Prep7S24

Bilal Tariq

2024-03-29

Reminder: Prep assignments are to be completed individually. Upload a final copy of the .Qmd and renamed .pdf to your private repo, and submit the renamed pdf to Gradescope before the deadline (Sunday night, 3/31/24, by midnight).

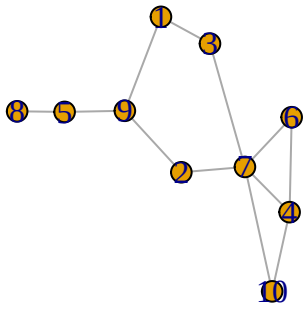
Reading

The associated reading for the week is Chapter 20 on networks and Chapter 17 on spatial data. For Chapter 17, the focus is on working with spatial data, which basically means, making maps and showing appropriate data on them.

1 - Basic Graph Concepts with a Toy Graph

Use the following generated graph to answer the questions that follow. Do NOT change the seed.

```
set.seed(231)
g <- erdos.renyi.game(n = 10, p = 0.3)
plot(g)
```



part a - How many vertices does the graph, g, have? How many edges?

Solution: It has 10 vertices and 12 edges

```
vertices <- vcount(g)
edges <- ecount(g)

cat(vertices, edges)
```

```
10 12
```

part b - Compute the diameter of the graph. Then identify a path with that length.

Hint: More than one path may have a length equal to the diameter. Remember that diameter is not the longest path possible. It is the longest of all the shortest paths. To identify a path, list the vertices involved such as 1-2-3 (this is not an actual path in this graph).

Solution: The diameter is 5. The path along this diameter is 4-7-2-9-5-8

```
diameter <- diameter(g)

cat(diameter,", ")
```

```
5 ,
```

```
#to identifyt the path with that legngth

path <- get_diameter(g)
cat(path)
```

```
4 7 2 9 5 8
```

> part c - Compute the degree for vertex 7 without using an R command. Explain what this number represents.

Solution: This will be the number of edges incident on it. In this case, the degree of vertex 7 will be 5. This just shows a degree of centrality, showing how connected vertex 7 is in the cluster.

part d - Looking at the plot of the graph, identify a vertex triple - three vertices which are connected by edges. Is your selected vertex triple closed - i.e. is it a triangle?

This is asking you to find a set of vertices that could form a triangle - at least 2 of the 3 potential edges should be there. Determining the fraction of actual triangles out of possible triangles is a measure of what is called the clustering present in the network.

You can list your triple with node numbers. E.G. 1-2-8 (not an actual triple in this graph). The idea would be that 1-2 is an edge and 2-8 is an edge. If 1-8 is also an edge, this would be a triangle.

Solution: 7 - 6 - 4 (closed triangle)

> part e - Imagine you have to walk along this graph. Walking around, what vertices are you most likely to visit often if you move around randomly? (What vertices help connect others a lot?)

No computations necessary here - you don't need to solve this formally - just think through it and give a guess. If you really want, you could look up a formal definition for a random walk on a graph. Basically, at any vertex, you have an equal probability of going to any vertices it has an edge to. For example, in this graph, if you were at vertex 3, you'd have a 50% chance of going to vertex 1 and 50% chance of going to vertex 7.

Solution: Vertex 7 and vertex 9 seem to be the most likely visits.

> part f - Compute betweenness centrality for all vertices. Identify the top 3 vertices in terms of betweeness. How do these vertices relate to those you listed in part e?

Solution: From the betweeness values, we see that 7 has a betweeness of 19.5 and 4 has a betweeness of 15.0. The vertex 9 has a betweenness of 12.0. The correspond to what I stated above except for 4.

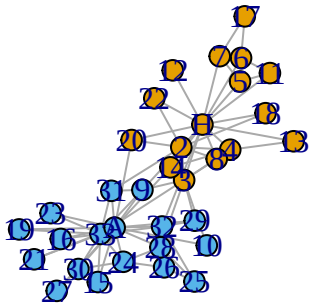
```
betweenness <- betweenness(g)

kable(betweenness)
```

x
3.0
12.0
4.0
0.5
8.0
0.0
19.5
0.0
15.0
0.0

2 - Exploring a Social Network

For this question, we'll explore a famous social network example known as "Zachary's Karate Club". This is information about a Karate club from 1970-1972 that split into 2 factions/groups based on a rift between members denoted "A" for "John A" and "H" for "Mr Hi" in the data set (these are not real names). Let's investigate a little bit!

```
# Visualize the network
# Igraph default plot, may not be "pretty"
data(karate)
karate <- upgrade_graph(karate) # used due to changes in igraph
plot(karate)
```



part a - Plot and describe the degree distribution of the karate network.

