Problem Set 4

Sachin Badole

12/1/2020

```
rm(list = ls(all = TRUE))
setwd("/Users/sachin/Downloads/Econ_725/Problem Sets/Problem Set 4")
library(knitr)
opts_chunk$set(tidy.opts=list(width.cutoff=60),tidy=TRUE)

suppressMessages(library(h2o))
suppressMessages(library(randomForest))
suppressMessages(library(neuralnet))
suppressMessages(library(data.table))
suppressMessages(library(stargazer))
suppressMessages(library(glmnet))
suppressMessages(library(knitr))
suppressMessages(library(tidyverse))
```

Question 1 Monte Carlos

a) Install the package "h2o" or both the "neuralnet" and "randomForest" packages in R.
b) Estimate three sets of three models. For each, set the seed to 0, the sample size to 1,000, and allocate 50% of the data to a test sample.

```
set.seed(123)
h2o.init()
```

```
##
## H2O is not running yet, starting it now...
##
## Note:  In case of errors look at the following log files:
##     /var/folders/bc/1xzwjdkd1rd3lpf6_2f5206c0000gn/T//Rtmprf9MPM/file8c11bc4c967/h2o_sachin_started_
##     /var/folders/bc/1xzwjdkd1rd3lpf6_2f5206c0000gn/T//Rtmprf9MPM/file8c11482976f/h2o_sachin_started_
##
##
## Starting H2O JVM and connecting: ... Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         4 seconds 276 milliseconds
##     H2O cluster timezone:       America/Chicago
##     H2O data parsing timezone:  UTC
##     H2O cluster version:        3.32.0.1
##     H2O cluster version age:    3 months and 11 days !!!
##     H2O cluster name:           H2O_started_from_R_sachin_osn291
```

```
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   1.78 GB
##     H2O cluster total cores:    4
##     H2O cluster allowed cores:  4
##     H2O cluster healthy:        TRUE
##     H2O Connection ip:          localhost
##     H2O Connection port:        54321
##     H2O Connection proxy:       NA
##     H2O Internal Security:      FALSE
##     H2O API Extensions:         Amazon S3, XGBoost, Algos, AutoML, Core V3, TargetEncoder, Core V4
##     R Version:                  R version 4.0.3 (2020-10-10)
```

```
## Warning in h2o.clusterInfo():
## Your H2O cluster version is too old (3 months and 11 days)!
## Please download and install the latest version from http://h2o.ai/download/
```

```
n=1000
perc_train=.5
specification1=T
error=F
sderror=.2

set.seed(0)
################################################
#Draw observable explanatory variables
x1 = rgamma(n,2,1); x2 = rnorm(n,0,2);
x3 = rweibull(n,2,2); x4 = rlogis(n,2,1);
x5 = rbeta(n,2,1);
x = cbind(x1,x2,x3,x4,x5)
################################################
#transform into independent random variables
# find the current correlation matrix
c1 = var(x)

# cholesky decomposition to get independence
chol1 = solve(chol(c1))

x= x%*%chol1
################################################
#generate random correlation matrix
R = matrix(runif(ncol(x)^2,-1,1), ncol=ncol(x))
RtR = R %*% t(R)
corr = cov2cor(RtR)
# check that it is positive definite
sum((eigen(corr)$values>0))==ncol(x)
```

```
## [1] TRUE
```

```
################################################
#transform according to this correlation matrix
x = x %*% chol(corr)

################################################
```

Defined the test and train data. c) You'll estimate three sets of three models. For each, set the seed to 0, the sample size to 1,000, and allocate 50% of the data to a test sample.

i) Estimate a neural net with 3 hidden layers, each with 64, 32, and 16 neurons respectively. Use 100 epochs.

