1.

```
library(mlbench)
install.packages("dplyr")
library(dplyr)
library(ggplot2)
library(reshape2)
data("BostonHousing")
housing <- BostonHousing
str(housing)

housing %>%
  ggplot(aes(x = medv)) +
  stat_density() +
  labs(x = "Median Value ($1000s)", y =
"Density", title = "Density Plot of
Median Value House Price in Boston") +
  theme_minimal()

summary(housing$medv)

housing %>%
  select(c(crim, rm, age, rad, tax, lstat,
medv)) %>%
  melt( id.vars = "medv") %>%
  ggplot(aes(x = value, y = medv, colour
= variable)) +
  geom_point(alpha = 0.7) +
  stat_smooth(aes(colour = "black")) +
  facet_wrap(~variable, scales = "free",
ncol = 2) +
  labs(x = "Variable Value", y = "Median
House Price ($1000s)") +
  theme_minimal()

library("caret")
set.seed(123) #random number
geneartion
to_train <- createDataPartition(y =
housing$medv, p = 0.75, list = FALSE)
to_test<-
createDataPartition(y=housing$medv,
p=0.25,list=FALSE)
train <- housing[to_train, ]
test <- housing[to_test, ]

first_lm <- lm( medv ~ crim +rm +tax
+lstat, data = train)

lm1_rsqu <-
summary(first_lm)$r.squared
print(paste("First linear model has an r-
squared value of ", round(lm1_rsqu, 3),
sep = ""))

second_lm <- lm(log(medv) ~ crim +rm
+ tax +lstat, data = train)

lm2_rsqu <-
summary(second_lm)$r.squared
print(paste("Our second linear model has
an r-squared value of ", round(lm2_rsqu,
3), sep = ""))

abs(mean(second_lm$residuals))


predicted <- predict(second_lm, newdata
= test)
results <- data.frame(predicted =
exp(predicted), original = test$medv)

results %>%
  ggplot(aes(x = predicted, y = original))
+
  geom_point() +
  stat_smooth() +
  labs(x = "Predicted Values", y =
"Original Values", title = "Predicted vs.
Original Values") +
  theme_minimal()
```

2.

```
library(KernelKnn)
data(ionosphere, package =
'KernelKnn')
ionosphere = ionosphere[, -2]
X = scale(ionosphere[, -c(34)])
y = ionosphere[, c(34)]
y = as.numeric(y)
train.idx = sample(1:length(y),
round(length(y) * 0.75))
test.idx = setdiff(1:length(y), train.idx)
train = X[train.idx, ]
test = X[test.idx, ]
train.labels = y[train.idx]
test.labels = y[test.idx]
accuracy = function (y_true, preds) {
  out = table(y_true,
              max.col(preds, ties.method =
"random"))
  acc = sum(diag(out))/sum(out)
  acc
}
predictions = KernelKnn(train, test,
train.labels,
                        k = 5 ,
                        method = 'euclidean',
                        weights_function = NULL,
                        regression = F,
                        Levels = unique(y))
acc = accuracy(test.labels, predictions)
paste('Accuracy is ', acc)
predictions = KernelKnn(train, test,
train.labels,
                        k = 10,
                        method = 'canberra',
                        weights_function =
'epanechnikov',
                        regression = F,
                        Levels = unique(y))
acc = accuracy(test.labels, predictions)
knn = KernelKnnCV(X, y,
                  k = 9 ,
                  folds = 5,
                  method = 'canberra',
                  weights_function =
'epanechnikov',
                  regression = F,
                  Levels = unique(y),
                  threads = 5)
acc_cv =
unlist(lapply(1:length(knn$preds),
                function(x)
accuracy(y[knn$folds[[x]]],

knn$preds[[x]])))
paste('Accuracy is ', mean(acc_cv))
paste('Accuracy is ', acc)
```

