IS 605 - Assignment 5

Dan Fanelli

February 29, 2016

Problem Set 1:

(1) Consider the unsolvable system Ax = b

Write R markdown script to compute t(A)*A and t(A)*b

- First, a function for doing least squares:

```
# the least squares function:
run_least_squares <- function(A, b, show_data) {
  cat("---- run_least_squares start: \n")
  at_a <- t(A) %*% A
  at_b <- t(A) %*% b
  ### Solve for ^x in R using the above two computed matrices.
  x_star <- solve(at_a, at_b)
  ### What is the squared error of this solution?
  a_times_x_star <- A %*% x_star
  a_times_x_star_minus_b <- a_times_x_star - b
  length <- sqrt(sum(a_times_x_star_minus_b^2))
  the_error <- b - A %*% x_star

  if(show_data){
    max_2_show <- 10
    cat("A: ", head(A, n = max_2_show), "\n")
    cat("b: ", head(b, n = max_2_show), "\n")
    cat("at: ", head(t(A), n = max_2_show), "\n")
    cat("at_a: ", at_a[1:max_2_show], "\n")
    cat("at_b: ", at_b, "\n")
    cat("x_star: ", x_star, "\n")
    cat("(if x_star is 0s then the the following is really not needed...) \n")
    cat("a_times_x_star: ", a_times_x_star, "\n")
    cat("a_times_x_star_minus_b: ", a_times_x_star_minus_b, "\n")
    cat("length = ", length, "\n")
    cat("the_error = ", the_error, "\n")
    cat("---- run_least_squares finished. ---- \n")
  }else{
    cat("x_star: ", x_star, "\n")
  }

  return (x_star)
}

eval_root_mean_squared_error <- function(observed, predicted){
  root_mean_squared_error <- sqrt(mean((observed-predicted)^2))
  cat("root_mean_squared_error: ", root_mean_squared_error, "\n")
  observed_mean <- mean(observed)
```

```
  cat("observed_mean: ", observed_mean, "\n")
  rmse_div_by_observed_mean <- root_mean_squared_error / observed_mean
  cat("rmse_div_by_observed_mean: ", rmse_div_by_observed_mean)
}
```

Run the script for the 2 equations:

```
A <- matrix(c(1,1,1,1,0,1,3,4), ncol=2)

# b, the first one that requires least squares (has no solution)
soln_b <- matrix(c(0,8,8,20), ncol=1)
soln_b_x_star <- run_least_squares(A,soln_b,TRUE)
```

```
## ---- run_least_squares start:
## A:  1 1 1 1 0 1 3 4
## b:  0 8 8 20
## at:  1 0 1 1 1 3 1 4
## at_a:  4 8 8 26 NA NA NA NA NA NA
## at_b:  36 112
## x_star:  1 4
## (if x_star is 0s then the the following is really not needed...)
## a_times_x_star:  1 5 13 17
## a_times_x_star_minus_b:  1 -3 5 -3
## length =  6.63325
## the_error =  -1 3 -5 3
## ---- run_least_squares finished. ----
```

```
soln_b_x_star
```

```
##      [,1]
## [1,]    1
## [2,]    4
```

```
# p, the second one that does not require least squares (has a solution)
soln_p <- matrix(c(1,5,13,17), ncol=1)
soln_p_x_star <- run_least_squares(A,soln_p,TRUE)
```

```
## ---- run_least_squares start:
## A:  1 1 1 1 0 1 3 4
## b:  1 5 13 17
## at:  1 0 1 1 1 3 1 4
## at_a:  4 8 8 26 NA NA NA NA NA NA
## at_b:  36 112
## x_star:  1 4
## (if x_star is 0s then the the following is really not needed...)
## a_times_x_star:  1 5 13 17
## a_times_x_star_minus_b:  0 0 0 0
## length =  0
## the_error =  0 0 0 0
## ---- run_least_squares finished. ----
```

```r
soln_p_x_star
```

```
##      [,1]
## [1,]    1
## [2,]    4
```

