Project Assignment 4: Clustering

BD5

11/14/2019

### Data preperation

library(rpart)  
library(rpart.plot)  
library(caret)  
library(dummies)  
library(class)  
library(corrplot)  
library(ROCR)  
library(tidyverse)  
library(car)  
library(cluster)  
  
setwd("~/Documents/2019 Fall/BD2/HW4")  
#setwd("/Users/zhoujiawang/Desktop/Brandeis Life/BigData2/ABB Data Project/Clustering")  
Data1 <- read.csv('clean\_V3.csv', header=TRUE)  
Data1$if\_review\_score96 <- ifelse(Data1$review\_scores\_rating>=96,"t","f")  
Data1$sqrt.host\_total\_listings\_count <- sqrt(Data1$host\_total\_listings\_count)   
Data1$sqrt.accommodates <- sqrt(Data1$accommodates)  
Data1$sqrt.security\_deposit <- sqrt(Data1$security\_deposit)  
Data1$sqrt.number\_of\_reviews <- sqrt(Data1$number\_of\_reviews)  
Data1 <- na.omit(Data1)  
  
selected.var <- c(3,5,8,9,11:15,17:20,34,35,37,43,44:48) # consistant with hw3  
outlier <- c(1242,1479,1555,102,2218,509,2133,3353,5741,3628,5976,4479,4692,965) # outliers from hw3  
data.clean <- Data1[-outlier,selected.var]  
attach(data.clean)  
  
# save the cleaned data for future use  
# write.csv(data.clean,'clean\_V4.csv')

This project continues the third assignment’s topic: figuring out the variables that can improve a listing’s review score. We start with selecting variables, dropping outliers, as we did in hw3.

### Clustering: hierarchical clustering

#### Further clean data for clustering: data.cluster

All numerical variables are normalized, and categorical variables are converted into several dummy variables, as we did in hw3.

# normalize numeric variables  
numeric.ind = unlist(lapply(data.clean,is.numeric))  
numeric.nor <- as.data.frame(sapply(data.clean[,numeric.ind], scale))  
data.cluster <- cbind(numeric.nor,data.clean[,!numeric.ind])  
#data.cluster <- dummy.data.frame(data.cluster,names=c('host\_identity\_verified','neighbourhood\_cleansed','property\_type', 'room\_type','bed\_type','instant\_bookable','cancellation\_policy','require\_guest\_phone\_verification','if\_review\_score96'),sep="\_")  
#write.csv(data.cluster,'cluster.csv')  
#duplicated.var <- c(14,16,41,64,67,72,74,80,82) # remove extra dummies (e.g. keep True column and drop the False column)  
#data.cluster <- data.cluster[,-duplicated.var]

#### Determine the best distance computation method

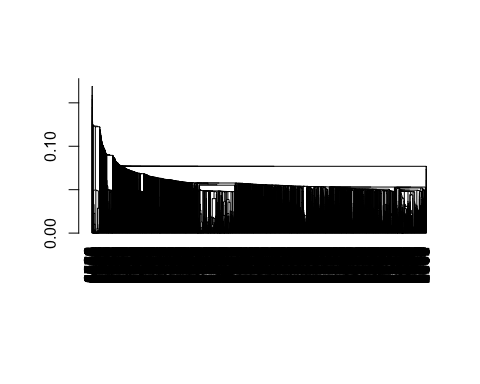
Since our data includes numerical and categorical variables, Gower’s similarity is the only method we can use.

# compute gower distance  
d.norm <- daisy(data.cluster[,1:21], metric = 'gower')

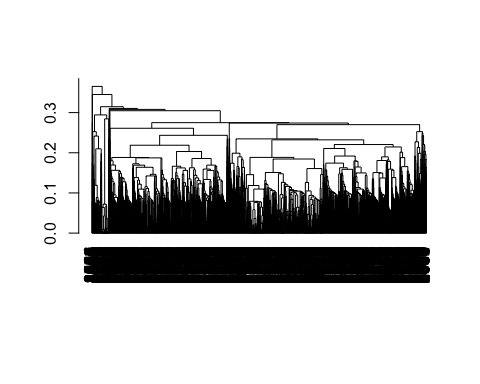
#### Determine the best cluster linking method

After selected distance calculating method, the next step is to determine the linkage method. Here we tried 6 different methods, and made the decision based on dendrograms and histograms - a clear dendrogram and evenly distributed clusters suggest a good cluster method. It turned out that “Ward.D” is the best, but “complete” also looks relatively acceptable. We kept both methods for further analysis.

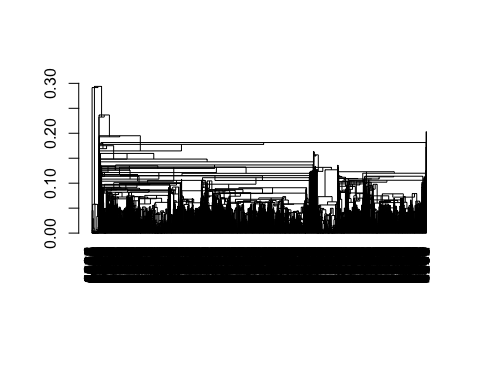
# determine the best method  
hc1 <- hclust(d.norm, method = "single")  
plot(hc1, hang = -1, ann = FALSE)



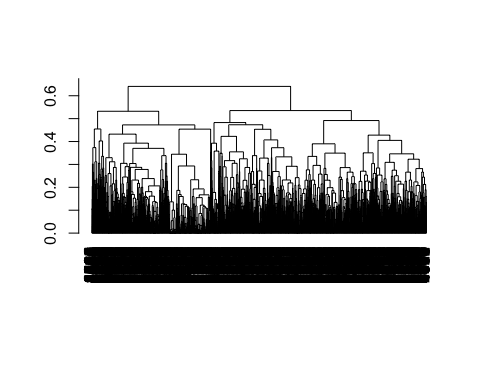
hc2 <- hclust(d.norm, method = "average")  
plot(hc2, hang = -1, ann = FALSE)



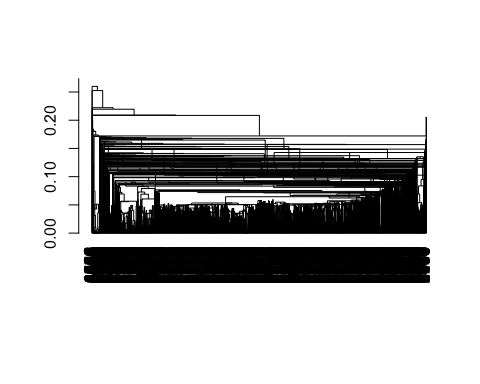
hc3 <- hclust(d.norm, method = "median")  
plot(hc3, hang = -1, ann = FALSE)



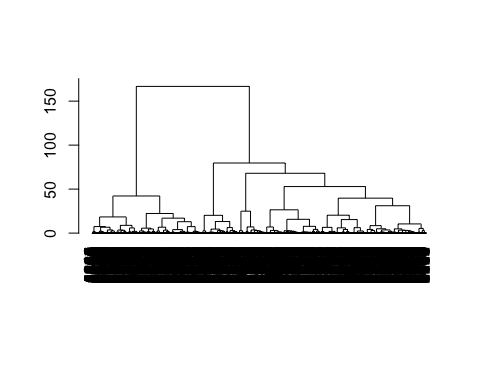
hc4 <- hclust(d.norm, method = "complete")  
plot(hc4, hang = -1, ann = FALSE)



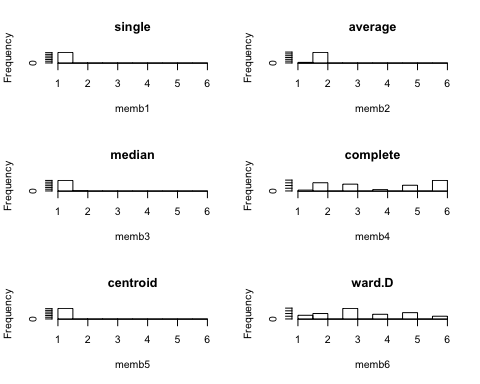
hc5 <- hclust(d.norm, method = "centroid")  
plot(hc5, hang = -1, ann = FALSE)



hc6 <- hclust(d.norm, method = "ward.D")  
plot(hc6, hang = -1, ann = FALSE)



# visually only ward.D is organized  
  
memb1 <- cutree(hc1, k = 6)  
memb2 <- cutree(hc2, k = 6)  
memb3 <- cutree(hc3, k = 6)  
memb4 <- cutree(hc4, k = 6)  
memb5 <- cutree(hc5, k = 6)  
memb6 <- cutree(hc6, k = 6)  
par(mfrow=c(3,2))   
hist(memb1,main="single")  
hist(memb2,main = "average")  
hist(memb3,main = "median")  
hist(memb4,main = "complete")  
hist(memb5,main = "centroid")  
hist(memb6,main = "ward.D")

