```r
rm(list=ls())
cat("\014") #clean console

#=====Basic======
library(tidyverse)
library(caret)
library(skimr)
library(ggplot2)
library(cluster)
library(psych)
library(ggcorrplot)
library(flexclust)

setwd('C:\\Users\\Jason Xu\\Desktop\\Framework_2\\Group project')
clean = read.csv('clean_dataset.csv', header = T)

#=====description of the variables=====
# order_id - (A unique number to identity the order)
# user_id - (A unique number to identify the user)
# order_number - (The number of times a user purchased at hunter) [question 1]
# order_dow - (Day of the Week the order was made)
# order_hour_of_day - (Time of the order ) [question 2]
# days_since_prior_order - (History of the order)
# product_id - (Id of the product) [ quesiton 2] [question 3]
```

```r
# add_to_cart_order - (Number of items added to cart)
# reordered - (If the reorder took place, □□□□) [question 1]
# department_id - (Unique number allocated to each department)
# department - (Names of the departments)
# product_name - (Name of the products)
# frequency - # of time a user purchased at hunter

# The variables we need to use to solve each problem:
# [question 1]: order_number, add_to_cart_order
# [question 2]: order_hour_of_day, product_id
# [question 3]: order_number, product_id, order_hour_of_day


# '''
# purchasing_count <- ecommerce %>%
#   group_by(user_id, department) %>%
#   summarise(count = n()) %>%
#   pivot_wider(names_from = department, values_from = count, values_fill = 0)
# '''

#======Dimension Reduction =======
#dropping id attributes
df <- clean %>% select(order_dow, order_hour_of_day, days_since_prior_order
```

```
, add_to_cart_order, reordered, first_time_purchase)#%>%left_join(purchse_frequency, by='user_id')%>%select(-user_id)

#=====Suitability for Factor Analysis=====

ggcorrplot(cor(df),colors = c('red','white','green'),type = 'lower')
KMO(r = cor(df)) #KMO = 0.5
cortest.bartlett(cor(df),n = 105273) # significan

#=====Factor Analysis=====
scree_plot=scree(cor(df),factors = T, pc=T)
# 2 components suggested
data.frame(factor = 1:ncol(df), eigen = eigen(cor(df))$values)
# 3 components suggested
fa.parallel(df,fa='fa',fm = 'pa')
# 3 components suggested

# 3 factors selected
fa = fa(r = df,nfactors = 3,fm = 'pa', rotate = 'none')
fa$Vaccounted # 64% explained
data.frame(communality = fa$communality)

print(fa$loadings,cut=0.1)
fa.diagram(fa,sort = T)

#=====Principal Componenet Analysis====
```

```

library(FactoMineR)
pca_facto = PCA(df,graph = F)
library(factoextra)
fviz_eig(pca_facto,ncp=11,addlabels = T)

pca = prcomp(df,scale. = F)
fviz_eig(pca,ncp = 11,addlabels = T)
pca_facto$eig[pca_facto$eig[,'eigenvalue']>1,]
# 3 components
pca_facto$eig

pca_facto = PCA(df,scale.unit = T,ncp = 3,graph = F)
pca_facto$var$contrib %>%
  round(2)
library(factoextra);library(gridExtra)
charts = lapply(1:3,FUN = function(x) fviz_contrib(pca_facto,choice = 'var', axes = x,title=paste('Dim',x)))
grid.arrange(grobs = charts)
fviz_pca_var(X = pca_facto,col.var = 'contrib',gradient.cols = c('red'),col.circle = 'steelblue',repel = T)
# 59% total variance explained

trainComponents = pca_facto$ind$coord


#=====Cluster Analysis=====
within_ss = sapply(X = 1:9, #1 to 9 means to find the best number of clusters from
```

```
 1 to 9
                        FUN = function(x) kmeans(clean,centers = x,iter.max = 100)$tot.withinss)

