Final Project

2022-12-07

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.8     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(tidyr)
library(dplyr)
library(ggplot2)
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
##
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
train<-read.csv("~/Documents/stat 432/Final Project/spaceship-titanic/train.csv")
head(train)
```

```
##   PassengerId HomePlanet CryoSleep Cabin   Destination Age   VIP RoomService
## 1     0001_01     Europa     False B/0/P   TRAPPIST-1e  39 False           0
## 2     0002_01      Earth     False F/0/S   TRAPPIST-1e  24 False         109
## 3     0003_01     Europa     False A/0/S   TRAPPIST-1e  58  True          43
## 4     0003_02     Europa     False A/0/S   TRAPPIST-1e  33 False           0
## 5     0004_01      Earth     False F/1/S   TRAPPIST-1e  16 False         303
## 6     0005_01      Earth     False F/0/P PSO J318.5-22  44 False           0
##   FoodCourt ShoppingMall  Spa VRDeck              Name Transported
## 1         0            0    0      0   Maham Ofracculy       False
## 2         9           25  549     44      Juanna Vines        True
## 3      3576            0 6715     49     Altark Susent       False
## 4      1283          371 3329    193      Solam Susent       False
## 5        70          151  565      2 Willy Santantines        True
## 6       483            0  291      0 Sandie Hinetthews        True
```

```r
dim(train)
```

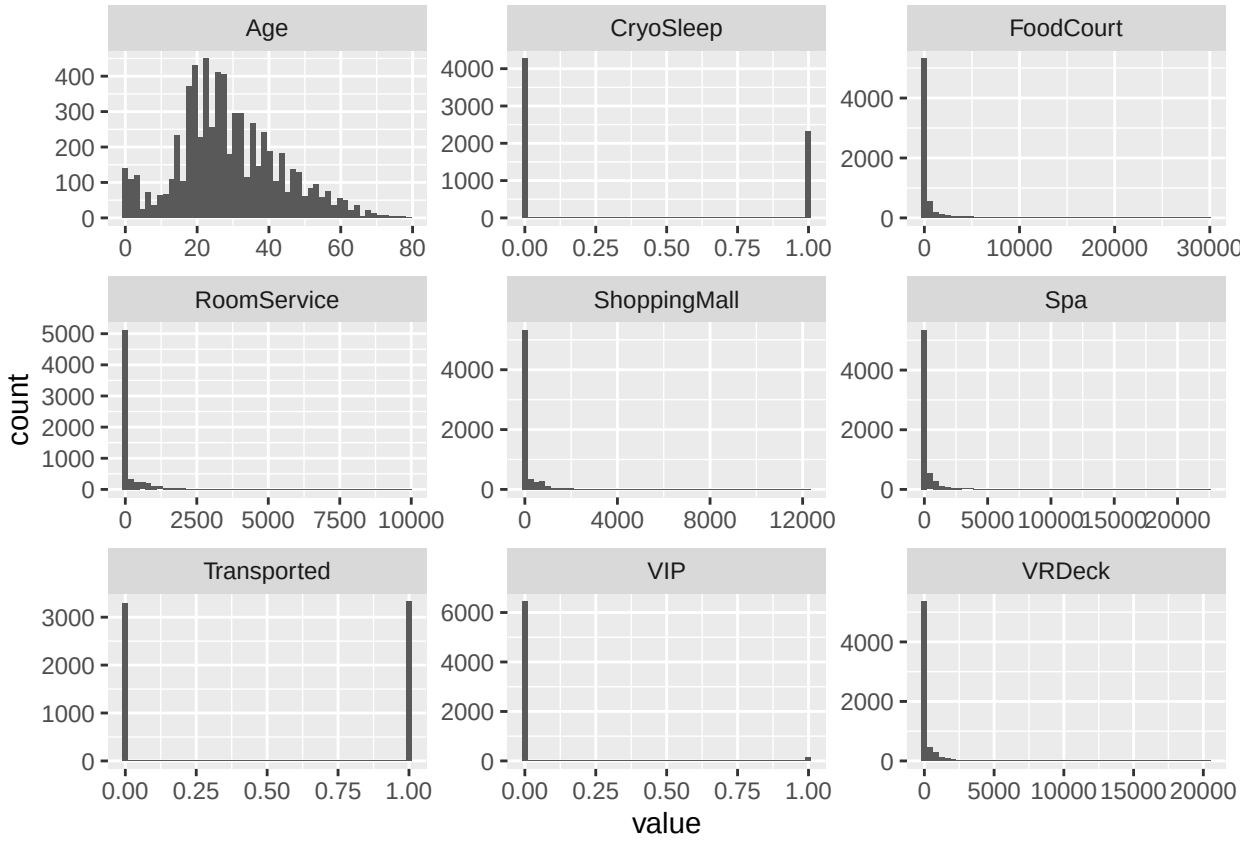
```
## [1] 8693   14
```