For n = 1000

```
y1 = x1+((x3*(x2^2))/10)+(x4*x1*x5)/10
y2 = x1+((x3*(x2^2))/10)+(x4*x1*x5)/10+rnorm(n,0,1)
y3 = log(abs((x1^4)/10)+abs(x4)+(x3^2))+x4*x2*sin(x5)+rnorm(n,0,1)
x <- cbind(x1,x2,x3,x4,x5,y1,y2,y3)
x=data.table(x)
names(x)=c("x1","x2","x3","x4","x5","y1","y2","y3")

set.seed(0)

smp_siz = floor(0.5*nrow(x))
train_ind=sample(nrow(x),size=smp_siz)
train =x[train_ind,]
test=x[-train_ind,]
rm(smp_siz)
set.seed(0)
h2o.init()
```

```
##  Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         4 seconds 646 milliseconds
##     H2O cluster timezone:       America/Chicago
##     H2O data parsing timezone:  UTC
##     H2O cluster version:        3.32.0.1
##     H2O cluster version age:    3 months and 11 days !!!
##     H2O cluster name:           H2O_started_from_R_sachin_osn291
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   1.77 GB
##     H2O cluster total cores:    4
##     H2O cluster allowed cores:  4
##     H2O cluster healthy:        TRUE
##     H2O Connection ip:          localhost
##     H2O Connection port:        54321
##     H2O Connection proxy:       NA
##     H2O Internal Security:      FALSE
##     H2O API Extensions:         Amazon S3, XGBoost, Algos, AutoML, Core V3, TargetEncoder, Core V4
##     R Version:                  R version 4.0.3 (2020-10-10)

## Warning in h2o.clusterInfo():
## Your H2O cluster version is too old (3 months and 11 days)!
## Please download and install the latest version from http://h2o.ai/download/
```

```
#h2o.train=as.h2o(train)
#h2o.test=as.h2o(test)

yvar <- c('y1','y2','y3')

# Run Machine Learning with Neural Net
neuraln_mse <- c()
for(i in yvar){
  classifier <- h2o.deeplearning(x=c('x1','x2','x3','x4','x5'),y = i, training_frame = as.h2o(train),
                                 validation_frame = as.h2o(test), activation = 'Rectifier', hidden = c(
                                epochs = 100, seed = 420, reproducible = TRUE, standardize = TRUE,
                                train_samples_per_iteration = -2)

# MSE for the Test set results
neuraln_mse[i] <- h2o.mse(classifier)
}
```

```
##   |                                                                      |
##   |                                                                      |
##   |                                                                      |
##   |                                                                      |
##   |                                                                      |
##   |                                                                      |
##   |                                                                      |
##   |                                                                      |
##   |                                                                      |
```

```
#data.frame(neuraln_mse)
```

ii) Estimate a series using the poly function. Set the degree to 3.
iii) Estimate a random forest. Use 1000 trees with 4 covariates sampled each time.
iv) Calculate the MSE on the test set for n=1000 and n=10,000.
v) For specification 1, which performs best? Why?

e) For specification 2, which performs best? Why?
f) For specification 3, which performs best? Why?

```
## Run the 3 poly models
y1_ploy=lm(y1~poly(x1,x2,x3,x4,x5,degree=3),data=train)
p=predict(y1_ploy,test)
y1_poly_mse=mean((p-test$y1)^2)

y2_ploy=lm(y2~poly(x1,x2,x3,x4,x5,degree=3),data=train)
p=predict(y2_ploy,test)
y2_poly_mse=mean((p-test$y2)^2)

y3_ploy=lm(y3~poly(x1,x2,x3,x4,x5,degree=3),data=train)
p=predict(y3_ploy,test)
y3_poly_mse=mean((p-test$y3)^2)

## Random Forest
y1_rf <- randomForest(y1~x1+x2+x3+x4+x5,data=train)
```

```
p=predict(y1_rf,test)
y1_rf_mse=mean((p-test$y1)^2)

y2_rf <- randomForest(y2~x1+x2+x3+x4+x5,data=train)
p=predict(y2_rf,test)
y2_rf_mse=mean((p-test$y2)^2)

y3_rf <- randomForest(y3~x1+x2+x3+x4+x5,data=train)
p=predict(y3_rf,test)
y3_rf_mse=mean((p-test$y3)^2)


poly_mse <- data.frame(poly_mse =c(y1_poly_mse, y2_poly_mse, y3_poly_mse), rf_mse = c(y1_rf_mse, y2_rf_

# MSE results for n = 1000
results_1000 <- data.frame(neuraln_mse, poly_mse)
names(results_1000)=c("Neuraln Net", "Poly Model", "Random Forest")
#results_1000
kable(results_1000)
```

