Group-4-Project-2

Yi Qian; Kailing Huang; Tarek Hatata; Shusen Wu; Bohui NI

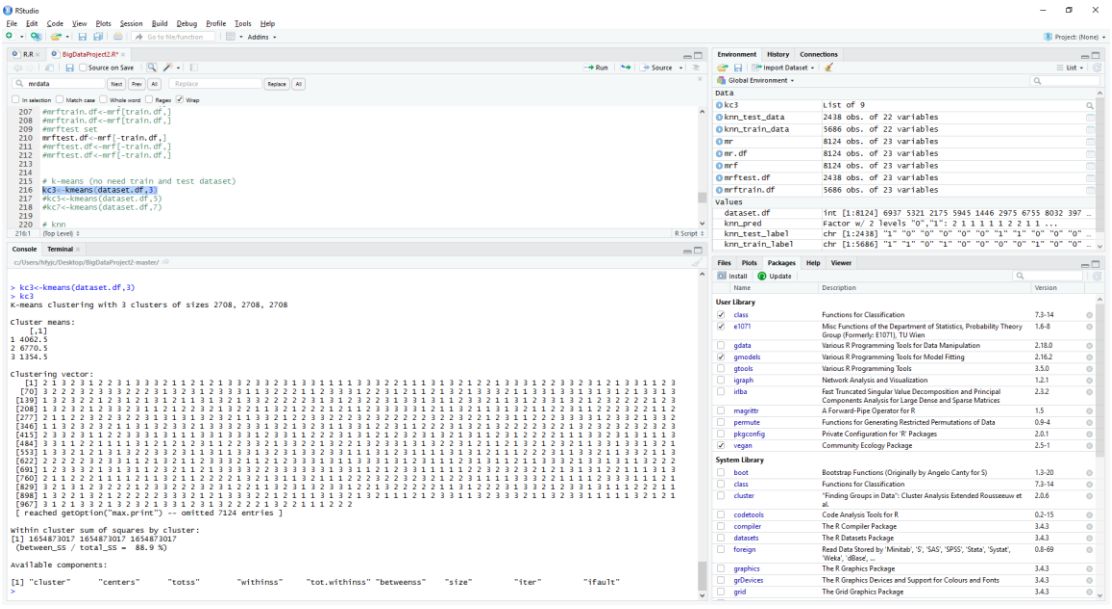
R function:

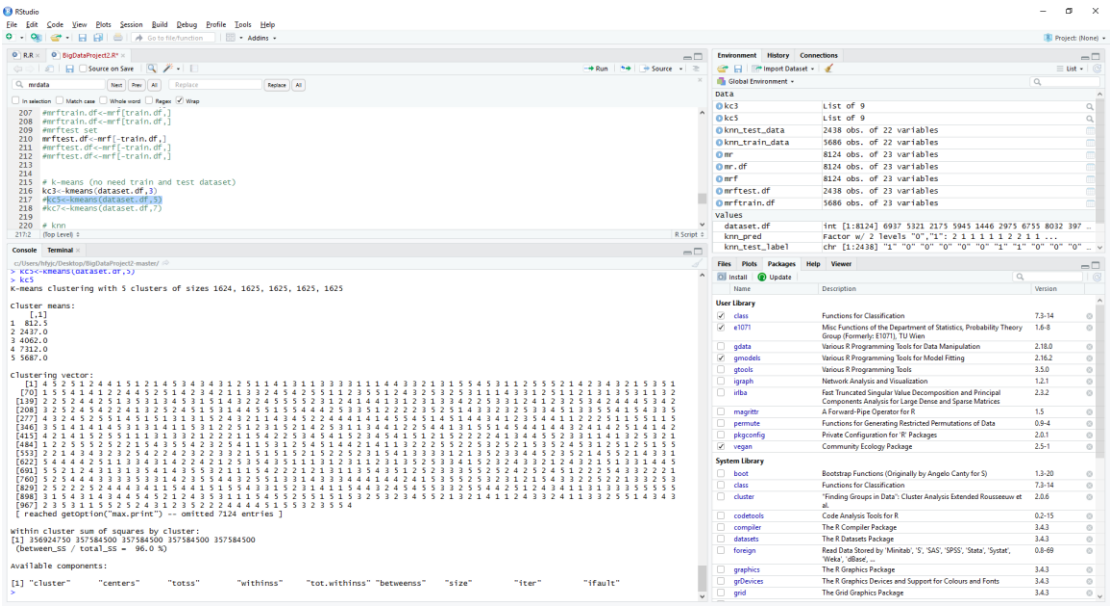
1. as.data.frame(x, row.names = NULL, optional = FALSE, ···)
   # S3 method for character
   as.data.frame(x, ···, stringsAsFactors = default.stringsAsFactors())
   # S3 method for list
   as.data.frame(x, row.names = NULL, optional = FALSE, ···, cut.names = FALSE, col.names = names(x), fix.empty.names = TRUE, stringsAsFactors = default.stringsAsFactors())
   # S3 method for matrix
   as.data.frame(x, row.names = NULL, optional = FALSE, ···, stringsAsFactors = default.stringsAsFactors())
   Functions to check if an object is a data frame, or coerce it if possible.
2. Sapply(X, FUN, ..., simplify = TRUE, USE.NAMES = TRUE)
   Sapply is equivalent to sapply, except that it preserves the dimension and dimension names of the argument X. It also preserves the dimension of results of the function FUN. It is intended for application to results e.g. of a call to by. Lapply is an analog to lapply insofar as it does not try to simplify the resulting list of results of FUN.
3. as.character(x, ...)
   Create or test for objects of type "character".
4. kmeans(x, centers, iter.max = 10, nstart = 1, algorithm = c("Hartigan-Wong", "Lloyd", "Forgy", "MacQueen"), trace=FALSE)
   The data given by x are clustered by the k-means method, which aims to partition the points into k groups such that the sum of squares from points to the assigned cluster centres is minimized. At the minimum, all cluster centres are at the mean of their Voronoi sets (the set of data points which are nearest to the cluster centre).
5. knn(train, test, cl, k = 1, l = 0, prob = FALSE, use.all = TRUE)
   k-nearest neighbour classification for test set from training set. For each row of the test set, the k nearest (in Euclidean distance) training set vectors are found, and the classification is decided by majority vote, with ties broken at random. If there are ties for the kth nearest vector, all candidates are included in the vote.
6. CrossTable(x, y, digits=3, max.width = 5, expected=FALSE, prop.r=TRUE, prop.c=TRUE, prop.t=TRUE, prop.chisq=TRUE, chisq = FALSE, fisher=FALSE, mcnemar=FALSE, resid=FALSE, sresid=FALSE, asresid=FALSE, missing.include=FALSE, format=c("SAS","SPSS"), dnn = NULL, ...)
   An implementation of a cross-tabulation function with output similar to S-Plus crosstabs() and SAS Proc Freq (or SPSS format) with Chi-square, Fisher and McNemar tests of the independence of all table factors.
7. vegdist(x, method="bray", binary=FALSE, diag=FALSE, upper=FALSE, na.rm = FALSE, ...)
   The function computes dissimilarity indices that are useful for or popular with community ecologists. All indices use quantitative data, although they would be named by the corresponding binary index, but you can calculate the binary index using an appropriate argument. If you do not find your favourite index here, you can see if it can be implemented using designdist.