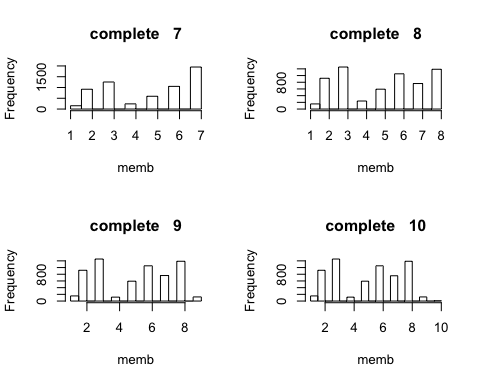
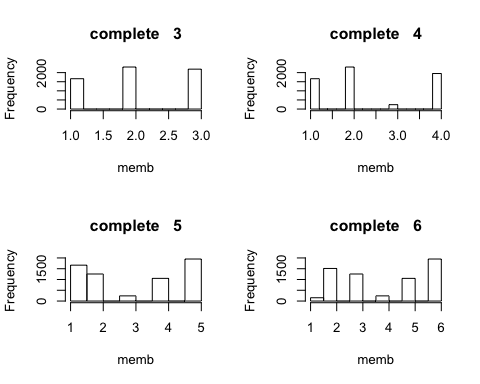
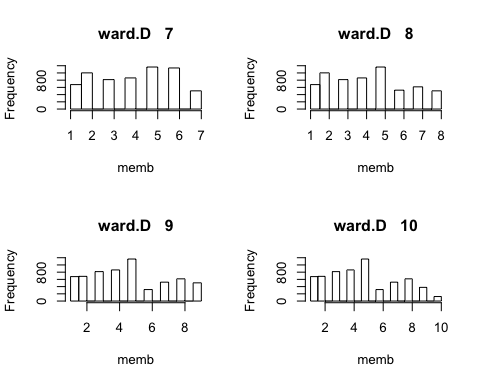
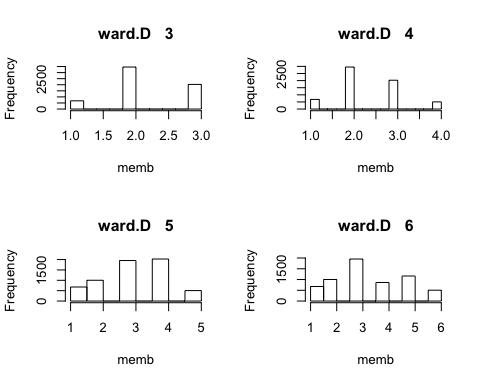


# again the histograms confirmed ward.D is the best

#### Determine the optimal k

Next we determined the optimal number of clusters. From the dendrograms we observed k = 3 to 10 are all acceptable, therefore we run a for loop to generate histograms. Again, evenly distributed clusters in a histogram show a good split. k = 7 and 8 are all visually good, so we kept 7 to prevent potential overfitting.

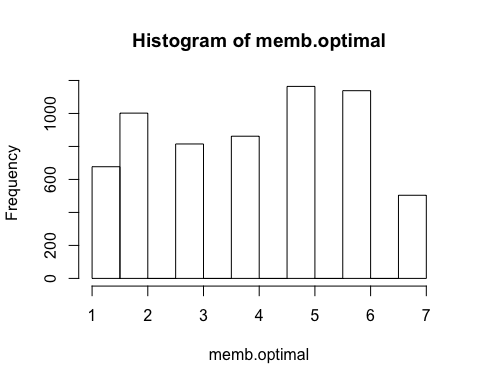
par(mfrow=c(2,2))   
d.norm <- daisy(data.cluster[,1:21], metric = 'gower')  
link.method <- c('ward.D', 'complete')  
for (m in link.method){  
 hc <- hclust(d.norm, method = m)  
 # visually 3-10 are acceptable  
 for (i in c(3:10)) {  
 memb <- cutree(hc, k = i)  
 main.title <- paste(m, ' ', i)  
 hist(memb, main=main.title)  
 }  
}



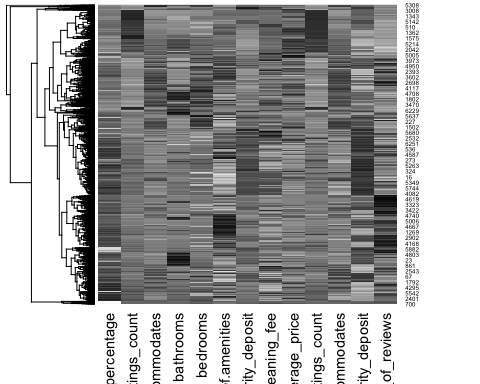
#### Display the best clustering model

link method = ward.D, k = 7 is the optimal. Due to the big dataset, heatmap is actually not so meaningful as it does not show all the data entries and the blocks are too small to look at.

# distance method=manhattan, cluster method=ward.D, k=8 is the optimal: most of the categories are equally distributed  
par(mfrow=c(1,1))  
d.norm <- daisy(data.cluster[,1:21], metric = 'gower')  
hc.optimal <- hclust(d.norm, method = 'ward.D')  
memb.optimal <- cutree(hc.optimal, k=7)   
#plot(hc.optimal)  
hist(memb.optimal)



# try heatmap, but not very meaningful due to too many listings  
heatmap(as.matrix(data.cluster[,1:13]), Colv = NA, hclustfun = hclust, col=rev(paste("grey",1:99,sep="")))



# heat map's input has to be numerical data, so I only show 1:13 columns which are numerical

#### Save the results

all commented out for the easiness of reading

# manually save the results to hc6.txt  
#cat(memb.optimal)  
# save results to csv  
#hc.pred <- read.table("hc.txt", quote="\"", comment.char="")  
#hc.pred <- t(hc.pred) # transpose  
#write.csv(hc.pred, 'hierCluster.csv')

### Clustering: KMeans

Start with organizing new data.

# normalize numeric variables  
numeric.ind = unlist(lapply(data.clean,is.numeric))  
numeric.nor <- as.data.frame(sapply(data.clean[,numeric.ind], scale))  
data.cluster <- cbind(numeric.nor,data.clean[,!numeric.ind])  
data.cluster <- dummy.data.frame(data.cluster,names=c('host\_identity\_verified','neighbourhood\_cleansed','property\_type', 'room\_type','bed\_type','instant\_bookable','cancellation\_policy','require\_guest\_phone\_verification','if\_review\_score96'),sep="\_")

## Warning in model.matrix.default(~x - 1, model.frame(~x - 1), contrasts =  
## FALSE): non-list contrasts argument ignored  
  
## Warning in model.matrix.default(~x - 1, model.frame(~x - 1), contrasts =  
## FALSE): non-list contrasts argument ignored  
  
## Warning in model.matrix.default(~x - 1, model.frame(~x - 1), contrasts =  
## FALSE): non-list contrasts argument ignored  
  
## Warning in model.matrix.default(~x - 1, model.frame(~x - 1), contrasts =  
## FALSE): non-list contrasts argument ignored  
  
## Warning in model.matrix.default(~x - 1, model.frame(~x - 1), contrasts =  
## FALSE): non-list contrasts argument ignored  
  
## Warning in model.matrix.default(~x - 1, model.frame(~x - 1), contrasts =  
## FALSE): non-list contrasts argument ignored  
  
## Warning in model.matrix.default(~x - 1, model.frame(~x - 1), contrasts =  
## FALSE): non-list contrasts argument ignored  
  
## Warning in model.matrix.default(~x - 1, model.frame(~x - 1), contrasts =  
## FALSE): non-list contrasts argument ignored  
  
## Warning in model.matrix.default(~x - 1, model.frame(~x - 1), contrasts =  
## FALSE): non-list contrasts argument ignored

#write.csv(data.cluster,'cluster.csv')  
duplicated.var <- c(14,16,41,64,67,72,74,80,82) # remove extra dummies (e.g. keep True column and drop the False column)  
data.cluster <- data.cluster[,-duplicated.var]

#### KMeans sample

set.seed(1)  
#from my side, if dummy variables are included, error reports: "NAs introduced by coercionError in do\_one(nmeth) : NA/NaN/Inf in foreign function call (arg 1)"  
km <- kmeans(data.cluster,7,20) # start from k=7, to be consistant with hierclustering  
# centroids  
# km$centers  
dist(km$centers)

## 1 2 3 4 5 6  
## 2 3.056658   
## 3 2.161347 3.418297   
## 4 5.335461 5.367213 4.989961   
## 5 7.664826 4.949078 7.668804 8.347658   
## 6 4.893927 4.193132 4.755684 6.137478 6.761486   
## 7 2.570958 3.588050 2.365134 5.368768 7.338377 5.032939

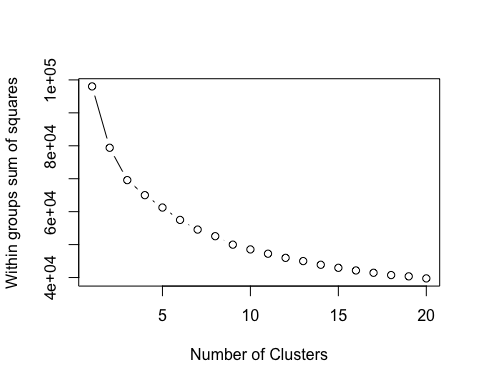
#### Determine the best k

Simply run a for loop to calculate WSS. From the graph below we observed that the WSS is improving less rapidly after k=7, therefore we use k=7 as the optimal k.

#to find the best K  
set.seed(1)  
  
wss <- 0  
  
# For 1 to 15 cluster centers  
for (i in 1:20) {  
 km.out <- kmeans(data.cluster, centers = i, nstart=20)  
 # Save total within sum of squares to wss variable  
 wss[i] <- km.out$tot.withinss  
}

## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations

plot(1:20, wss, type = "b",   
 xlab = "Number of Clusters",   
 ylab = "Within groups sum of squares")

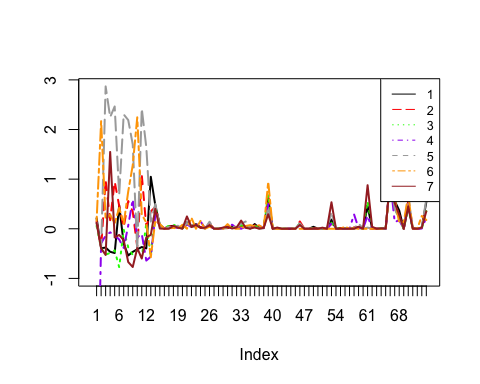


#K=7 and it is consistent with hierarchical cluster

#### Plot centroids - all variables

Each line represents a cluster. Note that for clarity, the y limit is set to be -1.