Show that the error e = b - p = [-1; 3;-5; 3]

```r
e <- soln_b - soln_p
e
```

```
##      [,1]
## [1,]   -1
## [2,]    3
## [3,]   -5
## [4,]    3
```

```r
eval_root_mean_squared_error(soln_b, soln_p)
```

```
## root_mean_squared_error:  3.316625
## observed_mean:  9
## rmse_div_by_observed_mean:  0.3685139
```

Show that the error e is orthogonal to p and to each of the columns of A.

```r
#orthogonal means dot product = 0
sum(e*soln_p)
```

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

```r
sum(e*A[,1])
```

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

```r
sum(e*A[,2])
```

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

Problem Set 2:

1. mpg: continuous
2. cylinders: multi-valued discrete
3. displacement: continuous
4. horsepower: continuous
5. weight: continuous
6. acceleration: continuous
7. model year: multi-valued discrete
8. origin: multi-valued discrete
9. car name: string (unique for each instance)

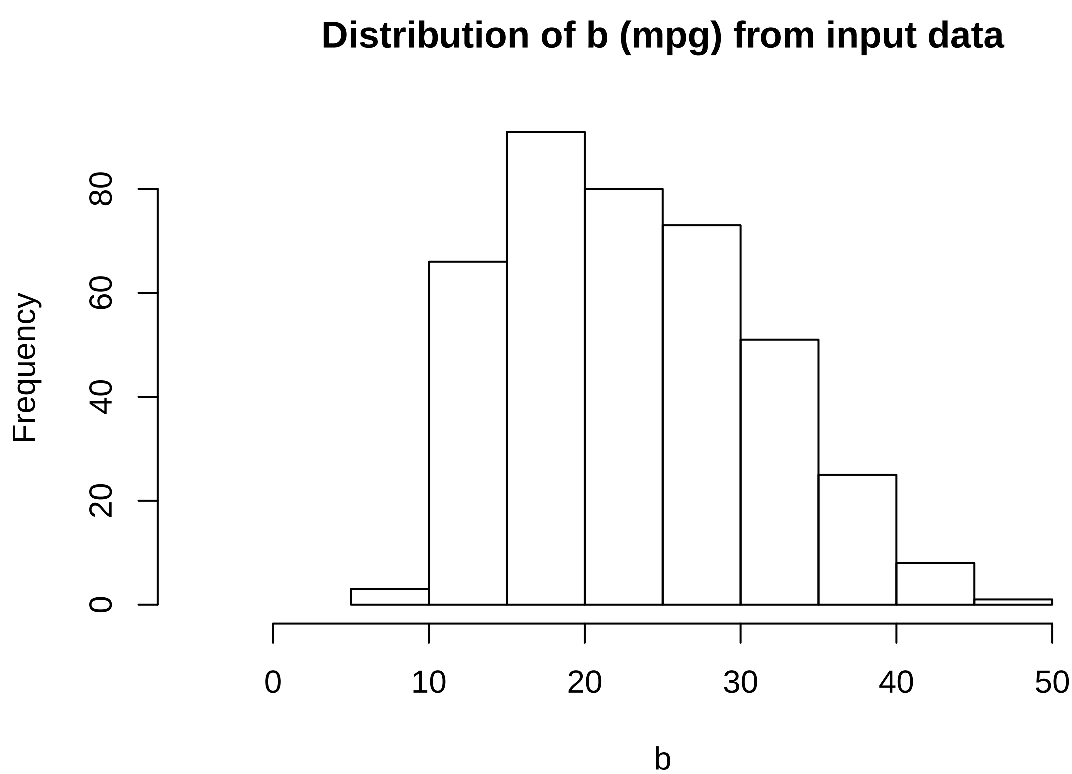
"Write an R markdown script that takes in the auto-mpg data, extracts an A matrix from the first 4 columns and b vector from the fifth (mpg) column."