	Neuraln Net	Poly Model	Random Forest
y1	0.408336	0.000000	0.9498763
y2	1.603857	1.314207	1.8083224
y3	1.109687	1.119598	4.9286909

For n = 10000

```
set.seed(0)
h2o.init(nthreads=-1)
```

```
##  Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         21 seconds 899 milliseconds
##     H2O cluster timezone:       America/Chicago
##     H2O data parsing timezone:  UTC
##     H2O cluster version:        3.32.0.1
##     H2O cluster version age:    3 months and 11 days !!!
##     H2O cluster name:           H2O_started_from_R_sachin_osn291
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   1.77 GB
##     H2O cluster total cores:    4
##     H2O cluster allowed cores:  4
##     H2O cluster healthy:        TRUE
##     H2O Connection ip:          localhost
##     H2O Connection port:        54321
##     H2O Connection proxy:       NA
##     H2O Internal Security:      FALSE
##     H2O API Extensions:         Amazon S3, XGBoost, Algos, AutoML, Core V3, TargetEncoder, Core V4
##     R Version:                  R version 4.0.3 (2020-10-10)
```

```
## Warning in h2o.clusterInfo():
## Your H2O cluster version is too old (3 months and 11 days)!
## Please download and install the latest version from http://h2o.ai/download/
```

```
n=10000
perc_train=.5
specification1=T
error=F
sderror=.2

set.seed(0)
################################################
#Draw observable explanatory variables
x1 = rgamma(n,2,1); x2 = rnorm(n,0,2);
x3 = rweibull(n,2,2); x4 = rlogis(n,2,1);
x5 = rbeta(n,2,1);
x = cbind(x1,x2,x3,x4,x5)
################################################
#transform into independent random variables
# find the current correlation matrix
c1 = var(x)

# cholesky decomposition to get independence
chol1 = solve(chol(c1))

x= x%*%chol1
################################################
#generate random correlation matrix
R = matrix(runif(ncol(x)^2,-1,1), ncol=ncol(x))
RtR = R %*% t(R)
corr = cov2cor(RtR)
# check that it is positive definite
sum((eigen(corr)$values>0))==ncol(x)
```

```
## [1] TRUE
```

```
################################################
#transform according to this correlation matrix
x = x %*% chol(corr)

################################################

y1 = x1+((x3*(x2^2))/10)+(x4*x1*x5)/10
y2 = x1+((x3*(x2^2))/10)+(x4*x1*x5)/10+rnorm(n,0,1)
y3 = log(abs((x1^4)/10)+abs(x4)+(x3^2))+x4*x2*sin(x5)+rnorm(n,0,1)
x <- cbind(x1,x2,x3,x4,x5,y1,y2,y3)
x=data.table(x)
names(x)=c("x1","x2","x3","x4","x5","y1","y2","y3")

set.seed(0)

smp_siz = floor(0.5*nrow(x))
train_ind=sample(nrow(x),size=smp_siz)
```

```
train =x[train_ind,]
test=x[-train_ind,]
rm(smp_siz)
set.seed(0)
h2o.init()
```

```
##  Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         22 seconds 76 milliseconds
##     H2O cluster timezone:       America/Chicago
##     H2O data parsing timezone:  UTC
##     H2O cluster version:        3.32.0.1
##     H2O cluster version age:    3 months and 11 days !!!
##     H2O cluster name:           H2O_started_from_R_sachin_osn291
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   1.77 GB
##     H2O cluster total cores:    4
##     H2O cluster allowed cores:  4
##     H2O cluster healthy:        TRUE
##     H2O Connection ip:          localhost
##     H2O Connection port:        54321
##     H2O Connection proxy:       NA
##     H2O Internal Security:      FALSE
##     H2O API Extensions:         Amazon S3, XGBoost, Algos, AutoML, Core V3, TargetEncoder, Core V4
##     R Version:                  R version 4.0.3 (2020-10-10)

## Warning in h2o.clusterInfo():
## Your H2O cluster version is too old (3 months and 11 days)!
## Please download and install the latest version from http://h2o.ai/download/
```

```
#h2o.train=as.h2o(train)
#h2o.test=as.h2o(test)

yvar <- c('y1','y2','y3')

# Run Machine Learning with Neural Net
neuraln_mse <- c()
for(i in yvar){
  classifier <- h2o.deeplearning(x=c('x1','x2','x3','x4','x5'),y = i, training_frame = as.h2o(train),
                                 validation_frame = as.h2o(test), activation = 'Rectifier', hidden = c(
                               epochs = 100, seed = 420, reproducible = TRUE, standardize = TRUE,
                               train_samples_per_iteration = -2)


# MSE for the Test set results
neuraln_mse[i] <- h2o.mse(classifier)
}
```

```
##   |                                                                              |
##   |                                                                              |
##   |                                                                              |
```

```
##   |                                                                      |
##   |                                                                      |
##   |                                                                      |
##   |                                                                      |
##   |                                                                      |
##   |                                                                      |
```

```
#data.frame(neuraln_mse)

#########################################################

## Run the 3 poly models
y1_ploy=lm(y1~poly(x1,x2,x3,x4,x5,degree=3),data=train)
p=predict(y1_ploy,test)
y1_poly_mse=mean((p-test$y1)^2)

y2_ploy=lm(y2~poly(x1,x2,x3,x4,x5,degree=3),data=train)
p=predict(y2_ploy,test)
y2_poly_mse=mean((p-test$y2)^2)

y3_ploy=lm(y3~poly(x1,x2,x3,x4,x5,degree=3),data=train)
p=predict(y3_ploy,test)
y3_poly_mse=mean((p-test$y3)^2)

## Random Forest
y1_rf <- randomForest(y1~x1+x2+x3+x4+x5,data=train)
p=predict(y1_rf,test)
y1_rf_mse=mean((p-test$y1)^2)

y2_rf <- randomForest(y2~x1+x2+x3+x4+x5,data=train)
p=predict(y2_rf,test)
y2_rf_mse=mean((p-test$y2)^2)

y3_rf <- randomForest(y3~x1+x2+x3+x4+x5,data=train)
p=predict(y3_rf,test)
y3_rf_mse=mean((p-test$y3)^2)


poly_mse <- data.frame(poly_mse =c(y1_poly_mse, y2_poly_mse, y3_poly_mse), rf_mse = c(y1_rf_mse, y2_rf_m

# MSE results for n = 10000
results_10000 <- data.frame(neuraln_mse, poly_mse)
names(results_10000)=c("Neuraln Net", "Poly Model", "Random Forest")

kable(results_10000)
```