# plot an empty scatter plot. since 14-74 are dummies, the variances are not that big  
plot(c(0), xaxt = 'n', ylab = "", type = "l", xlim = c(0,74), ylim = c(-1, max(km$centers)))  
# label x-axes  
axis(1, at = c(1:74))  
# plot centroids   
for (i in c(1:7)) {  
 lines(km$centers[i,], lty = i, lwd = 2,   
 col = switch(i, "black", "red", "green", "purple","dark grey", "orange",'brown'))  
}  
# name clusters  
legend('topright', legend=c(1:7),  
 col=c("black", "red", "green", "purple","dark grey", "orange",'brown'),  
 lty=c(1,5,3,4,2,6,1), cex=0.8)

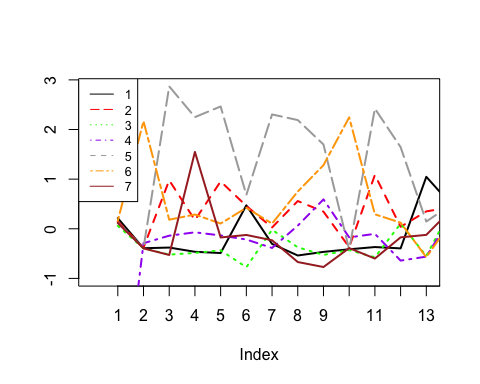


###Q3 Intrepretation

#### Plot centroids - look closer to 1-13th numerical variables

The 14th to 74th variables are all dummy variables and most of the clusters behave similarly after 13th. Therefore, we focused more on the 1st to 13th numerical variables.

plot(c(0), xaxt = 'n', ylab = "", type = "l", xlim = c(0,13), ylim = c(-1, max(km$centers)))  
# label x-axes  
axis(1, at = c(1:74))  
# plot centroids   
for (i in c(1:7)) {  
 lines(km$centers[i,], lty = i, lwd = 2,   
 col = switch(i, "black", "red", "green", "purple","dark grey", "orange",'brown'))  
}  
# name clusters  
legend('topleft', legend=c(1:7),  
 col=c("black", "red", "green", "purple","dark grey", "orange",'brown'),  
 lty=c(1,5,3,4,2,6,1), cex=0.8)

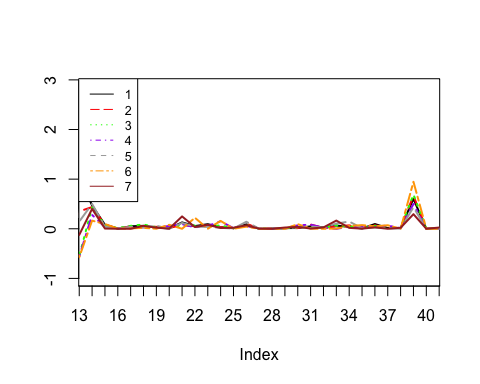


# print our x label names  
label <- names(data.cluster)[1:13]  
print(label)

## [1] "host\_response\_rate.percentage" "host\_total\_listings\_count"   
## [3] "accommodates" "bathrooms"   
## [5] "bedrooms" "Number.of.amenities"   
## [7] "security\_deposit" "cleaning\_fee"   
## [9] "average\_price" "sqrt.host\_total\_listings\_count"  
## [11] "sqrt.accommodates" "sqrt.security\_deposit"   
## [13] "sqrt.number\_of\_reviews"

#### Plot centroids - look closer to 14-40th numerical variables

plot(c(0), xaxt = 'n', ylab = "", type = "l", xlim = c(14,40), ylim = c(-1, max(km$centers)))  
# label x-axes  
axis(1, at = c(1:74))  
# plot centroids   
for (i in c(1:7)) {  
 lines(km$centers[i,], lty = i, lwd = 2,   
 col = switch(i, "black", "red", "green", "purple","dark grey", "orange",'brown'))  
}  
# name clusters  
legend('topleft', legend=c(1:7),  
 col=c("black", "red", "green", "purple","dark grey", "orange",'brown'),  
 lty=c(1,5,3,4,2,6,1), cex=0.8)

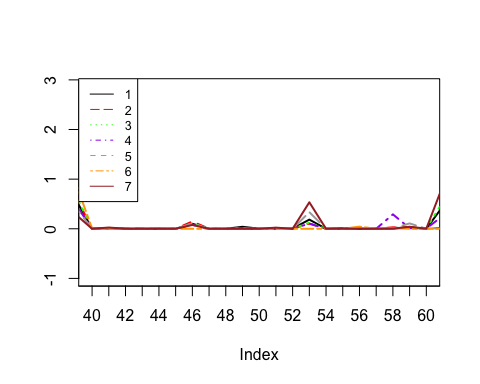


# print our x label names  
label <- names(data.cluster)[14:40]  
print(label)

## [1] "host\_identity\_verified\_t"   
## [2] "neighbourhood\_cleansed\_Back Bay"   
## [3] "neighbourhood\_cleansed\_Bay Village"   
## [4] "neighbourhood\_cleansed\_Beacon Hill"   
## [5] "neighbourhood\_cleansed\_Brighton"   
## [6] "neighbourhood\_cleansed\_Charlestown"   
## [7] "neighbourhood\_cleansed\_Chinatown"   
## [8] "neighbourhood\_cleansed\_Dorchester"   
## [9] "neighbourhood\_cleansed\_Downtown"   
## [10] "neighbourhood\_cleansed\_East Boston"   
## [11] "neighbourhood\_cleansed\_Fenway"   
## [12] "neighbourhood\_cleansed\_Hyde Park"   
## [13] "neighbourhood\_cleansed\_Jamaica Plain"   
## [14] "neighbourhood\_cleansed\_Leather District"   
## [15] "neighbourhood\_cleansed\_Longwood Medical Area"   
## [16] "neighbourhood\_cleansed\_Mattapan"   
## [17] "neighbourhood\_cleansed\_Mission Hill"   
## [18] "neighbourhood\_cleansed\_North End"   
## [19] "neighbourhood\_cleansed\_Roslindale"   
## [20] "neighbourhood\_cleansed\_Roxbury"   
## [21] "neighbourhood\_cleansed\_South Boston"   
## [22] "neighbourhood\_cleansed\_South Boston Waterfront"  
## [23] "neighbourhood\_cleansed\_South End"   
## [24] "neighbourhood\_cleansed\_West End"   
## [25] "neighbourhood\_cleansed\_West Roxbury"   
## [26] "property\_type\_Apartment"   
## [27] "property\_type\_Barn"

#### Plot centroids - look closer to 40-60th numerical variables

plot(c(0), xaxt = 'n', ylab = "", type = "l", xlim = c(40,60), ylim = c(-1, max(km$centers)))  
# label x-axes  
axis(1, at = c(1:74))  
# plot centroids   
for (i in c(1:7)) {  
 lines(km$centers[i,], lty = i, lwd = 2,   
 col = switch(i, "black", "red", "green", "purple","dark grey", "orange",'brown'))  
}  
# name clusters  
legend('topleft', legend=c(1:7),  
 col=c("black", "red", "green", "purple","dark grey", "orange",'brown'),  
 lty=c(1,5,3,4,2,6,1), cex=0.8)

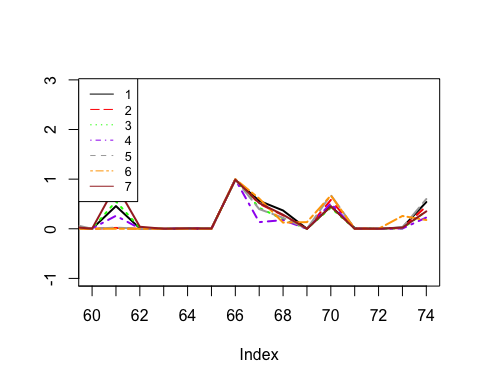


# print our x label names  
label <- names(data.cluster)[40:60]  
print(label)

## [1] "property\_type\_Barn"   
## [2] "property\_type\_Bed and breakfast"   
## [3] "property\_type\_Boat"   
## [4] "property\_type\_Boutique hotel"   
## [5] "property\_type\_Bungalow"   
## [6] "property\_type\_Casa particular (Cuba)"  
## [7] "property\_type\_Condominium"   
## [8] "property\_type\_Cottage"   
## [9] "property\_type\_Farm stay"   
## [10] "property\_type\_Guest suite"   
## [11] "property\_type\_Guesthouse"   
## [12] "property\_type\_Hostel"   
## [13] "property\_type\_Hotel"   
## [14] "property\_type\_House"   
## [15] "property\_type\_Houseboat"   
## [16] "property\_type\_Loft"   
## [17] "property\_type\_Other"   
## [18] "property\_type\_Resort"   
## [19] "property\_type\_Serviced apartment"   
## [20] "property\_type\_Townhouse"   
## [21] "property\_type\_Villa"

#### Plot centroids - look closer to 60-74th numerical variables

plot(c(0), xaxt = 'n', ylab = "", type = "l", xlim = c(60,74), ylim = c(-1, max(km$centers)))  
# label x-axes  
axis(1, at = c(1:74))  
# plot centroids   
for (i in c(1:7)) {  
 lines(km$centers[i,], lty = i, lwd = 2,   
 col = switch(i, "black", "red", "green", "purple","dark grey", "orange",'brown'))  
}  
# name clusters  
legend('topleft', legend=c(1:7),  
 col=c("black", "red", "green", "purple","dark grey", "orange",'brown'),  
 lty=c(1,5,3,4,2,6,1), cex=0.8)



# print our x label names  
label <- names(data.cluster)[60:74]  
print(label)

## [1] "property\_type\_Villa"   
## [2] "room\_type\_Private room"   
## [3] "room\_type\_Shared room"   
## [4] "bed\_type\_Couch"   
## [5] "bed\_type\_Futon"   
## [6] "bed\_type\_Pull-out Sofa"   
## [7] "bed\_type\_Real Bed"   
## [8] "instant\_bookable\_t"   
## [9] "cancellation\_policy\_moderate"   
## [10] "cancellation\_policy\_strict"   
## [11] "cancellation\_policy\_strict\_14\_with\_grace\_period"  
## [12] "cancellation\_policy\_super\_strict\_30"   
## [13] "cancellation\_policy\_super\_strict\_60"   
## [14] "require\_guest\_phone\_verification\_t"   
## [15] "if\_review\_score96\_t"

#####Interpretation:

First, we ranked all clusters according to the average price from low to high which is index #10 showing in the plot above and we have the following ranking:

Average price of Cluster # 1< #6<#4<#3<#5<#7<#2

Based on our observation from the visualization, cluster 1, 6, 4’s average prices are below average versus, cluster 3 is around average, cluster 4’s average house price is about one standard deviation higher than the total average price whereas cluster 7 and cluster 2 have average room/house prices almost 2 standard deviations from the mean price. In this case, we could start with categorizing these clusters into 4 groups by price: Cheap (houses belong to clusters: 1,6,4); economic (cluster 3); high end (cluster 4); and expansive (cluster 2 and 7). Furthermore, to have more specific categories within these four. We divide the index plot into four parts, so that we could have more clear look on characteristics of each cluster.