```r
#drop missing values
train = train[(!apply(train == '', 1, any)), ]
# split column and add new columns to df
train=train %>%
  drop_na()%>%
  separate('Cabin', c('Deck', 'Num', 'Side'), sep='/') %>%
  separate('PassengerId',c('group', 'people'), sep = '_')
#train
colSums(is.na(train) | train == '')
```

```
##        group       people   HomePlanet    CryoSleep         Deck          Num
##            0            0            0            0            0            0
##         Side  Destination          Age          VIP  RoomService    FoodCourt
##            0            0            0            0            0            0
## ShoppingMall          Spa       VRDeck         Name  Transported
##            0            0            0            0            0
```

```r
#replace 1 or 0 to VIP, CryoSleep, Transported
train$VIP=as.numeric(as.logical(train$VIP))
train$CryoSleep=as.numeric(as.logical(train$CryoSleep))
train$Transported=as.numeric(as.logical(train$Transported))
```

```r
#Convert wide to long
#Small Multiple Chart
p <- train %>%
    keep(is.numeric) %>%
    gather() %>%
    ggplot(aes(value)) +
    facet_wrap(~ key, scales = "free") +
    geom_histogram(bins = 50)
p
```




```
table(train$HomePlanet)
```

```
##
##  Earth Europa   Mars
##   3566   1673   1367
```

```
table(train$Destination)
```

```
##
##   55 Cancri e PSO J318.5-22   TRAPPIST-1e
##          1407           623          4576
```

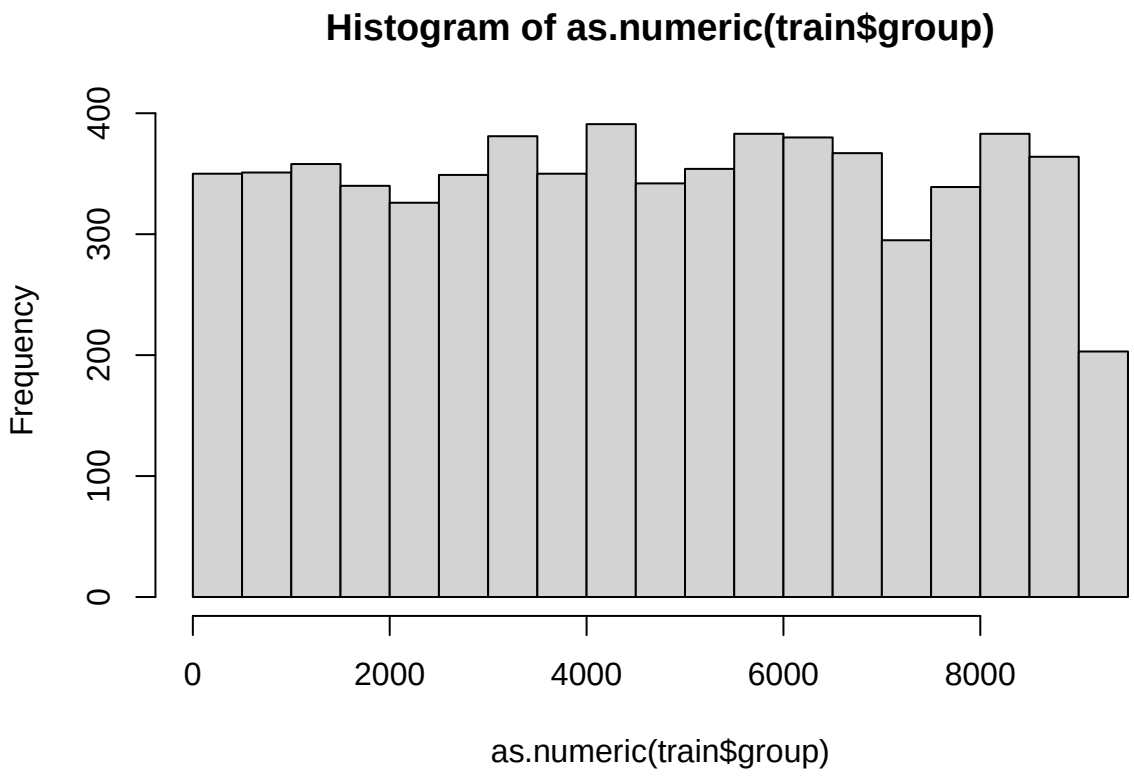
```
table(train$people)
```