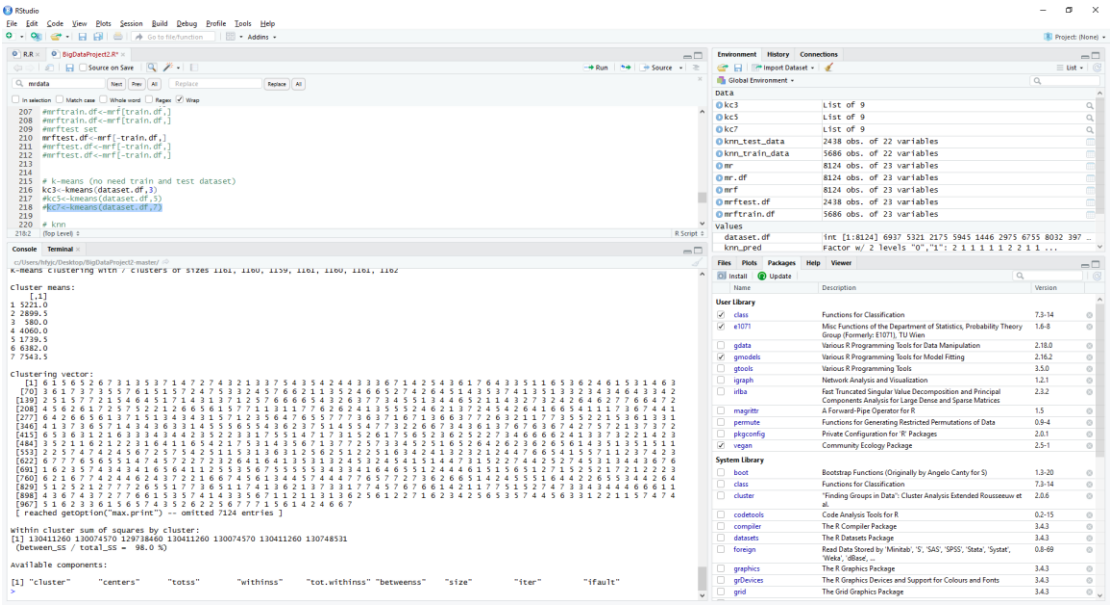
8. hclust(d, method = "complete", members = NULL)
   Hierarchical cluster analysis on a set of dissimilarities and methods for analyzing it.
9. pairs(formula, data = NULL, ⋯, subset, na.action = stats::na.pass)
   A matrix of scatterplots is produced
10. lm(formula, data, subset, weights, na.action, method = "qr", model = TRUE, x = FALSE, y = FALSE, qr = TRUE, singular.ok = TRUE, contrasts = NULL, offset, ⋯)
    lm is used to fit linear models. It can be used to carry out regression, single stratum analysis of variance and analysis of covariance (although aov may provide a more convenient interface for these).
11. predict(object, newdata, se.fit = FALSE, scale = NULL, df = Inf, interval = c("none", "confidence", "prediction"), level = 0.95, type = c("response", "terms"), terms = NULL, na.action = na.pass, pred.var = res.var/weights, weights = 1, ...)
    Predicted values based on linear model object.
12. glm(formula, family = gaussian, data, weights, subset, na.action, start = NULL, etastart, mustart, offset, control = list(⋯), model = TRUE, method = "glm.fit", x = FALSE, y = TRUE, contrasts = NULL, ⋯)
13. sample(x, size, replace = FALSE, prob = NULL)
    sample.int(n, size = n, replace = FALSE, prob = NULL, useHash = (!replace && is.null(prob) && size <= 2="" n="" &&="">
    sample takes a sample of the specified size from the elements of x using either with or without replacement.
14. svm(formula, data = NULL, ..., subset, na.action = na.omit, scale = TRUE)
    svm is used to train a support vector machine. It can be used to carry out general regression and classification (of nu and epsilon-type), as well as density-estimation. A formula interface is provided.

Table with results as specified in 3:

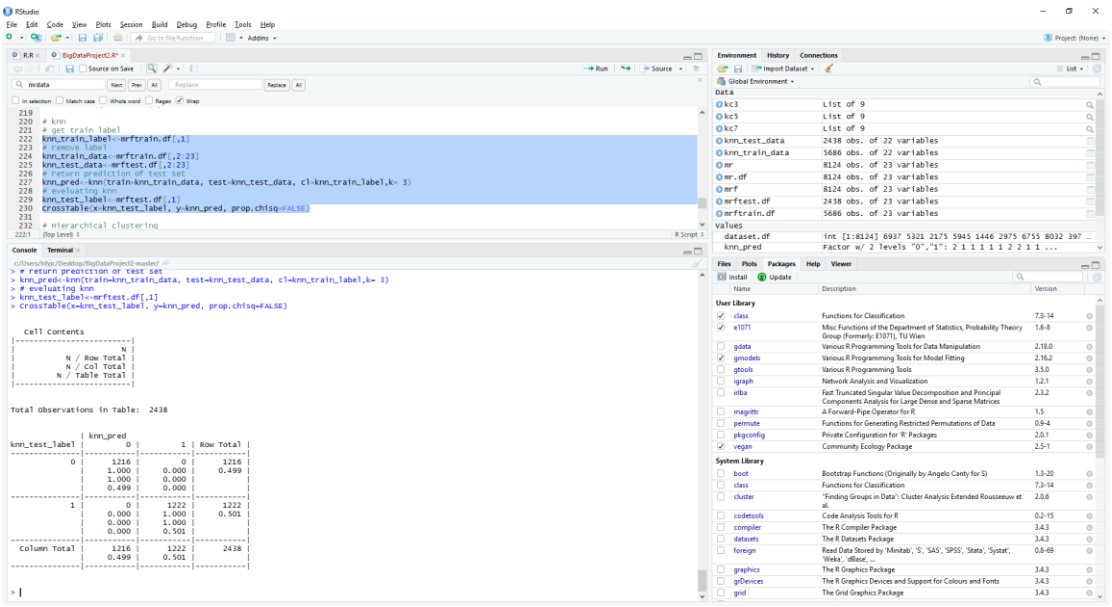
K-means




RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
mrdata
#mrftrain.df<-mrf[train.df,]
#mrftrain.df<-mrf[train.df,]
#mrftest set
mrftest.df<-mrf[-train.df,]
#mrftest.df<-mrf[-train.df,]
#mrftest.df<-mrf[-train.df,]
# k-means (no need train and test dataset)
kc3<-kmeans(dataset.df,3)
#kc5<-kmeans(dataset.df,5)
#kc7<-kmeans(dataset.df,7)
knn
Console Terminal
> kc5<-kmeans(dataset.df,5)
> kc5
K-means clustering with 5 clusters of sizes 1624, 1625, 1625, 1625, 1625
Cluster means:
[,1]
1 812.5
2 2437.0
3 4062.0
4 7312.0
5 5687.0
Clustering vector:
[1] 4 5 2 5 1 2 4 4 1 5 1 2 1 4 5 3 4 3 4 3 1 2 5 1 1 4 1 3 1 1 3 3 3 3 1 1 1 4 4 3 3 2 1 3 1 5 5 4 5 3 1 1 2 5 5 5 2 1 4 2 3 4 3 2 1 5 3 5 1
[70] 1 5 5 4 1 4 1 2 2 4 4 5 2 5 1 4 2 3 4 2 1 1 3 3 2 4 5 4 2 5 5 1 1 2 3 5 5 1 2 4 3 2 5 3 2 5 3 1 1 1 4 3 3 1 2 5 1 1 2 1 3 1 3 5 3 1 1 3 2
[139] 2 2 5 2 4 4 2 5 1 3 5 3 1 3 4 5 3 1 5 1 4 3 2 2 4 5 5 5 5 2 3 1 2 4 1 4 4 1 3 1 2 3 1 3 3 4 2 2 5 3 3 1 2 4 1 2 3 2 5 3 4 2 4 4 4 5 3 4 2
[208] 3 2 5 2 4 5 4 2 2 4 1 3 2 5 2 4 5 1 5 3 1 4 4 5 5 1 5 5 4 4 4 2 5 3 3 5 1 2 2 2 2 3 5 2 5 1 4 3 3 2 3 2 5 3 3 4 5 1 3 3 5 5 4 1 5 4 3 3 5
[277] 4 3 2 4 5 2 5 5 1 4 5 1 5 1 3 1 3 1 5 2 4 3 2 1 1 4 3 4 5 2 2 4 4 4 1 4 1 4 5 5 4 5 1 4 5 1 4 4 3 4 1 2 3 5 4 4 1 1 2 2 2 5 1 1 5 5 1 1 5
[346] 3 5 1 4 1 4 1 4 5 3 1 3 1 4 1 5 1 3 1 2 2 5 1 2 3 1 5 2 1 4 2 5 3 1 1 3 4 4 1 2 2 5 4 4 1 3 1 5 5 1 4 5 4 4 1 4 4 3 2 4 1 4 2 5 1 4 1 4 2
[415] 4 2 1 4 1 5 2 5 5 1 1 1 3 1 3 3 2 1 2 2 2 1 1 5 4 2 2 5 3 4 5 4 1 5 1 2 1 5 2 2 2 2 4 1 3 4 4 5 5 2 3 3 1 1 4 1 3 2 5 3 2 1
[484] 1 2 2 5 5 5 2 2 1 5 4 2 5 4 2 3 2 5 4 1 1 5 3 1 2 5 4 5 1 4 4 2 1 4 1 1 2 2 2 5 5 2 2 5 2 1 5 3 5 2 4 5 3 1 2 5 1 2 5 1 5 5
[553] 2 2 1 4 3 4 3 2 3 2 5 4 2 2 4 2 3 2 2 3 3 2 1 5 1 5 1 2 1 5 2 2 5 2 2 3 1 5 4 1 3 3 3 3 1 2 1 3 5 2 5 3 3 4 5 2 3 5 2 1 4 5 5 2 1 4 3 3 1
[622] 5 4 4 4 4 2 5 1 1 3 3 4 3 1 4 2 2 4 2 1 2 5 3 5 4 3 5 1 1 1 3 1 2 3 1 1 2 3 1 3 5 2 5 3 3 4 1 5 2 3 2 4 3 3 2 1 2 4 3 2 1 5 1 3 3 1 4 4 5
[691] 5 5 2 1 2 4 3 1 3 1 3 5 4 1 4 3 5 5 3 2 1 1 1 5 4 2 2 2 1 2 1 3 1 1 3 5 4 3 5 1 2 5 2 3 3 3 5 5 2 5 4 2 5 2 4 5 1 2 2 2 5 4 3 3 2 2 2 1
[760] 5 2 5 4 4 4 3 3 3 3 5 3 3 1 4 2 3 5 5 4 4 3 2 5 5 1 3 3 1 4 3 3 3 4 4 4 1 4 4 2 4 1 5 3 5 5 2 5 3 2 3 1 2 1 5 4 3 3 2 2 5 2 2 1 3 3 2 5 3
[829] 2 5 2 2 2 5 2 4 4 4 3 4 1 1 5 4 4 1 5 1 5 5 4 3 3 1 5 2 3 1 4 1 1 5 4 4 3 2 4 5 4 5 5 3 3 2 5 5 4 4 2 5 1 2 4 3 4 1 1 3 1 3 3 3 5 5 5 5 5
[898] 3 1 5 4 3 1 4 3 4 4 5 4 5 2 1 2 4 3 5 3 1 1 1 5 4 5 5 2 5 5 1 5 1 5 3 2 5 3 4 5 5 2 1 3 2 1 4 1 1 2 4 3 3 2 4 1 1 3 3 2 5 5 1 4 3 4 3
[967] 2 3 5 3 1 1 5 5 2 5 2 4 3 1 2 3 5 2 2 2 4 4 4 5 1 5 5 3 2 3 5 5 4
[reached getOption("max.print") -- omitted 7124 entries]
within cluster sum of squares by cluster:
[1] 356924750 357584500 357584500 357584500 357584500
(between_SS / total_SS = 96.0 %)
Available components:
[1] "cluster" "centers" "totss" "withinss" "tot.withinss" "betweenss" "size" "iter" "ifault"
Environment History Connections
Import Dataset
Global Environment
Data
kc3 List of 9
kc5 List of 9
knn_test_data 2438 obs. of 22 variables
knn_train_data 5686 obs. of 22 variables
mr 8124 obs. of 23 variables
mr.df 8124 obs. of 23 variables
mrf 8124 obs. of 23 variables
mrftest.df 2438 obs. of 23 variables
mrftrain.df 5686 obs. of 23 variables
Values
dataset.df int [1:8124] 6937 5321 2175 5945 1446 2975 6755 8032 397 ...
knn_pred Factor w/ 2 levels "0","1": 2 1 1 1 1 1 2 2 1 1 ...
knn_test_label chr [1:2438] "1" "0" "0" "0" "0" "0" "1" "1" "0" "0" "0" ...
Files Plots Packages Help Viewer
Install Update
Name Description Version
User Library
class Functions for Classification 7.3-14
e1071 Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien 1.6-8
gdata Various R Programming Tools for Data Manipulation 2.18.0
gmodels Various R Programming Tools for Model Fitting 2.16.2
gtools Various R Programming Tools 3.5.0
igraph Network Analysis and Visualization 1.2.1
irlba Fast Truncated Singular Value Decomposition and Principal Components Analysis for Large Dense and Sparse Matrices 2.3.2
magrittr A Forward-Pipe Operator for R 1.5
permute Functions for Generating Restricted Permutations of Data 0.9-4
pkgconfig Private Configuration for 'R' Packages 2.0.1
vegan Community Ecology Package 2.5-1
System Library
boot Bootstrap Functions (Originally by Angelo Canty for S) 1.3-20
class Functions for Classification 7.3-14
cluster "Finding Groups in Data": Cluster Analysis Extended Rousseeuw et al. 2.0.6
codetools Code Analysis Tools for R 0.2-15
compiler The R Compiler Package 3.4.3
datasets The R Datasets Package 3.4.3
foreign Read Data Stored by 'Minitab', 'S', 'SAS', 'SPSS', 'Stata', 'Systat', 'Weka', 'dBase', ... 0.8-69
graphics The R Graphics Package 3.4.3
grDevices The R Graphics Devices and Support for Colours and Fonts 3.4.3
grid The Grid Graphics Package 3.4.3