Cluster 2: It is not hard to find out the for houses/rooms that belong to cluster 2, they all have high accommodates, bathrooms bedrooms and cleaning fees. In this case, we could infer that houses which belong to cluster one, do not necessarily have the highest value per square feet, and it is expensive because it is very big in size. Categorical wise, cluster 2 contains more property type of houses and fewer apartments than other clusters which make sense since they generally have more rooms. Location wise, it more located in Leather District, cleansed South Boston and south Boston Waterfront. However, in general they have least number of properties that have review score more than 96.

Cluster 7: In contrast to cluster 1, houses from cluster 7 most likely because it’s luxurious and are operated by pretty experienced Airbnb hosts and located in expansive neighborhood of Boston area. From the index 4, 5, and 6 we could see that despite the high prices, these houses/rooms do not have that much bedroom, or bathroom, which indicate high price per square feet. In addition, the cluster also has very high number of total listings, which indicates that the owner of those Airbnb houses/rooms are experienced hosts. Location wise, it contains most properties in Hyde Park and north end, which is a very expansive neighbor nearby Boston downtown. Property type wise, this cluster more lofts and resorts than other clusters and most of the time these are entire houses so guests have more privacy. As the result, this cluster has a highest number of properties with more than 96 review score.

Cluster 5: Houses/rooms from this cluster has one biggest feature which is that the owners of these houses are highly experienced, owns 2 standard deviations higher than average number of listings compare with other owners and have most properties located in downtown. Besides, they have average number of rooms and amenities. However, they are able to charge guests higher prices most likely because of experience and location. In this case, these houses and rooms are able to charge relatively higher prices than houses/rooms that belong to “Cheap” categories. Categorical wise, properties in this cluster tend to be: not able to book instant, have more number of apartments than other clusters and have more number of properties in downtown than other clusters.

Cluster 3: In contrast to cluster5, houses/rooms that belong to cluster 3 have much less experienced owners and in addition, compared with houses with similar number of bedrooms and accommodates, these houses have less bathrooms. As the result, due to less experience and fewer bathrooms, these houses are charged in much lower prices than that of other houses. Categorical wise, it has more Condominium, and relative low possibilities to have review score more than 96.

Cluster 1: The average house price within cluster 1 is the lowest among all clusters. In addition, average number amenities are high relative to clusters in the same price range. Categorical wise, more of the properties tend to: be private rooms, instant bookable, have more Hostel and locate more in Drochester.

Cluster 6: Cluster 6 has second lowest average prices compare with other low cost “cheap” clusters. In addition, cluster 6 has low Number of amenities but high security deposit. As the result, cluster 6 is low price but not liquidity friendly. Categorical wise, it tend to: have more private room, have flexible cancellation policy and have pretty good reviews.

Cluster 4: This cluster has the highest average prices among total three “cheap” category. However, it has relatively higher average accommodates and bedrooms. Categorical wise, it tends to be: entire apartment or house, Futon bed type and pull out sofa, and have least cancellation\_policy\_strict\_14\_with\_grace\_period. In this case, it’s spacious and economic type of houses and rooms.

To Summarize all our findings, we would like to label all categories as the following:

Cluster 1: Best Value houses: Low price, high number of amenities, low cleaning fee and security deposit

Cluster 2: Spacious high-end houses: more bedrooms, accommodates and bathrooms; high price

Cluster 3: Bathroom shortage average price houses: reasonable price, and feature, but low number of bathrooms.

Cluster 4: Economic friendly spacious houses: relative more spacious than houses belong to “cheap” label.

Cluster 5: Experienced host houses: featured with hosts who have large number of listings with high cleaning fee, high deposit and high price

Cluster 6: Cheap flexible houses: low price, high deposit, but privacy and flexible cancellation policies.

Cluster 7: Luxurious best View houses: High price, locate in Hyde Park and north end which have good view and location, and run by experienced owners.

#### Save the clustering results

all commented out for the easiness of reading

# manually save the results to kMeans.txt  
#cat(km$cluster)  
  
# save results to csv  
#kmeans.pred <- read.table("kMeans.txt", quote="\"", comment.char="")  
#kmeans.pred <- t(kmeans.pred) # transpose  
#write.csv(kmeans.pred, 'kMeans.csv')

### Start to put back into model

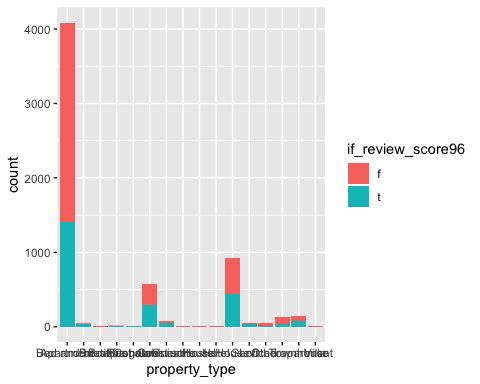
setwd("~/Documents/2019 Fall/BD2/HW4")  
Data1new <- read.csv('clean\_V4.csv', header=TRUE)  
Data1new$if\_review\_score96<-as.factor(Data1new$if\_review\_score96)  
Data1new$HierCluster<-as.factor(Data1new$HierCluster)  
Data1new$KMeansCluster<-as.factor(Data1new$KMeansCluster)  
Data1new<-Data1new[!Data1new$property\_type %in% c("Casa particular (Cuba)","Aparthotel","Barn","Cottage","Resort","Farm stay","Houseboat"),]  
Data1new<-Data1new[!Data1new$bed\_type %in% c("Couch"),]  
#delete a special property type Casa(cuba) seems like a outlier  
summary(Data1new$property\_type)

## Aparthotel Apartment Barn   
## 0 4087 0   
## Bed and breakfast Boat Boutique hotel   
## 44 6 13   
## Bungalow Casa particular (Cuba) Condominium   
## 5 0 575   
## Cottage Farm stay Guest suite   
## 0 0 80   
## Guesthouse Hostel Hotel   
## 6 10 11   
## House Houseboat Loft   
## 927 0 54   
## Other Resort Serviced apartment   
## 49 0 125   
## Townhouse Villa   
## 150 9

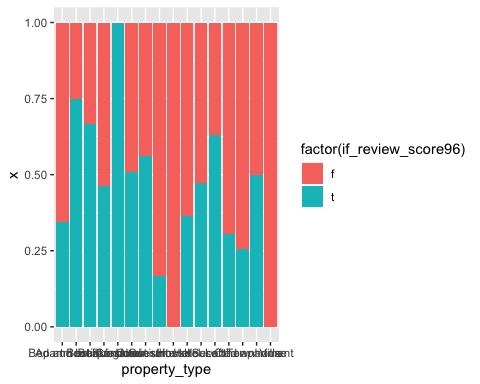
summary(Data1new$bed\_type)

## Airbed Couch Futon Pull-out Sofa Real Bed   
## 20 0 24 14 6093

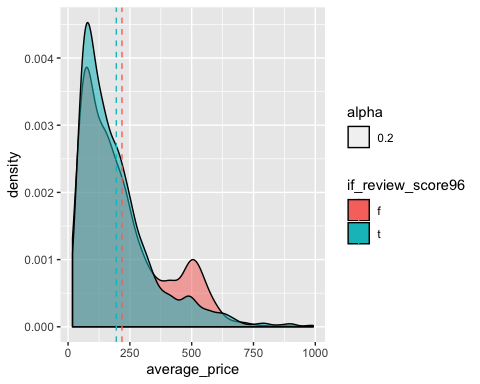
mu <- Data1new %>%   
 group\_by(if\_review\_score96) %>%  
 summarise(grp.mean = mean(average\_price))  
  
ggplot(Data1new,aes(x=property\_type))+geom\_bar(aes(fill=if\_review\_score96))



Data1new %>%  
 group\_by(property\_type,if\_review\_score96) %>%tally() %>%group\_by(property\_type) %>%  
 mutate(x = n / sum(n)) %>%ggplot() +  
 geom\_col(aes(x = property\_type,y = x,fill = factor(if\_review\_score96)), position = "stack")



ggplot(Data1new,aes(x=average\_price))+geom\_density(aes(fill=if\_review\_score96,alpha=0.2))+  
 geom\_vline(aes(xintercept = grp.mean, color = if\_review\_score96), data = mu, linetype = "dashed")



Some property types contain small amount of listings. Among the “common” property types, apartments have a lower review\_score>96 ratio, bed and breakfast more likely to earn a higher than 96 review score. A fat tail is found at price around 500, high score happens more often at price range from 100 to 200.

set.seed(7)  
Train <- sample(c(1:nrow(Data1new)), round(nrow(Data1new)\*0.6,0))  
Logtrain <- Data1new[ Train, ]  
Logtest <- Data1new[ -Train, ]

First, rerun the logistic model with our cluster factors in and detect with the new variable in, can we cast out more unnecessary variables? Will the Clustering variables invalid the original variables?

