Fakebook

Adityo_DasGupta

2023-03-21

Fakebook networking task

The objective of this task is to explore the concept of centrality within networks, specifically in the context of determining the optimal seat or location to occupy during a bus trip from downtown San Francisco to Fakebook. Our ability to connect with other individuals is limited to those within our immediate vicinity and not extend beyond that.

Visualise the grid with all possible seats

```
library(gt)
library(gtExtras)
library(visNetwork)
library(networkD3)
library(ggplot2)
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
```

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

```
## ── Attaching packages
## ───────────────────────────────────────
## tidyverse 1.3.2 ──
```

```
## ✔ tibble  3.1.8     ✔ purrr   0.3.5
## ✔ tidyr   1.2.1     ✔ stringr 1.5.0
## ✔ readr   2.1.3     ✔ forcats 0.5.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

```
library(igraph)
```

```
##
## Attaching package: 'igraph'
##
## The following objects are masked from 'package:purrr':
##
##     compose, simplify
##
## The following object is masked from 'package:tidyr':
##
##     crossing
##
## The following object is masked from 'package:tibble':
##
##     as_data_frame
##
## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union
##
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
##
## The following object is masked from 'package:base':
##
##     union
```

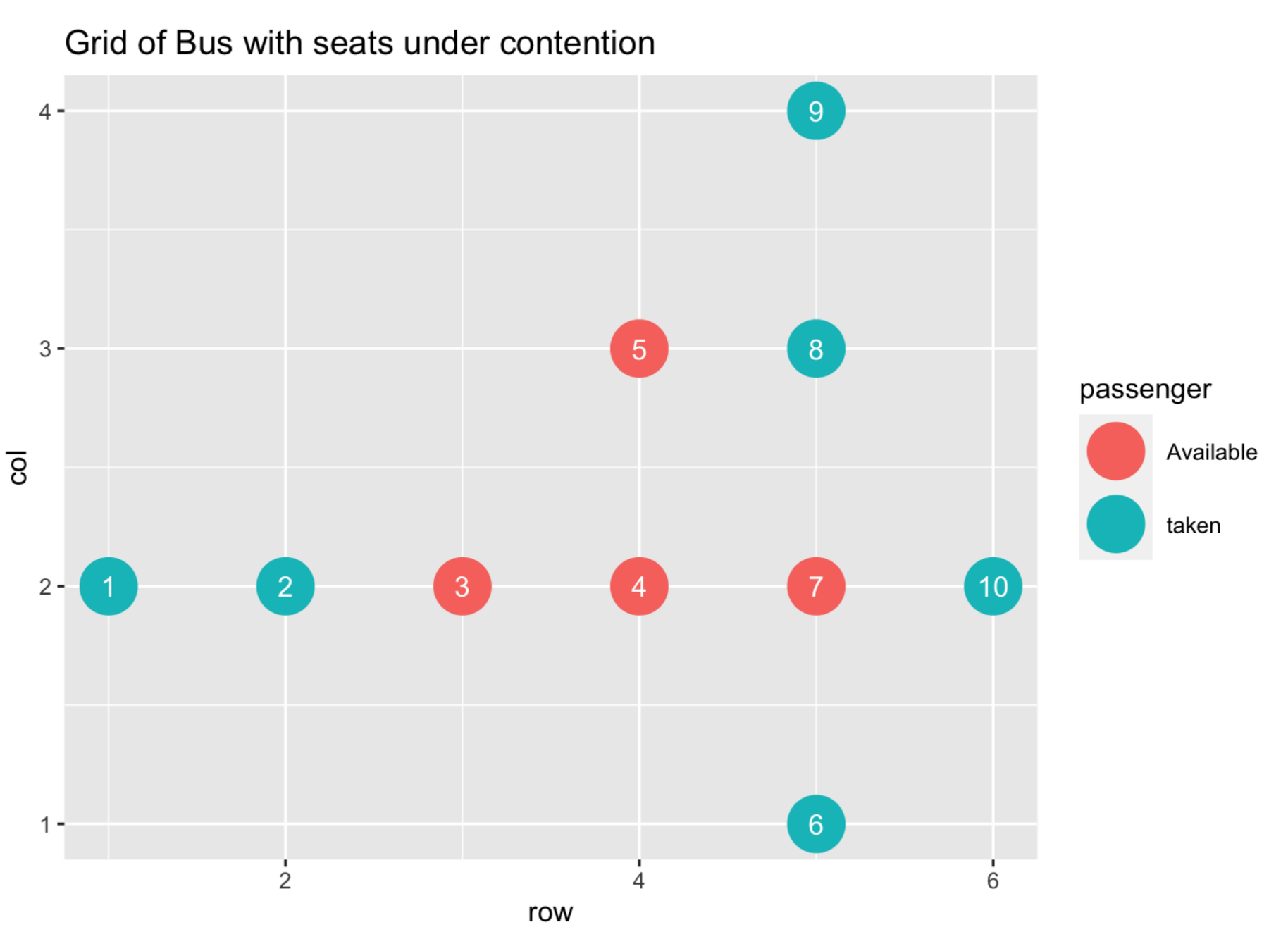
```
library(tidygraph)
```