3.

```
install.packages("tm")
install.packages("wordcloud")
install.packages("e1071")
library(tm)
library(wordcloud)
library(e1071)
sms_spam_df <-
read.csv(file="E:\\sms_spam.csv",stringsAsFactors=F)
View(sms_spam_df)
str(sms_spam_df)
sms_corpus <-
VCorpus(VectorSource(sms_spam_df$text))
print(sms_corpus)
inspect(sms_corpus[1:2])
clean_corpus <- tm_map(sms_corpus,
content_transformer(tolower))
clean_corpus <- tm_map(clean_corpus,
removeNumbers)
clean_corpus <- tm_map(clean_corpus,
removePunctuation)
stopwords()[1:15]
clean_corpus <- tm_map(clean_corpus, removeWords,
stopwords())
clean_corpus <- tm_map(clean_corpus,
stripWhitespace)
inspect(clean_corpus[1:3])
sms_dtm <- DocumentTermMatrix(clean_corpus)
str(sms_dtm)
spam_indices <- which(sms_spam_df$type ==
"spam")
ham_indices <- which(sms_spam_df$type == "ham")
wordcloud(clean_corpus[ham_indices], min.freq=40)
wordcloud(clean_corpus[spam_indices], min.freq=40)
sms_raw_train <- sms_spam_df[1:4169,]
sms_raw_test <- sms_spam_df[4170:5559,]
sms_dtm_train <- sms_dtm[1:4169,]
sms_dtm_test <- sms_dtm[4170:5559,]
sms_corpus_train <- clean_corpus[1:4169]
sms_corpus_test <- clean_corpus[4170:5559]
spam <- subset(sms_raw_train, type == "spam")
ham <- subset(sms_raw_train, type == "ham")
five_times_words <- findFreqTerms(sms_dtm_train,
5)
sms_train <- DocumentTermMatrix(sms_corpus_train,
                                control=list(dictionary =
five_times_words))
sms_test <- DocumentTermMatrix(sms_corpus_test,
                               control=list(dictionary =
five_times_words))
convert_count <- function(x) {
  y <- ifelse(x > 0, 1,0)
  y <- factor(y, levels=c(0,1), labels=c("No", "Yes"))
  y
}
sms_train <- apply(sms_train, 2, convert_count)
sms_test <- apply(sms_test, 2, convert_count)

sms_classifier <- naiveBayes(sms_train,
factor(sms_raw_train$type))
sms_test_pred <- predict(sms_classifier,
newdata=sms_test)
k=table(sms_test_pred, sms_raw_test$type)
k
accuracy = sum(diag(k))/sum(k)*100
accuracy
```

4.

```
data(iris)
help(iris)
iris_dataset<-iris
View(iris_dataset)

head(iris_dataset,7)

colnames(iris_dataset)<-
c("Sepal.Length","Sepal.Width","Petal.Length","Petal.Width
","Species")
head(iris_dataset,5)

library(caret)
index <- createDataPartition(iris_dataset$Species, p=0.80,
list=FALSE)
testset <- iris_dataset[-index,]
trainset <- iris_dataset[index,]

dim(trainset)

str(trainset)

summary(trainset)

levels(trainset$Species)

hist(trainset$Sepal.Width)

par(mfrow=c(1,4))
for(i in 1:4) {
  boxplot(trainset[,i], main=names(trainset)[i])
}
library(ggplot2)
g <- ggplot(data=trainset, aes(x = Petal.Length, y =
Petal.Width))
print(g)
g <-g +
  geom_point(aes(color=Species, shape=Species)) +
  xlab("Petal Length") +
  ylab("Petal Width") +
  ggtitle("Petal Length-Width")+
  geom_smooth(method="lm")
print(g)
box <- ggplot(data=trainset, aes(x=Species,
y=Sepal.Length)) +
  geom_boxplot(aes(fill=Species)) +
  ylab("Sepal Length") +
  ggtitle("Iris Boxplot") +
  stat_summary(fun.y=mean, geom="point", shape=5,
size=4)
print(box)

library(ggthemes)
histogram <- ggplot(data=iris, aes(x=Sepal.Width)) +
  geom_histogram(binwidth=0.2, color="black",
aes(fill=Species)) +
  xlab("Sepal Width") +
  ylab("Frequency") +
  ggtitle("Histogram of Sepal Width")+
  theme_economist()
print(histogram)


library(ggthemes)
facet <- ggplot(data=trainset, aes(Sepal.Length,
y=Sepal.Width, color=Species))+
  geom_point(aes(shape=Species), size=1.5) +
  geom_smooth(method="lm") +
  xlab("Sepal Length") +
  ylab("Sepal Width") +
  ggtitle("Faceting") +
  theme_fivethirtyeight() +
  facet_grid(. ~ Species) # Along rows
print(facet)

'p;;;o'
```