19MID0020 Cycle-Sheet:3

Importing the libraries

```
library(datasets)
library(ggplot2)
library("e1071") ## Naive-Bayes classifier

library(caTools)
library(class)

library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ClusterR)
```

```
## Loading required package: gtools
```

```
##
## Attaching package: 'gtools'
```

```
## The following object is masked from 'package:e1071':
##
##     permutations
```

```
library(cluster)

library(arules)
```

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

```
##
## Attaching package: 'arules'
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

```
library(arulesViz)
library(RColorBrewer)
```

Importing the data-set

```
df = iris
head(df)
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
	<dbl>	<dbl>	<dbl>	<dbl>	<fct>
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa

6 rows

Pre-Processing the data-set

```
str(df)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
## Number of NaN values in the data-set
sum(is.nan(as.matrix(df)))
```

```
## [1] 0
```

Splitting into train and test data

```
n = nrow(df)
split = sample(c(TRUE, FALSE), n, replace=TRUE, prob=c(0.75, 0.25))

df_train = df[split, ]
df_test  = df[!split, ]
```

```
print(nrow(df_train))
```

```
## [1] 110
```

```
print(nrow(df_test))
```

```
## [1] 40
```

1)Naive Bayes Classifier

```
## Fitting the training data into the model
model = naiveBayes(Species~.,data=df_train)
pred  = predict(model,df_test)
```

```
cat("Predicted value from the trainning data")
```

```
## Predicted value from the trainning data
```

```
table(pred)
```

```
## pred
##     setosa versicolor  virginica
##         10         18         12
```

```
cat("\nActual value in the test-data")
```

```
##
## Actual value in the test-data
```

```
table(df_test$Species)
```

```
##
##     setosa versicolor  virginica
##         10         17         13
```

```
## Confusion matrix
table(pred,df_test$Species)
```

```
##
## pred         setosa versicolor virginica
##   setosa         10          0         0
##   versicolor      0         16         2
##   virginica       0          1        11
```

2)Linear Regression

```
## Separating into Dependent and In-Dependent attributes
x = df_train$Sepal.Width
y = df_train$Sepal.Length

## Fitting the trainning data into the model
linear_model = lm(x~y, data=df_train)
summary(linear_model)
```

```
##
## Call:
## lm(formula = x ~ y, data = df_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.16301 -0.25017 -0.01235  0.26956  1.08305
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.62359    0.28572  12.683   <2e-16 ***
## y           -0.09212    0.04832  -1.906   0.0592 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4311 on 108 degrees of freedom
## Multiple R-squared:  0.03256,    Adjusted R-squared:  0.0236
## F-statistic: 3.635 on 1 and 108 DF,  p-value: 0.05925
```