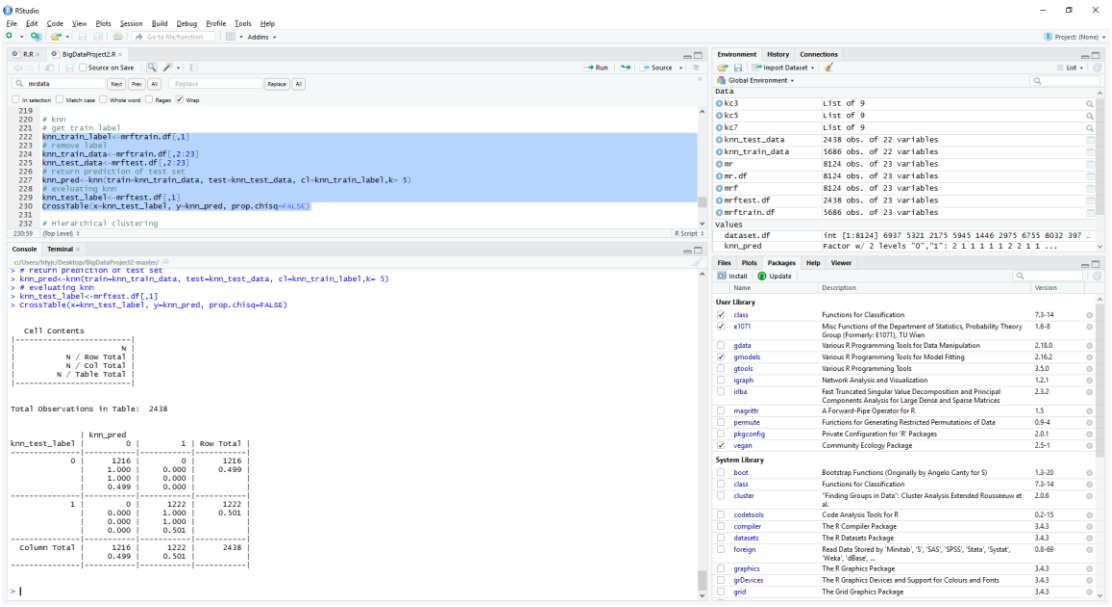

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
mrdata
#mrftrain.df<-mrf[train.df,]
#mrftrain.df<-mrf[train.df,]
#mrftest set
mrftest.df<-mrf[-train.df,]
#mrftest.df<-mrf[-train.df,]
#mrftest.df<-mrf[-train.df,]
# k-means (no need train and test dataset)
kc3<-kmeans(dataset.df,3)
#kc5<-kmeans(dataset.df,5)
#kc7<-kmeans(dataset.df,7)
knn
Console Terminal
K-means clustering with 7 clusters of sizes 1161, 1160, 1159, 1161, 1160, 1161, 1162
Cluster means:
[,1]
1 5221.0
2 2899.5
3 580.0
4 4060.0
5 1739.5
6 6382.0
7 7543.5
Clustering vector:
[1] 6 1 5 6 5 2 6 7 3 1 3 5 3 7 1 4 7 2 7 4 3 2 1 3 3 7 5 4 3 5 4 2 4 4 3 3 3 6 7 1 4 2 5 4 3 6 1 7 6 4 3 3 5 1 1 6 5 3 6 2 4 6 1 5 3 1 4 6 3
[70] 3 6 1 7 3 7 3 5 5 7 6 1 5 1 5 7 2 4 7 5 3 3 2 4 5 7 6 6 2 1 1 3 5 2 4 6 6 5 2 7 4 2 6 4 5 1 4 3 5 3 7 4 1 3 5 1 3 3 2 3 4 3 4 6 4 3 3 4 2
[139] 2 5 1 5 7 7 2 1 5 4 6 4 5 1 7 1 4 3 1 3 7 1 2 5 7 6 6 6 6 5 4 3 2 6 3 7 7 3 4 5 5 1 3 4 4 6 5 2 1 1 4 3 2 7 3 2 4 2 6 4 6 2 7 7 6 6 4 7 2
[208] 4 5 6 2 6 1 7 2 5 7 5 2 2 1 2 6 6 5 6 1 5 7 7 1 1 3 1 5 7 7 6 2 6 2 4 1 3 5 5 2 4 6 2 1 3 7 2 4 5 4 2 6 4 1 6 6 5 4 1 1 1 7 3 6 7 4 4 1
[277] 6 4 2 6 6 5 6 1 3 7 1 5 1 3 4 3 4 3 1 5 7 1 2 3 5 6 4 7 6 5 5 7 7 7 3 6 3 7 1 6 7 1 3 6 6 3 7 7 2 6 3 2 1 1 7 7 3 5 5 2 2 1 5 3 6 1 3 3 1
[346] 4 1 3 7 3 6 5 7 1 4 3 4 3 6 3 3 1 4 5 5 5 6 5 5 4 3 6 2 3 7 5 1 4 5 5 4 7 7 3 2 2 6 6 7 3 4 3 6 1 3 7 6 7 6 3 6 7 4 2 7 5 7 2 1 3 7 3 7 2
[415] 6 5 3 6 3 1 2 1 6 3 3 3 4 3 4 4 2 3 5 2 2 3 3 1 2 7 5 5 1 4 7 1 7 3 1 5 2 6 1 7 5 6 5 2 3 6 2 5 2 2 7 3 4 6 6 6 6 2 4 1 3 3 7 3 2 2 1 4 2 3
[484] 3 5 2 1 1 6 2 1 2 2 3 1 6 4 1 1 6 5 4 2 1 7 5 3 1 4 3 5 6 7 1 3 7 7 2 5 7 3 3 4 5 2 5 1 6 5 2 6 4 2 6 2 3 6 2 6 5 6 1 4 3 5 1 3 5 1 5 1 1
[553] 2 2 5 7 4 7 4 2 4 5 6 7 2 5 7 5 4 2 5 1 1 5 3 1 3 6 3 1 2 5 6 2 5 1 2 2 5 1 6 3 4 2 4 1 3 2 3 2 1 2 4 4 7 6 6 5 4 1 5 5 7 1 1 2 3 7 4 2 3
[622] 6 7 7 7 6 5 6 5 5 1 4 7 4 5 7 2 2 7 2 3 2 6 4 1 6 4 1 3 5 3 1 3 2 4 5 4 1 5 3 4 4 7 3 1 5 2 2 7 4 4 2 5 2 7 4 5 3 1 3 4 4 3 6 7 6
[691] 1 6 2 3 5 7 4 3 4 3 4 1 6 5 6 4 1 1 2 5 5 3 5 6 7 5 5 5 5 3 4 3 3 4 1 6 4 6 5 5 1 2 4 4 4 6 1 5 1 5 6 5 1 2 7 1 5 2 5 2 1 7 2 1 2 2 2 3
[760] 6 2 1 6 7 7 4 2 4 4 6 2 4 3 7 2 2 1 6 6 7 4 5 6 1 3 4 4 5 7 4 4 4 7 7 6 5 7 7 2 7 3 6 2 6 6 5 1 4 2 4 5 5 5 1 6 4 4 2 2 6 5 5 3 4 4 2 6 4
[829] 5 1 2 5 2 1 2 7 7 7 2 6 5 5 1 7 7 3 6 5 1 1 7 4 1 3 6 2 1 3 7 3 3 1 7 7 4 5 7 6 7 6 6 1 4 2 1 1 7 7 5 1 5 2 7 4 7 3 3 4 3 4 4 4 6 6 6 1 1
[898] 4 3 6 7 4 3 7 2 7 7 6 6 1 5 3 5 7 4 1 4 3 3 5 6 7 1 1 2 1 1 3 1 3 6 2 5 6 1 2 2 7 1 6 2 3 4 2 5 6 5 3 5 7 4 4 5 6 3 3 1 2 2 1 1 5 7 4 7 4
[967] 5 1 6 2 1 3 6 1 5 6 5 7 4 3 5 2 6 2 2 5 6 7 2 7 1 5 6 1 4 2 4 6 6 7
[reached getOption("max.print") -- omitted 7124 entries]
within cluster sum of squares by cluster:
[1] 130411260 130074570 129738460 130411260 130074570 130411260 130748531
(between_SS / total_SS = 98.0 %)
Available components:
[1] "cluster" "centers" "totss" "withinss" "tot.withinss" "betweenss" "size" "iter" "ifault"
Environment History Connections
Import Dataset
Global Environment
Data
kc3 List of 9
kc5 List of 9
kc7 List of 9
knn_test_data 2438 obs. of 22 variables
knn_train_data 5686 obs. of 22 variables
mr 8124 obs. of 23 variables
mr.df 8124 obs. of 23 variables
mrf 8124 obs. of 23 variables
mrftest.df 2438 obs. of 23 variables
mrftrain.df 5686 obs. of 23 variables
Values
dataset.df int [1:8124] 6937 5321 2175 5945 1446 2975 6755 8032 397 ...
knn_pred Factor w/ 2 levels "0","1": 2 1 1 1 1 1 2 2 1 1 ...
Files Plots Packages Help Viewer
Install Update
Name Description Version
User Library
class Functions for Classification 7.3-14
e1071 Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien 1.6-8
gdata Various R Programming Tools for Data Manipulation 2.18.0
gmodels Various R Programming Tools for Model Fitting 2.16.2
gtools Various R Programming Tools 3.5.0
igraph Network Analysis and Visualization 1.2.1
irlba Fast Truncated Singular Value Decomposition and Principal Components Analysis for Large Dense and Sparse Matrices 2.3.2
magrittr A Forward-Pipe Operator for R 1.5
permute Functions for Generating Restricted Permutations of Data 0.9-4
pkgconfig Private Configuration for 'R' Packages 2.0.1
vegan Community Ecology Package 2.5-1
System Library
boot Bootstrap Functions (Originally by Angelo Canty for S) 1.3-20
class Functions for Classification 7.3-14
cluster "Finding Groups in Data": Cluster Analysis Extended Rousseeuw et al. 2.0.6
codetools Code Analysis Tools for R 0.2-15
compiler The R Compiler Package 3.4.3
datasets The R Datasets Package 3.4.3
foreign Read Data Stored by 'Minitab', 'S', 'SAS', 'SPSS', 'Stata', 'Systat', 'Weka', 'dBase', ... 0.8-69
graphics The R Graphics Package 3.4.3
grDevices The R Graphics Devices and Support for Colours and Fonts 3.4.3
grid The Grid Graphics Package 3.4.3

