uber analysis

Krishnan Raman

10/6/2020

```
knitr::opts_chunk$set(echo = TRUE)
rm(list=ls())
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(igraph)
```

```
##
## Attaching package: 'igraph'

## 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
```

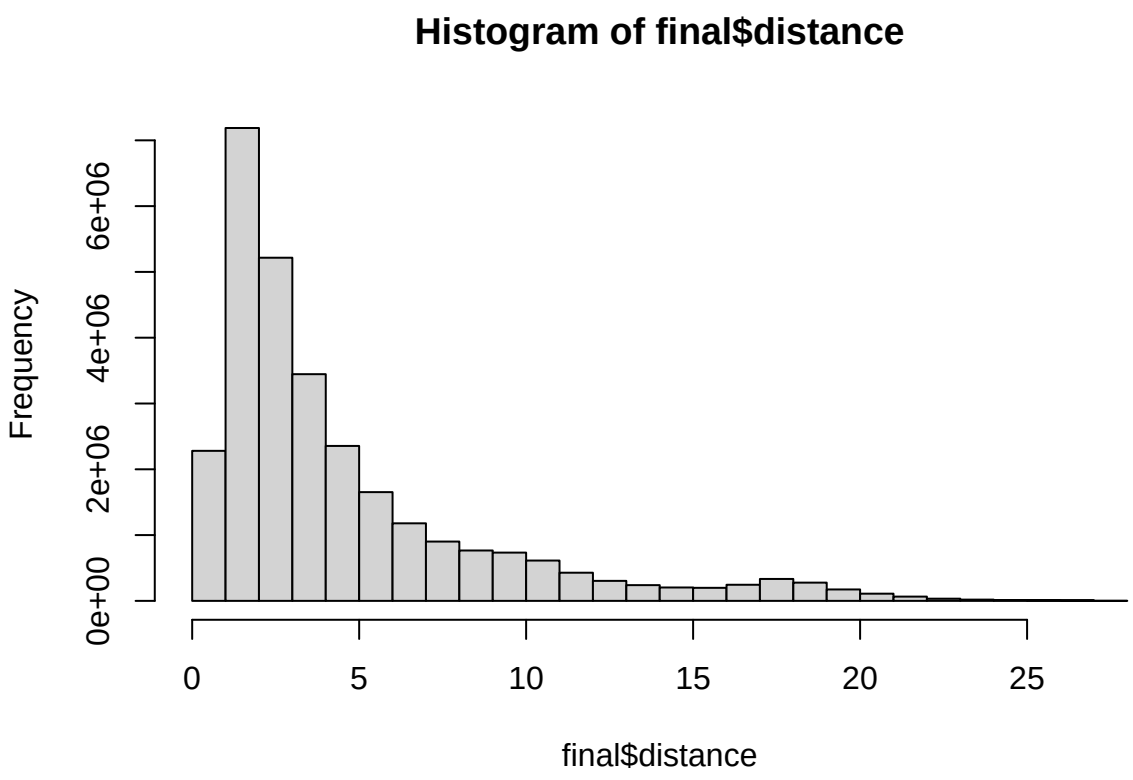
```
# CHANGE THIS TO LOCAL DRIVE WHERE uber_nyc_data.csv is located
# set nrows to 40 million
setwd("~/Desktop/695/uber-tlc-foil-response/uber-trip-data/")
df<-read.csv("uber_nyc_data.csv", nrows=1000*1000*40)

# convert factor vars to formatted numbers
df$distance = as.double(as.character(df$trip_distance))
```

```
## Warning: NAs introduced by coercion
```