ratio_ss = sapply(X = 1:9,
                  FUN = function(x) {
                    km = kmeans(clean, centers = x,iter.max = 100)
                    ratio = km$betweenss/km$totss
                    return(ratio)
                  })

dat = data.frame(clusters=1:9,within_ss, ratio_ss)
dat #check the result

#=====Plot======
par(mar = c(4, 4, 2, 2)) # set the figure margin

# Elbow in Ratio Plot
ggplot(dat,aes(x=clusters,y=ratio_ss))+
  geom_line(color='steelblue',size=1.4)+
  scale_x_continuous(breaks=1:9, minor_breaks = 1:9)+
  geom_vline(xintercept=4) # 4 clusters would be the best number of clusters

# Elbow in Within_ss (within sum of squares Plot)
ggplot(dat,aes(x=clusters,y=within_ss))+
```

```r
  geom_line(color='steelblue',size=1.4)+
  scale_x_continuous(breaks=1:9, minor_breaks = 1:9)+
  geom_vline(xintercept=4) # 4 clusters would be the best number of clusters

# Conclusion: We decide to choose 4 clusters for further analysis

#=====Analysis part=====
set.seed(100)
km = kmeans(clean, centers = 4,iter.max = 100, nstart =1) # k-means analysis
table(km$cluster)
k_cluster = km$cluster #□□□□□□

data2 = cbind(clean,k_cluster) # □□□□□□□□□□□cluster
data2

data2 %>%
  select(order_id:first_time_purchase, k_cluster) %>% # select the range of columns
  group_by(k_cluster) %>%
  summarize_all(function(x) round(mean(x, na.rm=T),4)) %>%
  data.frame() #□□□□cluster□□□□□□□□, □□□□insights

#=====Cluster then Predict Using Regression=====
#-----research question 1-----
```

```
#split data
set.seed(100)
split = createDataPartition(y=clean$add_to_cart_order,p = 0.7,list = F,groups = 100)
train = clean[split,]
test = clean[-split,]

#regression predict
linear = lm(add_to_cart_order~.,train)
summary(linear)
sseLinear = sum(linear$residuals^2); sseLinear

#test dataset
predLinear = predict(linear,newdata=test)
sseLinear = sum((predLinear-test$add_to_cart_order)^2); sseLinear

#remove outcome
trainNorm = subset(train,select=-c(add_to_cart_order)) # remove outcome
testNorm = subset(test,select=-c(add_to_cart_order)) # remove outcome

# Since our data is clean already, so we do not need to normalize the data again

set.seed(100)
km = kmeans(x = trainNorm,centers = 2,iter.max=100,nstart=1)
#km$center

```

```r
# Total within sum of squares Plot
within_ss = sapply(1:10,FUN = function(x) kmeans(x = trainNorm,centers = x,iter.max = 100,nstart = 1)$tot.withinss)
ggplot(data=data.frame(cluster = 1:10,within_ss),aes(x=cluster,y=within_ss))+
  geom_line(col='steelblue',size=1.2)+
  geom_point()+
  scale_x_continuous(breaks=seq(1,10,1))+
  geom_vline(xintercept=2)# result: 2 clsuters

# Ratio Plot
ratio_ss = sapply(1:10,FUN = function(x) {km = kmeans(x = trainNorm,centers = x,iter.max = 100,nstart = 1)
km$betweenss/km$totss} )
ggplot(data=data.frame(cluster = 1:10,ratio_ss),aes(x=cluster,y=ratio_ss))+
  geom_line(col='steelblue',size=1.2)+
  geom_point()+
  scale_x_continuous(breaks=seq(1,10,1))+
  geom_vline(xintercept=2)# result: 2 clsuters

# Apply Clustering Solution from Train to Test
km_kcca = as.kcca(km,trainNorm) # flexclust uses objects of the classes kcca
clusterTrain = predict(km_kcca)
clusterTest = predict(km_kcca,newdata=testNorm)
```

```r

table(clusterTrain) # check train cluster
table(clusterTest) # check test cluster

# Split train and test based on cluster membership
train1 = subset(train,clusterTrain==1)
