October 18, 2021

The results below are generated from an R script.

```
### Please enter your answers for Assessed Coursework 1 below.
##
####################################
# Question 1
############
# Comments:
#.   In-built matrix operations used are:
#.        t(x).    - takes transpose of matrix x
#.        %*%.     - used for matrix multiplication
#.        solve(x) - finds inverse of x (square matrix)
#
#.   No defensive programming!

# Estimate the parameter vector from the (augmented) data matrix 'x' and
# the value vector 'y'.
linreg <- function(x, y) {
  beta_est = solve(t(x) %*% x) %*% t(x) %*% y
  return(beta_est)
}


# Calculate the value vector for a set of data (augmented!) in the matrix
# 'preddata' using the parameter vector 'params'
predict <- function(preddata, params) {
  y_est = preddata %*% params
  return(y_est)
}

# Calculate the Residual Sum of Squares (RSS) between two vectors.
rss <- function(y, yhat) {
  rss_val <- sum((y-yhat) * (y-yhat))
  return(rss_val)
}

############
# Test functions:
# Take beta = (1.1,2.3,-3.1,1.2) for beta_0 ... beta_3 of a 3 variable
# problem.  Now generate 5 'sample'value' points using the data:
# x_1 = (1.1,2.1,3.2),    x_2 = (-1.2, 0.5, 1.2), x_3 = (0, 1.2, 3.0),
# x_4 = (2.1, 2.4, -1.2), x_5 = (1.0, 1.0, 1.0).
# From this we see that y should be: (0.96, -1.77, 0.98, -2.95, 1.5)
#
# Test Data:
```

```r
# Create the X matrix (with augmented 1 in first column):
x <- rbind(c(1, 1.1, 2.1, 3.2), c(1,-1.2, 0.5, 1.2), c(1 ,0, 1.2, 3.0),
           c(1, 2.1, 2.4, -1.2), c(1, 1.0, 1.0, 1.0))
beta <- c(1.1, 2.3, -3.1, 1.2)
y_actual <- c(0.96, -1.77, 0.98, -2.95, 1.5)
```

```r
# Perform test: linreg
#   Should return a good approximation to beta vector.
beta_estimate <- linreg(x, y_actual)

# Check the 'total' difference between the actual and estimated beta values.
# Ideally should be zero (i.e. very small due to rounding).
sum(abs(beta_estimate - beta))
```

```
## [1] 1.132427e-14
```

```r
# Perform test: predict
#   We can use this method to calculate y_actual - should be exact
y_estimated <- predict(x, beta)
# Below should return zero (i.e. very small due to rounding).
sum(abs(y_estimated - y_actual))
```

```
## [1] 1.110223e-15
```

```r
# Perform test: rss
#   Take two vectors and calculate their RSS:
vec1 <- c(1,2,3,4,5)
vec2 <- c(2,1,5,1,3)

# Calculated RSS value should be:
# (2-1)^2 + (1-2)^2 + (5-3)^2 + (1-4)^2 + (3-5)^2 = 19
rss_calc <- rss(vec1, vec2)
rss_calc
```

```
## [1] 19
```

```r
####################################
# Question 2
############
# Set the seed for reproducability & import library for multidim. normal
# distribution:
library(mvtnorm)
set.seed(101)
# We can allocate 'p' to have various integer values (>1).
# Also define n, the number of data points (n > p)
p <- 10
n <- 150

# Calculate the 'beta' vector from the uniform U[0,1] distribution
beta <- runif(p+1, min = 0, max = 1)

# Create the data matrix X & augment:
```

```r
x <- matrix(rmvnorm(n, rep(0, p), diag(p)), byrow=T, nrow=n)
x <- cbind(1, x)

# Calculate response vector:
y <- predict(x, beta)

# Now we start to test our functions:
#   - Calculate 'our' estimate for beta: beta_hat
#.  - Use RSS to estimate how good the agreement is for beta and beta_hat.