Logtrain$HierCluster<-as.factor(Logtrain$HierCluster)  
Logtrain$KMeansCluster<-as.factor(Logtrain$KMeansCluster)  
Logtest$HierCluster<-as.factor(Logtest$HierCluster)  
Logtest$KMeansCluster<-as.factor(Logtest$KMeansCluster)  
  
logfit1 <- glm(if\_review\_score96~., data=Logtrain, family='binomial')  
logfit2 <- glm(if\_review\_score96~., data=Logtrain[-c(23,24)], family='binomial')  
  
# make predictions  
probabilities1 <- predict(logfit1, newdata = Logtest, type='response')  
confusionMatrix(as.factor(ifelse(probabilities1 > 0.5,"t","f")), as.factor(Logtest$if\_review\_score96))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction f t  
## f 1232 474  
## t 293 461  
##   
## Accuracy : 0.6882   
## 95% CI : (0.6695, 0.7065)  
## No Information Rate : 0.6199   
## P-Value [Acc > NIR] : 8.911e-13   
##   
## Kappa : 0.3126   
##   
## Mcnemar's Test P-Value : 8.063e-11   
##   
## Sensitivity : 0.8079   
## Specificity : 0.4930   
## Pos Pred Value : 0.7222   
## Neg Pred Value : 0.6114   
## Prevalence : 0.6199   
## Detection Rate : 0.5008   
## Detection Prevalence : 0.6935   
## Balanced Accuracy : 0.6505   
##   
## 'Positive' Class : f   
##

probabilities2 <- predict(logfit2, newdata = Logtest, type='response')  
confusionMatrix(as.factor(ifelse(probabilities2 > 0.5,"t","f")), as.factor(Logtest$if\_review\_score96))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction f t  
## f 1231 469  
## t 294 466  
##   
## Accuracy : 0.6898   
## 95% CI : (0.6711, 0.7081)  
## No Information Rate : 0.6199   
## P-Value [Acc > NIR] : 2.567e-13   
##   
## Kappa : 0.3171   
##   
## Mcnemar's Test P-Value : 2.991e-10   
##   
## Sensitivity : 0.8072   
## Specificity : 0.4984   
## Pos Pred Value : 0.7241   
## Neg Pred Value : 0.6132   
## Prevalence : 0.6199   
## Detection Rate : 0.5004   
## Detection Prevalence : 0.6911   
## Balanced Accuracy : 0.6528   
##   
## 'Positive' Class : f   
##

anova(logfit1,logfit2,test ="Chisq")

## Analysis of Deviance Table  
##   
## Model 1: if\_review\_score96 ~ host\_response\_rate.percentage + host\_total\_listings\_count +   
## host\_identity\_verified + neighbourhood\_cleansed + property\_type +   
## room\_type + accommodates + bathrooms + bedrooms + bed\_type +   
## Number.of.amenities + security\_deposit + cleaning\_fee + instant\_bookable +   
## cancellation\_policy + require\_guest\_phone\_verification +   
## average\_price + sqrt.host\_total\_listings\_count + sqrt.accommodates +   
## sqrt.security\_deposit + sqrt.number\_of\_reviews + HierCluster +   
## KMeansCluster  
## Model 2: if\_review\_score96 ~ host\_response\_rate.percentage + host\_total\_listings\_count +   
## host\_identity\_verified + neighbourhood\_cleansed + property\_type +   
## room\_type + accommodates + bathrooms + bedrooms + bed\_type +   
## Number.of.amenities + security\_deposit + cleaning\_fee + instant\_bookable +   
## cancellation\_policy + require\_guest\_phone\_verification +   
## average\_price + sqrt.host\_total\_listings\_count + sqrt.accommodates +   
## sqrt.security\_deposit + sqrt.number\_of\_reviews  
## Resid. Df Resid. Dev Df Deviance Pr(>Chi)   
## 1 3613 4219.6   
## 2 3625 4240.6 -12 -21.043 0.04976 \*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Although the fact that our model accuracy improved 0.002 after adding the Clustering variables. Anova Chi-Square test shows a P value around 0.10, which indicates that our New Model and Original Model are not that statistically different. We need to explore this further.

#### Detect variable significance after adding the clustering factors

After summarized our second logistic regression, we decide to drop unsignificant variables such as number of bathrooms, number of bedrooms, bed type, host\_identity\_verified, room\_type, security\_deposit, cleaning\_fee.

Although it seems our newly added depended variable: the Kmeans and Hierarchical clusterings seem not significant as well, let’s leave them in the model.

