Practice7S24

Sofia Guttmann

2024-04-17

Reminder: Practice assignments may be completed working with other individuals.

Reading

The associated reading for the week is Chapter 20 on Networks and Chapter 17 on Spatial Data.

Practicing Academic Integrity

If you worked with others or used resources outside of provided course material (anything besides our textbook, course materials in the repo, labs, R help menu) to complete this assignment, please acknowledge them below using a bulleted list.

I acknowledge the following individuals with whom I worked on this assignment:

Name(s) and corresponding problem(s)

-

I used the following sources to help complete this assignment:

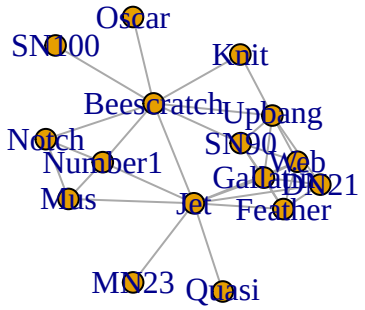
Source(s) and corresponding problem(s)

-

1 - Dolphins

A pod of dolphins has been studied for years in New Zealand to try to understand their social interactions. While the whole network is available (you can plot it if you like) it's hard to visualize. So, we'll look at a small sample of the network. We took what is called a "snowball sample" starting with the dolphin called "Notch". We took all the dolphins he was connected to, and then all the dolphins they were connected to, and then all the dolphins all those dolphins were connected to, and found this induced subgraph. (This is a snowball sample with 3 waves, with one originating vertex).

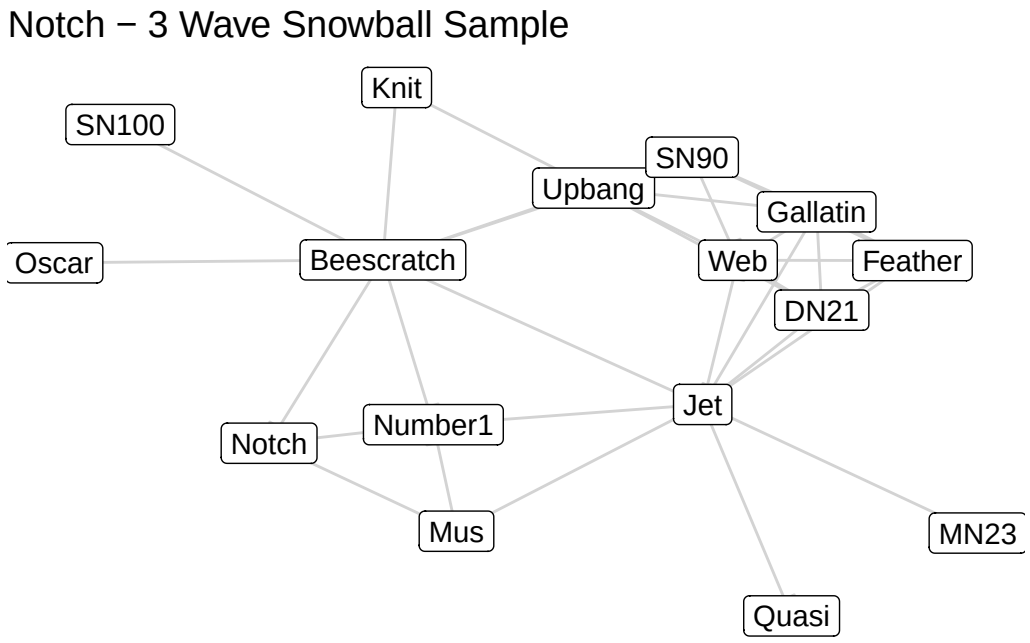
```
dolphins <- read_graph("https://awagaman.people.amherst.edu/stat240/dolphins.gml",
                       format = "gml")

# focus on Notch - vertex 27
# take snowball sample by finding neighbors sequentially
myvert1 <- neighbors(dolphins, 27)
myvert2 <- neighbors(dolphins, myvert1)
myvert3 <- neighbors(dolphins, myvert2)

# put set of all needed vertices together
# duplicates do not need removed
combined <- c(27, myvert1, myvert2, myvert3)

# create sub-graph
dolphin_Notch <- induced_subgraph(dolphins, combined, impl = "copy_and_delete")
plot(dolphin_Notch)
```



part a - Improve the visualization of the Notch-based subgraph by plotting it using ggplot2.