Hint: ggplot2 requires data to be in a data.frame to plot, so the code below will get you the degree distribution in data set to use, along with faction information for later use.

```
faction <- get.vertex.attribute(karate)$Faction
```

```
Warning: `get.vertex.attribute()` was deprecated in igraph 2.0.0.
i Please use `vertex_attr()` instead.
```

```
kdegree <- degree(karate)
karatedata <- data.frame(kdegree, faction)
```

Solution:

```
df <- karatedata$kdegree
```

part b - Identify the individuals with the top 5 highest degree values. Do they include "John A" and "Mr Hi"?

Solution:

part c - Determine the eigenvector centrality of all vertices. Identify the individuals with the top 5 eigenvector centrality values. Do they include "John A" and "Mr Hi"?

Hint: The igraph command is eigen_centrality.

Solution:

part d - We investigated k-means as a method of clustering observations to find natural groups. Clustering can be performed on networks, though very different methods are needed to do so. Run the code below to perform a clustering analysis on this network. Then use the provided output to assess whether the clusters found match the provided factions assigned in the network by the researcher who studied the karate club. What did you find? Describe the findings in a few sentences.

Note: This clustering analysis may not find just 2 clusters.

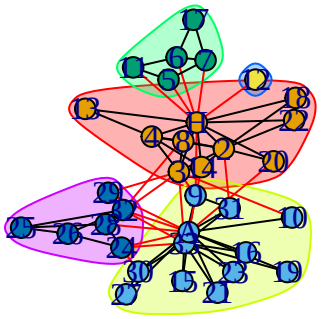
```
# run clustering on karate network
# this is just one example clustering algorithm for a network
kclusters <- leading.eigenvector.community(karate)
# number of clusters
length(kclusters)
```

```
[1] 5
```

```
# size of each cluster
sizes(kclusters)
```

```
Community sizes
 1  2  3  4  5
10 12  5  1  6
```

```r
#how to plot the solution
plot(kclusters, karate)
```



```r
# Get cluster memberships to compare to faction
karatedata <- karatedata %>%
  mutate(clusters = as.numeric(membership(kclusters)))
mosaic::tally(clusters ~ faction, data = karatedata)
```

```
        faction
clusters  1  2
       1 10  0
       2  0 12
       3  5  0
       4  1  0
       5  0  6
```

Solution:

3 - Spatial data basics

Chapter 17 introduces a number of spatial data specific terms.

part a - What are two examples of spatial data structure formats?

Solution: Shapefile and KML

part b - What does `EPSG` stand for in `EPSG codes`? Why is it important to know the `EPSG` number, if applicable for your map?

Solution: It stands for Eurpoean Petroleum Survey group and it's important to know them as they can be used to identify coordinates in data so in our scenario, we can make a plot of the geospatial data.

part c - What is the primary package used in the textbook to work with spatial data?

Solution: We will be working with the sf package.

part d - What term/concept did you find hardest to understand from the reading?

Solution: I currently can't seem to understand the concept of layers such as the st_layers function.

4 - Reproducing a map

Section 17.1 introduces *shapefiles*, and includes an example of working with a shapefile to re-create Snow's cholera map. This exercise mostly follows along with the text code.

Setup

Load the **sf** package in the `setup` code chunk. Verify your working directory is the folder *this* .Rmd file is in. Then, create a *data* subfolder in this directory.