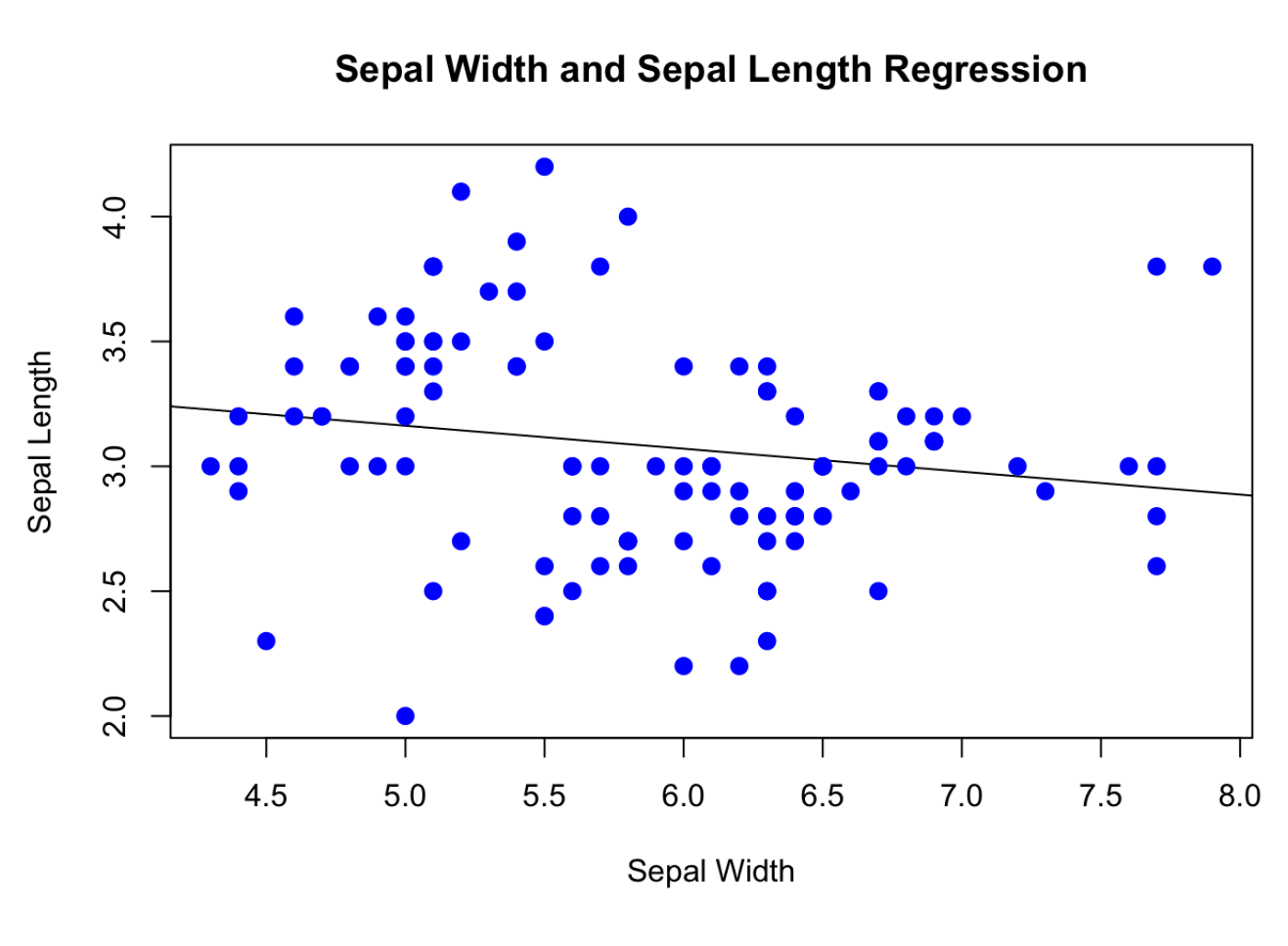
```
result = predict(linear_model,df_test)
```

```
## Warning: 'newdata' had 40 rows but variables found have 110 rows
```

```
summary(result)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.896   3.034   3.089   3.085   3.154   3.227
```

```
# Plot the chart.
plot(y,x,col = "blue",main = "Sepal Width and Sepal Length Regression",abline(linear_model),
     cex = 1.3,pch = 16, xlab = "Sepal Width", ylab = "Sepal Length")
```



3)K-Nearest Neighbor Classifier

```
## Scaling the independent feature values for both training and testing data
df_train_scale = scale(df_train[, 1:4])
df_test_scale  = scale(df_test[, 1:4])
```

```
## Fitting the trainning data into the model
classifier_knn <- knn(train = df_train_scale, test = df_test_scale,
                      cl = df_train$Species,k = 3)
classifier_knn
```

```
##  [1] setosa     setosa     setosa     setosa     setosa     setosa
##  [7] setosa     setosa     setosa     setosa     versicolor versicolor
## [13] versicolor versicolor versicolor versicolor versicolor versicolor
## [19] versicolor versicolor versicolor versicolor versicolor versicolor
## [25] versicolor versicolor versicolor virginica  virginica  versicolor
## [31] virginica  virginica  virginica  virginica  virginica  versicolor
## [37] virginica  virginica  virginica  virginica
## Levels: setosa versicolor virginica
```

```
## confusion matrix
cm <- table(df_test$Species, classifier_knn)
cm
```

```
##             classifier_knn
##              setosa versicolor virginica
##   setosa         10          0         0
##   versicolor      0         17         0
##   virginica       0          2        11
```

```
## accuracy of the classifier
misClassError <- mean(classifier_knn == df_test$Species)
print(paste('Accuracy =', misClassError))
```

```
## [1] "Accuracy = 0.95"
```

