Assignment 2

Emery Dittmer

2023-03-09

Excercise #2

For this exercise we are investigating the position that we should sit on for the bus ride to Fakebook from Downtown San Francisco!

The Problem

We need to pick an optimal set on the bus. We can talk to people within a fixed range so we will need to be casreful how we pick the seat.

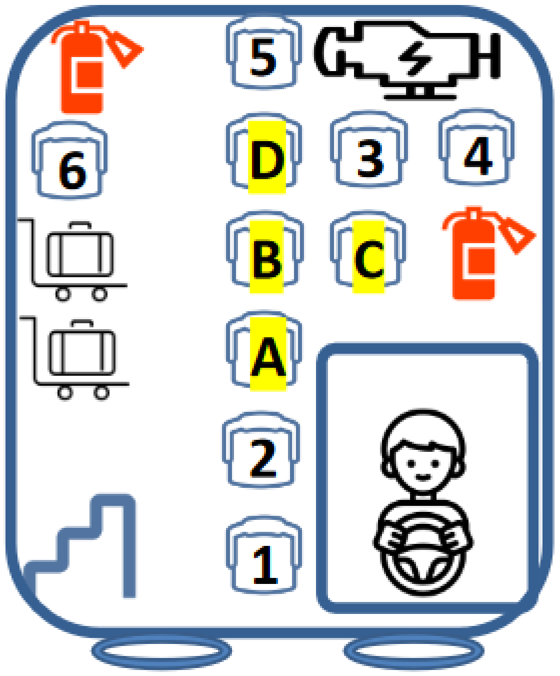


Figure 1: Bus Network Illustartion.

Data acquisition & preprocessing

We can reduce this problem to a set of coordinates and use X- Y Cartesian plane to measure distances and proximity. We will simplify the bus problem slightly by: 1- Ignoring the alley in the bus therefore seat 6 and D are adjacent in this model.

2- Creating a 4x6 grid where seats are either: taken, available or un-available. The un-available seats do not exist.