- model mpg as a function of displacement, horsepower, weight, and acceleration
- read the data and show a quick summary/histogram of the target variable b
- strip all but the 4 important columns
- strip all the bad rows

```
data <- read.table("auto-mpg.data", sep="")
names(data) <- c("mpg", "cylinders", "displacement", "horsepower", "weight", "acceleration", "model_yea
b <- as.matrix(data[,c("mpg")])
summary(b)
```

```
##        V1
##  Min.   : 9.00
##  1st Qu.:17.50
##  Median :23.00
##  Mean   :23.51
##  3rd Qu.:29.00
##  Max.   :46.60
```

```
hist(b, xlim=c(-5,55), main="Distribution of b (mpg) from input data")
```



```r
data <- data[,c("displacement","horsepower","weight","acceleration")]
#head(data)
data <- subset(data, displacement != '?' && horsepower != '?' && weight != '?' && acceleration != '?')
# convert to a matrix and show head
A <- data.matrix(data[1:4])
head(A)
```

```
##   displacement horsepower weight acceleration
## 1          307         17   3504         12.0
## 2          350         35   3693         11.5
## 3          318         29   3436         11.0
## 4          304         29   3433         12.0
## 5          302         24   3449         10.5
## 6          429         42   4341         10.0
```

"Using the least squares approach, your code should compute the best fitting solution. That is, find the best fitting equation that expresses mpg in terms of the other 4 variables"

- So, using the function from above, without outputting all the data, let's find the equation **co-efficeints** and take a peek at the **variable data**:

```r
b_x_star_ps2 <- run_least_squares(A,b,TRUE)
```

```
## ---- run_least_squares start:
## A:  307 350 318 304 302 429 454 440 455 390 17 35 29 29 24 42 47 46 48 40 3504 3693 3436 3433 3449 4
## b:  18 15 18 16 17 15 14 14 14 15
## at:  307 17 3504 12 350 35 3693 11.5 318 29 3436 11 304 29 3433 12 302 24 3449 10.5 429 42 4341 10 4
## at_a:  19206864 3363237 261373948 1136422 3363237 1406759 55916308 326824.8 261373948 55916308
## at_b:  1550039 520099.7 25614041 149295
## x_star:  -0.01125847 0.06559461 -0.001099502 1.613001
## (if x_star is 0s then the the following is really not needed...)
## a_times_x_star:  13.16212 12.8444 12.28718 14.0611 11.31855 9.282169 7.701385 7.033087 9.290658 7.71
## a_times_x_star_minus_b:  -4.837878 -2.155596 -5.712821 -1.938903 -5.681453 -5.717831 -6.298615 -6.96
## length =  122.4976
## the_error =  4.837878 2.155596 5.712821 1.938903 5.681453 5.717831 6.298615 6.966913 4.709342 7.2895
## ---- run_least_squares finished. ----
```

```r
b_x_star_ps2
```

```
##                      [,1]
## displacement -0.011258466
## horsepower    0.065594608
## weight       -0.001099502
## acceleration  1.613001401
```

```r
displacement_coefficient <- b_x_star_ps2[1]
horsepower_coefficient <- b_x_star_ps2[2]
weight_coefficient <- b_x_star_ps2[3]
acceleration_coefficient <- b_x_star_ps2[4]

displacements <- A[,c("displacement")]
```

```r
horsepowers <- A[,c("horsepower")]
weights <- A[,c("weight")]
accelerations <- A[,c("acceleration")]

head(displacements)
```

```
##   1   2   3   4   5   6
## 307 350 318 304 302 429
```

```r
head(horsepowers)
```

```
##  1  2  3  4  5  6
## 17 35 29 29 24 42
```

```r
head(weights)
```

```
##    1    2    3    4    5    6
## 3504 3693 3436 3433 3449 4341
```

```r
head(accelerations)
```

```
##    1    2    3    4    5    6
## 12.0 11.5 11.0 12.0 10.5 10.0
```