4)K-Means Clustering

```
## in-dependent feature
x = df %>% select(-Species)

## dependent feature
y = df$Species
```

```
k_means = kmeans(x, centers = 3, nstart = 20)

k_means
```

```
## K-means clustering with 3 clusters of sizes 38, 62, 50
##
## Cluster means:
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1     6.850000    3.073684     5.742105    2.071053
## 2     5.901613    2.748387     4.393548    1.433871
## 3     5.006000    3.428000     1.462000    0.246000
##
## Clustering vector:
##   [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##  [38] 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [75] 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 1 1 1 1 2 1 1 1 1
## [112] 1 1 2 2 1 1 1 1 2 1 2 1 2 1 1 2 2 1 1 1 1 1 2 1 1 1 1 2 1 1 1 2 1 1 1 2 1
## [149] 1 2
##
## Within cluster sum of squares by cluster:
## [1] 23.87947 39.82097 15.15100
##  (between_SS / total_SS =  88.4 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
## Size of each cluster
k_means$size
```

```
## [1] 38 62 50
```

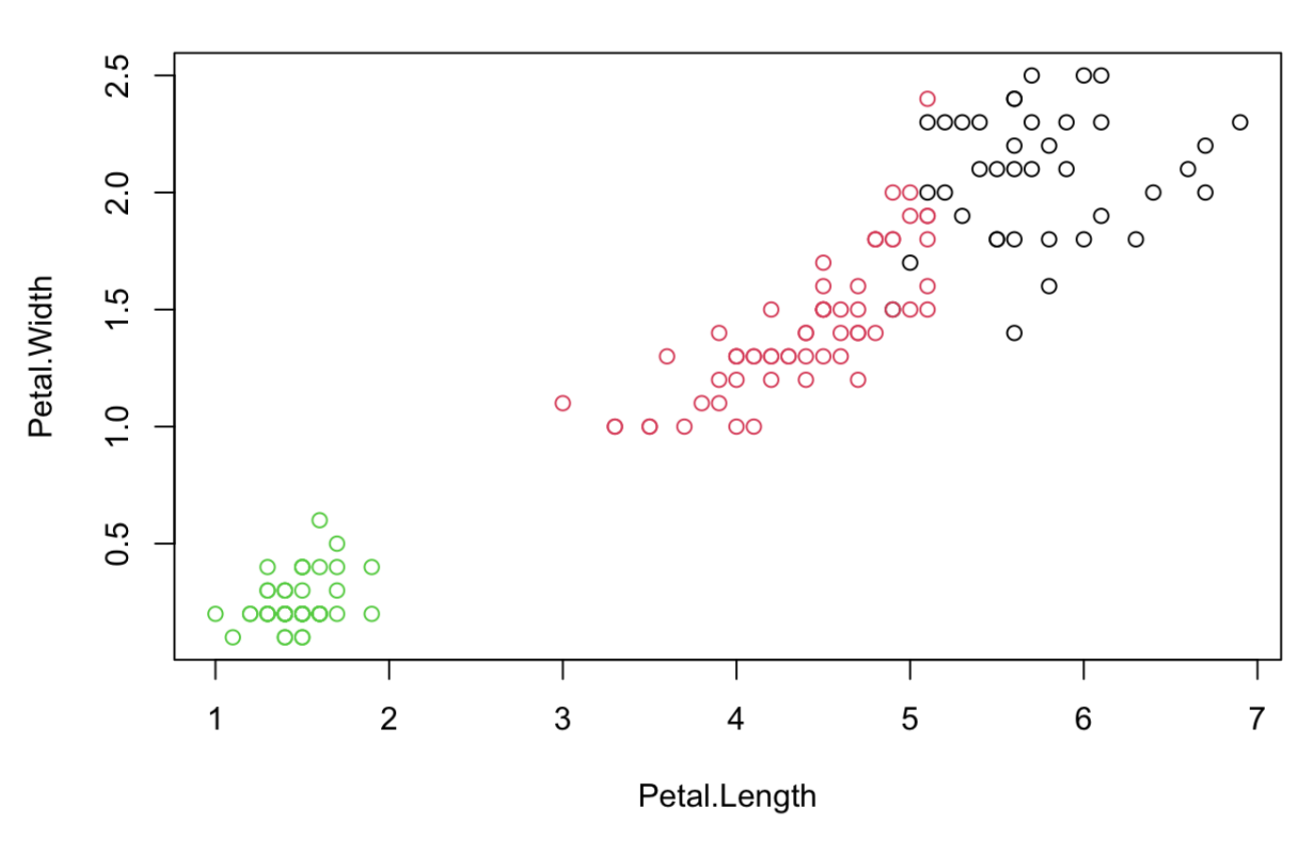
```
## Cluster identification for each observation
k_means$cluster
```

```
##   [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##  [38] 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [75] 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 1 1 1 1 2 1 1 1 1
## [112] 1 1 2 2 1 1 1 1 2 1 2 1 2 1 1 2 2 1 1 1 1 1 2 1 1 1 1 2 1 1 1 2 1 1 1 2 1
## [149] 1 2
```