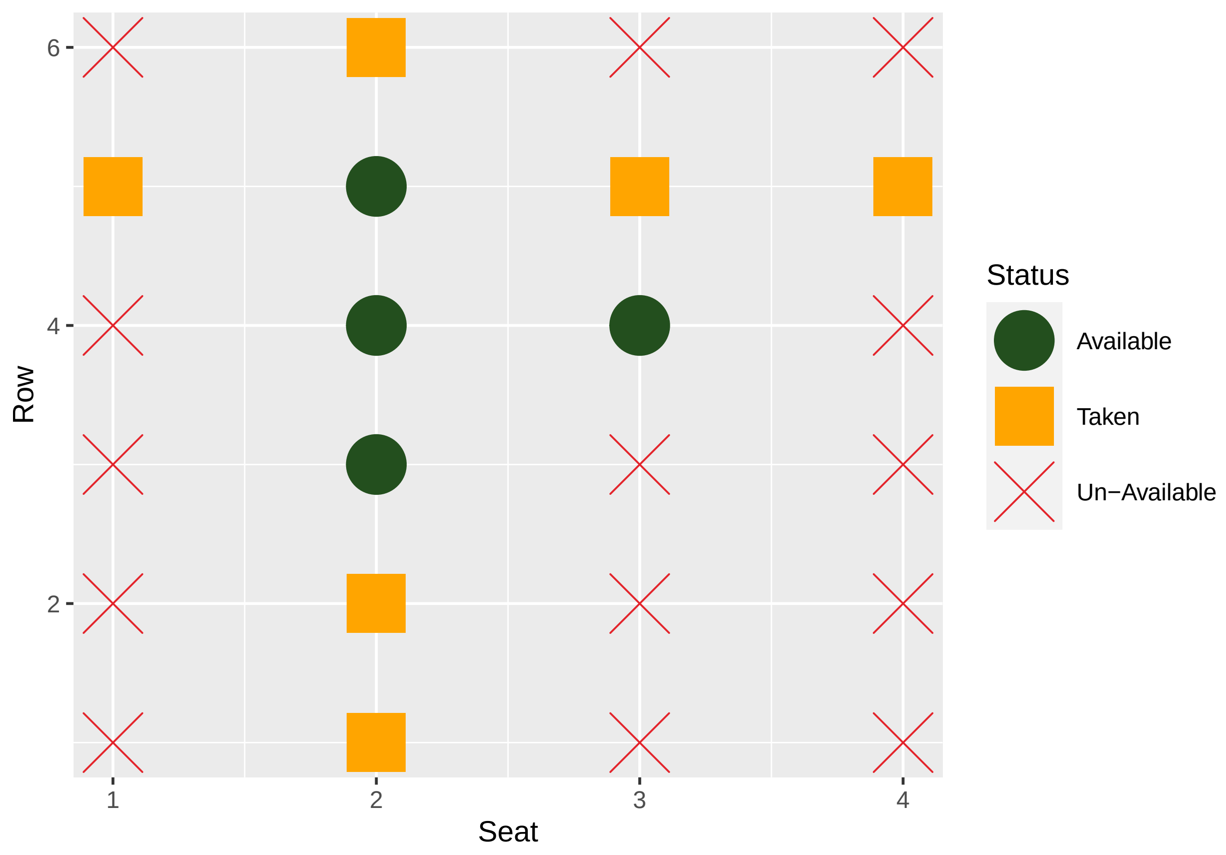
```
library(ggplot2)
```

```
#These are the seats withi the rows
x_dimention=4
#These are the number of rows on the bus
y_dimention=6

Coords <- data.frame(
   Seat = rep(c(1:x_dimention),times=y_dimention),
   Row = rep(c(1:y_dimention),each=x_dimention),
   Status=rep('Available',x_dimention*y_dimention)
   )


#set some conditions First un-available (non existent seats)
unavail="Un-Available"
#remove non existant seats on left
Coords$Status[Coords$Row != 5 & Coords$Seat == 1] <- unavail
#remove on right
Coords$Status[(Coords$Row != 5 & Coords$Seat == 4)] <- unavail
#remove driver cabin
Coords$Status[(Coords$Row < 4 & Coords$Seat > 2)] <- unavail
#remove seats over engine
Coords$Status[(Coords$Row == 6 & Coords$Seat > 2)] <- unavail

#now lets mark the occupied or taken seats
taken="Taken"
#first two rows and last row
Coords$Status[(Coords$Row <3 | Coords$Row >5) & Coords$Status == 'Available' ] <- taken
#seat number on on the left side
Coords$Status[Coords$Seat <2 & Coords$Status == 'Available' ] <- taken
#seat number on on the left side
Coords$Status[Coords$Seat >2 & Coords$Status == 'Available' & Coords$Row >4 ] <- taken
```

Lets Vislualize

Let's take a look at our data!

```
#plotting the seats
ggplot(data=Coords,aes(x=Seat,y=Row))+
  geom_point(aes(shape = Status, color = Status),size=10)+
  scale_shape_manual(values = c(19,15,4,1))+
  scale_color_manual(values=c("#234F1E", "#FFA500", "#E3242B"))
```