part a - Run the code below line-by-line to understand what each part is doing. You can use *command + enter* or *ctrl + enter* to run one selected or highlighted set of code at a time. Confirm that you get a figure similar to that of Figure 17.3 in the textbook.

```
# Download SnowGIS_SHP zip file
download.file("http://rtwilson.com/downloads/SnowGIS_SHP.zip",
              destfile = "data/SnowGIS_SHP.zip")

# Unzip file in same folder
unzip(zipfile = "data/SnowGIS_SHP.zip",
      exdir = "data")
```

```
# Create filepath to unzipped files so we don't need to re-type
data_path <- "data/SnowGIS_SHP"

# List files in SnowGIS_SHP
list.files(data_path)
```

```
 [1] "Cholera_Deaths.dbf"          "Cholera_Deaths.prj"
 [3] "Cholera_Deaths.sbn"          "Cholera_Deaths.sbx"
 [5] "Cholera_Deaths.shp"          "Cholera_Deaths.shx"
 [7] "OSMap.tfw"                   "OSMap.tif"
 [9] "OSMap_Grayscale.tfw"         "OSMap_Grayscale.tif"
[11] "OSMap_Grayscale.tif.aux.xml" "OSMap_Grayscale.tif.ovr"
[13] "Pumps.dbf"                   "Pumps.prj"
[15] "Pumps.sbx"                   "Pumps.shp"
[17] "Pumps.shx"                   "README.txt"
[19] "SnowMap.tfw"                 "SnowMap.tif"
[21] "SnowMap.tif.aux.xml"         "SnowMap.tif.ovr"
```

```r
# List layers
st_layers(data_path)
```

```
Driver: ESRI Shapefile
Available layers:
      layer_name geometry_type features fields                        crs_name
1 Cholera_Deaths         Point      250      2 OSGB36 / British National Grid
2          Pumps         Point        8      1 OSGB36 / British National Grid
```

```r
# Load second layer
cholera_deaths <- st_read(data_path, layer = "Cholera_Deaths")
```

```
Reading layer `Cholera_Deaths' from data source
  `C:\Users\mbila\OneDrive\Documents\GitHub\stat231bilal\Homework\Prep7\data\SnowGIS_SHP'
  using driver `ESRI Shapefile'
Simple feature collection with 250 features and 2 fields
Geometry type: POINT
Dimension:     XY
Bounding box:  xmin: 529160.3 ymin: 180857.9 xmax: 529655.9 ymax: 181306.2
Projected CRS: OSGB36 / British National Grid
```

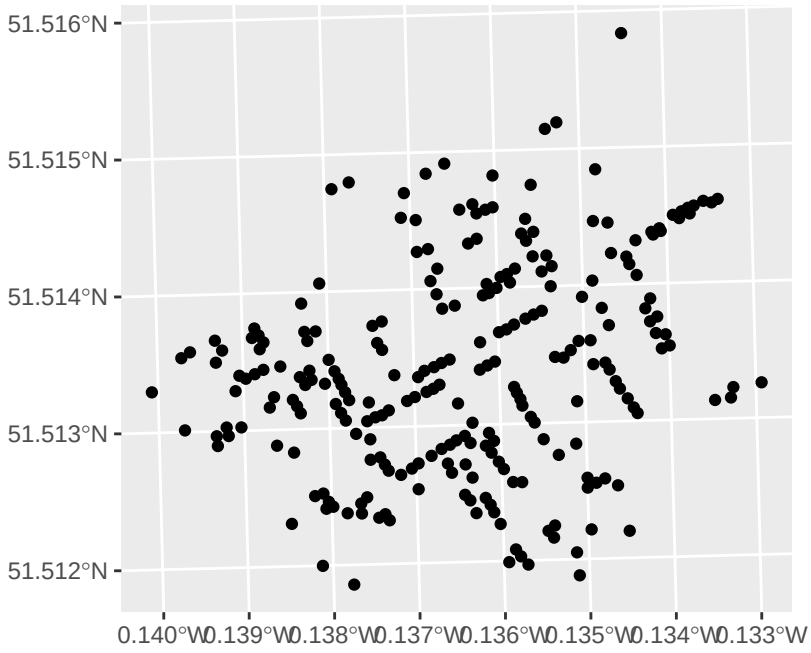
```r
class(cholera_deaths)
```

```
[1] "sf"         "data.frame"
```

```r
head(cholera_deaths)
```

```
Simple feature collection with 6 features and 2 fields
Geometry type: POINT
Dimension:     XY
Bounding box:  xmin: 529308.7 ymin: 181006 xmax: 529336.7 ymax: 181031.4
Projected CRS: OSGB36 / British National Grid
  Id Count                   geometry
1  0     3 POINT (529308.7 181031.4)
2  0     2 POINT (529312.2 181025.2)
3  0     1 POINT (529314.4 181020.3)
4  0     1 POINT (529317.4 181014.3)
5  0     4 POINT (529320.7 181007.9)
6  0     2   POINT (529336.7 181006)
```

```
# Context-less plot
ggplot(cholera_deaths) +
  geom_sf()
```