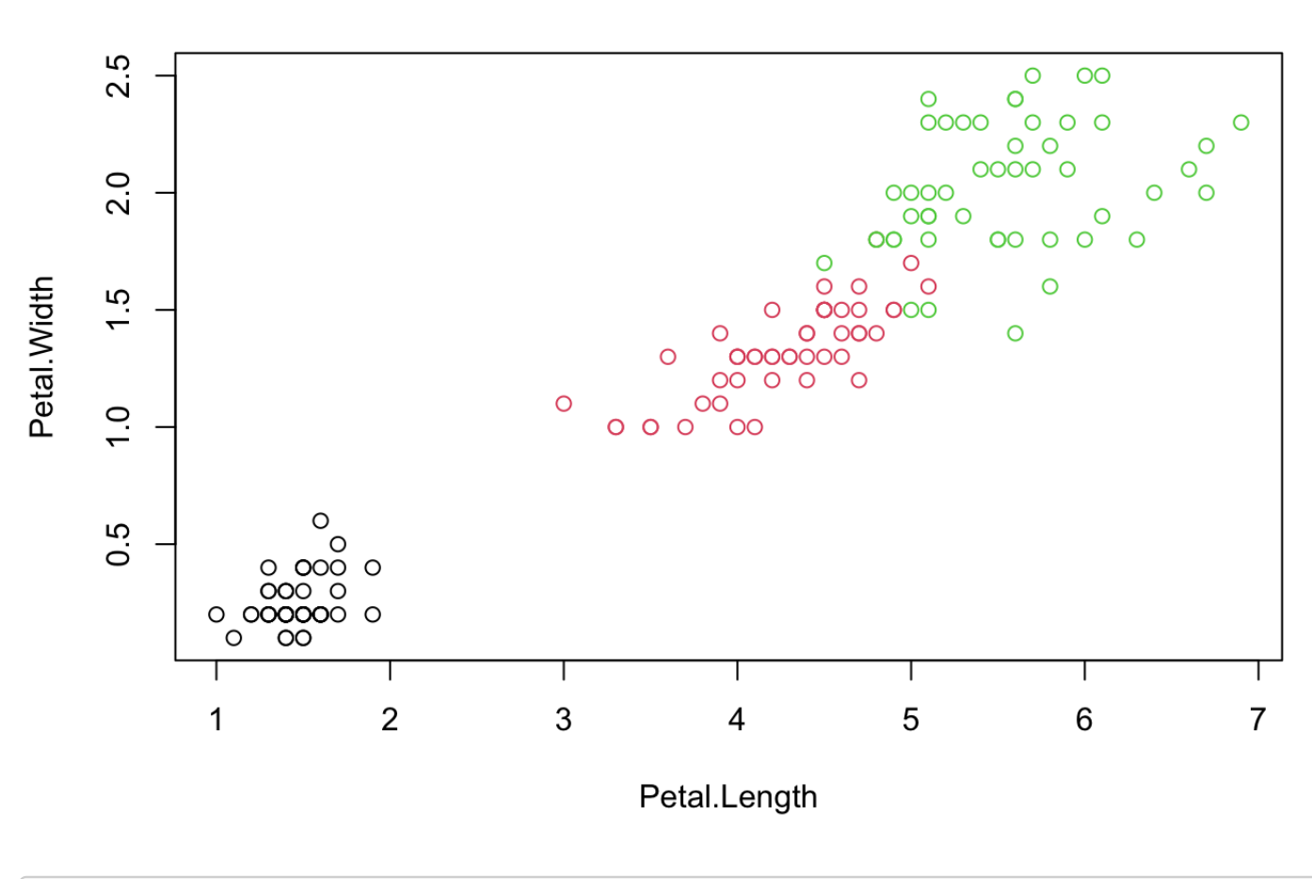
```
## Confusion Matrix
cm <- table(y, k_means$cluster)
cm
```

```
##
## y             1  2  3
##   setosa      0  0 50
##   versicolor  2 48  0
##   virginica  36 14  0
```