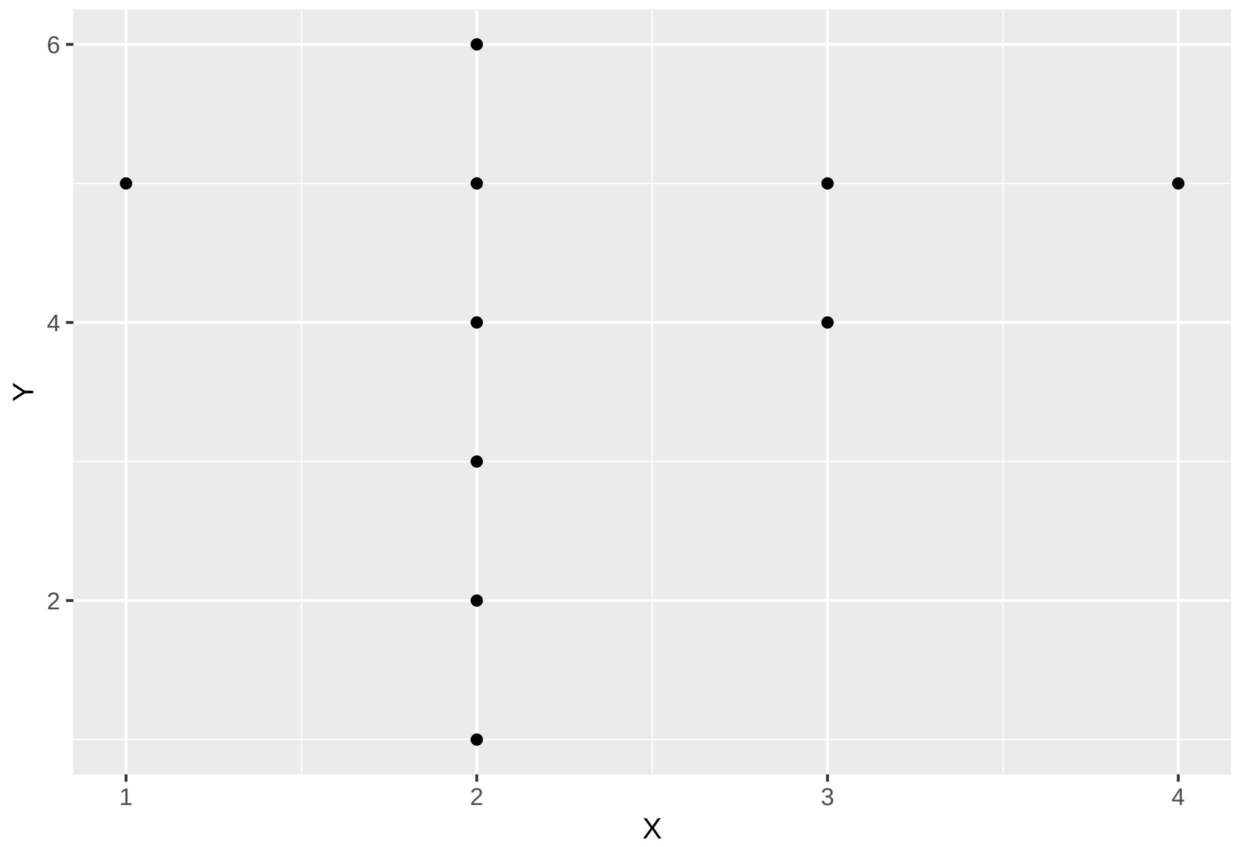
Looks like we have our bus, the seats available and taken! Now lets filter our data frame to have only the useful coordiates

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
Cords_Simple_Status <- Coords %>%
  filter(Status != unavail) %>%
  rename("Y" = "Row", "X"= "Seat")

Cords_Simple <- Cords_Simple_Status[, c("X", "Y")]
```

```
#plotting the seats
ggplot(data=Cords_Simple,aes(x=X,y=Y))+
  geom_point()
```



Now lest look at the distance between points to see which seat is the most central in our network

```
library(cluster)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v tibble  3.1.8     v purrr   0.3.4
## v tidyr   1.2.1     v stringr 1.4.1
## v readr   2.1.3     v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
##  Dissimilarities using Euclidean metric
dist_seats <- daisy(Cords_Simple, metric = "euclidean")
#convert matrix to dataframe
dist_seats=as.data.frame(as.matrix(dist_seats))
#bind the status and coords to each row
dist_seats=cbind(Cords_Simple_Status,dist_seats)