	Neuraln Net	Poly Model	Random Forest
y1	0.012978	0.0000000	0.4168588
y2	0.997011	0.9421907	1.3058079
y3	1.081388	1.0338544	3.1064397

Answer for above queston 1:

According to above calculation, the polynomial model performs best under the first specification y1 in the 1000 and 10000 cases. While poly is able to catch non-linear correlations among parameters, both Neural net and Random Forest based on linear models. For second specification y2, again polynomial model performs the best under the n=1000 case. but produces different MSE with polys under n=10000 case. It might because that polynomial model is able to catch and parse out more information than the random forest and neural net. The prediction performance of polynomial and random forest may be disturbed by randomness of the observations. Lastly third specification y3, neural net stands out good under n=1000 case and polynomial model performs better under n=10000 (and mse for poly and neural some what similar for n=1000). The 3rd function is much more complicated, which may cause the neural net to be affected by randomness with small training set. However, it outperforms the other two models with larger sample set which offers more information to learn from.

Question 2

2) Go back to problem set 3. In addition to the five models you estimated there, estimate a neural net with the same five predictors you used in question 6. Use 300 epochs. How does it perform, in terms of MSE, relative to the cross-validated flexible logit model with those predictors?

```
# Merge data from problem set 3.
datam <- read.csv("data_airm.csv")

# create testing and training datasets
testmarketind <- sample.int(nrow(datam), 1000, replace = F)
markettest <- datam[testmarketind,]
markettrain <- datam[-testmarketind,]

lpm_carrier_market_in <- lm(carrier_market_in ~ num_competitors, data = markettrain)
logit_carrier_market_in <- glm(carrier_market_in ~ num_competitors, family = binomial(link = "logit"), 
probit_carrier_market_in <- glm(carrier_market_in ~ num_competitors, family = binomial(link = "probit")

probs <- rep(0,times = length(unique(markettrain$num_competitors)))

for (i in 1:length(unique(markettrain$num_competitors))){
  nc <- unique(markettrain$num_competitors)[i]

  # calculate joint frequency of carrier_market_in and number of carriers i
  if_ncarr_carrier_market_in <- ifelse(((markettrain$num_competitors == nc)
                        & (markettrain$carrier_market_in == 1)),1,0)
  sum_ncarr_carrier_market_in <- sum(if_ncarr_carrier_market_in)

  # calculate frequency of number of carriers
  if_ncarr <- ifelse(markettrain$num_competitors == nc, 1, 0)
  sum_ncarr <- sum(if_ncarr)

  probs[i] <- (sum_ncarr_carrier_market_in/nrow(markettrain)) / (sum_ncarr/nrow(markettrain))
}
probs_nums <- data.frame(cbind(probs, unique(markettrain$num_competitors)))
names(probs_nums) <- c("probs","num_competitors")
probs_nums <- probs_nums[order(probs_nums$num_competitors),]

#
```

```r
covars <- datam[, c("num_competitors", "average_distance_m.x", "market_size.x",
                    "hub_route.x", "vacation_route.x", "slot_controlled.x",
                    "market_income.x")]
covar_train <- covars[-testmarketind,]
covar_test <- covars[testmarketind,]
testpoly <- poly(as.matrix(covar_test), degree = 2, raw = T)
carrier_market_in_train <- datam$carrier_market_in[-testmarketind]
carrier_market_in_test <- datam$carrier_market_in[testmarketind]
glm_mat <- cbind(carrier_market_in_train, poly(as.matrix(covar_train), degree = 2, raw = T))