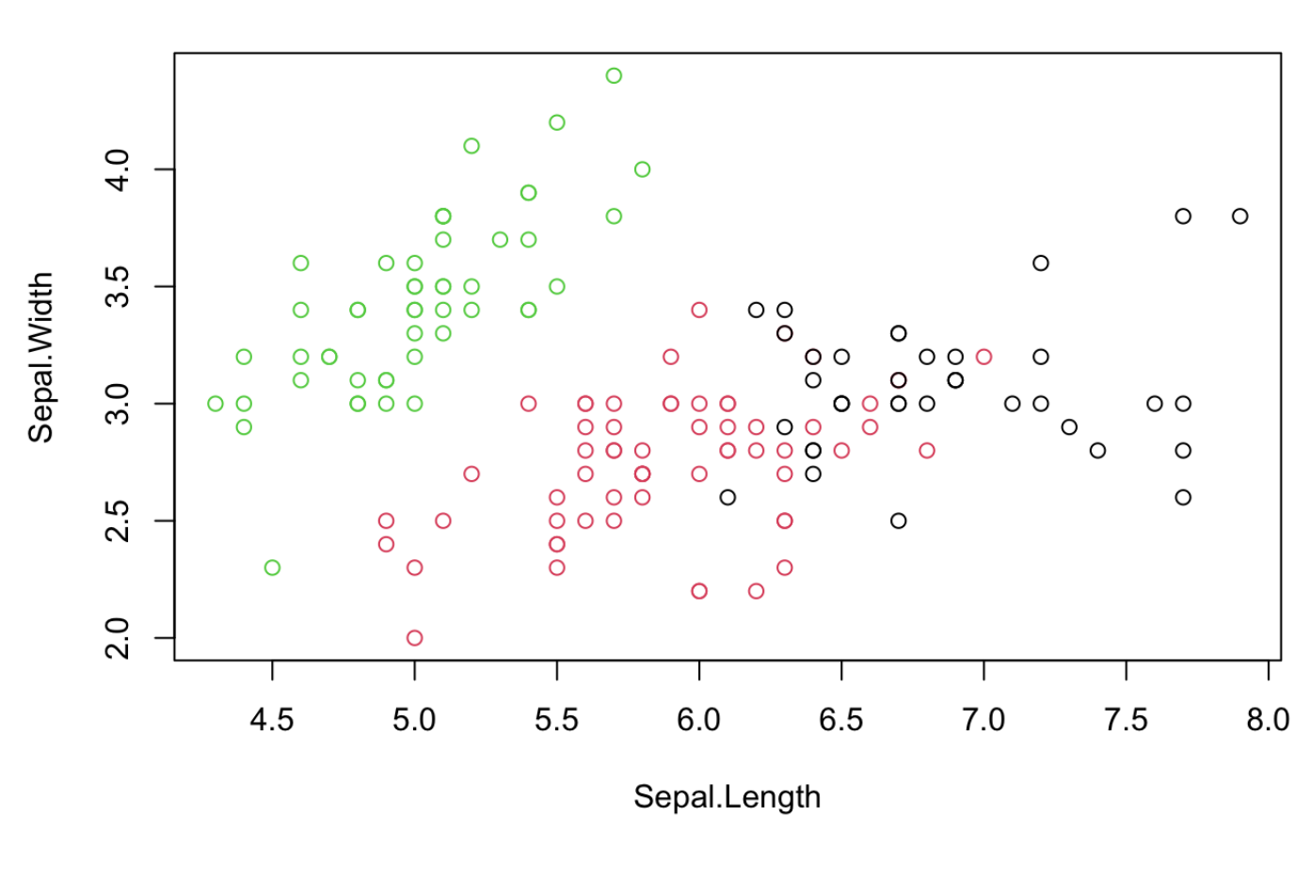
```
plot(iris[c("Petal.Length","Petal.Width")],col=k_means$cluster)
```



```
plot(iris[c("Petal.Length","Petal.Width")],col=y)
```



```
plot(iris[c("Sepal.Length","Sepal.Width")],col=k_means$cluster)
```