# summarize the fit  
data.frame(summary(logfit1)$coefficients, odds = exp(coef(logfit1)))

## Estimate Std..Error  
## (Intercept) -3.632030e+00 1.132583e+00  
## host\_response\_rate.percentage 2.720083e-02 7.481208e-03  
## host\_total\_listings\_count 4.490540e-03 8.301706e-04  
## host\_identity\_verifiedt 8.398259e-02 9.120323e-02  
## neighbourhood\_cleansedBack Bay -8.129820e-02 2.334958e-01  
## neighbourhood\_cleansedBay Village -4.836393e-01 4.903157e-01  
## neighbourhood\_cleansedBeacon Hill 4.231221e-01 2.535548e-01  
## neighbourhood\_cleansedBrighton 1.178561e-01 2.196388e-01  
## neighbourhood\_cleansedCharlestown 2.851180e-01 2.839072e-01  
## neighbourhood\_cleansedChinatown 2.827434e-01 3.158759e-01  
## neighbourhood\_cleansedDorchester 3.699037e-01 1.998714e-01  
## neighbourhood\_cleansedDowntown -1.448728e-01 2.389565e-01  
## neighbourhood\_cleansedEast Boston 2.499611e-01 2.185660e-01  
## neighbourhood\_cleansedFenway 4.124792e-01 2.297199e-01  
## neighbourhood\_cleansedHyde Park 7.486124e-01 4.045808e-01  
## neighbourhood\_cleansedJamaica Plain 5.377750e-01 2.044644e-01  
## neighbourhood\_cleansedLeather District -8.195016e-01 1.275113e+00  
## neighbourhood\_cleansedLongwood Medical Area -1.341301e+01 4.941851e+02  
## neighbourhood\_cleansedMattapan 5.334299e-01 3.671288e-01  
## neighbourhood\_cleansedMission Hill 6.862042e-02 2.879834e-01  
## neighbourhood\_cleansedNorth End 6.051009e-01 2.597080e-01  
## neighbourhood\_cleansedRoslindale 6.521834e-01 3.108333e-01  
## neighbourhood\_cleansedRoxbury 2.681165e-01 2.188365e-01  
## neighbourhood\_cleansedSouth Boston 4.615387e-01 2.331718e-01  
## neighbourhood\_cleansedSouth Boston Waterfront 3.029998e-01 3.422737e-01  
## neighbourhood\_cleansedSouth End 4.475657e-01 2.199531e-01  
## neighbourhood\_cleansedWest End 1.031367e+00 3.160272e-01  
## neighbourhood\_cleansedWest Roxbury 2.618598e-01 3.747037e-01  
## property\_typeBed and breakfast 1.830463e+00 4.779491e-01  
## property\_typeBoat 2.231528e-01 1.078773e+00  
## property\_typeBoutique hotel 8.697736e-01 7.139492e-01  
## property\_typeBungalow 1.459026e+01 6.232615e+02  
## property\_typeCondominium 3.454281e-01 1.339433e-01  
## property\_typeGuest suite 1.318108e-01 3.083403e-01  
## property\_typeGuesthouse -1.270504e+00 1.179175e+00  
## property\_typeHostel -1.289870e+01 3.586235e+02  
## property\_typeHotel 6.903495e-01 1.048441e+00  
## property\_typeHouse 5.717683e-02 1.244423e-01  
## property\_typeLoft 7.306872e-01 4.255449e-01  
## property\_typeOther 4.759284e-01 4.941963e-01  
## property\_typeServiced apartment -7.419988e-01 3.191459e-01  
## property\_typeTownhouse 5.599896e-02 2.460462e-01  
## property\_typeVilla -1.331457e+01 2.788033e+02  
## room\_typePrivate room -1.577028e-01 2.432209e-01  
## room\_typeShared room -1.938354e-01 4.503490e-01  
## accommodates 1.492239e-01 9.163697e-02  
## bathrooms 1.512976e-02 9.860805e-02  
## bedrooms -9.654338e-02 7.412451e-02  
## bed\_typeFuton 1.464959e+00 8.641730e-01  
## bed\_typePull-out Sofa 1.002993e+00 9.999746e-01  
## bed\_typeReal Bed 1.397488e+00 6.930279e-01  
## Number.of.amenities 3.449351e-02 5.143757e-03  
## security\_deposit 4.466239e-04 2.806660e-04  
## cleaning\_fee 1.751965e-03 1.247452e-03  
## instant\_bookablet -2.265244e-01 1.410111e-01  
## cancellation\_policymoderate 1.721918e-01 1.286634e-01  
## cancellation\_policystrict 7.531919e-01 5.818272e-01  
## cancellation\_policystrict\_14\_with\_grace\_period -3.039606e-01 1.245762e-01  
## cancellation\_policysuper\_strict\_30 -1.040560e+00 5.905964e-01  
## cancellation\_policysuper\_strict\_60 -1.365788e+00 1.215446e+00  
## require\_guest\_phone\_verificationt -5.776131e-01 2.771527e-01  
## average\_price 1.401017e-03 5.372492e-04  
## sqrt.host\_total\_listings\_count -2.723452e-01 2.757604e-02  
## sqrt.accommodates -1.013855e+00 3.891439e-01  
## sqrt.security\_deposit -1.858153e-02 8.758711e-03  
## sqrt.number\_of\_reviews 5.987730e-02 1.239209e-02  
## HierCluster2 -2.003895e-01 2.303074e-01  
## HierCluster3 -3.392930e-01 2.198998e-01  
## HierCluster4 -2.964552e-02 2.357664e-01  
## HierCluster5 -2.409192e-01 2.483480e-01  
## HierCluster6 -3.321170e-01 2.446905e-01  
## HierCluster7 4.607724e-01 5.029071e-01  
## KMeansCluster2 2.112665e-01 3.805550e-01  
## KMeansCluster3 2.198545e-01 1.698018e-01  
## KMeansCluster4 8.424761e-01 5.461410e-01  
## KMeansCluster5 1.533435e+00 6.219357e-01  
## KMeansCluster6 -1.393236e-02 1.385471e-01  
## KMeansCluster7 1.551972e+00 4.391219e-01  
## z.value Pr...z..  
## (Intercept) -3.20685597 1.341942e-03  
## host\_response\_rate.percentage 3.63588717 2.770256e-04  
## host\_total\_listings\_count 5.40917705 6.331501e-08  
## host\_identity\_verifiedt 0.92082905 3.571397e-01  
## neighbourhood\_cleansedBack Bay -0.34817839 7.277062e-01  
## neighbourhood\_cleansedBay Village -0.98638348 3.239450e-01  
## neighbourhood\_cleansedBeacon Hill 1.66876007 9.516494e-02  
## neighbourhood\_cleansedBrighton 0.53659062 5.915504e-01  
## neighbourhood\_cleansedCharlestown 1.00426477 3.152510e-01  
## neighbourhood\_cleansedChinatown 0.89510924 3.707287e-01  
## neighbourhood\_cleansedDorchester 1.85070854 6.421150e-02  
## neighbourhood\_cleansedDowntown -0.60627286 5.443336e-01  
## neighbourhood\_cleansedEast Boston 1.14364162 2.527723e-01  
## neighbourhood\_cleansedFenway 1.79557485 7.256216e-02  
## neighbourhood\_cleansedHyde Park 1.85034079 6.426445e-02  
## neighbourhood\_cleansedJamaica Plain 2.63016377 8.534374e-03  
## neighbourhood\_cleansedLeather District -0.64268963 5.204255e-01  
## neighbourhood\_cleansedLongwood Medical Area -0.02714167 9.783467e-01  
## neighbourhood\_cleansedMattapan 1.45297749 1.462300e-01  
## neighbourhood\_cleansedMission Hill 0.23827912 8.116646e-01  
## neighbourhood\_cleansedNorth End 2.32992770 1.980997e-02  
## neighbourhood\_cleansedRoslindale 2.09817719 3.588950e-02  
## neighbourhood\_cleansedRoxbury 1.22519104 2.205032e-01  
## neighbourhood\_cleansedSouth Boston 1.97939316 4.777176e-02  
## neighbourhood\_cleansedSouth Boston Waterfront 0.88525589 3.760186e-01  
## neighbourhood\_cleansedSouth End 2.03482340 4.186865e-02  
## neighbourhood\_cleansedWest End 3.26353991 1.100297e-03  
## neighbourhood\_cleansedWest Roxbury 0.69884505 4.846489e-01  
## property\_typeBed and breakfast 3.82982768 1.282330e-04  
## property\_typeBoat 0.20685799 8.361208e-01  
## property\_typeBoutique hotel 1.21825695 2.231263e-01  
## property\_typeBungalow 0.02340953 9.813236e-01  
## property\_typeCondominium 2.57891184 9.911208e-03  
## property\_typeGuest suite 0.42748491 6.690262e-01  
## property\_typeGuesthouse -1.07745120 2.812787e-01  
## property\_typeHostel -0.03596725 9.713085e-01  
## property\_typeHotel 0.65845329 5.102469e-01  
## property\_typeHouse 0.45946452 6.459006e-01  
## property\_typeLoft 1.71706275 8.596771e-02  
## property\_typeOther 0.96303528 3.355298e-01  
## property\_typeServiced apartment -2.32495172 2.007454e-02  
## property\_typeTownhouse 0.22759527 8.199609e-01  
## property\_typeVilla -0.04775615 9.619106e-01  
## room\_typePrivate room -0.64839323 5.167306e-01  
## room\_typeShared room -0.43041161 6.668962e-01  
## accommodates 1.62842502 1.034348e-01  
## bathrooms 0.15343329 8.780566e-01  
## bedrooms -1.30244873 1.927630e-01  
## bed\_typeFuton 1.69521471 9.003470e-02  
## bed\_typePull-out Sofa 1.00301804 3.158522e-01  
## bed\_typeReal Bed 2.01649600 4.374814e-02  
## Number.of.amenities 6.70589772 2.001719e-11  
## security\_deposit 1.59130073 1.115419e-01  
## cleaning\_fee 1.40443520 1.601893e-01  
## instant\_bookablet -1.60642937 1.081796e-01  
## cancellation\_policymoderate 1.33831244 1.807946e-01  
## cancellation\_policystrict 1.29452845 1.954830e-01  
## cancellation\_policystrict\_14\_with\_grace\_period -2.43995732 1.468900e-02  
## cancellation\_policysuper\_strict\_30 -1.76187991 7.808959e-02  
## cancellation\_policysuper\_strict\_60 -1.12369333 2.611431e-01  
## require\_guest\_phone\_verificationt -2.08409650 3.715139e-02  
## average\_price 2.60776033 9.113674e-03  
## sqrt.host\_total\_listings\_count -9.87615465 5.282108e-23  
## sqrt.accommodates -2.60534687 9.178129e-03  
## sqrt.security\_deposit -2.12149113 3.388049e-02  
## sqrt.number\_of\_reviews 4.83189827 1.352374e-06  
## HierCluster2 -0.87009578 3.842481e-01  
## HierCluster3 -1.54294412 1.228443e-01  
## HierCluster4 -0.12574106 8.999369e-01  
## HierCluster5 -0.97008725 3.320030e-01  
## HierCluster6 -1.35729451 1.746877e-01  
## HierCluster7 0.91621767 3.595527e-01  
## KMeansCluster2 0.55515367 5.787896e-01  
## KMeansCluster3 1.29477124 1.953992e-01  
## KMeansCluster4 1.54259803 1.229283e-01  
## KMeansCluster5 2.46558451 1.367899e-02  
## KMeansCluster6 -0.10056045 9.198994e-01  
## KMeansCluster7 3.53426290 4.089141e-04  
## odds  
## (Intercept) 2.646241e-02  
## host\_response\_rate.percentage 1.027574e+00  
## host\_total\_listings\_count 1.004501e+00  
## host\_identity\_verifiedt 1.087610e+00  
## neighbourhood\_cleansedBack Bay 9.219187e-01  
## neighbourhood\_cleansedBay Village 6.165356e-01  
## neighbourhood\_cleansedBeacon Hill 1.526721e+00  
## neighbourhood\_cleansedBrighton 1.125082e+00  
## neighbourhood\_cleansedCharlestown 1.329919e+00  
## neighbourhood\_cleansedChinatown 1.326765e+00  
## neighbourhood\_cleansedDorchester 1.447595e+00  
## neighbourhood\_cleansedDowntown 8.651323e-01  
## neighbourhood\_cleansedEast Boston 1.283976e+00  
## neighbourhood\_cleansedFenway 1.510558e+00  
## neighbourhood\_cleansedHyde Park 2.114065e+00  
## neighbourhood\_cleansedJamaica Plain 1.712193e+00  
## neighbourhood\_cleansedLeather District 4.406512e-01  
## neighbourhood\_cleansedLongwood Medical Area 1.495560e-06  
## neighbourhood\_cleansedMattapan 1.704769e+00  
## neighbourhood\_cleansedMission Hill 1.071030e+00  
## neighbourhood\_cleansedNorth End 1.831437e+00  
## neighbourhood\_cleansedRoslindale 1.919728e+00  
## neighbourhood\_cleansedRoxbury 1.307499e+00  
## neighbourhood\_cleansedSouth Boston 1.586513e+00  
## neighbourhood\_cleansedSouth Boston Waterfront 1.353914e+00  
## neighbourhood\_cleansedSouth End 1.564499e+00  
## neighbourhood\_cleansedWest End 2.804899e+00  
## neighbourhood\_cleansedWest Roxbury 1.299344e+00  
## property\_typeBed and breakfast 6.236773e+00  
## property\_typeBoat 1.250012e+00  
## property\_typeBoutique hotel 2.386371e+00  
## property\_typeBungalow 2.170041e+06  
## property\_typeCondominium 1.412594e+00  
## property\_typeGuest suite 1.140892e+00  
## property\_typeGuesthouse 2.806902e-01  
## property\_typeHostel 2.501302e-06  
## property\_typeHotel 1.994412e+00  
## property\_typeHouse 1.058843e+00  
## property\_typeLoft 2.076507e+00  
## property\_typeOther 1.609508e+00  
## property\_typeServiced apartment 4.761612e-01  
## property\_typeTownhouse 1.057597e+00  
## property\_typeVilla 1.650270e-06  
## room\_typePrivate room 8.541036e-01  
## room\_typeShared room 8.237935e-01  
## accommodates 1.160933e+00  
## bathrooms 1.015245e+00  
## bedrooms 9.079705e-01  
## bed\_typeFuton 4.327365e+00  
## bed\_typePull-out Sofa 2.726429e+00  
## bed\_typeReal Bed 4.045026e+00  
## Number.of.amenities 1.035095e+00  
## security\_deposit 1.000447e+00  
## cleaning\_fee 1.001754e+00  
## instant\_bookablet 7.972999e-01  
## cancellation\_policymoderate 1.187906e+00  
## cancellation\_policystrict 2.123768e+00  
## cancellation\_policystrict\_14\_with\_grace\_period 7.378900e-01  
## cancellation\_policysuper\_strict\_30 3.532568e-01  
## cancellation\_policysuper\_strict\_60 2.551795e-01  
## require\_guest\_phone\_verificationt 5.612364e-01  
## average\_price 1.001402e+00  
## sqrt.host\_total\_listings\_count 7.615913e-01  
## sqrt.accommodates 3.628176e-01  
## sqrt.security\_deposit 9.815900e-01  
## sqrt.number\_of\_reviews 1.061706e+00  
## HierCluster2 8.184119e-01  
## HierCluster3 7.122737e-01  
## HierCluster4 9.707896e-01  
## HierCluster5 7.859051e-01  
## HierCluster6 7.174034e-01  
## HierCluster7 1.585298e+00  
## KMeansCluster2 1.235241e+00  
## KMeansCluster3 1.245895e+00  
## KMeansCluster4 2.322110e+00  
## KMeansCluster5 4.634068e+00  
## KMeansCluster6 9.861642e-01  
## KMeansCluster7 4.720772e+00

summary(logfit1)