```
##
## Attaching package: 'tidygraph'
##
## The following object is masked from 'package:igraph':
##
##     groups
##
## The following object is masked from 'package:stats':
##
##     filter
```

```
bus <- data.frame(
  col = rep(c(1:4),times=6),
  row = rep(c(1:6),each=4),
  passenger=rep('Available',4*6))


bus$passenger[bus$row != 5 & bus$col == 1] <- 'NA'
bus$passenger[(bus$row != 5 & bus$col == 4)] <- 'NA'
bus$passenger[(bus$row < 4 & bus$col > 2)] <- 'NA'
bus$passenger[(bus$row == 6 & bus$col > 2)] <- 'NA'


bus$passenger[(bus$row <3 | bus$row >5) & bus$passenger == 'Available' ] <- 'taken'
bus$passenger[bus$col <2 & bus$passenger == 'Available' ] <- 'taken'
bus$passenger[bus$col >2 & bus$passenger == 'Available' & bus$row >4 ] <- 'taken'


ggplot(data=bus,aes(x=row,y=col))+
  geom_point(aes(color = passenger),size=5)+
  labs(title="Grid of Bus no modification")
```

Visualise the grid with only the seats under contention

```
bus=bus[c(bus$passenger!='NA'),]
bus1=rowid_to_column(bus, "id")

ggplot(data=bus1,aes(x=row,y=col,color=passenger,label=id))+
  geom_point(size=10)+
  geom_text(hjust=.5, vjust=0.5,color="white")+
  labs(title="Grid of Bus with seats under contention")
```



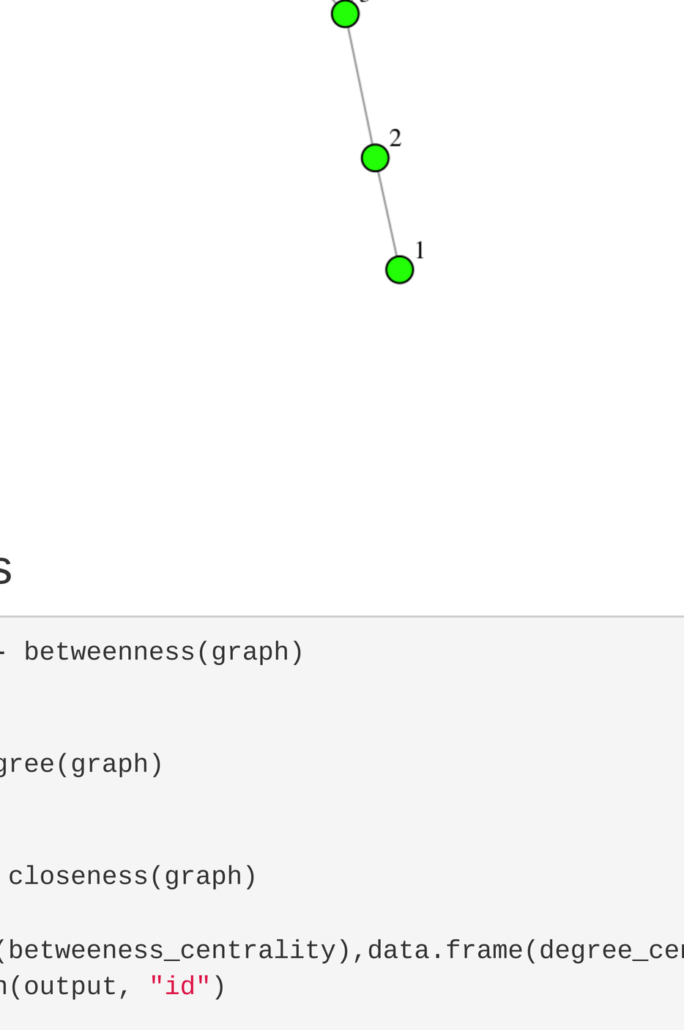
Get all distances between nodes and keep only which are feasible