Knn：



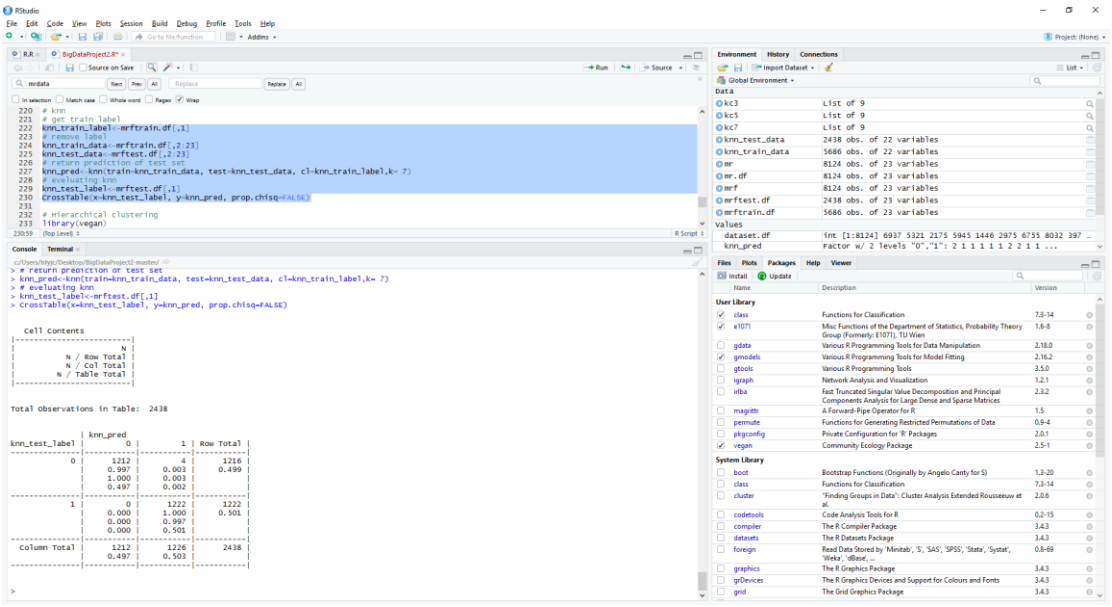
```
# knn
# get train label
knn_train_label<-mrftrain.df[,1]
# remove label
knn_train_data<-mrftrain.df[,2:23]
knn_test_data<-mrftest.df[,2:23]
# return prediction of test set
knn_pred<-knn(train=knn_train_data, test=knn_test_data, cl=knn_train_label,k= 3)
# evaluating knn
knn_test_label<-mrftest.df[,1]
CrossTable(x=knn_test_label, y=knn_pred, prop.chisq=FALSE)

# Hierarchical clustering
```

```
> # return prediction of test set
> knn_pred<-knn(train=knn_train_data, test=knn_test_data, cl=knn_train_label,k= 3)
> # evaluating knn
> knn_test_label<-mrftest.df[,1]
> CrossTable(x=knn_test_label, y=knn_pred, prop.chisq=FALSE)


   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  2438


             | knn_pred
knn_test_label |         0 |         1 | Row Total |
---------------|-----------|-----------|-----------|
             0 |      1216 |         0 |      1216 |
               |     1.000 |     0.000 |     0.499 |
               |     1.000 |     0.000 |           |
               |     0.499 |     0.000 |           |
---------------|-----------|-----------|-----------|
             1 |         0 |      1222 |      1222 |
               |     0.000 |     1.000 |     0.501 |
               |     0.000 |     1.000 |           |
               |     0.000 |     0.501 |           |
---------------|-----------|-----------|-----------|
  Column Total |      1216 |      1222 |      2438 |
               |     0.499 |     0.501 |           |
---------------|-----------|-----------|-----------|
```



```
# knn
# get train label
knn_train_label<-mrftrain.df[,1]
# remove label
knn_train_data<-mrftrain.df[,2:23]
knn_test_data<-mrftest.df[,2:23]
# return prediction of test set
knn_pred<-knn(train=knn_train_data, test=knn_test_data, cl=knn_train_label,k= 5)
# evaluating knn
knn_test_label<-mrftest.df[,1]
CrossTable(x=knn_test_label, y=knn_pred, prop.chisq=FALSE)

# Hierarchical clustering
```