Solution:

```
Notch_network <- ggnetwork(dolphin_Notch)
ggplot(data = Notch_network, aes(x = x, y = y,
xend = xend, yend = yend)) +
geom_edges(arrow = arrow(type = "closed", length = unit(8, "pt")),
color = "lightgray") +
geom_nodes() +
geom_nodelabel(aes(label = label)) +
labs(title = "Notch - 3 Wave Snowball Sample") +
theme_blank()
```



part b - Notch was used to generate this graph, but is Notch the most central dolphin in the network? To help answer this question, compute the degree of each node. Which dolphins have the 3 largest degrees? Share their names and associated degree values in a nice table.

If there are ties for the top 3, include the tied dolphins.

Solution:

```
# Put id and label in a data set for later use
id <- vertex_attr(dolphin_Notch)$id
label <- vertex_attr(dolphin_Notch)$label
dolphin_data <- data.frame(id, label)
```

```r
# find degrees
ddegree <- degree(dolphin_Notch)
dolphin_data <- dolphin_data %>%
mutate(degree = ddegree)
# make a table
dolphin_data %>%
select(-id) %>%
arrange(desc(degree)) %>%
rename("Dolphin" = label, "Degree" = degree) %>%
# three way tie for third
head(5) %>%
kable(booktabs = TRUE)
```

Dolphin	Degree
Jet	9
Beescratch	8
Gallatin	6
Upbang	6
Web	6

```r
# make a table
```

part c - Based on the snowball sampling methodology, what is the furthest any dolphin could be from Notch in this network? What is the furthest apart any two dolphins could be as a result? No code is necessary to answer this.

Solution: The furthest any dolphin could be from Notch is 3. Notch » Dolphin 1 » Dolphin 2 » Dolphin 3. The maximum distance any two dolphins could be apart in the network is 6

part d - What is the diameter of this network? Interpret the value you obtain.

Solution: This means the longest of all the shortest paths between the vertices is 3.

```r
diameter(dolphin_Notch)
```

```
[1] 3
```

part e - Notch was used to generate the network, but is Notch on many of the shortest paths between dolphins? Compute the betweenness centrality of each node. Use a nice table (hint: kable) to show which dolphins have the top 3 betweenness centrality values.

If there are ties for the top 3, include the tied dolphins.

Solution:

```
btw <- betweenness(dolphin_Notch)
dolphin_data <- dolphin_data %>%
mutate(btw = btw)

dolphin_data %>%
select(-id, -degree) %>%
arrange(desc(btw)) %>%
rename("Dolphin" = label, "Betweenness" = btw) %>%

head(3) %>%
kable(booktabs = TRUE)
```

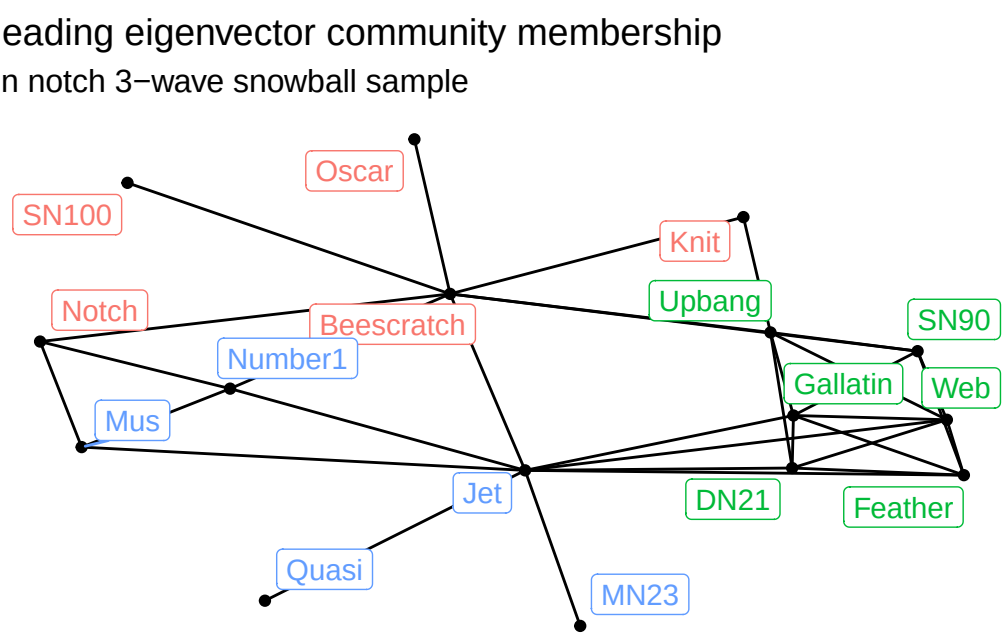
Dolphin	Betweenness
Jet	45.533333
Beescratch	43.533333
Upbang	8.566667

part f - Run a network clustering algorithm of your choice on the dolphin_Notch network. Plot the results including the cluster membership using ggplot2. How many clusters were found?