```
##
##   01   02   03   04   05   06   07   08
## 4734 1062  432  178   99   60   29   12
```

```
table(train$Transported)
```

```
##
##    0    1
## 3279 3327
```

```
hist(as.numeric(train$group))
```



1. Viewing distribution: 2. Skewness

Classification Method: Cross-validation Using AUC # Use the glmnet package to fit Lasso & use AUC as the criteria to select the best tuning parameter. Followings are shown below: - Mutating data - Plot the cv results log(lambda) vs mse - Report the best lambda for Lasso using lambda.min or lambda.1se - What is the corresponding AUC? - Apply the best model to the testing data and report the prediction AUC with package ROCR - Does the model fits well?

```
library(glmnet)
```

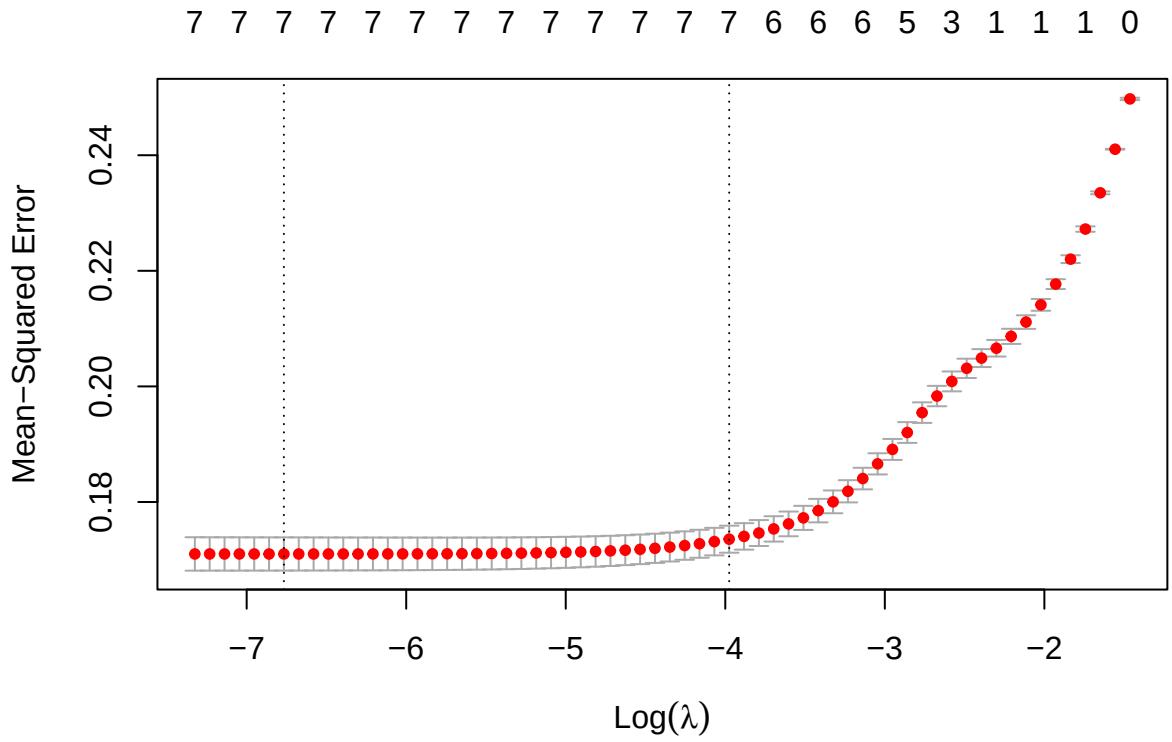
```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
## Loaded glmnet 4.1-4
```