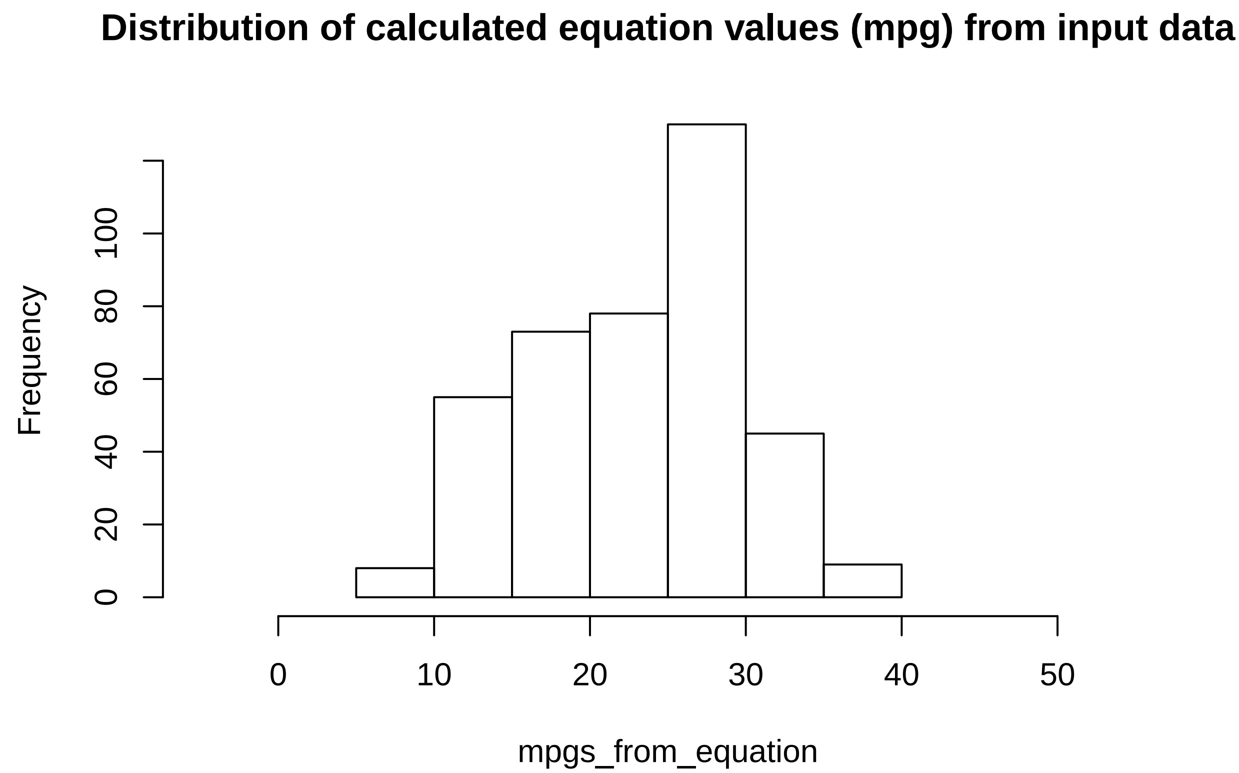
- So the best fit least-squares **equation** is:
- **mpg = (-0.011258 * displacement) + (0.065595 * horsepower) + (-0.0010995 * weight) + (1.613 * acceleration)**
- Let's calculate a collection of mpgs using the above **equation**:

```r
mpgs_from_equation <- (displacement_coefficient*displacements) + (horsepower_coefficient*horsepowers) +

summary(mpgs_from_equation)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   7.033  17.430  24.560  23.040  27.830  39.720
```

```r
hist(mpgs_from_equation, xlim=c(-5,55), main="Distribution of calculated equation values (mpg) from inpu
```



"Finally, calculate the fitting error between the predicted mpg of your model and the actual mpg."

- So the original was slightly skewed right, this is skewed left…and this seems to be represented in the summary and histograms of the b * x_star, where the error values are weighted towards the negative side…

```
eval_root_mean_squared_error(b, mpgs_from_equation)
```

```
## root_mean_squared_error:  6.14025
## observed_mean:  23.51457
## rmse_div_by_observed_mean:  0.2611253
```