Solution:

```
Notch_cl <- leading.eigenvector.community(dolphin_Notch)
Notch_membership <- membership(Notch_cl)

dolphin_Notch <- set_vertex_attr(dolphin_Notch,
name = "membership",
value = Notch_membership)
Notch_network2 <- ggnetwork(dolphin_Notch) %>%
mutate(membership = factor(membership))

set.seed(231)
Notch_network2 %>%
ggplot(aes(x = x, y = y, xend = xend, yend = yend)) +
geom_edges() +

geom_nodes() +
```

```
geom_nodelabel_repel(aes(label = label, color = membership)) +
guides(color = "none") +
labs(title = "leading eigenvector community membership",
subtitle = "in notch 3-wave snowball sample") +
theme_blank()
```



part g - Create an induced subgraph for a dolphin of your choice from the original network, using a snowball sample with TWO waves. Plot the network nicely using ggplot2.

Solution:

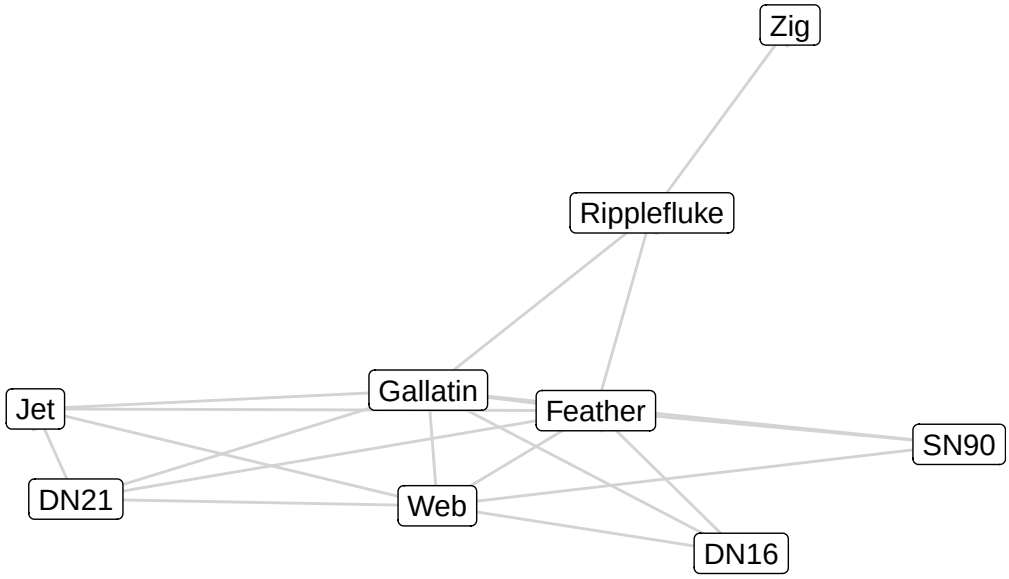
```
get.vertex.attribute(dolphins)$label
```

```
 [1] "Beak"        "Beescratch"  "Bumper"      "CCL"         "Cross"
 [6] "DN16"        "DN21"        "DN63"        "Double"      "Feather"
[11] "Fish"        "Five"        "Fork"        "Gallatin"    "Grin"
[16] "Haecksel"    "Hook"        "Jet"         "Jonah"       "Knit"
[21] "Kringel"     "MN105"       "MN23"        "MN60"        "MN83"
[26] "Mus"         "Notch"       "Number1"     "Oscar"       "Patchback"
[31] "PL"          "Quasi"       "Ripplefluke" "Scabs"       "Shmuddel"
[36] "SMN5"        "SN100"       "SN4"         "SN63"        "SN89"
[41] "SN9"         "SN90"        "SN96"        "Stripes"     "Thumper"
```

```
[46] "Topless"     "TR120"       "TR77"        "TR82"        "TR88"
[51] "TR99"        "Trigger"     "TSN103"      "TSN83"       "Upbang"
[56] "Vau"         "Wave"        "Web"         "Whitetip"    "Zap"
[61] "Zig"         "Zipfel"
```

```
# focus on Ripplefluke - vertex 33
# take snowball sample by finding neighbors sequentially; TWO waves
myvertrf1 <- neighbors(dolphins, 33)
myvertrf2 <- neighbors(dolphins, myvertrf1)
# put set of all needed vertices together
# duplicates do not need removed
combinedrf <- c(33, myvertrf1, myvertrf2)
# create sub-graph
dolphin_Ripplefluke <- induced_subgraph(dolphins, combinedrf, impl = "copy_and_delete")
# Create ggnetwork object from igraph object
RF_network <- ggnetwork(dolphin_Ripplefluke)
# use to plot; set label as label (variable name)
set.seed(231)
ggplot(data = RF_network, aes(x = x, y = y,
xend = xend, yend = yend)) +
geom_edges(arrow = arrow(type = "closed", length = unit(8, "pt")),
color = "lightgray") +
geom_nodes() +
geom_nodelabel(aes(label = label)) +
labs(title = "Ripplefluke - 2 Wave Snowball Sample") +
theme_blank()
```

Ripplefluke − 2 Wave Snowball Sample



2 - Mapping spatial data

Reproduce the map you created in Lab 9's Part 3 - `Your turn` - where you made your own map. Be sure to provide only the minimal code needed to generate it (don't load extra packages, etc.)