```
#Mutating data frame
data= train %>%
  select('CryoSleep','Age':'VRDeck', 'Transported')
#Plot the cv results log(lambda) vs mse.
#alpha=1 is the lasso penalty,
lasso.fit = cv.glmnet(x = data.matrix(data[, -9]), y = data$Transported,
                      alpha=1 ,ty.measure = "auc")
plot(lasso.fit)
```




```
#We can view the selected lambda's and the corresponding coefficients
coef(lasso.fit, s = "lambda.min")
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                         s1
## (Intercept)   3.889683e-01
## CryoSleep     4.573091e-01
## Age          -1.137125e-03
## VIP           .
## RoomService  -1.064900e-04
## FoodCourt     6.328830e-05
## ShoppingMall  1.084045e-04
## Spa          -6.588644e-05
## VRDeck       -6.314614e-05
```

```r
#lambda.min is the value of lambda that gives minimum mean cross-validated error
lasso.fit$lambda.min
```

```
## [1] 0.001151662
```

```r
# lambda.lse, which gives the most regularized model such that error is within one standard error of th
lasso.fit$lambda.1se
```

```
## [1] 0.01876921
```

```r
lassopred = predict(lasso.fit, as.matrix(data[, -9]), s = "lambda.min")
library(ROCR)
roc <- prediction(lassopred, data$Transported)
# The prediction AUC
performance(roc, measure = "auc")@y.values[[1]]
```

```
## [1] 0.8344001
```

The prediction AUC is decent, which is around 80%. It might be better than random guess. Running this multiple times, the result is a bit different.

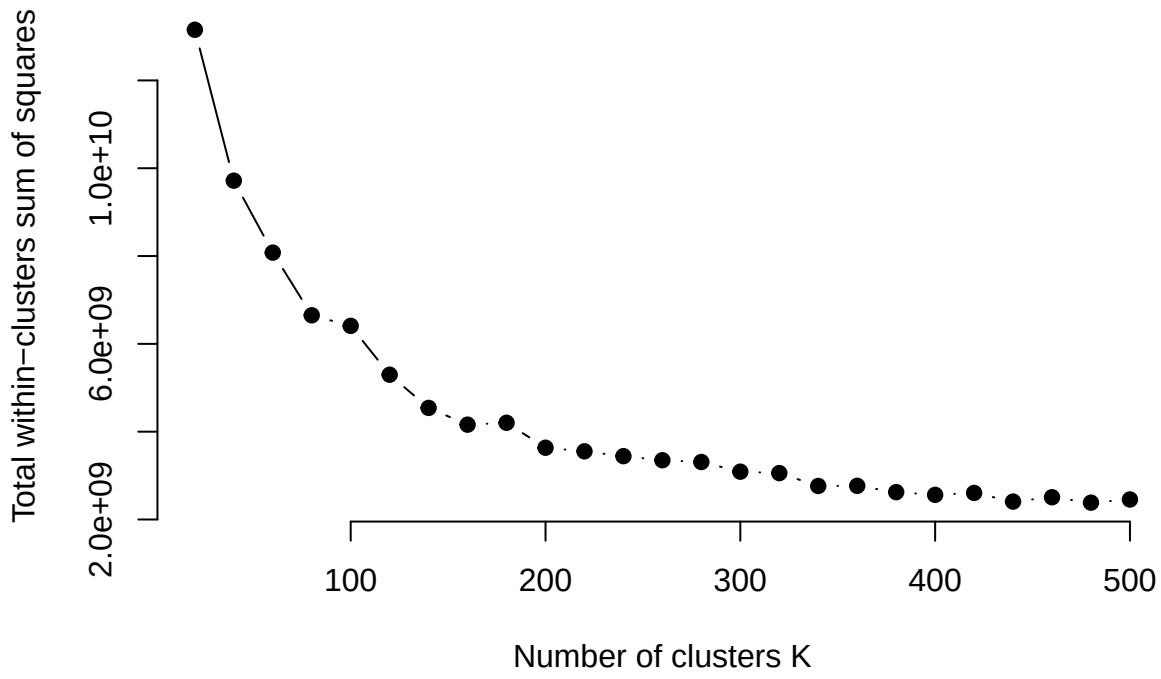
Perform at least three clustering algorithms to the training data.

K-means