L1cvnet <- cv.glmnet(x = poly(as.matrix(covar_train), degree = 2, raw = T),
                     y = carrier_market_in_train, alpha = 1, family = "binomial", type.measure = "mse")
L1logitnet <- glmnet(x =  poly(as.matrix(covar_train), degree = 2, raw = T),
                     y = carrier_market_in_train, family = "binomial")
L1cvmin <- L1cvnet$lambda.min

# Find MSE for each model
mselpm <- mean((carrier_market_in_test - predict(lpm_carrier_market_in, markettest))^2)
mselogit <- mean((carrier_market_in_test - predict(logit_carrier_market_in, markettest, type = "response
mseprobit <- mean((carrier_market_in_test - predict(probit_carrier_market_in, markettest, type = "respo

nonpartest <- merge(markettest, probs_nums, by = "num_competitors")
msenonpar <- mean((nonpartest$carrier_market_in - nonpartest$probs)^2)
mseL1logit <- mean((carrier_market_in_test - predict(L1logitnet,testpoly,type = "response", s = L1cvmin

# For Neural Net

mkt_var <- c("carrier_market_in","num_competitors", "average_distance_m.x", "market_size.x",
                    "hub_route.x", "vacation_route.x", "slot_controlled.x",
                    "market_income.x")
data_st <- datam[,c("carrier_market_in","num_competitors", "average_distance_m.x", "market_size.x",
                    "hub_route.x", "vacation_route.x", "slot_controlled.x", "market_income.x")]

maxs2 <- apply(data_st, 2, max)
mins2 <- apply(data_st, 2, min)
data_st <- as.data.frame(scale(data_st,center = mins2, scale = maxs2 - mins2))

nn_test<- data_st[testmarketind,mkt_var]
nn_train <- data_st[-testmarketind,mkt_var]

suppressMessages(library(h2o))
h2o.init(nthreads = -1)
```

```
##  Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         2 minutes 26 seconds
##     H2O cluster timezone:       America/Chicago
##     H2O data parsing timezone:  UTC
##     H2O cluster version:        3.32.0.1
##     H2O cluster version age:    3 months and 11 days !!!
##     H2O cluster name:           H2O_started_from_R_sachin_osn291
##     H2O cluster total nodes:    1
```

```
##     H2O cluster total memory:   1.77 GB
##     H2O cluster total cores:    4
##     H2O cluster allowed cores:  4
##     H2O cluster healthy:        TRUE
##     H2O Connection ip:          localhost
##     H2O Connection port:        54321
##     H2O Connection proxy:       NA
##     H2O Internal Security:      FALSE
##     H2O API Extensions:         Amazon S3, XGBoost, Algos, AutoML, Core V3, TargetEncoder, Core V4
##     R Version:                  R version 4.0.3 (2020-10-10)

## Warning in h2o.clusterInfo():
## Your H2O cluster version is too old (3 months and 11 days)!
## Please download and install the latest version from http://h2o.ai/download/
```

```
classifier2 <- h2o.deeplearning(x=mkt_var[-1],y = 'carrier_market_in', training_frame = as.h2o(nn_train
                                validation_frame = as.h2o(nn_test), activation = 'Rectifier', hidden = c(
                                epochs = 300, train_samples_per_iteration = -2)
```

```
##   |                                                                              |
##   |                                                                              |
##   |                                                                              |
```

```
nn_mse2 <- h2o.mse(classifier2)

mses <- round(c(mselpm, mselogit, mseprobit, msenonpar, mseL1logit,nn_mse2),4)
models <- c("Linear Probability", "Logit", "Probit", "Nonparametric", "L_1 Regularized","Neural Network
mses <- data.frame("Model" = models, "MSE" = mses)

kable(mses)
```