In 2-4 sentences, interpret the visualization. What stands out as the central message? Note: You shouldn't say what colors are representing what feature; this is obvious to the viewer, assuming there's an appropriate legend and title. (Add one if you don't have one!) Rather, share what information you extract from the visualization.

Be sure to load all appropriate packages in the setup chunk above.

Solution:

```
library(gapminder)
data(gapminder)
head(gapminder)

world_map <- maps::map("world", plot = FALSE, fill = TRUE) %>%
st_as_sf()

world_mapdata <- world_map %>% left_join(gapminder, by = c("ID" = "country"))
names(world_mapdata)
ggplot(data = world_mapdata) +
geom_sf(aes(fill = lifeExp)) +
scale_fill_viridis(option = "turbo", direction = -1) +
theme_void() +
labs(fill = "life expectancy",
title = "life expectancy by country",
subtitle = "from gapminder package") +
theme(legend.position = "top")
```

3 - Leaflet - Based on MDSR 17.2

In this final problem, we'll explore making an interactive map with leaflet. (You can revisit the last question on the lab for assistance with this as well.) We will use the package *pdxTrees* to access a data set which contains information on trees in Portland parks (be sure to install this package).

```
# loads data set
tree_data <- get_pdxTrees_parks()
```

```
# shows all common names for trees
unique(tree_data$Common_Name)
```

part a - Pick three common names and make a new data set for those trees.

Note: Don't select Douglas-Fir, Norway Maple, or Western Redcedar (~ 1000 or more each). These have too many observations and we want the map to be readable.

Solution:

```
# Example with just 2 - Use other common names!
# Make sure you can do the wrangling to get three names instead of just 2!

my_trees <- tree_data %>%
  filter(Common_Name == "Box Elder" | Common_Name == "Oregon Ash")
my_trees <- tree_data %>%
filter(Common_Name == "Box Elder" | Common_Name == "Oregon Ash")
# Example solution with 3 types of trees as specified
my_trees2 <- tree_data %>%
filter(Common_Name == "Willow" | Common_Name == "Limber Pine" | Common_Name == "White Spru
my_trees2 <- tree_data %>%
filter(Common_Name %in% c("Willow", "Wingnut", "White Spruce"))
```

We are going to create a leaflet map that plots the trees by Common_Name and allows you to pick between which ones you want to view (either, both, or neither), using circles to mark them on the map.

part b - The example code below runs for the demo with 2 Common_Names. Adapt this for your Common_names, and be sure to add the third you chose! Feel free to change colors, etc. as you like.

Solution:

```r
leaflet(my_trees) %>%
  addTiles() %>%
  addCircleMarkers(data = filter(my_trees, Common_Name == "Oregon Ash"),
                   group = "Oregon Ash",
                   lng = ~ Longitude,
                   lat = ~ Latitude,
                   color = "blue") %>%
  addCircleMarkers(data = filter(my_trees, Common_Name == "Pin Oak"),
                   group = "Box Elder",
                   lng = ~ Longitude,
                   lat = ~ Latitude,
                   color = "white") %>%
  addLayersControl(overlayGroups = c("Oregon Ash", "Box Elder"),
                  options = layersControlOptions(collapsed = FALSE))
# Example solution of code with 3 types of trees
leaflet(my_trees2) %>%
addTiles() %>%
addCircleMarkers(data = filter(my_trees2, Common_Name == "Willow"),
group = "Willow",
lng = ~ Longitude,
lat = ~ Latitude,
color = "blue") %>%
addCircleMarkers(data = filter(my_trees2, Common_Name == "Limber Pine"),
group = "Wingnut",
lng = ~ Longitude,
lat = ~ Latitude,
color = "orange") %>%
addCircleMarkers(data = filter(my_trees2, Common_Name == "White Spruce"),
group = "White Spruce",
lng = ~ Longitude,
lat = ~ Latitude,
color = "green") %>%
addLayersControl(overlayGroups = c("Willow", "Wingnut", "White Spruce"),
options = layersControlOptions(collapsed = FALSE))
```

part c - Write a few sentences about what you learned from your map.

Solution: I learned only the Box Elder is present. The second is mostly Willow with a little bit of Spruce.