train2 = subset(train,clusterTrain==2)

test1 = subset(test,clusterTest==1)
test2 = subset(test,clusterTest==2)

# Predict for each Cluster then Combine
lm1 = lm(add_to_cart_order~.,train1)
lm2 = lm(add_to_cart_order~.,train2)

pred1 = predict(lm1,newdata=test1)
pred2 = predict(lm2,newdata=test2)

sse1 = sum((test1$add_to_cart_order-pred1)^2); sse1
sse2 = sum((test2$add_to_cart_order-pred2)^2); sse2

predOverall = c(pred1,pred2)
qualityOverall = c(test1$add_to_cart_order,test2$add_to_cart_order)

sseOverall = sum((predOverall - qualityOverall)^2); sseOverall # get the result of sse, can be used to prove our conclusion is more persuasive
```

```
# Compare Results
paste('SSE for model on entire data', sseLinear)
paste('SSE for model on clusters', sseOverall)


#-----research question 2-----
#split data
set.seed(100)
split = createDataPartition(y=clean$product_id,p = 0.7,list = F,groups = 100)
train = clean[split,]
test = clean[-split,]

#regression predict
linear = lm(product_id~.,train)
summary(linear)
sseLinear = sum(linear$residuals^2); sseLinear

#test dataset
predLinear = predict(linear,newdata=test)
sseLinear = sum((predLinear-test$product_id)^2); sseLinear

#remove outcome
trainNorm = subset(train,select=-c(product_id)) # remove outcome
testNorm = subset(test,select=-c(product_id)) # remove outcome

```

```
# Since our data is clean already, so we do not need to normalize the data again

set.seed(100)
km = kmeans(x = trainNorm,centers = 2,iter.max=100,nstart=1)
#km$center

# Total within sum of squares Plot
within_ss = sapply(1:10,FUN = function(x) kmeans(x = trainNorm,centers = x,iter.max = 100,nstart = 1)$tot.withinss)
ggplot(data=data.frame(cluster = 1:10,within_ss),aes(x=cluster,y=within_ss))+
  geom_line(col='steelblue',size=1.2)+
  geom_point()+
  scale_x_continuous(breaks=seq(1,10,1))+
  geom_vline(xintercept=2) # result: 2 clsuters

# Ratio Plot
ratio_ss = sapply(1:10,FUN = function(x) {km = kmeans(x = trainNorm,centers = x,iter.max = 100,nstart = 1)
km$betweenss/km$totss} )
ggplot(data=data.frame(cluster = 1:10,ratio_ss),aes(x=cluster,y=ratio_ss))+
  geom_line(col='steelblue',size=1.2)+
  geom_point()+
  scale_x_continuous(breaks=seq(1,10,1))+
  geom_vline(xintercept=2) # result: 2 clsuters

```

```r
# Apply Clustering Solution from Train to Test
km_kcca = as.kcca(km,trainNorm) # flexclust uses objects of the classes kcca
clusterTrain = predict(km_kcca)
clusterTest = predict(km_kcca,newdata=testNorm)

table(clusterTrain) # check train cluster
table(clusterTest) # check test cluster

# Split train and test based on cluster membership
train1 = subset(train,clusterTrain==1)
train2 = subset(train,clusterTrain==2)

test1 = subset(test,clusterTest==1)
test2 = subset(test,clusterTest==2)

# Predict for each Cluster then Combine
lm1 = lm(product_id~.,train1)
lm2 = lm(product_id~.,train2)

pred1 = predict(lm1,newdata=test1)
pred2 = predict(lm2,newdata=test2)

sse1 = sum((test1$product_id-pred1)^2); sse1
sse2 = sum((test2$product_id-pred2)^2); sse2

predOverall = c(pred1,pred2)
```

```r
qualityOverall = c(test1$product_id,test2$product_id)

sseOverall = sum((predOverall - qualityOverall)^2); sseOverall # get the result of sse, can be used to prove our conclusion is more persuasive

# Compare Results
paste('SSE for model on entire data', sseLinear)
paste('SSE for model on clusters', sseOverall)

```