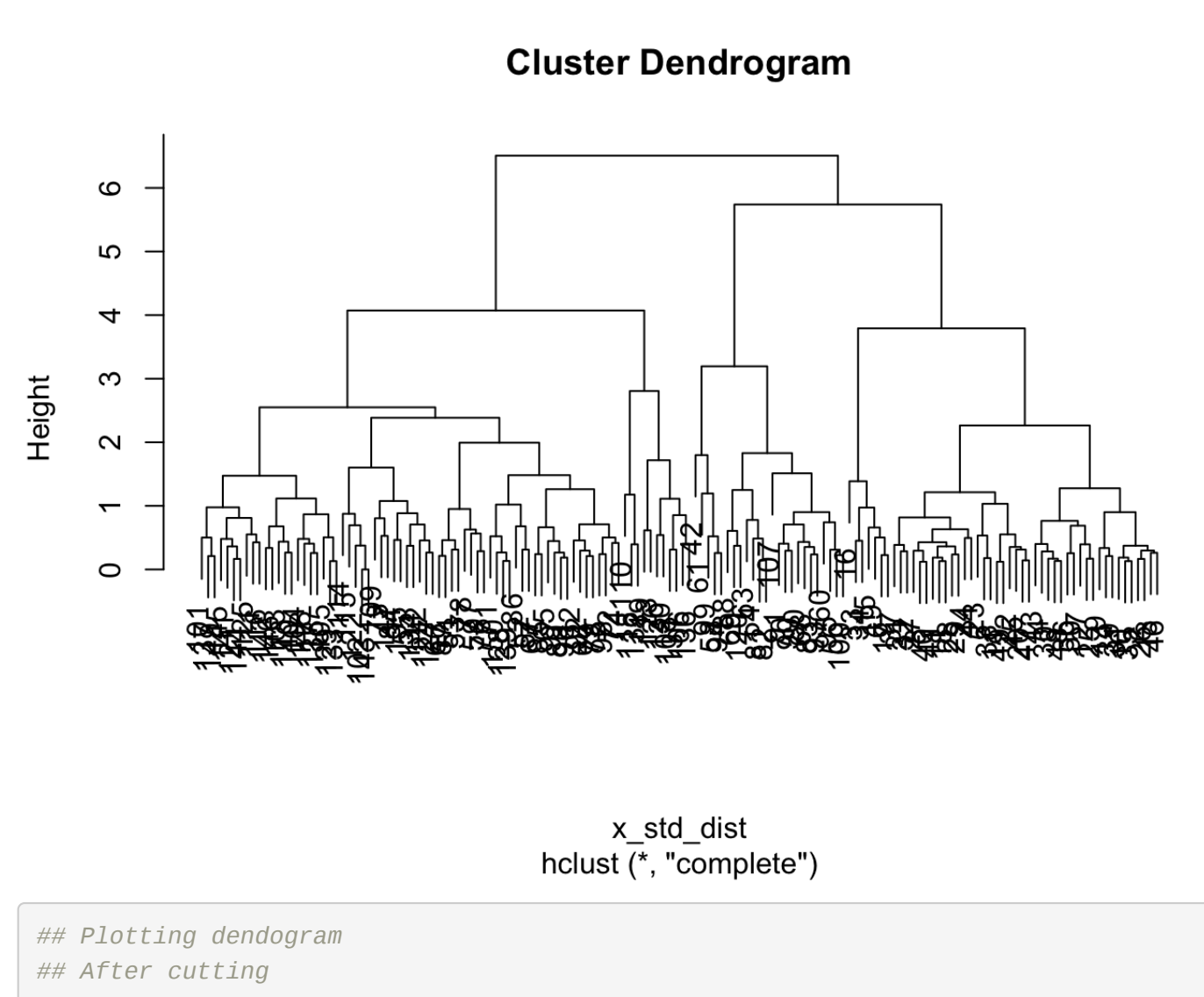
5)Hierarchial Clustering

```
## Standardizing the values
x_std_data = scale(x)

## Finding distance matrix
x_std_dist = dist(x_std_data)
```

```
myclust = hclust(x_std_dist,method="complete")
myclust
```

```
##
## Call:
## hclust(d = x_std_dist, method = "complete")
##
## Cluster method   : complete
## Distance         : euclidean
## Number of objects: 150
```

```
## Plotting dendogram
## Before cutting
plot(myclust)
```