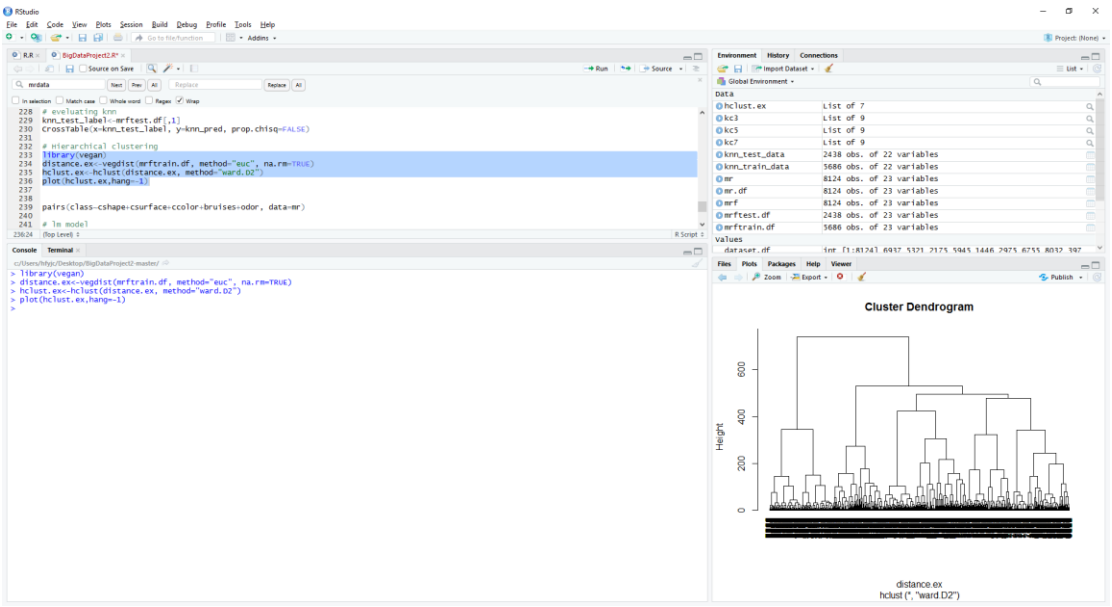
```
> # return prediction of test set
> knn_pred<-knn(train=knn_train_data, test=knn_test_data, cl=knn_train_label,k= 5)
> # evaluating knn
> knn_test_label<-mrftest.df[,1]
> CrossTable(x=knn_test_label, y=knn_pred, prop.chisq=FALSE)


   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  2438


             | knn_pred
knn_test_label |         0 |         1 | Row Total |
---------------|-----------|-----------|-----------|
             0 |      1216 |         0 |      1216 |
               |     1.000 |     0.000 |     0.499 |
               |     1.000 |     0.000 |           |
               |     0.499 |     0.000 |           |
---------------|-----------|-----------|-----------|
             1 |         0 |      1222 |      1222 |
               |     0.000 |     1.000 |     0.501 |
               |     0.000 |     1.000 |           |
               |     0.000 |     0.501 |           |
---------------|-----------|-----------|-----------|
  Column Total |      1216 |      1222 |      2438 |
               |     0.499 |     0.501 |           |
---------------|-----------|-----------|-----------|
```



```
# knn
# get train label
knn_train_label<-mrftrain.df[,1]
# remove label
knn_train_data<-mrftrain.df[,2:23]
knn_test_data<-mrftest.df[,2:23]
# return prediction of test set
knn_pred<-knn(train=knn_train_data, test=knn_test_data, cl=knn_train_label,k= 7)
# evaluating knn
knn_test_label<-mrftest.df[,1]
CrossTable(x=knn_test_label, y=knn_pred, prop.chisq=FALSE)

# Hierarchical clustering
library(vegan)
```

```
> # return prediction of test set
> knn_pred<-knn(train=knn_train_data, test=knn_test_data, cl=knn_train_label,k= 7)
> # evaluating knn
> knn_test_label<-mrftest.df[,1]
> CrossTable(x=knn_test_label, y=knn_pred, prop.chisq=FALSE)


   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  2438


             | knn_pred
knn_test_label |         0 |         1 | Row Total |
---------------|-----------|-----------|-----------|
             0 |      1212 |         4 |      1216 |
               |     0.997 |     0.003 |     0.499 |
               |     1.000 |     0.003 |           |
               |     0.497 |     0.002 |           |
---------------|-----------|-----------|-----------|
             1 |         0 |      1222 |      1222 |
               |     0.000 |     1.000 |     0.501 |
               |     0.000 |     0.997 |           |
               |     0.000 |     0.501 |           |
---------------|-----------|-----------|-----------|
  Column Total |      1212 |      1226 |      2438 |
               |     0.497 |     0.503 |           |
---------------|-----------|-----------|-----------|

>
```

Hierarchical clustering

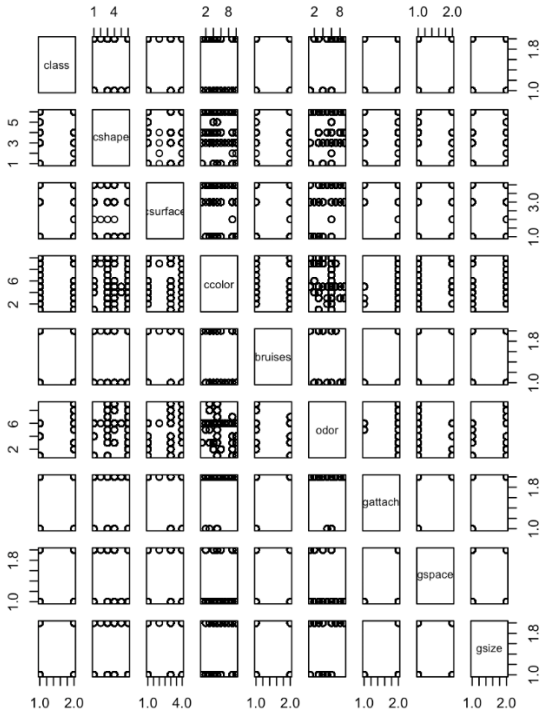


Discussion of results from the experiments that you run in part 3b above. Which clustering version gives the best results? What seems to be the best number of clusters for each method?

A clustering of the samples into 3 and 5 gives the best results, because k-nearest neighbor classification for test set from training set have a good accuracy, they all have 100% accuracy rate.