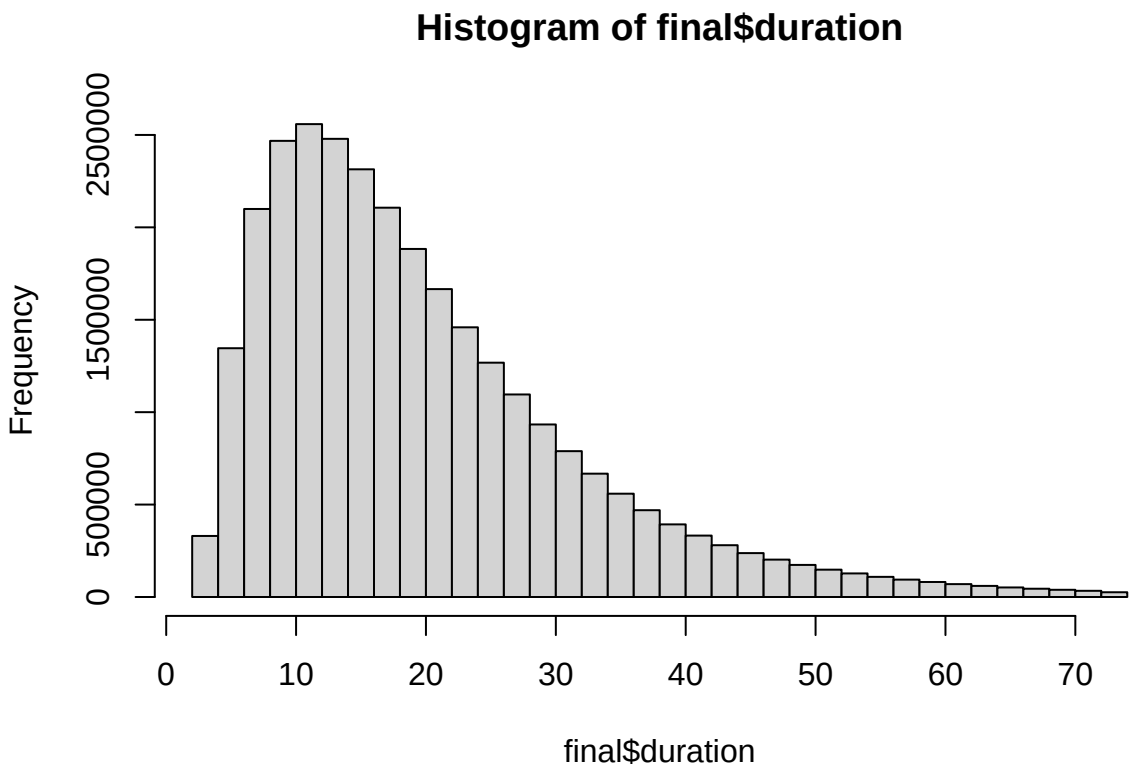
```
df$duration = as.double(as.difftime(as.character(df$trip_duration), format = "%H:%M:%S", units = "mins")

# find 1% & 99% quantiles, eliminate anything beyond
# this helps with cancelled trips, overly long trips & other weird outlier cases
durq = quantile(df$duration,c(0.01, 0.99), names=F, na.rm=T)
disq = quantile(df$distance,c(0.01, 0.99), names=F, na.rm=T)

df2 = df[df$duration > durq[1] & df$duration < durq[2] & df$distance > disq[1] & df$distance < disq[2] &
df2 = select(df2,2:4, 7:8)

# remove NAs & prev dataframes
final = df2[complete.cases(df2), ]
rm(df,df2)
```