#.    It is a good estimate as rss < 10^-30.
beta_hat <- linreg(x, y)
rss(beta, beta_hat)
```

```
## [1] 8.597833e-31
```

```r
###################################
# Question 3
############
# Read in the data ftom the Diabetes file:
diabetes = read.csv('./DiabetesData.csv', header = TRUE, sep = ",")

# View sample of data and check total number of records (found 768 records):
head(diabetes)
```

```
##   pregnant glucose pressure triceps insulin mass pedigree age diabetes
## 1        6     148       72      35      NA 33.6    0.627  50      pos
## 2        1      85       66      29      NA 26.6    0.351  31      neg
## 3        8     183       64      NA      NA 23.3    0.672  32      pos
## 4        1      89       66      23      94 28.1    0.167  21      neg
## 5        0     137       40      35     168 43.1    2.288  33      pos
## 6        5     116       74      NA      NA 25.6    0.201  30      neg
```

```r
dim(diabetes)
```

```
## [1] 768   9
```

```r
# Remove rows that contain na/null values and check number of records remaining
# (found 392 remaining records):
diabetes_na <- na.omit(diabetes)
dim(diabetes_na)
```

```
## [1] 392   9
```

```r
# Remove the 'diabetes' column and put it into a seperate response vector (y)
# In this new vector replace 'pos' & 'neg' by 1 and -1 respectively as the model
# requires numeric data.
# The independent variables are then placed into the data matrix (x) and
# augmented (with 1 in first column).
y <- diabetes_na[9]
y <- ifelse(y=='pos', 1, -1)
x <- data.matrix(cbind(1, diabetes_na[-9]))

# Now get our estimated beta: beta_est, based on the diabetes dataset.
# Show the estimated parameters:
beta_est <- linreg(x, y)
beta_est
```

```
##                diabetes
## 1        -3.2053536202
## pregnant  0.0259054770
## glucose   0.0128171872
## pressure  0.0001092992
## triceps   0.0033550479
## insulin  -0.0002466728
## mass      0.0186501287
## pedigree  0.3143837887
## age       0.0117561689

# Calculate y_calc using beta_est and the independent variables in x
y_calc <- predict(x, beta_est)

# Calculate RSS for y and y_calc (rss found was 227.38):
rss(y, y_calc)

## [1] 227.3771

# Comments on Question 3:
# 1. The RSS found was very large for such a (relatively small) dataset
#.   (RSS > 227).
#.   So, the Linear Regression model doesn't appear the appropriate one to use
#.   in the case for diabetes analysis.
# 2. Using the Linear Regression model to interpret data for diabetes would
#.   appear to be inappropriate.  Linear Regression is used to model data where
#.   the output value parameter has a linear (or near linear) relationship to
#.   the independent variables and is a real number.  In this case the we are
#.   really looking at (binary) classified data - which is either "positive"
#.   (i.e. has diabetes) or "negative" (i.e. doesnt have diabetes).  For this
#.   Logistic Regression would be more suitable.
# 3. The estimated parameters (beta_est) is shown - see above in the code.
#.   Except for the constant term (beta_0), the parameters have small values.
```

The R session information (including the OS info, R version and all packages used):

```
sessionInfo()

## R version 4.0.0 (2020-04-24)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 20.04 LTS
##
## Matrix products: default
## BLAS/LAPACK: /usr/lib/x86_64-linux-gnu/openblas-pthread/libopenblasp-r0.3.8.so
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C               LC_TIME=en_US.UTF-8
##  [4] LC_COLLATE=en_US.UTF-8     LC_MONETARY=en_US.UTF-8    LC_MESSAGES=C
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C                  LC_ADDRESS=C
## [10] LC_TELEPHONE=C             LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
```

```
## [1] mvtnorm_1.1-3
##
## loaded via a namespace (and not attached):
##  [1] compiler_4.0.0 magrittr_1.5   tools_4.0.0    tinytex_0.23   stringi_1.4.6
##  [6] highr_0.8      knitr_1.36     stringr_1.4.0  xfun_0.26      evaluate_0.14
```

```
Sys.time()
```

```
## [1] "2021-10-18 18:56:59 UTC"
```