```r
train$Num = as.numeric(train$Num)
train$group = as.numeric(train$group)
train$people = as.numeric(train$people)
train$HomePlanet = as.numeric(as.factor(train$HomePlanet))
train$Deck = as.numeric(as.factor(train$Deck))
train$Side = as.numeric(as.factor(train$Side))
train$Destination = as.numeric(as.factor(train$Destination))
train = train[, -16]
```

```r
set.seed(1)

# function to compute total within-cluster sum of square
wss <- function(k) {
  kmeans(train[, -16], k, nstart = 1, iter.max = 30)$tot.withinss
}

# Compute and plot wss for k = 1 to k = 15
k.values <- seq(20, 500, 20)

# extract wss for 2-15 clusters
wss_values <- map_dbl(k.values, wss)
```

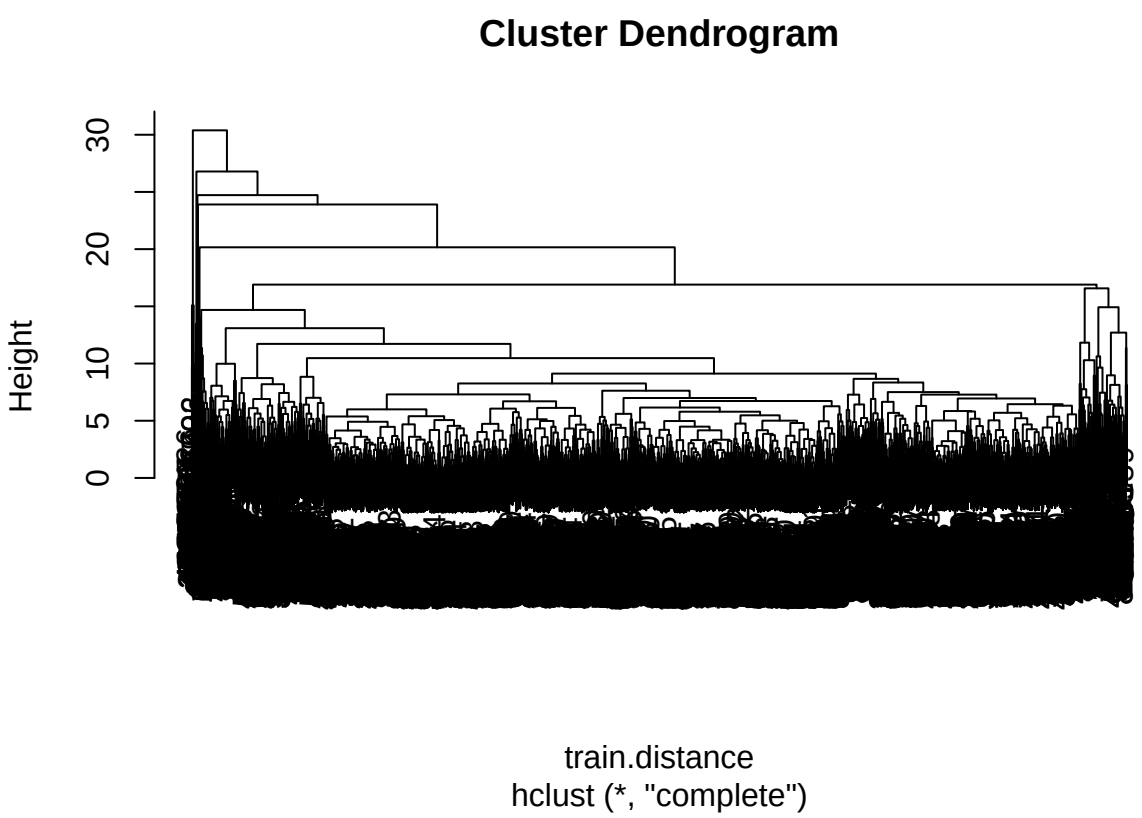
```
plot(k.values, wss_values,
       type="b", pch = 19, frame = FALSE,
       xlab="Number of clusters K",
       ylab="Total within-clusters sum of squares")
```



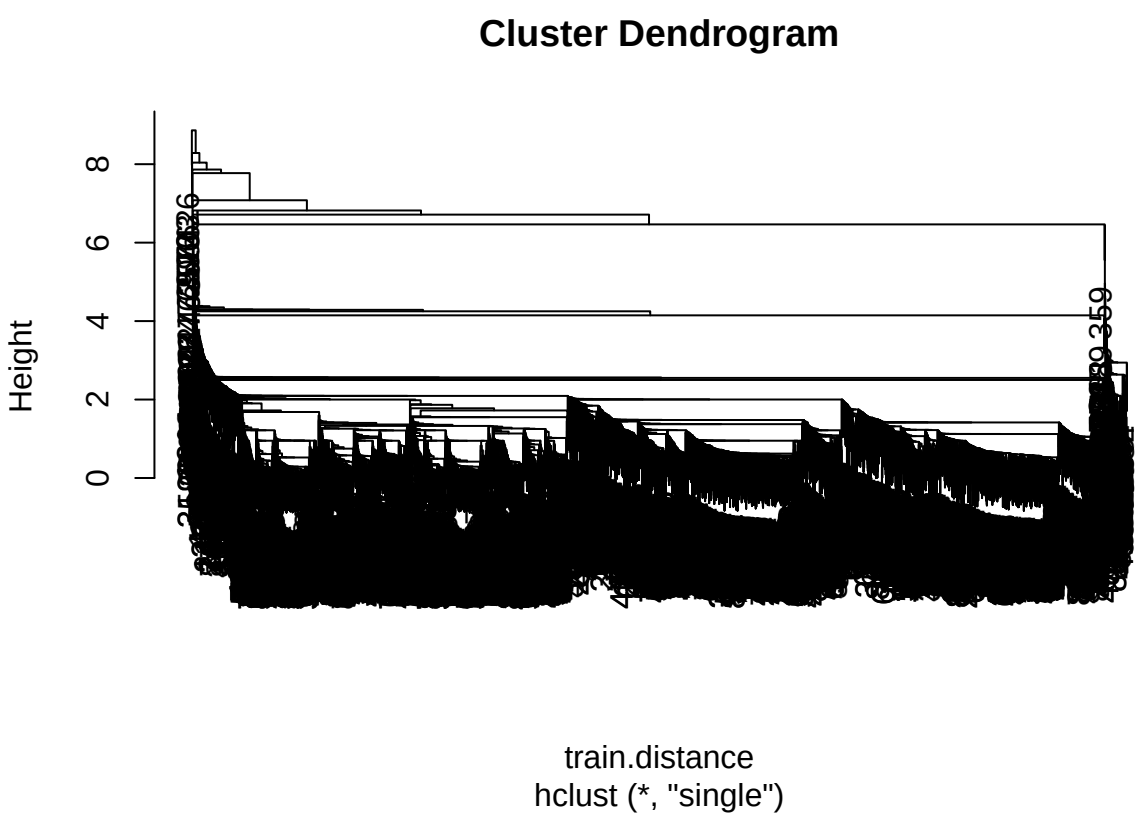
choose k = 80

Heirarchial Clustering

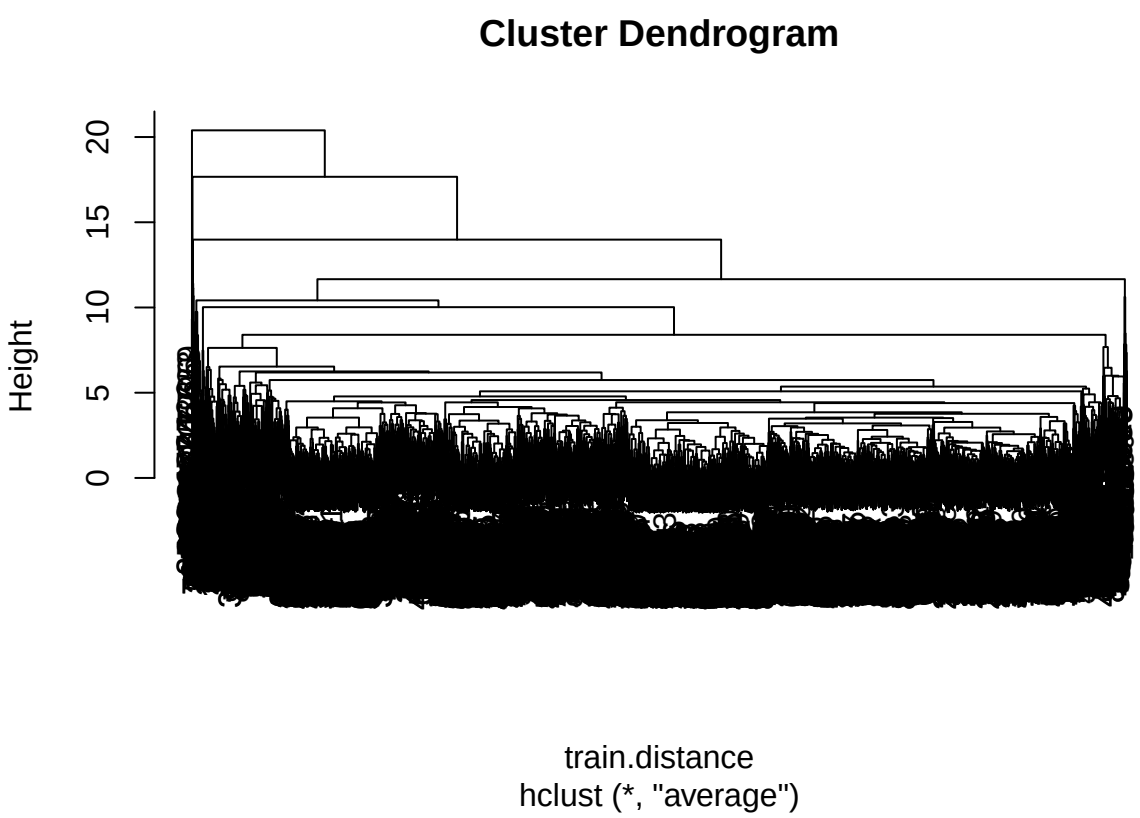
```
set.seed(1)

