Homework 5

BY: NOA SHADMON

ID: 5980

Honor Code: ``The codes and results derived by using these codes constitute my own work. I have consulted the following resources regarding this assignment:'' (ADD: names of persons or web resources, if any, excluding the instructor, TAs, and materials posted on course website)

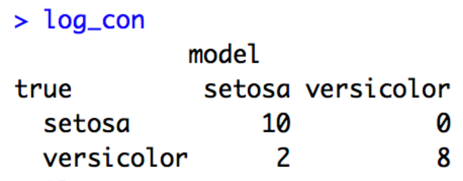
Kathleen Zhen, code posted on piazza, and discussion notes (posted by Nick)

Problem 1

The first time we run the code, we get $\beta_0 =$ -5.071102 and $\beta_1 = 2.016006$ with $\sigma^2 = 0.9827592$. For all 10 runs, the theoretical confidence interval for β_0 is always greater than that for bootstrap procedure. This is also true for β_1. This is again seen when we calculate the average values for the slope and intercept coefficients for the theoretical and bootstrap samples. The average theoretical β_0= 0.830346 and the average bootstrap $\beta_0 = 0.6838061$. The average theoretical β_1= 0.1230048 and the average bootstrap $\beta_1 = 0.1046841$.

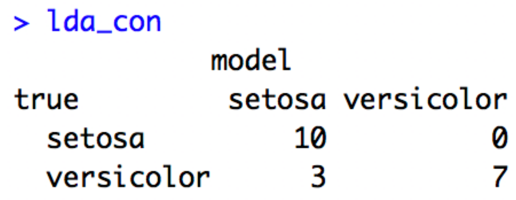
Problem 2

Using the logistic model, we created a confusion model. Of the 20 plants (using Sepal.Length as the predictor), the system predicted 12 setosa plants and 8 versicolor plants even though there are 10 of each. This model is 90% accurate (code to calculate accuracy attached).



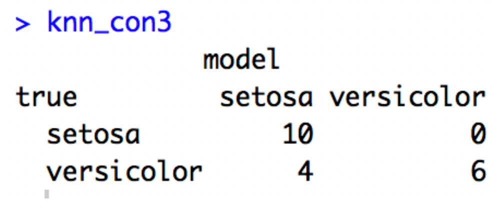
```
> log_con
            model
true         setosa versicolor
  setosa         10          0
  versicolor      2          8
```

Using the linear discriminant analysis the system predicted 13 setosa plants and 7 versicolor plants (using Sepal.Length as the predictor). This model predicted 85% correctly.



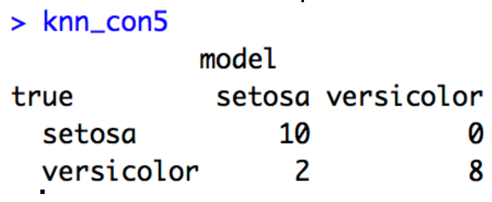
```
> lda_con
            model
true         setosa versicolor
  setosa         10          0
  versicolor      3          7
```

Using the k nearest neighbor's algorithm, when k=3 the confusion matrix is as such:



```
> knn_con3
            model
true         setosa versicolor
  setosa         10          0
  versicolor      4          6
```

Here, the system predicted 14 setosas and 6 versicolor. This model is 80% accurate. When k=5, the confusion shows a prediction of 12 setosas and 8 versicolors. This model is 90% accurate.



```
> knn_con5
            model
true         setosa versicolor
  setosa         10          0
  versicolor      2          8
```

From the tables and using the calculated accuracy values, we can conclude that the logistic model and k nearest neighbors (with k = 5) is the best for classification prediction when sepal.length is the predictor.