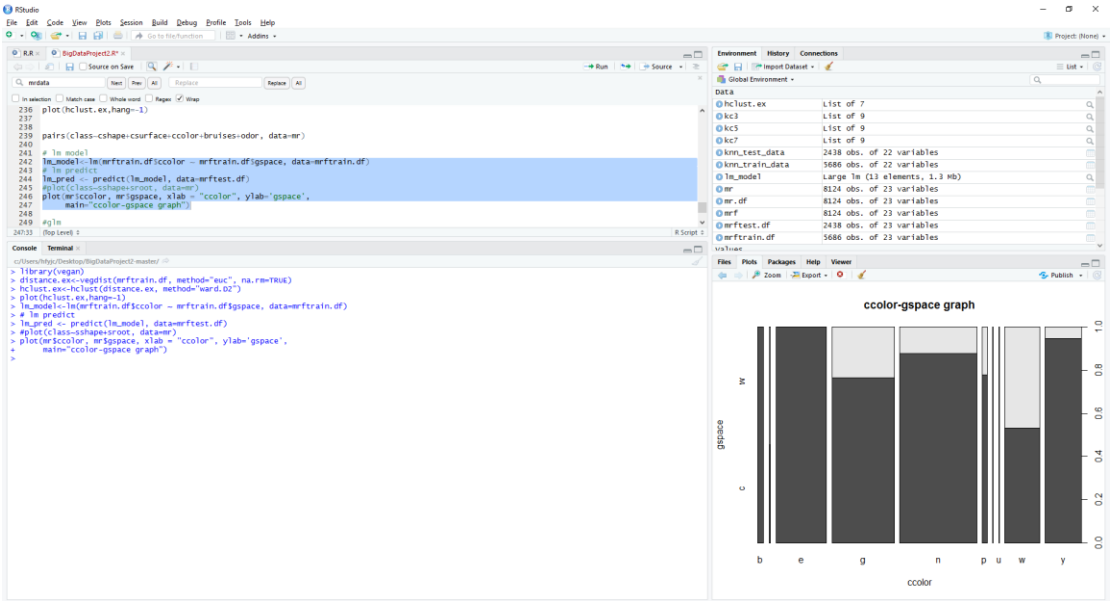
In our opinion, the KNN give us best result. Undefinedly, the other method also can give us a good result. But the KNN accuracy rate close to 100%.

Table from 4 above

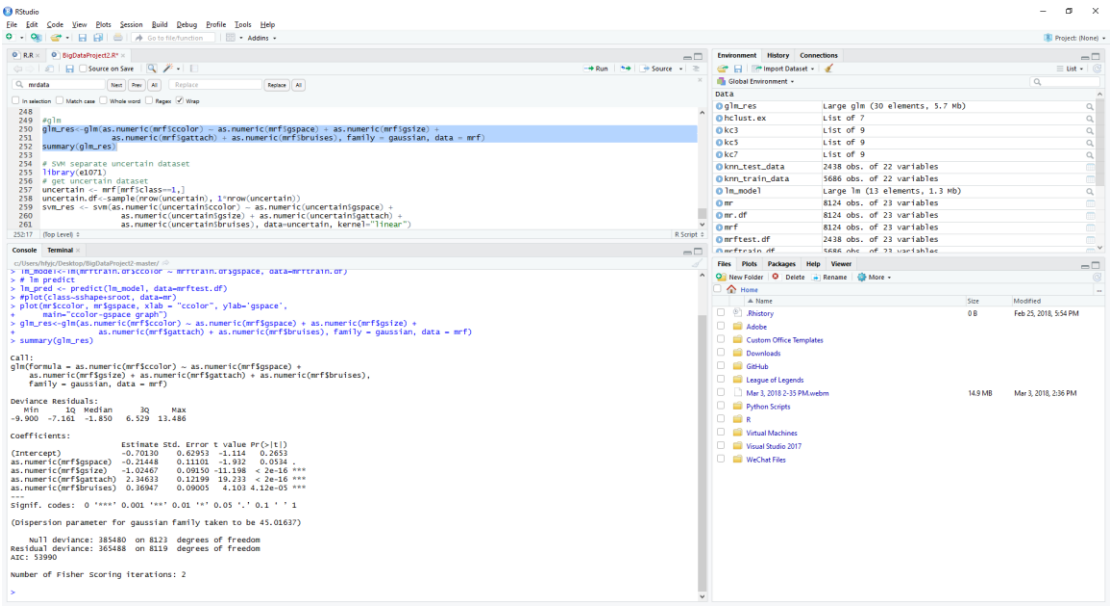


From above pair picture, we get some class could be compare and utilized.

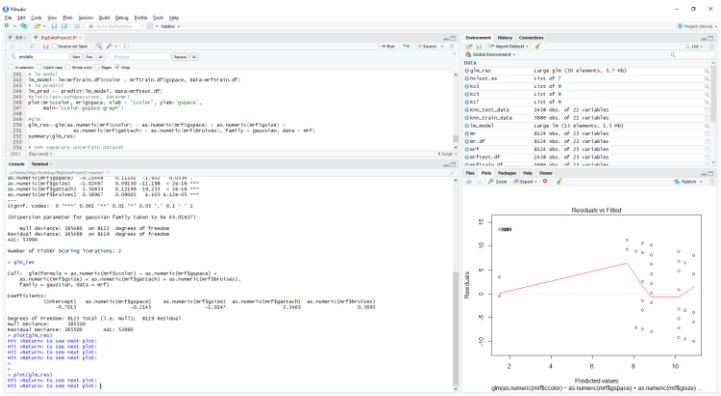
In lm model, we pick color and space:

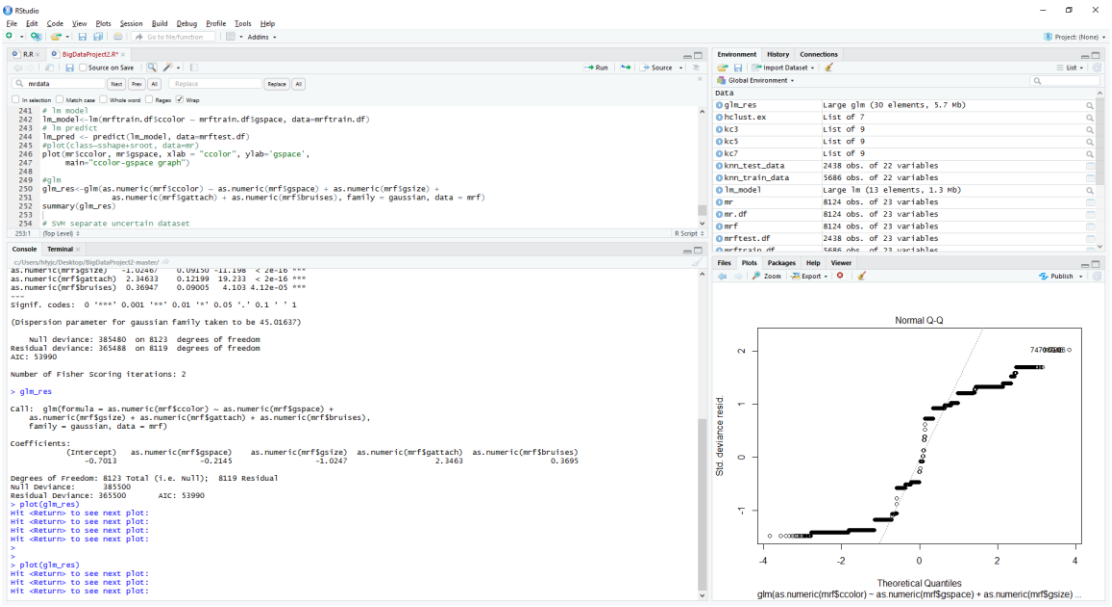


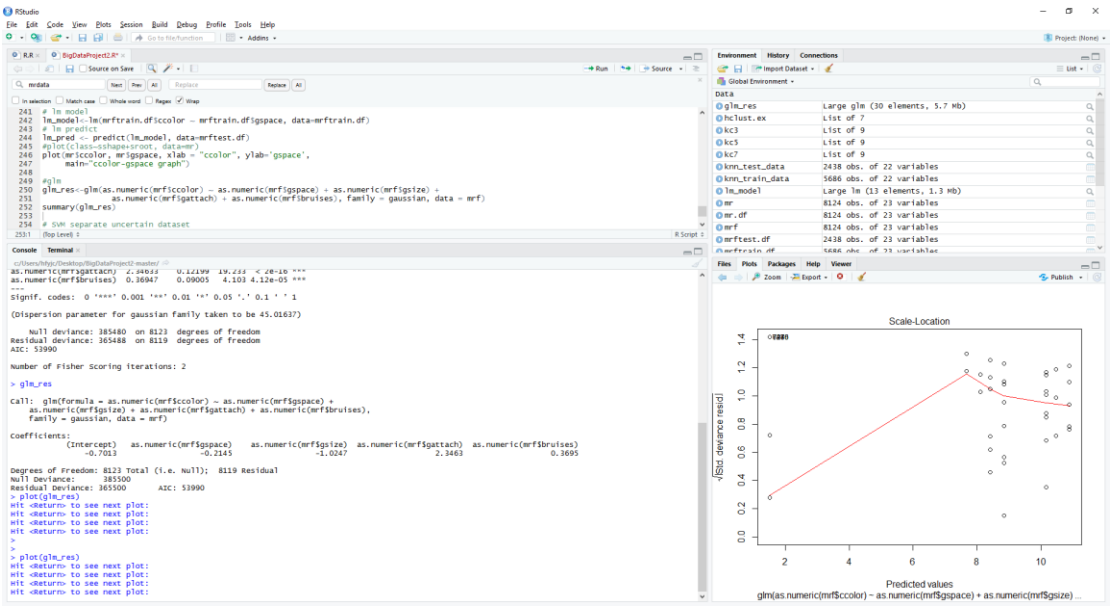
In glm model, we can pick more class: color, space, size, attach, bruises:

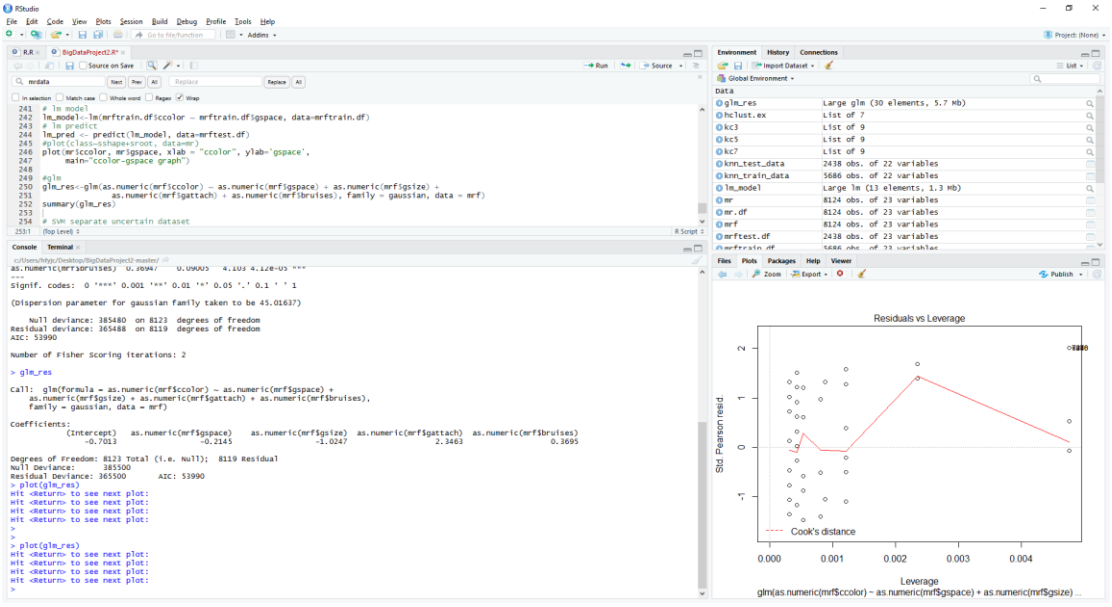


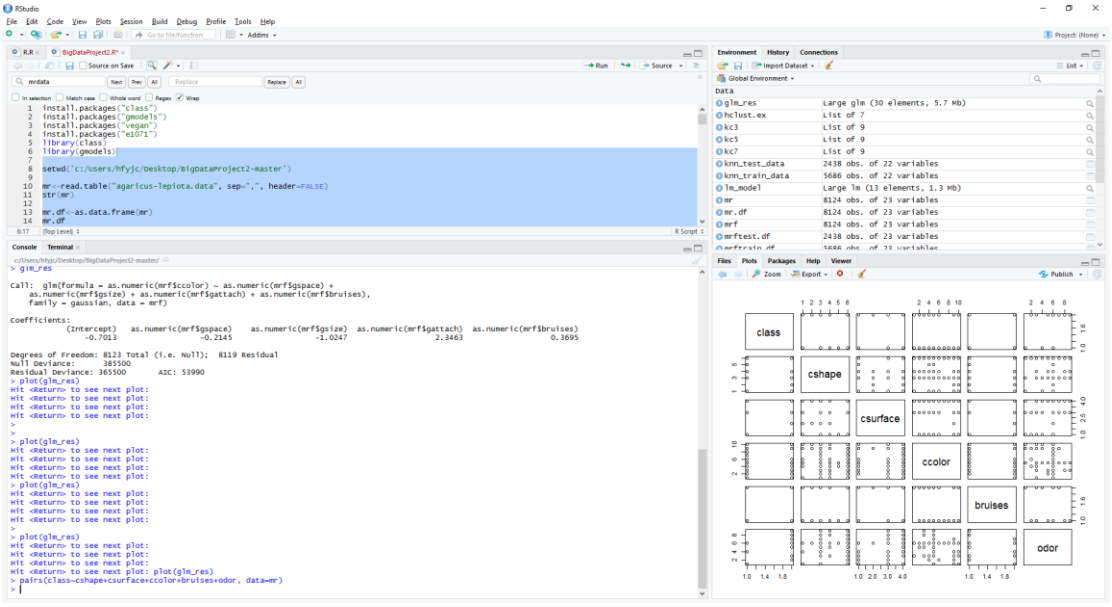
Plots with discussion of results: which plot helped you understand the data best and why?






Scale-Location
√|Std. deviance resid.|
Predicted values
glm(as.numeric(mrf$ccolor) ~ as.numeric(mrf$gspace) + as.numeric(mrf$gsize) ...


Residuals vs Leverage
Std. Pearson resid.
Cook's distance
Leverage
glm(as.numeric(mrf$ccolor) ~ as.numeric(mrf$gspace) + as.numeric(mrf$gsize) ...


class
cshape
csurface
ccolor
bruises
odor

The above picture is best plot for us to understand the data. Because, from this picture, they show all relationship between different class. From this picture, we can find which class can help us to distinguish the kind of mushroom. We compare the picture, and try to find which two class have linear relationship, and then use those data further.

Analysis of what this project helped you learn about data science, e.g., the exploration of data which is what you have been doing – 1 point. You must argue persuasively.

First, we learn some new function and how to use it. Based on those function, we can get some graph and data what we need for analyses. Then the most important things is know how to analyses those graph, and data. We have those data, then convert to different table or graph, then we make assumption from that information. And then we try to prove it or find some new viewpoint. We work this as a loop, so that improve our ability of integrating and analyzing data.