compute fare based on linear combination of time and distance

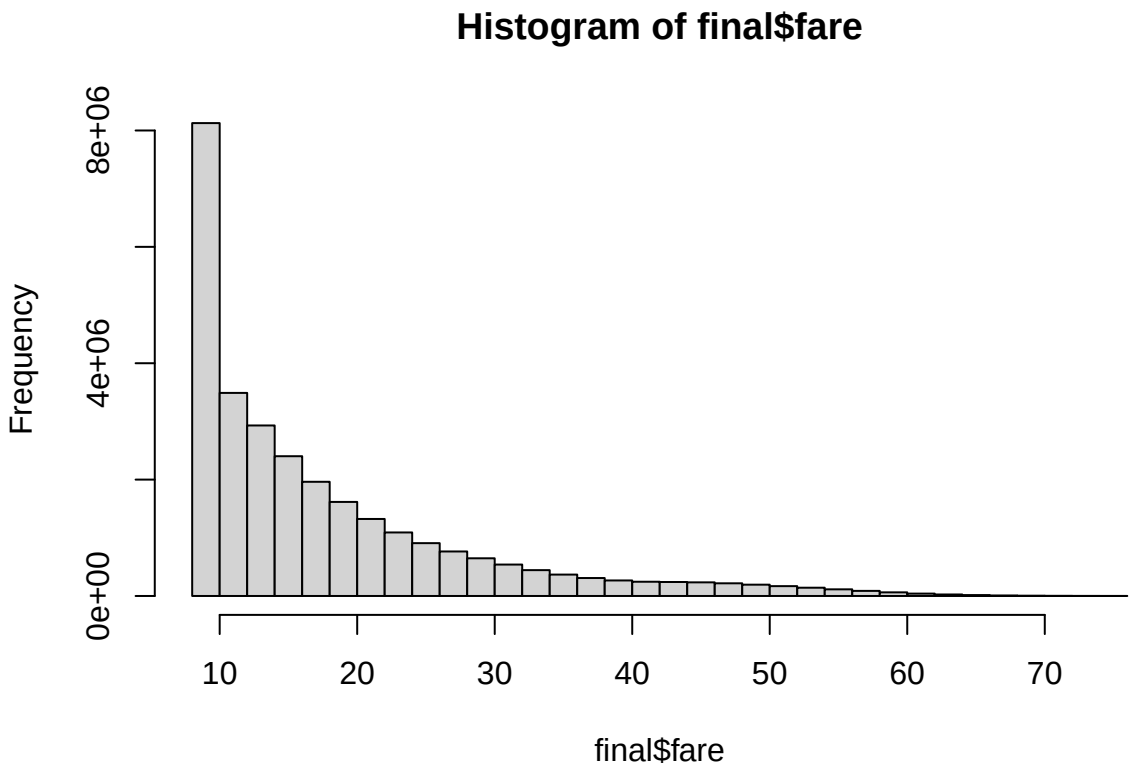
```
base_fare = 2.55
per_minute = 0.35
per_mile = 1.75
min_fare = 8
final$fare <- mapply(function(dis,dur) {
  max(min_fare, base_fare + per_minute*dur + per_mile*dis)
}, final$distance, final$duration)
# distance distribution, duration distribution
hist(final$distance)
```



```r
hist(final$duration)
```




```r
hist(final$fare)
```



```r
# get summary stats
summary(final)
```

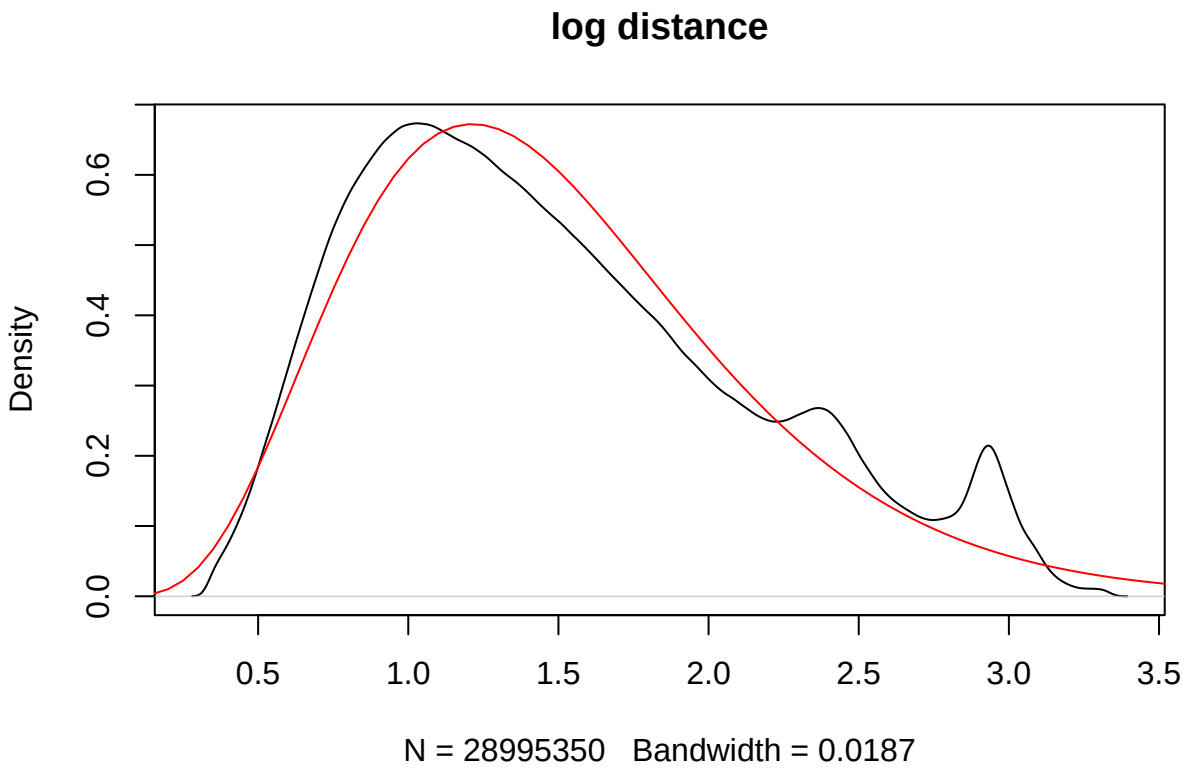
```
##   origin_taz        destination_taz    pickup_datetime       distance
##  Length:28995350    Length:28995350    Length:28995350    Min.   : 0.40
##  Class :character   Class :character   Class :character   1st Qu.: 1.68
##  Mode  :character   Mode  :character   Mode  :character   Median : 2.96
##                                                           Mean   : 4.60
##                                                           3rd Qu.: 5.73
##                                                           Max.   :27.17
##     duration           fare
##  Min.   : 2.867   Min.   : 8.000
##  1st Qu.:10.800   1st Qu.: 9.533
##  Median :16.833   Median :13.960
##  Mean   :19.774   Mean   :17.724
##  3rd Qu.:25.617   3rd Qu.:21.829
##  Max.   :73.750   Max.   :75.823
```