APPENDIX

```
library(readr)
library(broom)
library(MASS)
library(class)
#PROBLEM 1
resample = function(data) {
  n = nrow(data)
  # Sample row numbers (i) rather than values (e_i)
  idx = sample(n, n, replace = TRUE)

  # Use row numbers to get new residuals (e2_i).
  res_samp = data$.resid[idx]

  # y2_i =  b_0 + b_1 * x_i    + e2_i
  y_samp =  data$.fitted       + res_samp

  # Insert new response (y_i) into data frame, keeping old covariates (x_i)
  data$gift_aid = y_samp

  # Fit the same model with new data (y2_i, x_i).
  new_mod = lm(gift_aid ~ x, data)

  return (coef(new_mod))
}

prob1 = function(seed) {
  set.seed(seed) # only set the seed once, at the beginning

  # Part 1
  x = rchisq(n = 100, df = 6)
  e = rnorm(n = 100, mean = 0, sd = 1)
  y = -5 + 2*x + e

  # Part 2
  mod = lm(y ~ x)
  summ = summary(mod)
  print(summ$coefficients)
  sigma = summ$sigma
  print(sigma)
  resid = augment(mod)
  # Part 3
  theo = confint(mod)
  boot = sapply(1:400, function(i) resample(resid))
  ci_intercept = quantile(boot[1, ], c(0.05, 0.95))
  ci_slope     = quantile(boot[2, ], c(0.05, 0.95))

  theo_diff_int = abs(theo[3] - theo[1])
  theo_diff_x = abs(theo[4] - theo[2])
  diff_int = abs(ci_intercept[[1]] - ci_intercept[[2]])
  diff_slope = abs(ci_slope[[1]] - ci_slope[[2]])

  ci_widths = data.frame(theo_diff_int, theo_diff_x, diff_int, diff_slope)

  # Return widths of both the theoretical and bootstrap confidence intervals:
  return (ci_widths)
}

all_ci_widths = sapply(1:10, prob1)
#average of theoretical intercept confidence interval
theo_avg_int = mean(as.numeric(as.vector(all_ci_widths[1,])))
#average of theoretical slope confidence interval
theo_avg_x = mean(as.numeric(as.vector(all_ci_widths[2,])))
#average of boot intercept confidence interval
diff_avg_int = mean(as.numeric(as.vector(all_ci_widths[3,])))
#average of boot slope confidence interval
diff_avg_slope = mean(as.numeric(as.vector(all_ci_widths[4,])))

#part 2
data(iris)

data = iris[1:100,]
test = rbind(data[41:50,], data[91:100,])
training = rbind(data[1:40,], data[51:90,])
test = droplevels(test)
training = droplevels(training)
log_model = glm(Species ~ Sepal.Length, training,family = binomial)

# Predict for test data. Use type = "response" to get class probabilities.
log_pred = predict(log_model, test, type = "response")
# Convert predictions to 1 or 2, for category 1 or 2 respectively.
log_pred = (log_pred > 0.5) + 1
log_pred = levels(training$Species)[log_pred]

log_con = tabcle(true = test$Species, model = log_pred)
acc_log = sum(log_con[1],log_con[4])/sum(log_con[1],log_con[2],log_con[3],log_con[4])

lda = lda(Species~Sepal.Length, training)
```

```
lda_pred = predict(lda, test, type = "response")
lda_pred = levels(training$Species)[lda_pred$class]
lda_con = table(true = test$Species, model = lda_pred)
acc_lda = sum(lda_con[1],lda_con[4])/sum(lda_con[1],lda_con[2],lda_con[3],lda_con[4])

knn_pred3 = knn(
  # Note the use of [ ] rather than $ or [[ ]].
  #
  # The knn() function expects a matrix or data frame for the train and test
  # arguments. Using $ or [[ ]] would get a vector rather than a data frame.
  #
  train = training["Sepal.Length"], # 1-col data frame
  test  = test["Sepal.Length"],  # 1-col data frame
  cl    = training$Species,                    # vector
  k     = 3
)

knn_con3 = table(true = test$Species, model = knn_pred3)
acc_knn3 =
sum(knn_con3[1],knn_con3[4])/sum(knn_con3[1],knn_con3[2],knn_con3[3],knn_con3[4])

knn_pred5 = knn(

  train = training["Sepal.Length"], # 1-col data frame
  test  = test["Sepal.Length"],  # 1-col data frame
  cl    = training$Species,                    # vector
  k     = 5
)

knn_con5 = table(true = test$Species, model = knn_pred5)
acc_knn5 =
sum(knn_con5[1],knn_con5[4])/sum(knn_con5[1],knn_con5[2],knn_con5[3],knn_con5[4])
```