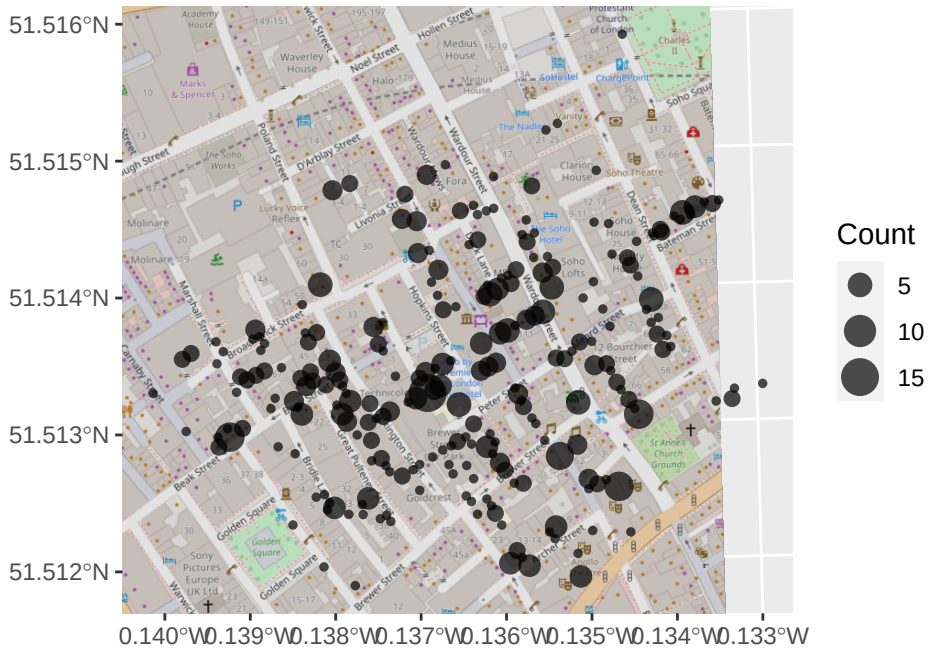
Solution: Yes, we got exactly a map like displayed in fig 17.3

part b - Now use the **ggspatial** package to overlay the London street map. Make sure you (install then) load the **ggspatial** package in the `setup` code chunk before running the code. What is wrong with this map?

```
# if you get an error that a package is missing, you may
# need to install some dependencies
ggplot(cholera_deaths) +
  annotation_map_tile(type = "osm", zoomin = 0) +
  geom_sf(aes(size = Count), alpha = 0.7)
```

```
Loading required namespace: raster
```

```
Zoom: 17
```

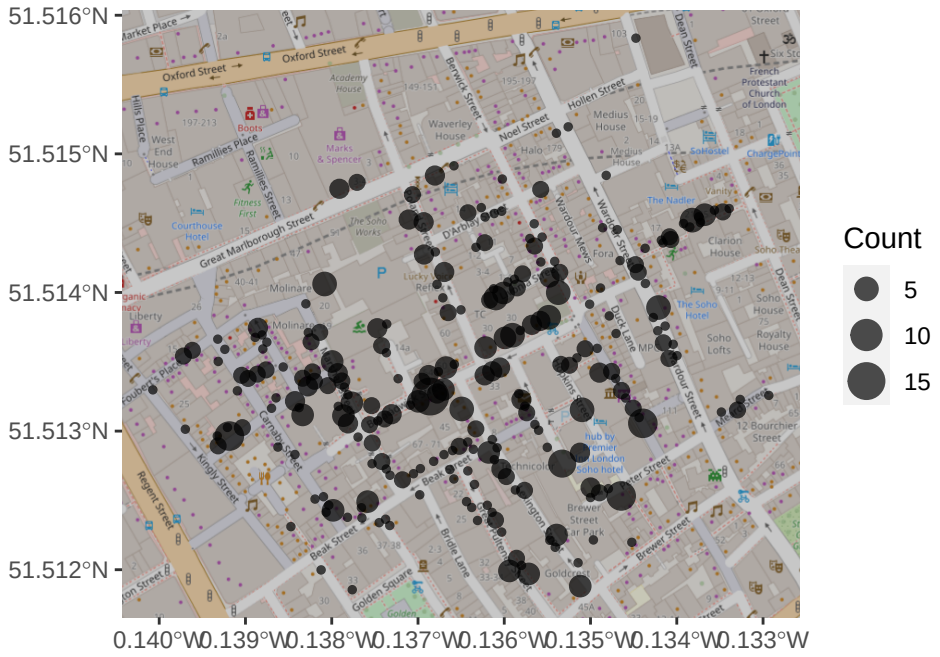


Solution: The dots representing the deaths are off by hundreds of metres.