log transform for positive variates

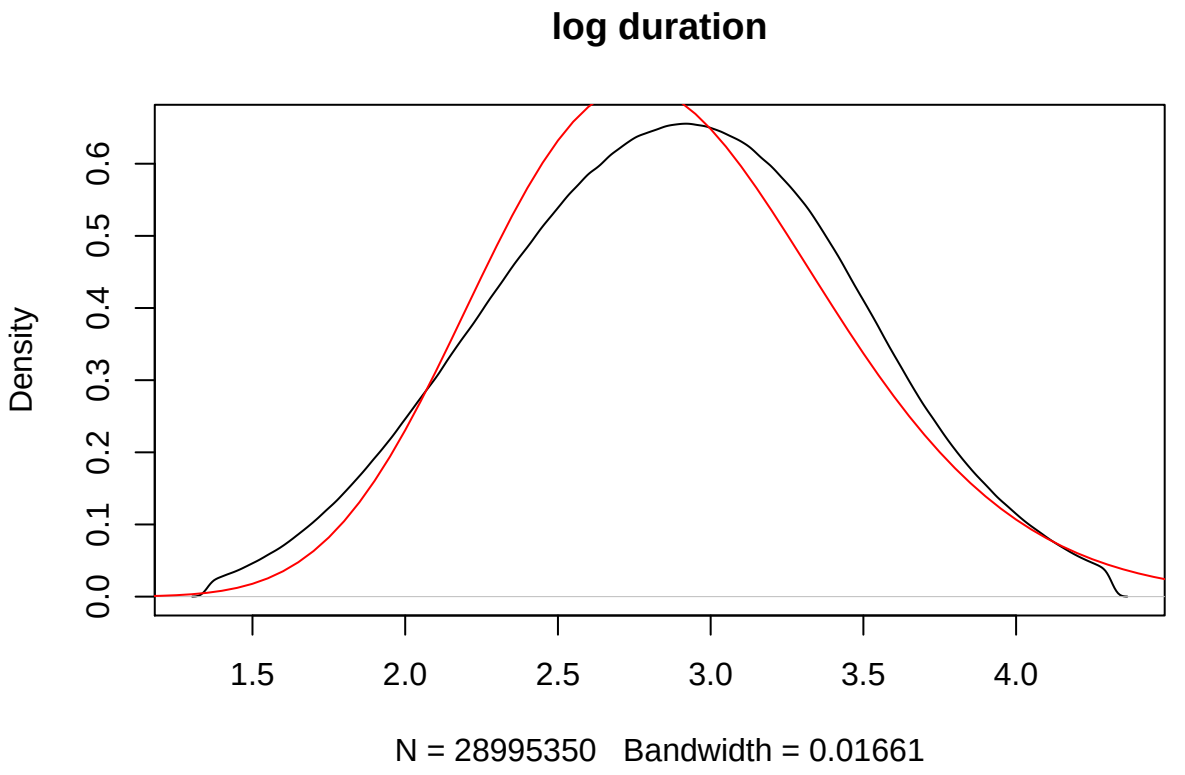
```r
final$logdist = log(1.0+final$distance)
final$logdur = log(1.0 + final$duration)
final$logfare = log(final$fare)
```