##   
## Call:  
## glm(formula = if\_review\_score96 ~ ., family = "binomial", data = Logtrain)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.2124 -0.9176 -0.5538 1.0169 2.5764   
##   
## Coefficients:  
## Estimate Std. Error  
## (Intercept) -3.632e+00 1.133e+00  
## host\_response\_rate.percentage 2.720e-02 7.481e-03  
## host\_total\_listings\_count 4.490e-03 8.302e-04  
## host\_identity\_verifiedt 8.398e-02 9.120e-02  
## neighbourhood\_cleansedBack Bay -8.130e-02 2.335e-01  
## neighbourhood\_cleansedBay Village -4.836e-01 4.903e-01  
## neighbourhood\_cleansedBeacon Hill 4.231e-01 2.536e-01  
## neighbourhood\_cleansedBrighton 1.179e-01 2.196e-01  
## neighbourhood\_cleansedCharlestown 2.851e-01 2.839e-01  
## neighbourhood\_cleansedChinatown 2.827e-01 3.159e-01  
## neighbourhood\_cleansedDorchester 3.699e-01 1.999e-01  
## neighbourhood\_cleansedDowntown -1.449e-01 2.390e-01  
## neighbourhood\_cleansedEast Boston 2.500e-01 2.186e-01  
## neighbourhood\_cleansedFenway 4.125e-01 2.297e-01  
## neighbourhood\_cleansedHyde Park 7.486e-01 4.046e-01  
## neighbourhood\_cleansedJamaica Plain 5.378e-01 2.045e-01  
## neighbourhood\_cleansedLeather District -8.195e-01 1.275e+00  
## neighbourhood\_cleansedLongwood Medical Area -1.341e+01 4.942e+02  
## neighbourhood\_cleansedMattapan 5.334e-01 3.671e-01  
## neighbourhood\_cleansedMission Hill 6.862e-02 2.880e-01  
## neighbourhood\_cleansedNorth End 6.051e-01 2.597e-01  
## neighbourhood\_cleansedRoslindale 6.522e-01 3.108e-01  
## neighbourhood\_cleansedRoxbury 2.681e-01 2.188e-01  
## neighbourhood\_cleansedSouth Boston 4.615e-01 2.332e-01  
## neighbourhood\_cleansedSouth Boston Waterfront 3.030e-01 3.423e-01  
## neighbourhood\_cleansedSouth End 4.476e-01 2.200e-01  
## neighbourhood\_cleansedWest End 1.031e+00 3.160e-01  
## neighbourhood\_cleansedWest Roxbury 2.619e-01 3.747e-01  
## property\_typeBed and breakfast 1.830e+00 4.779e-01  
## property\_typeBoat 2.232e-01 1.079e+00  
## property\_typeBoutique hotel 8.698e-01 7.139e-01  
## property\_typeBungalow 1.459e+01 6.233e+02  
## property\_typeCondominium 3.454e-01 1.339e-01  
## property\_typeGuest suite 1.318e-01 3.083e-01  
## property\_typeGuesthouse -1.271e+00 1.179e+00  
## property\_typeHostel -1.290e+01 3.586e+02  
## property\_typeHotel 6.903e-01 1.048e+00  
## property\_typeHouse 5.718e-02 1.244e-01  
## property\_typeLoft 7.307e-01 4.255e-01  
## property\_typeOther 4.759e-01 4.942e-01  
## property\_typeServiced apartment -7.420e-01 3.191e-01  
## property\_typeTownhouse 5.600e-02 2.460e-01  
## property\_typeVilla -1.331e+01 2.788e+02  
## room\_typePrivate room -1.577e-01 2.432e-01  
## room\_typeShared room -1.938e-01 4.503e-01  
## accommodates 1.492e-01 9.164e-02  
## bathrooms 1.513e-02 9.861e-02  
## bedrooms -9.654e-02 7.412e-02  
## bed\_typeFuton 1.465e+00 8.642e-01  
## bed\_typePull-out Sofa 1.003e+00 1.000e+00  
## bed\_typeReal Bed 1.397e+00 6.930e-01  
## Number.of.amenities 3.449e-02 5.144e-03  
## security\_deposit 4.466e-04 2.807e-04  
## cleaning\_fee 1.752e-03 1.247e-03  
## instant\_bookablet -2.265e-01 1.410e-01  
## cancellation\_policymoderate 1.722e-01 1.287e-01  
## cancellation\_policystrict 7.532e-01 5.818e-01  
## cancellation\_policystrict\_14\_with\_grace\_period -3.040e-01 1.246e-01  
## cancellation\_policysuper\_strict\_30 -1.041e+00 5.906e-01  
## cancellation\_policysuper\_strict\_60 -1.366e+00 1.215e+00  
## require\_guest\_phone\_verificationt -5.776e-01 2.772e-01  
## average\_price 1.401e-03 5.372e-04  
## sqrt.host\_total\_listings\_count -2.723e-01 2.758e-02  
## sqrt.accommodates -1.014e+00 3.891e-01  
## sqrt.security\_deposit -1.858e-02 8.759e-03  
## sqrt.number\_of\_reviews 5.988e-02 1.239e-02  
## HierCluster2 -2.004e-01 2.303e-01  
## HierCluster3 -3.393e-01 2.199e-01  
## HierCluster4 -2.965e-02 2.358e-01  
## HierCluster5 -2.409e-01 2.483e-01  
## HierCluster6 -3.321e-01 2.447e-01  
## HierCluster7 4.608e-01 5.029e-01  
## KMeansCluster2 2.113e-01 3.806e-01  
## KMeansCluster3 2.199e-01 1.698e-01  
## KMeansCluster4 8.425e-01 5.461e-01  
## KMeansCluster5 1.533e+00 6.219e-01  
## KMeansCluster6 -1.393e-02 1.385e-01  
## KMeansCluster7 1.552e+00 4.391e-01  
## z value Pr(>|z|)   
## (Intercept) -3.207 0.001342 \*\*   
## host\_response\_rate.percentage 3.636 0.000277 \*\*\*  
## host\_total\_listings\_count 5.409 6.33e-08 \*\*\*  
## host\_identity\_verifiedt 0.921 0.357140   
## neighbourhood\_cleansedBack Bay -0.348 0.727706   
## neighbourhood\_cleansedBay Village -0.986 0.323945   
## neighbourhood\_cleansedBeacon Hill 1.669 0.095165 .   
## neighbourhood\_cleansedBrighton 0.537 0.591550   
## neighbourhood\_cleansedCharlestown 1.004 0.315251   
## neighbourhood\_cleansedChinatown 0.895 0.370729   
## neighbourhood\_cleansedDorchester 1.851 0.064211 .   
## neighbourhood\_cleansedDowntown -0.606 0.544334   
## neighbourhood\_cleansedEast Boston 1.144 0.252772   
## neighbourhood\_cleansedFenway 1.796 0.072562 .   
## neighbourhood\_cleansedHyde Park 1.850 0.064264 .   
## neighbourhood\_cleansedJamaica Plain 2.630 0.008534 \*\*   
## neighbourhood\_cleansedLeather District -0.643 0.520426   
## neighbourhood\_cleansedLongwood Medical Area -0.027 0.978347   
## neighbourhood\_cleansedMattapan 1.453 0.146230   
## neighbourhood\_cleansedMission Hill 0.238 0.811665   
## neighbourhood\_cleansedNorth End 2.330 0.019810 \*   
## neighbourhood\_cleansedRoslindale 2.098 0.035889 \*   
## neighbourhood\_cleansedRoxbury 1.225 0.220503   
## neighbourhood\_cleansedSouth Boston 1.979 0.047772 \*   
## neighbourhood\_cleansedSouth Boston Waterfront 0.885 0.376019   
## neighbourhood\_cleansedSouth End 2.035 0.041869 \*   
## neighbourhood\_cleansedWest End 3.264 0.001100 \*\*   
## neighbourhood\_cleansedWest Roxbury 0.699 0.484649   
## property\_typeBed and breakfast 3.830 0.000128 \*\*\*  
## property\_typeBoat 0.207 0.836121   
## property\_typeBoutique hotel 1.218 0.223126   
## property\_typeBungalow 0.023 0.981324   
## property\_typeCondominium 2.579 0.009911 \*\*   
## property\_typeGuest suite 0.427 0.669026   
## property\_typeGuesthouse -1.077 0.281279   
## property\_typeHostel -0.036 0.971308   
## property\_typeHotel 0.658 0.510247   
## property\_typeHouse 0.459 0.645901   
## property\_typeLoft 1.717 0.085968 .   
## property\_typeOther 0.963 0.335530   
## property\_typeServiced apartment -2.325 0.020075 \*   
## property\_typeTownhouse 0.228 0.819961   
## property\_typeVilla -0.048 0.961911   
## room\_typePrivate room -0.648 0.516731   
## room\_typeShared room -0.430 0.666896   
## accommodates 1.628 0.103435   
## bathrooms 0.153 0.878057   
## bedrooms -1.302 0.192763   
## bed\_typeFuton 1.695 0.090035 .   
## bed\_typePull-out Sofa 1.003 0.315852   
## bed\_typeReal Bed 2.016 0.043748 \*   
## Number.of.amenities 6.706 2.00e-11 \*\*\*  
## security\_deposit 1.591 0.111542   
## cleaning\_fee 1.404 0.160189   
## instant\_bookablet -1.606 0.108180   
## cancellation\_policymoderate 1.338 0.180795   
## cancellation\_policystrict 1.295 0.195483   
## cancellation\_policystrict\_14\_with\_grace\_period -2.440 0.014689 \*   
## cancellation\_policysuper\_strict\_30 -1.762 0.078090 .   
## cancellation\_policysuper\_strict\_60 -1.124 0.261143   
## require\_guest\_phone\_verificationt -2.084 0.037151 \*   
## average\_price 2.608 0.009114 \*\*   
## sqrt.host\_total\_listings\_count -9.876 < 2e-16 \*\*\*  
## sqrt.accommodates -2.605 0.009178 \*\*   
## sqrt.security\_deposit -2.121 0.033880 \*   
## sqrt.number\_of\_reviews 4.832 1.35e-06 \*\*\*  
## HierCluster2 -0.870 0.384248   
## HierCluster3 -1.543 0.122844   
## HierCluster4 -0.126 0.899937   
## HierCluster5 -0.970 0.332003   
## HierCluster6 -1.357 0.174688   
## HierCluster7 0.916 0.359553   
## KMeansCluster2 0.555 0.578790   
## KMeansCluster3 1.295 0.195399   
## KMeansCluster4 1.543 0.122928   
## KMeansCluster5 2.466 0.013679 \*   
## KMeansCluster6 -0.101 0.919899   
## KMeansCluster7 3.534 0.000409 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 4951.2 on 3690 degrees of freedom  
## Residual deviance: 4219.6 on 3613 degrees of freedom  
## AIC: 4375.6  
##   
## Number of Fisher Scoring iterations: 13