```
distances <- daisy(bus1[,c('row','col')], metric = "euclidean")
distances=as.data.frame(as.matrix(distances))


distances=cbind(bus1,distances)

#pivot to make into tabular form
distances <- distances %>%
  pivot_longer(where(is.numeric) & (!contains(c("id","row","col"))),names_to = "to_seat_id")

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


distances=distances[distances$Distance<=sqrt(2),]

distances%>%
  gt()
```

id	col	row	passenger	to_seat_id	Distance
1	2	1	taken	1	0.000000
1	2	1	taken	2	1.000000
2	2	2	taken	1	1.000000
2	2	2	taken	2	0.000000
2	2	2	taken	3	1.000000
3	2	3	Available	2	1.000000
3	2	3	Available	3	0.000000
3	2	3	Available	4	1.000000
3	2	3	Available	5	1.414214
4	2	4	Available	3	1.000000
4	2	4	Available	4	0.000000
4	2	4	Available	5	1.000000
4	2	4	Available	6	1.414214
4	2	4	Available	7	1.000000
4	2	4	Available	8	1.414214
5	3	4	Available	3	1.414214
5	3	4	Available	4	1.000000
5	3	4	Available	5	0.000000
5	3	4	Available	7	1.414214
5	3	4	Available	8	1.000000
5	3	4	Available	9	1.414214
6	1	5	taken	4	1.414214
6	1	5	taken	6	0.000000
6	1	5	taken	7	1.000000
6	1	5	taken	10	1.414214
7	2	5	Available	4	1.000000
7	2	5	Available	5	1.414214
7	2	5	Available	6	1.000000
7	2	5	Available	7	0.000000
7	2	5	Available	8	1.000000
7	2	5	Available	10	1.000000
8	3	5	taken	4	1.414214
8	3	5	taken	5	1.000000
8	3	5	taken	7	1.000000
8	3	5	taken	8	0.000000
8	3	5	taken	9	1.000000
8	3	5	taken	10	1.414214
9	4	5	taken	5	1.414214
9	4	5	taken	8	1.000000
9	4	5	taken	9	0.000000
10	2	6	taken	6	1.414214
10	2	6	taken	7	1.000000
10	2	6	taken	8	1.414214
10	2	6	taken	10	0.000000

Visualise the network map

```
final=distances[,c('id','to_seat_id')]

final=final[final$id!=final$to_seat_id,]
final$to_seat_id=as.numeric(final$to_seat_id)

unique_combinations <- t(apply(final, 1, function(id) sort(id)))
unique_combinations <- unique(unique_combinations)

final_1=data.frame(unique_combinations)

graph <- graph_from_data_frame(final_1, directed=FALSE)

plot(graph, layout=layout.fruchterman.reingold,
     vertex.size = 10,
     vertex.label = V(graph)$name,
     vertex.label.cex = 0.8,
     vertex.label.dist = 1.5,
     vertex.label.color = "black",
     vertex.color = "green")
```



Get the centralities

```
betweeness_centrality <- betweenness(graph)


degree_centrality <- degree(graph)


closeness_centrality <- closeness(graph)

output=cbind(data.frame(betweeness_centrality),data.frame(degree_centrality),data.frame(closeness_centrality))
output_1=rowid_to_column(output, "id")
output_1%>%
  gt()
```

id	betweeness_centrality	degree_centrality	closeness_centrality
1	0.0000000	1	0.03333333
2	8.0000000	2	0.04545455
3	14.0000000	3	0.06250000
4	9.0333333	5	0.07142857
5	8.6000000	5	0.07142857
6	0.9333333	3	0.05263158
7	3.2666667	5	0.06250000
8	4.6333333	5	0.06250000
9	0.0000000	2	0.05000000
10	0.5333333	3	0.04761905

We observe that the node with ID=7 which is seat ‘D’ in the original problem is best possible seat