Model	MSE
Linear Probability	0.1128
Logit	0.0933
Probit	0.0944
Nonparametric	0.0909
L_1 Regularized	0.0840
Neural Network	0.0591

Answer for above question 2 :

From the table above, we observe that the Neural Network perform better than the other method. It may because the NN classifier catches more information among the factors by building hidden layers.

Question 3

3) Group markets using kmeans and agglomerative hierarchical clustering.

a) Load the file "PS4 mkt.R".

b) For c) through e), use the average price, average distance, nonstop miles, number of carriers, and hhi as covariates.
c) Use the kmeans function to cluster the origin and destination pairs in the data into 2 clusters. Calculate summary statistics for each of them. How would you best characterize these clusters qualitatively?
d) Now, use the kmeans function to cluster the origin and destination pairs in the data into 4 clusters.
e) Use the hclust function to perform agglomerative clustering. Plot the dendrogram. Using the cutree function with k = 4, compare the results of this clustering algorithm with those in part d.

```
load('PS4_mkt.R')

set.seed(0)
cov.cl <- c("average_price","average_distance","nonstop_miles","num_carriers","hhi")
datamcluster <- datam[cov.cl]
kmeanscluster <- kmeans(datamcluster, 2, nstart=10)
datam$cluster <- kmeanscluster$cluster
# Summarize
m <- datam %>%
group_by(cluster) %>%
  select(average_price,average_distance,nonstop_miles,num_carriers,hhi,cluster) %>%
  summarise_all(mean)

s <- datam %>%
group_by(cluster) %>%
  select(average_price,average_distance,nonstop_miles,num_carriers,hhi,cluster) %>%
  summarise_all(sd)
```

Mean

```
kable(m)
```

cluster	average_price	average_distance	nonstop_miles	num_carriers	hhi
1	269.8549	1457.3090	1350.2452	5.251104	3902.914
2	201.3623	788.9524	758.5206	2.960175	8585.706

SD

```
kable(s)
```

cluster	average_price	average_distance	nonstop_miles	num_carriers	hhi
1	75.23428	817.1190	794.8811	1.227682	1106.542
2	86.94902	523.5572	508.5733	1.519989	1298.442

Answer for above question 3 :