gamma priors

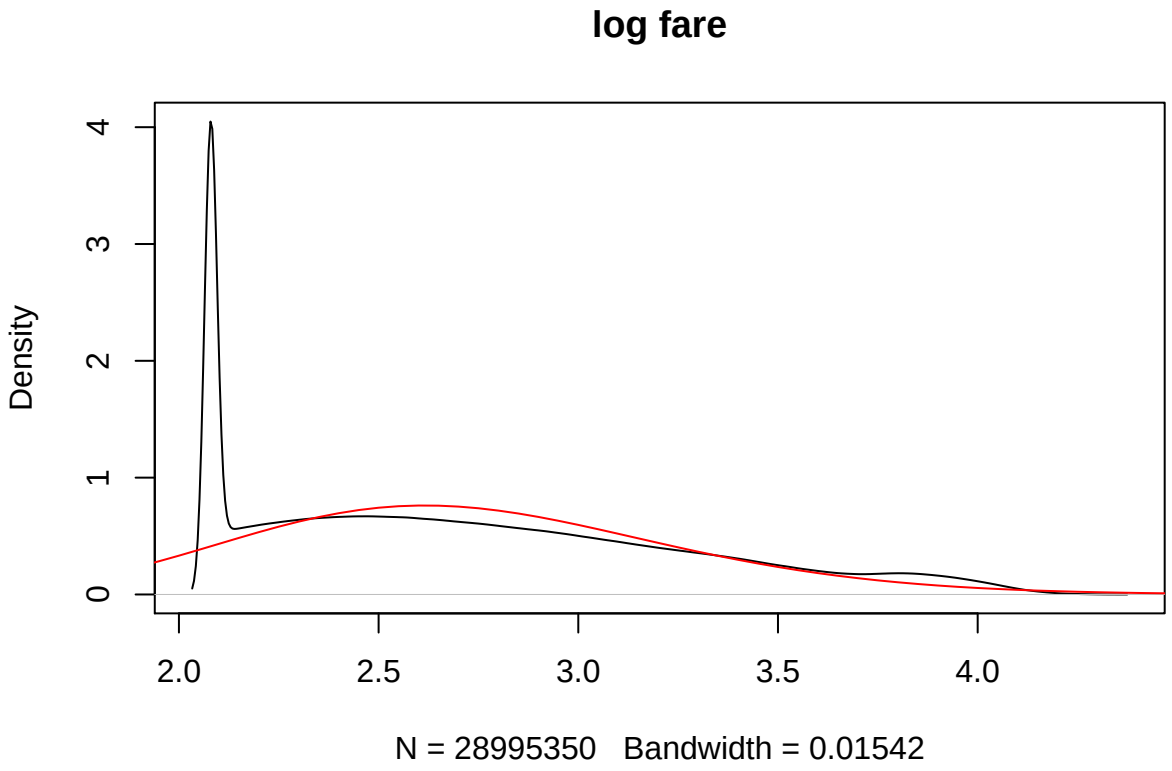
```r
bestgammafit = function(x,title) {
dist_scale = var(x)/mean(x)
dist_shape = mean(x)/dist_scale
plot(density(x), main=title)
px=seq(0,5,0.05)
py=dgamma(px,scale=dist_scale, shape=dist_shape)
lines(px,py, col='red')
}

bestgammafit(final$logdist, "log distance")
```




```
bestgammafit(final$logdur, "log duration")
```



```
bestgammafit(final$logfare, "log fare")
```




```
origdest<- group_split(final %>% group_by(origin_taz,destination_taz))
# get the median fare per src-dest tuple
mid<-sapply(1:length(origdest), function(i) {quantile(origdest[[i]]$fare)[3]})
sorted<-sort(unname(mid), decreasing = TRUE, index.return=TRUE)
topk=20
indices<-sorted$ix[1:topk]
origins<-sapply(indices, function(i) { origdest[[i]]$origin_taz[1] })
dest<-sapply(indices, function(i) { origdest[[i]]$destination_taz[1] })
common<-intersect(origins, dest)
x<-sapply(1:topk, function(i) { (dest[i] %in% common) & (origins[i] %in% common) })
length(x[x==1])
```