train.distance = dist(scale(train[, -16]))
hc_complete <- hclust(train.distance, method = "complete")
hc_single <- hclust(train.distance, method = "single")
hc_average <- hclust(train.distance, method = "average")

plot(hc_complete)
```




```
plot(hc_single)
```




```
plot(hc_average)
```



Spectral Clustering

```
set.seed(1)

library(ClusterR)
```

```
## Loading required package: gtools

##
## Attaching package: 'gtools'

## The following object is masked from 'package:glmnet':
##
##     na.replace
```

```
train.scale = scale(train[, -16])

opt_gmm = Optimal_Clusters_GMM(train.scale, max_clusters = 20, criterion = "BIC",

                               dist_mode = "maha_dist", seed_mode = "random_subset",

                               km_iter = 10, em_iter = 10, var_floor = 1e-10,
```

```
                                plot_data = T)
```

6 clusters

Supervised Learning

AdaBoost

```
library(adabag)
```

```
## Loading required package: rpart

## Loading required package: caret

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift

## Loading required package: foreach

##
## Attaching package: 'foreach'

## The following objects are masked from 'package:purrr':
##
##     accumulate, when

## Loading required package: doParallel

## Loading required package: iterators

## Loading required package: parallel
```

```
library(caret)
```

```
# train a model using our training data
train$Transported <- as.factor(train$Transported)
model_adaboost <- boosting(Transported~., data=train, boos=TRUE, mfinal=50)

#use model to make predictions on test data
pred_test = predict(model_adaboost, train)
```

```r
# Returns the prediction values of test data along with the confusion matrix
pred_test

1 - pred_test$error
```

```r
# train a model using our training data
train$Transported <- as.factor(train$Transported)
model_adaboost <- boosting(Transported~., data=train, boos=TRUE, mfinal=100)

#use model to make predictions on test data
pred_test = predict(model_adaboost, train)

1 - pred_test$error
```

```
## [1] 0.8268241
```

```r
# train a model using our training data
train$Transported <- as.factor(train$Transported)
model_adaboost <- boosting(Transported~., data=train, boos=TRUE, mfinal= 10)

#use model to make predictions on test data
pred_test = predict(model_adaboost, train)

1 - pred_test$error
```

```
## [1] 0.8125946
```

```r
# train a model using our training data
train$Transported <- as.factor(train$Transported)
model_adaboost <- boosting(Transported~., data=train, boos=TRUE, mfinal= 25)

#use model to make predictions on test data
pred_test = predict(model_adaboost, train)

1 - pred_test$error
```

```
## [1] 0.8271269
```

```r
# train a model using our training data
train$Transported <- as.factor(train$Transported)
model_adaboost <- boosting(Transported~., data=train, boos=TRUE, mfinal= 75)

#use model to make predictions on test data
pred_test = predict(model_adaboost, train)

1 - pred_test$error
```

```
## [1] 0.8256131
```