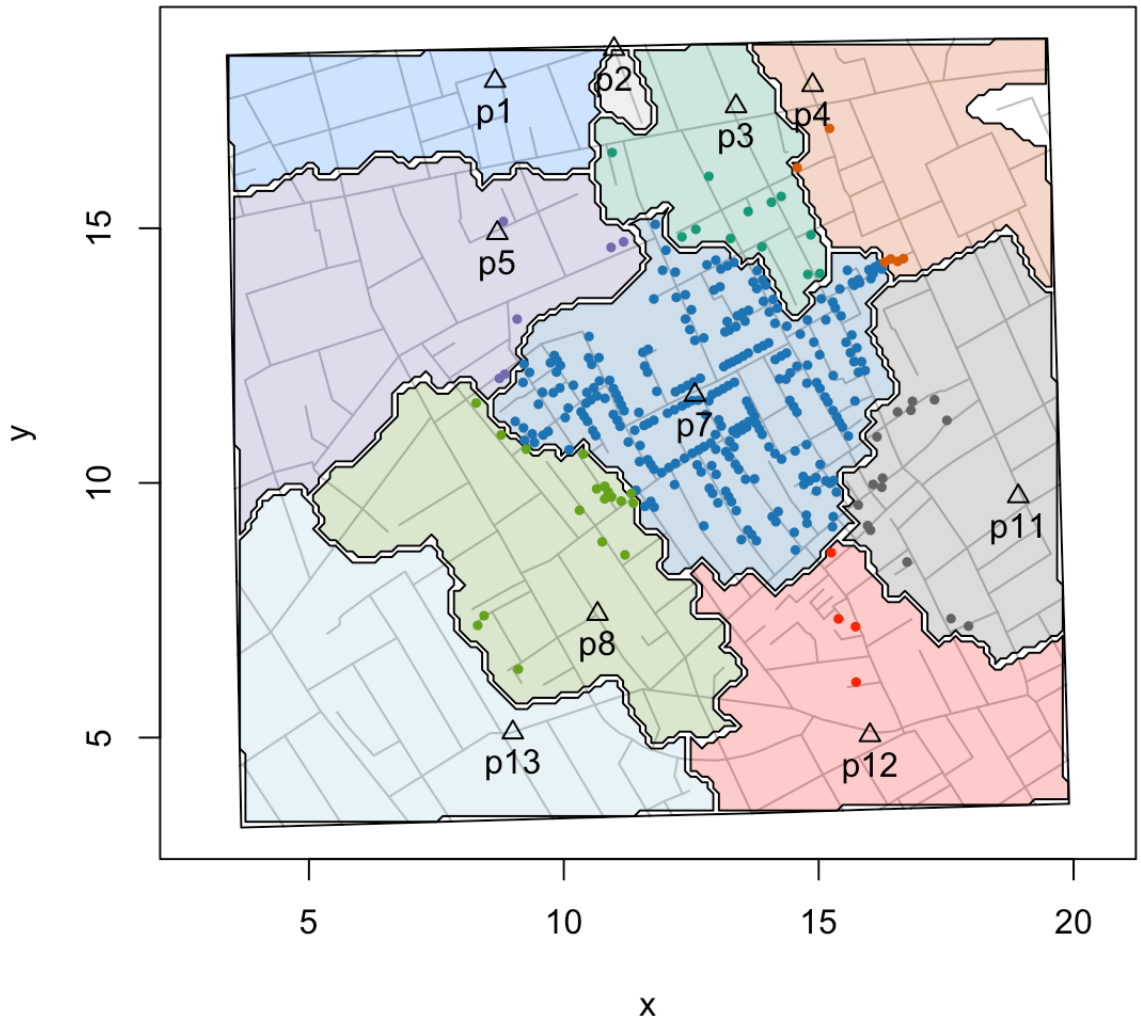


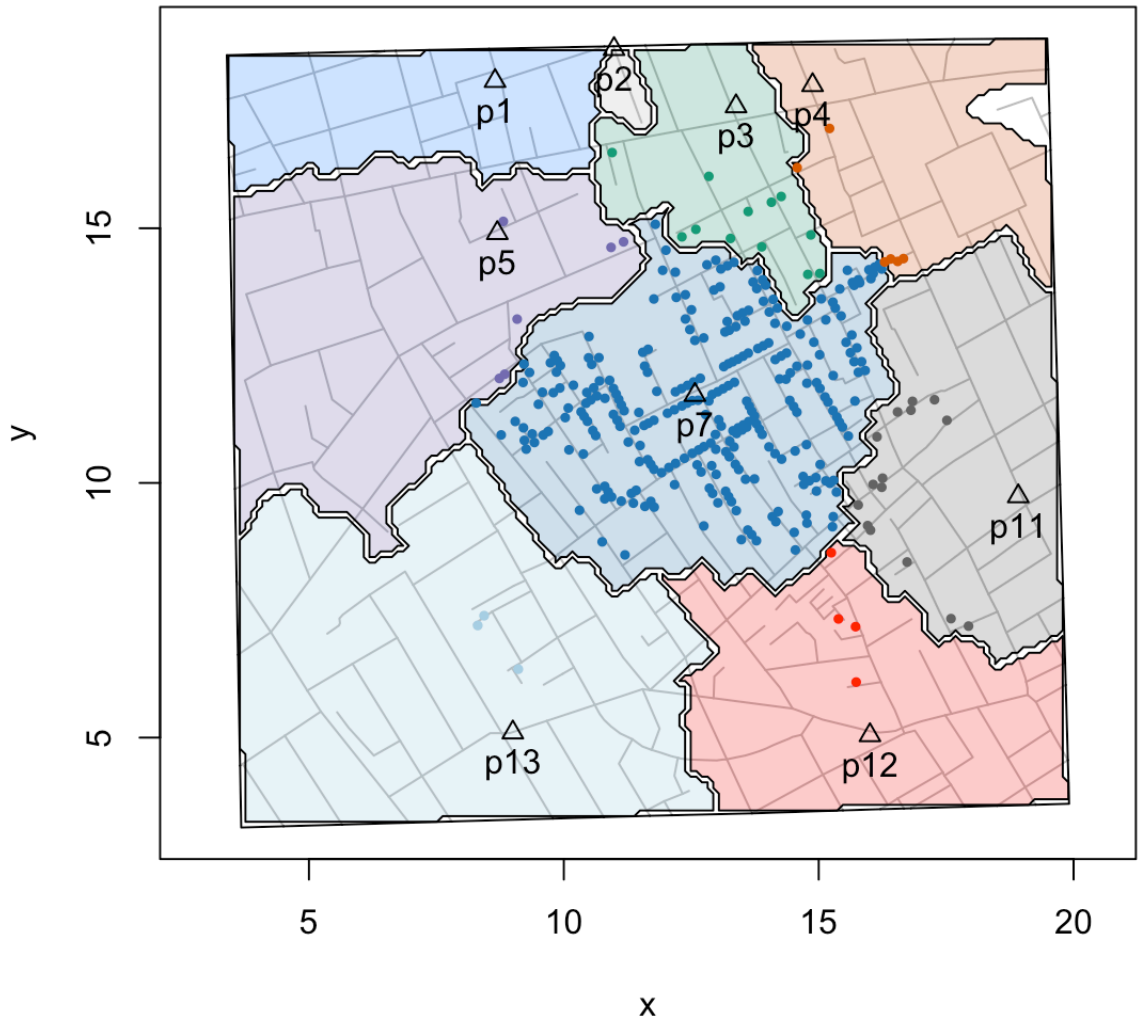
scenarios

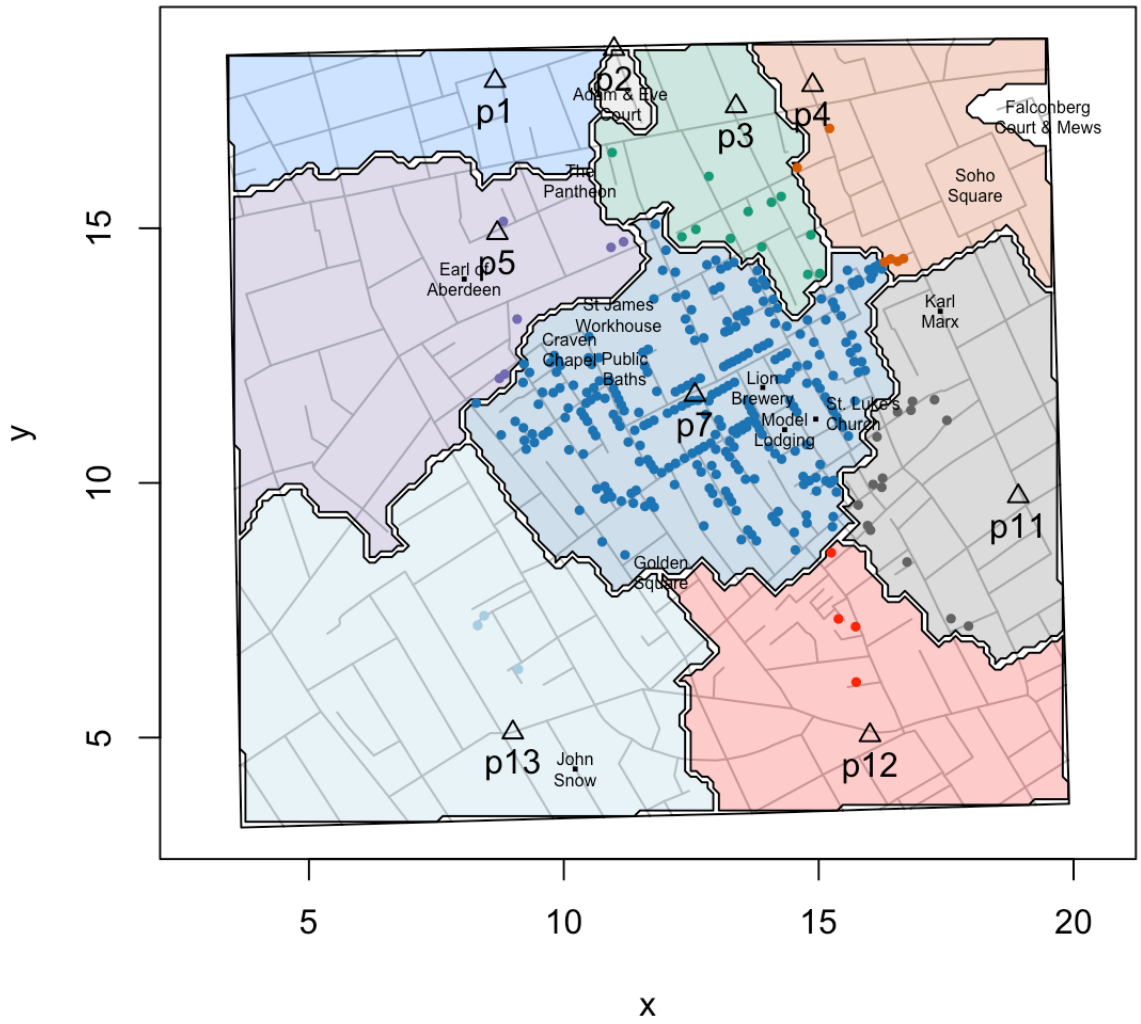

p1
p2
p3
p4
p5
p6
p7
p8
p9
p10
p11
p12
p13
y
x
5
10
15
20
5
10
15


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


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


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


p1
p2
p3
p4
p5
p7
p11
p12
p13
y
x
5
10
15
20
5
10
15


p1
p2
p3
p4
p5
p7
p11
p12
p13
Adam & Eve Court
The Pantheon
Falconberg Court & Mews
Soho Square
Earl of Aberdeen
St James Workhouse
Craven Chapel Public Baths
Lion Brewery
Model Lodging
St. Luke's Church
Karl Marx
Golden Square
John Snow
y
x
5
10
15
20

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

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