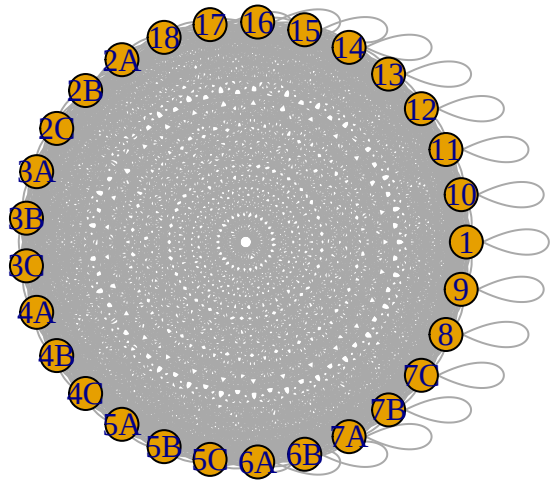
```
## [1] 14
```

```
sorted$x[1:topk]
```

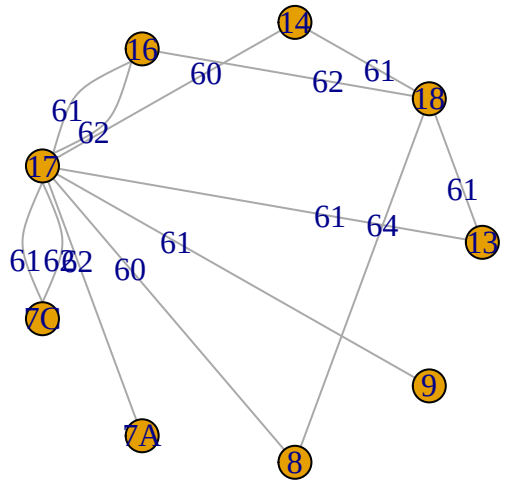
```
##  [1] 63.68917 61.95667 61.89250 61.81667 61.62417 61.23333 61.15750 61.05833
##  [9] 61.02042 61.02042 60.98833 60.11625 60.01125 59.88875 59.66417 59.12167
## [17] 58.29625 58.27583 57.83833 57.68667
```

```
# make graph of original dataset
mylen = length(origdest)
edgematrix<-matrix(NA,nrow=mylen,ncol=3)
for(i in 1:mylen) {
  edgematrix[i,] = c(origdest[[i]]$origin_taz[1],origdest[[i]]$destination_taz[1], quantile(origdest[[i]
```

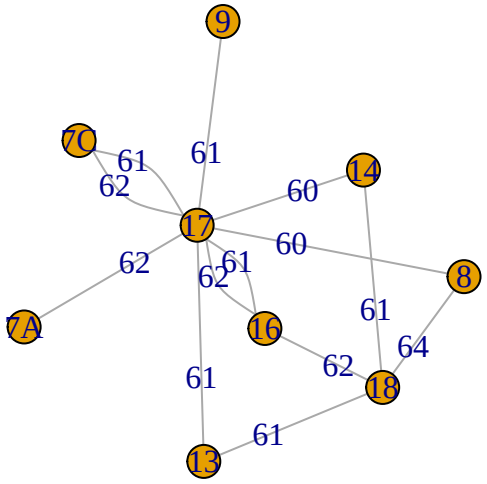
```
}
g<-graph_from_edgelist(edgematrix[,1:2], directed=FALSE)
plot(g, layout=layout.circle)
```



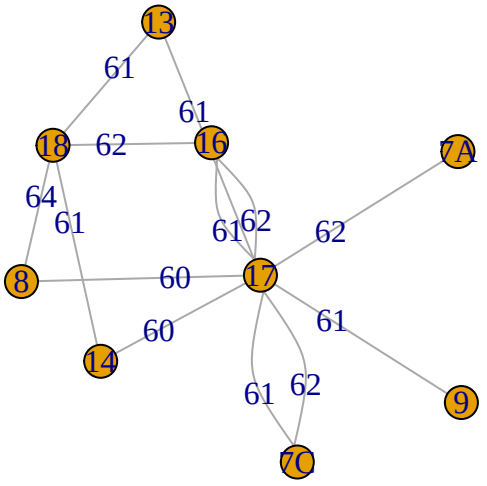

```
highfare_matrix = edgematrix[as.numeric(edgematrix[,3]) > 60,]
highfare_weights = round(as.numeric(highfare_matrix[,3]))
g2<-graph_from_edgelist(highfare_matrix[,1:2], directed=FALSE)
plot(g2,edge.label= highfare_weights, layout=layout.circle)
```




```
plot(g2,edge.label=highfare_weights, layout=layout.davidson.harel)
```




```
plot(g2,edge.label=highfare_weights, layout=layout.fruchterman.reingold)
```