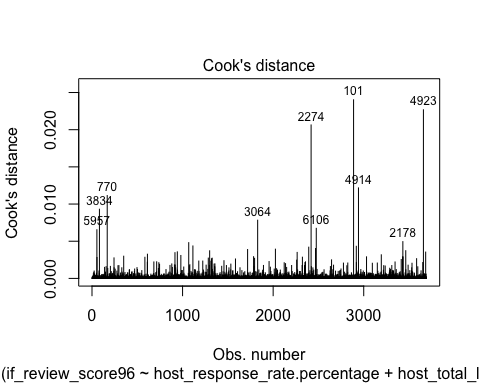
Now reduce the variables and rerun it again.

logfit3 <- glm(if\_review\_score96 ~ host\_response\_rate.percentage + host\_total\_listings\_count +  
 neighbourhood\_cleansed + property\_type + accommodates + Number.of.amenities +   
 instant\_bookable + cancellation\_policy + require\_guest\_phone\_verification + average\_price+  
 sqrt.host\_total\_listings\_count + sqrt.accommodates + sqrt.security\_deposit +  
 sqrt.number\_of\_reviews + HierCluster +KMeansCluster,   
 data=Logtrain, family='binomial')  
  
probabilities3 <- predict(logfit3, newdata = Logtest, type='response')  
confusionMatrix(as.factor(ifelse(probabilities3 > 0.5,"t","f")), as.factor(Logtest$if\_review\_score96))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction f t  
## f 1222 480  
## t 303 455  
##   
## Accuracy : 0.6817   
## 95% CI : (0.6629, 0.7001)  
## No Information Rate : 0.6199   
## P-Value [Acc > NIR] : 9.611e-11   
##   
## Kappa : 0.2989   
##   
## Mcnemar's Test P-Value : 3.180e-10   
##   
## Sensitivity : 0.8013   
## Specificity : 0.4866   
## Pos Pred Value : 0.7180   
## Neg Pred Value : 0.6003   
## Prevalence : 0.6199   
## Detection Rate : 0.4967   
## Detection Prevalence : 0.6919   
## Balanced Accuracy : 0.6440   
##   
## 'Positive' Class : f   
##

To improve model further, detect outliers and delete.

plot(logfit3, which = 4, id.n = 10)



Logtrain2 <- Logtrain[!(rownames(Logtrain) %in%  
 c(1280,2374,101,3064,2274,770,3295,4923,4914,2178)),]

After deleting outliers the new model becomes better performed, let’s redo the Anova test again, first apply our original Best-Performed Logfit1 model into the outlier-removed new dataset.

logfit1.2 <- glm(if\_review\_score96~., data=Logtrain2[-c(23,24)], family='binomial')  
logfit3.2 <- glm(if\_review\_score96 ~ host\_response\_rate.percentage + host\_total\_listings\_count +  
 neighbourhood\_cleansed + property\_type + accommodates + Number.of.amenities +   
 instant\_bookable + cancellation\_policy + average\_price + sqrt.host\_total\_listings\_count +  
 sqrt.accommodates + sqrt.security\_deposit + sqrt.number\_of\_reviews + HierCluster +  
 KMeansCluster, data=Logtrain2, family='binomial')  
probabilities3.2 <- predict(logfit3.2, newdata = Logtest, type='response')  
confusionMatrix(as.factor(ifelse(probabilities3.2 > 0.5,"t","f")), as.factor(Logtest$if\_review\_score96))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction f t  
## f 1231 482  
## t 294 453  
##   
## Accuracy : 0.6846   
## 95% CI : (0.6658, 0.7029)  
## No Information Rate : 0.6199   
## P-Value [Acc > NIR] : 1.314e-11   
##   
## Kappa : 0.3035   
##   
## Mcnemar's Test P-Value : 1.908e-11   
##   
## Sensitivity : 0.8072   
## Specificity : 0.4845   
## Pos Pred Value : 0.7186   
## Neg Pred Value : 0.6064   
## Prevalence : 0.6199   
## Detection Rate : 0.5004   
## Detection Prevalence : 0.6963   
## Balanced Accuracy : 0.6459   
##   
## 'Positive' Class : f   
##

probabilities1.2 <- predict(logfit1.2, newdata = Logtest, type='response')  
confusionMatrix(as.factor(ifelse(probabilities1.2 > 0.5,"t","f")), as.factor(Logtest$if\_review\_score96))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction f t  
## f 1229 468  
## t 296 467  
##   
## Accuracy : 0.6894   
## 95% CI : (0.6707, 0.7077)  
## No Information Rate : 0.6199   
## P-Value [Acc > NIR] : 3.514e-13   
##   
## Kappa : 0.3166   
##   
## Mcnemar's Test P-Value : 6.149e-10   
##   
## Sensitivity : 0.8059   
## Specificity : 0.4995   
## Pos Pred Value : 0.7242   
## Neg Pred Value : 0.6121   
## Prevalence : 0.6199   
## Detection Rate : 0.4996   
## Detection Prevalence : 0.6898   
## Balanced Accuracy : 0.6527   
##   
## 'Positive' Class : f   
##

anova(logfit1.2,logfit3.2,test ="Chisq")

## Analysis of Deviance Table  
##   
## Model 1: if\_review\_score96 ~ host\_response\_rate.percentage + host\_total\_listings\_count +   
## host\_identity\_verified + neighbourhood\_cleansed + property\_type +   
## room\_type + accommodates + bathrooms + bedrooms + bed\_type +   
## Number.of.amenities + security\_deposit + cleaning\_fee + instant\_bookable +   
## cancellation\_policy + require\_guest\_phone\_verification +   
## average\_price + sqrt.host\_total\_listings\_count + sqrt.accommodates +   
## sqrt.security\_deposit + sqrt.number\_of\_reviews  
## Model 2: if\_review\_score96 ~ host\_response\_rate.percentage + host\_total\_listings\_count +   
## neighbourhood\_cleansed + property\_type + accommodates + Number.of.amenities +   
## instant\_bookable + cancellation\_policy + average\_price +   
## sqrt.host\_total\_listings\_count + sqrt.accommodates + sqrt.security\_deposit +   
## sqrt.number\_of\_reviews + HierCluster + KMeansCluster  
## Resid. Df Resid. Dev Df Deviance Pr(>Chi)   
## 1 3617 4216.1   
## 2 3616 4211.0 1 5.1028 0.02389 \*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

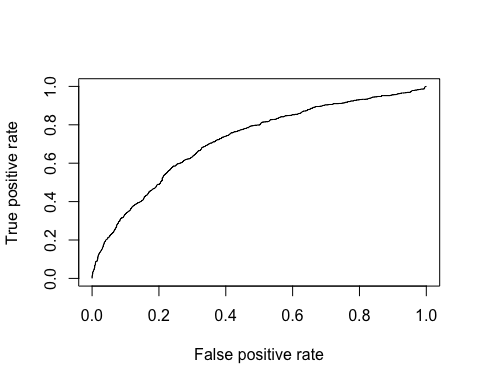
Now we rechecked our model comparison, we can easily see that our model accuracy increased from 0.698 to 0.700, still a 0.002 increase.

The increase of accuracy comes mostly from increase in sensitivity from 0.826 to 0.83. Sensitivity = The proportion of observed positives that were predicted to be positive. In other words, of all the review scores that were truly lower than 96, what percentage did we find? The increase in Sensitivity means our model becomes better on interpreting the score<96 listings.

Actually, our original model has already been very accurate on predicting the score<96 situations, our original model’s sensitivity 0.826 has already been high enough when compared with only 0.49 Specificity.

The result of Anova Chi-Square test finally got P value drop to 0.02 (lower than 0.05). That is to say these two models are statistically different from each other.

# make predictions  
library(ROCR)  
# Compute AUC for predicting Class with the variable CreditHistory.Critical  
perf <- performance(prediction(probabilities3.2,Logtest$if\_review\_score96), measure = "tpr", x.measure = "fpr")  
plot(perf)



auc<-performance(prediction(probabilities3.2,Logtest$if\_review\_score96), measure = "auc")@y.values[[1]]  
auc

## [1] 0.721167