Above cluster splits samples into one with higher price, longer distance and nonstop miles, more carriers

and a smaller HHI, the other group with the opposite features. Intuitively, this clustering separate larger, commercial airports with smaller, private ones.

```
kmeanscluster_4gr <- kmeans(datamcluster, 4, nstart=10)
datam$cluster4 <- kmeanscluster_4gr$cluster

# Summarize
m1 <- datam %>%
group_by(cluster4) %>%
  select(average_price,average_distance,nonstop_miles,num_carriers,hhi,cluster) %>%
  summarise_all(mean)
```

```
## Adding missing grouping variables: `cluster4`
```

```
s1 <- datam %>%
group_by(cluster4) %>%
  select(average_price,average_distance,nonstop_miles,num_carriers,hhi,cluster) %>%
  summarise_all(sd)
```

```
## Adding missing grouping variables: `cluster4`
```

Mean

```
kable(m1)
```

cluster4	average_price	average_distance	nonstop_miles	num_carriers	hhi	cluster
1	186.3920	764.4205	743.8937	2.371831	9337.658	2.000000
2	236.8167	924.8106	865.4837	4.456747	6122.403	1.441523
3	245.4983	1067.1173	964.1475	5.137898	3730.731	1.000000
4	333.0826	2447.6873	2305.0910	5.880634	3130.358	1.000835

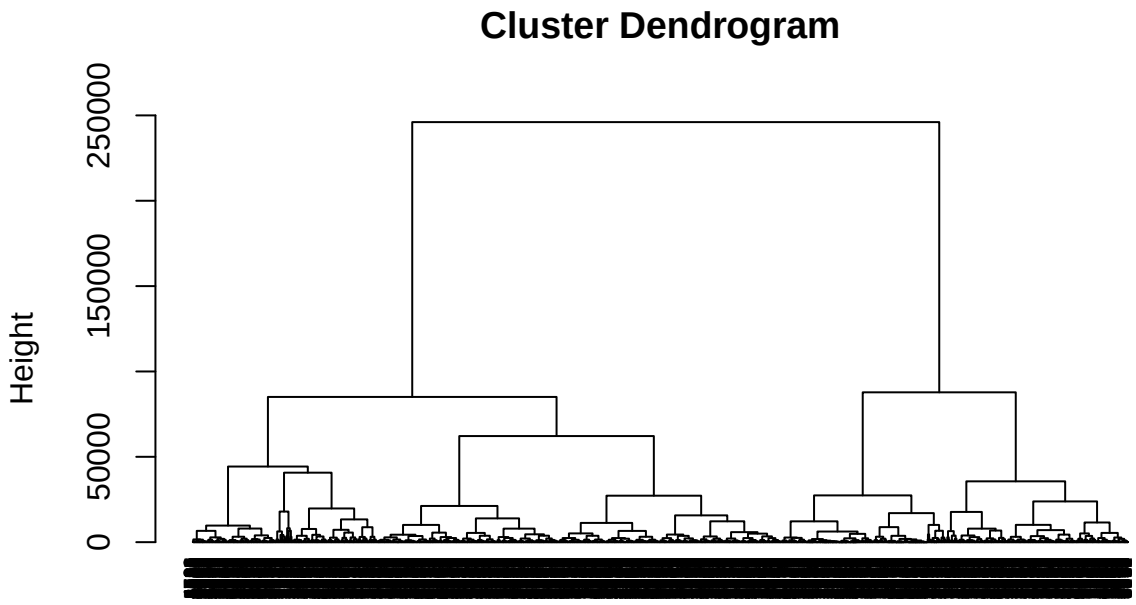
SD

```
kable(s1)
```

cluster4	average_price	average_distance	nonstop_miles	num_carriers	hhi	cluster
1	87.89204	531.2752	513.9796	1.326181	730.0173	0.0000000
2	69.33488	513.4296	500.9306	1.107312	816.4512	0.4967405
3	52.89275	361.9145	342.9395	1.049788	723.5483	0.0000000
4	77.03146	690.6919	699.1706	1.282857	901.1335	0.0288916

```
d <- dist(datamcluster, method = "euclidean")
hc <- hclust(d, method = "ward.D2")
```

```r
plot(hc,cex=0.6,hang=-1)
```




```r
ct4 <- cutree(hc, k = 4)
table(kmean4=kmeanscluster_4gr$cluster,hclust4 = ct4)
```

```
##       hclust4
## kmean4    1    2    3    4
##      1 1085    0  333    2
##      2    1  499  857   88
##      3    0 1967    0  107
##      4    0  185    0 1013
```