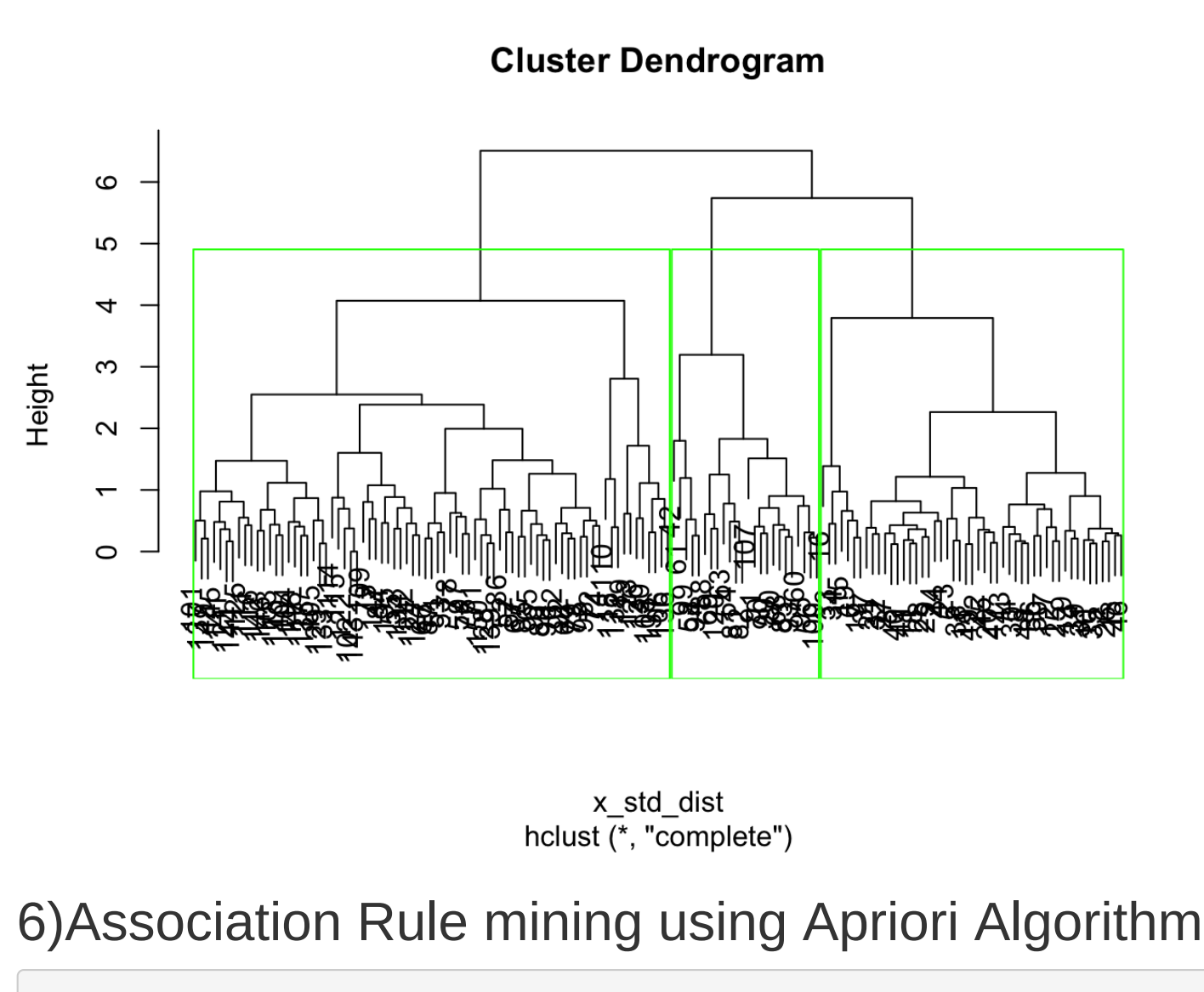
```
## Plotting dendogram
## After cutting
plot(myclust)

## Choosing no. of clusters
# Cutting tree by height
abline(h = 110, col = "green")

# Cutting tree by no. of clusters
fit <- cutree(myclust, k = 3 )
fit
```

```
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [38] 1 1 1 1 2 1 1 1 1 1 1 1 1 3 3 3 2 3 2 3 2 3 2 2 3 2 3 2 3 3 3 3 2 2 2 3 3 3 3
##  [75] 3 3 3 3 3 2 2 2 2 3 3 3 2 3 2 2 3 2 2 2 2 3 3 3 2 2 3 3 3 3 3 3 2 3 3 3 3
## [112] 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [149] 3 3
```

```
table(fit)
```

```
## fit
##  1  2  3
## 49 24 77
```

```
rect.hclust(myclust, k = 3, border = "green")
```