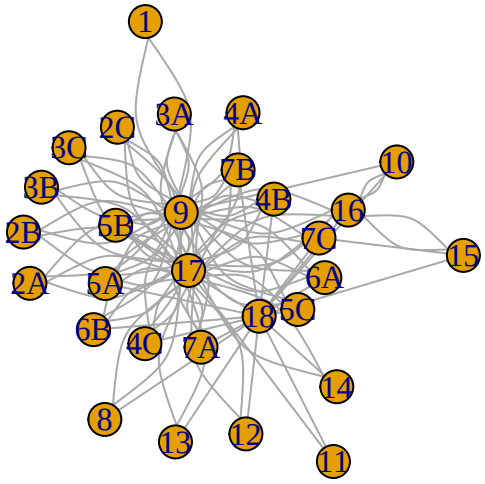



```
for(fares in seq(60,40,-2)) {
  highfare_matrix = edgematrix[as.numeric(edgematrix[,3]) > fares,]
  g2<-graph_from_edgelist(highfare_matrix[,1:2], directed=FALSE)
  lengths = c()
  for (v in V(g2)$name) {
    res = all_simple_paths(g2,v)
    lengths = c(lengths, sapply(1:length(res), function(j) { length(res[[j]]) }))
  }
  m = max(lengths)
  cat(sprintf("Fare: %f, Max length: %f\n", fares, m))
  if (m >= 9) break;
}
```

```
## Fare: 60.000000, Max length: 5.000000
## Fare: 58.000000, Max length: 5.000000
## Fare: 56.000000, Max length: 5.000000
## Fare: 54.000000, Max length: 7.000000
## Fare: 52.000000, Max length: 7.000000
## Fare: 50.000000, Max length: 8.000000
```

```
## Fare: 48.000000, Max length: 8.000000
## Fare: 46.000000, Max length: 9.000000
```

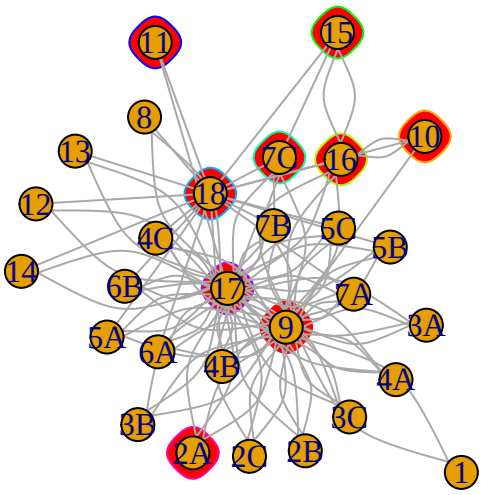
```
# visualize the graph with simple path of length 10
highfare_matrix = edgematrix[as.numeric(edgematrix[,3]) > 44,]
highfare_weights = round(as.numeric(highfare_matrix[,3]))
g3<-graph_from_edgelist(highfare_matrix[,1:2], directed=FALSE)
plot(g3,layout=layout.fruchterman.reingold)
```




```
paths_of_length_10 = c()
found=FALSE
for (src in V(g3)$name) {
    res = all_simple_paths(g3,src)
    for(i in 1:length(res)) {
      if (length(res[[i]]) == 9) {
        last = res[[i]][9]$name
        d = distances(g3,v=last,to=src)
        if (d[1] == 1) {
          print(res[[i]])
          paths_of_length_10 = c(paths_of_length_10, res[[i]])
          found=TRUE
          break
        }
      }
    }
    if(found) break
}
```

```
## + 9/29 vertices, named, from 0e45c13:
## [1] 9  10 16 15 7C 18 11 17 2A
```

```
plot(g3, mark.groups=c("9", "10", "16", "15", "7C", "18", "11", "17", "2A"), mark.col="red")
```




```
plot(make_undirected_graph(c("9", "10","10","16", "16","15", "15","7C", "7C","18", "18","11", "11","17"
```