#pivot to make into tabular form
dist_seats <- dist_seats %>%
  pivot_longer(where(is.numeric) & (!contains(c("X","Y"))),names_to = "to_seat_id")

#quick change the name of value to distance
dist_seats <- dist_seats %>%
  rename("Distance" = "value")
```

```
#get the from seat id
lcords=nrow(Cords_Simple_Status)
dist_seats <- cbind(from_seat_id = rep(c(1:lcords),each=lcords), dist_seats)

#remove all duplicated rows 55 of them (10 choose 2 is 45)
dist_seats=unique(dist_seats)
#remove all the self relationships and if the seat is taken
dist_seats <- dist_seats %>%
  filter(to_seat_id != from_seat_id) %>%
  filter(Status != taken)
```

Now we have the distance between each of the availabel seats and the taken or occiped seats. We just need to apply the rules of connections (diagonal, front,back, ect) and we will be able to summarize the table to get the stregth of each seat based on the connections. We will filter all of the connections who are further than sqrt 2 away from the current seat

```
library(visNetwork)
library(networkD3)
#get all connections within distance
dist_seats <- dist_seats %>%
  filter(Distance <= sqrt(2)) %>%
  rename(from = from_seat_id , to = to_seat_id)

#crete the edges
edges <- unique(select(dist_seats, from, to))
edges$to <- as.numeric(edges$to)
edges
```

```
##    from to
## 1     3  2
## 2     3  4
## 3     3  5
## 4     4  3
## 5     4  5
## 6     4  6
## 7     4  7
## 8     4  8
## 9     5  3
## 10    5  4
## 11    5  7
## 12    5  8
## 13    5  9
## 14    7  4
## 15    7  5
## 16    7  6
## 17    7  8
## 18    7 10
```

```
#Create the values of the seat values. we are using dist_seats since we dont care about the occupied se
seat_vals <- dist_seats %>%
  group_by(from) %>%
  summarize(value=n())

nodes <- Cords_Simple_Status
nodes <- rowid_to_column(nodes, "id")
```

```r
nodes <- nodes %>%
  left_join(seat_vals, by = c("id" = "from")) %>%
  replace(is.na(.),1)
nodes$name <- as.character(nodes$id)
nodes <- nodes %>%
  mutate(title = paste("Coords:",paste(X, Y,sep=","),"ID:",id,"size:",value, sep = " ")) %>%
  rename(group = Status,label = name)



visNetwork(nodes, edges)%>%
  visGroups(groupname="Available", color="#234F1E") %>%
  visGroups(groupname="Available", color="#E3242B")  %>%
  visLegend()
```

```
## PhantomJS not found. You can install it with webshot::install_phantomjs(). If it is installed, please
```

Now let's try to find the measures for each seat ## Centrality Measures

Degree Centrality

Is the number of links incident upon a node (i.e., the number of ties that a node has).

```r
#attach(mtcars)
#relabel seats
seat_labs <- data.frame(
  label=c("A","B","C","D"),
  seat_id=c(3,4,5,7)
   )

seat_vals_table <- seat_vals %>%
  rename(seat_id = from) %>%
  left_join(seat_labs,by='seat_id') %>%
  rename(Seat = label,Degree_Centrality = value) %>%
  select(-seat_id)

#formaint to make it look nice
seat_vals_table <- seat_vals_table[, c("Seat", "Degree_Centrality")]
seat_vals_table = seat_vals_table[order(-seat_vals_table$Degree_Centrality),]
seat_vals_table
```

```
## # A tibble: 4 x 2
##   Seat  Degree_Centrality
##   <chr>             <int>
## 1 B                     5
## 2 C                     5
## 3 D                     5
## 4 A                     3
```

Closeness centrality

is a way of detecting nodes that are able to spread information very efficiently through a graph. The closeness centrality of a node measures its average farness (inverse distance) to all other nodes

Betweenness centrality

s a way of detecting the amount of influence a node has over the flow of information in a graph.