6)Association Rule mining using Apriori Algorithms

```
data('Groceries')
```

```
## using apriori() function
rules = apriori(Groceries, parameter = list(supp = 0.01, conf = 0.2))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.2    0.1    1 none FALSE            TRUE       5    0.01      1
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 98
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [88 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [232 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
rules
```

```
## set of 232 rules
```

```
## using inspect() function
inspect(rules[1:10])
```

```
##      lhs                rhs                support    confidence coverage
## [1]  {}              => {whole milk}       0.25551601 0.2555160  1.00000000
## [2]  {hard cheese}   => {whole milk}       0.01006609 0.4107884  0.02450432
## [3]  {butter milk}   => {other vegetables} 0.01037112 0.3709091  0.02796136
## [4]  {butter milk}   => {whole milk}       0.01159126 0.4145455  0.02796136
## [5]  {ham}           => {whole milk}       0.01148958 0.4414062  0.02602949
## [6]  {sliced cheese} => {whole milk}       0.01077783 0.4398340  0.02450432
## [7]  {oil}           => {whole milk}       0.01128622 0.4021739  0.02806304
## [8]  {onions}        => {other vegetables} 0.01423488 0.4590164  0.03101169
## [9]  {onions}        => {whole milk}       0.01209964 0.3901639  0.03101169
## [10] {berries}       => {yogurt}           0.01057448 0.3180428  0.03324860
##      lift     count
## [1]  1.000000 2513
## [2]  1.607682   99
## [3]  1.916916  102
## [4]  1.622385  114
## [5]  1.727509  113
## [6]  1.721356  106
## [7]  1.573968  111
## [8]  2.372268  140
## [9]  1.526965  119
## [10] 2.279848  104
```

```
# using itemFrequencyPlot() function
arules::itemFrequencyPlot(Groceries, topN = 20,
                          col = brewer.pal(8, 'Pastel2'),
                          main = 'Relative Item Frequency Plot',
                          type = "relative",
                          ylab = "Item Frequency (Relative)")
```