part c - Set the coordinates from the cholera data as the `espg:27700` coordinate system using `st_set_crs()`, then transform them to the `espg:4326` system using `st_transform()`, and finally plot the new, correctly projected data. What does "crs" stand for in this code?

```
cholera_latlong <- cholera_deaths %>%
  st_set_crs(27700) %>%
  st_transform(4326)

ggplot(cholera_latlong) +
  annotation_map_tile(type = "osm", zoomin = 0) +
  geom_sf(aes(size = Count), alpha = 0.7)
```

```
Zoom: 17
```



Solution: It stands or coordinate reference system.

part d - Repeat the layer loading and coordinate transformation procedure to add the water pumps to the plot. Looking at your plot, do you agree that the water pump on Broad Street seems to have been the problem for this outbreak?

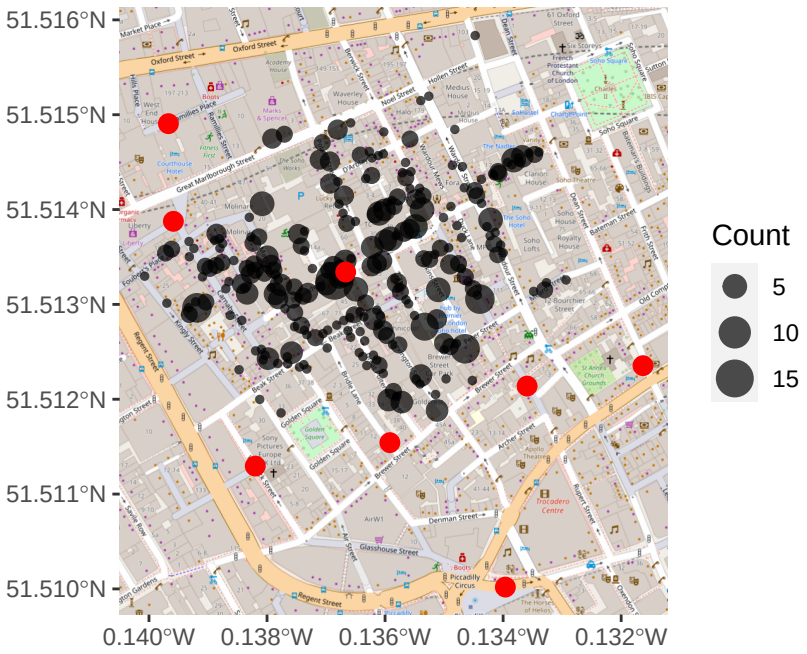
```
pumps_latlong <- st_read(data_path, layer = "Pumps") %>%
  st_set_crs(27700) %>%
  st_transform(4326)
```

```
Reading layer `Pumps' from data source
  `C:\Users\mbila\OneDrive\Documents\GitHub\stat231bilal\Homework\Prep7\data\SnowGIS_SHP'
  using driver `ESRI Shapefile'
Simple feature collection with 8 features and 1 field
Geometry type: POINT
Dimension:     XY
Bounding box:  xmin: 529183.7 ymin: 180660.5 xmax: 529748.9 ymax: 181193.7
Projected CRS: OSGB36 / British National Grid
```

```
ggplot(cholera_latlong) +
  annotation_map_tile(type = "osm", zoomin = 0) +
```

```
    geom_sf(aes(size = Count), alpha = 0.7) +
    geom_sf(data = pumps_latlong, size = 3, color = "red")
```

```
Zoom: 17
```



Solution: Yes, it seems as though the water pump on broad street was indeed the cause of the cholera outbreak as it has a high centrality and is in the middle of the region where the cholera cases were recorded.

> part e - Finally, try out the code below to create a dynamic map using the **leaflet** package (install and load as before!). Zoom in and out of the map to confirm that (1) there is a death in the middle of Hopkins Street, and (2) there is a pump near the intersection of Broadwick Street and Lexington Street. What seem to be the main types of businesses along the modern day Kingly street (look left of the pump)?

```
# create dynamic map
leaflet() %>%
  addTiles() %>%
  addCircleMarkers(data = cholera_latlong,
                   radius =  ~ Count,
```

```
                    color = "navy",
                    stroke = FALSE,
                    fillOpacity = 0.7) %>%
  addCircleMarkers(data = pumps_latlong,
                    radius = 6,
                    color = "red",
                    stroke = FALSE,
                    fillOpacity = 0.7)
```

Solution: There seems to be a lot of clothing brands, a shoe shop, and a lot of food brands on Kingly street.