STAT420 Homework 3

Alok K. Shukla

9/12/2016

Contents

Assignment Solutions

Exercise 1 (Using `lm`)

For this exercise we will use the `faithful` dataset. This is a default dataset in `R`, so there is no need to load it. You should use `?faithful` to learn about the background of this dataset.

(a) Suppose we would like to predict the duration of an eruption of the Old Faithful geyser in Yellowstone National Park based on the waiting time before an eruption. Fit a simple linear model in `R` that accomplishes this task. Store the results in a variable called `faithful_model`. Output the result of calling `summary()` on `faithful_model`.

Solution

```r
faithful_model = lm(eruptions ~ waiting, data = faithful)
summary(faithful_model)
```

```
##
## Call:
## lm(formula = eruptions ~ waiting, data = faithful)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.29917 -0.37689  0.03508  0.34909  1.19329
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.874016   0.160143  -11.70   <2e-16 ***
## waiting      0.075628   0.002219   34.09   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4965 on 270 degrees of freedom
## Multiple R-squared:  0.8115, Adjusted R-squared:  0.8108
## F-statistic:  1162 on 1 and 270 DF,  p-value: < 2.2e-16
```

(b) Output only the estimated regression coefficients. Interpret β_0 and $\hat{\beta}_1$ in the *context of the problem*. Be aware that only one of those is an estimate.

Solution

```
faithful_model["coefficients"]
```

```
## $coefficients
## (Intercept)     waiting
## -1.87401599  0.07562795
```

β_0 tells us the mean duration of an eruption that occurred without any waiting, which doesn't make sense in contect of the problem; if there was no wait time, its essentially the same eruption.

$\hat{\beta}_1 = 0.0756$ tells us that for an increase in waiting time of one minute, the estimated mean duration of eruption increases by 0.075 minutes.

(c) Use your model to predict the duration of an eruption based on a waiting time of **80** minutes. Do you feel confident in this prediction? Briefly explain.

Solution

```
predict(faithful_model, data.frame(waiting = 80))
```

```
##       1
## 4.17622
```

I feel confident about this prediction, looking at the actual values in `faithful`, the predicted value seems about right. Thats becuase the predictor is in range.

(d) Use your model to predict the duration of an eruption based on a waiting time of **120** minutes. Do you feel confident in this prediction? Briefly explain.

Solution

```
predict(faithful_model, data.frame(waiting = 120))
```

```
##        1
## 7.201338
```

I dont feel confident about this prediction, looking at the actual values in `faithful`, I can say nothing about its accuracy. Thats becuase the predictor is not in range.

(e) Calculate the RSS for this model.

Solution

```
sum(faithful_model$residuals^2)
```

```
## [1] 66.56178
```

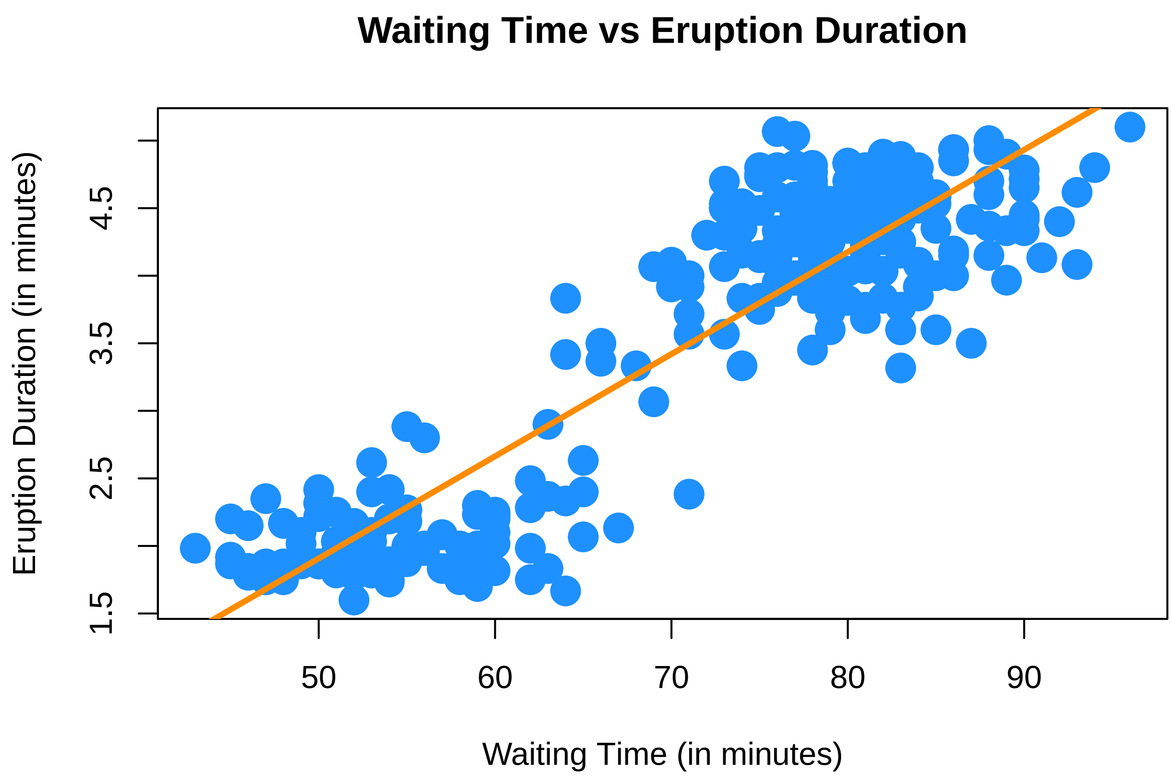
```
sum((faithful_model$fitted.values - faithful$eruptions)^2)
```

```
## [1] 66.56178
```

(f) Create a scatterplot of the data and add the fitted regression line. Make sure your plot is well labeled and is somewhat visually appealing.

Solution

```
plot(eruptions ~ waiting, data = faithful, xlab = "Waiting Time (in minutes)",
    ylab = "Eruption Duration (in minutes)", main = "Waiting Time vs Eruption Duration",
    pch = 20, cex = 3, col = "dodgerblue")
abline(faithful_model, lwd = 3, col = "darkorange")
```



(g) Report the value of R^2 for the model. Do so directly. Do not simply copy and paste the value from the full output in the console after running `summary()` in part **(a)**.

Solution

```
SST = sum((faithful$eruptions - mean(faithful$eruptions))^2)
SSReg = sum((faithful_model$fitted.values - mean(faithful$eruptions))^2)
R2 = SSReg/SST
R2
```

```
## [1] 0.8114608
```

```
summary(faithful_model)$r.squared
```

```
## [1] 0.8114608
```

Exercise 2 (Writing Functions)

This exercise is a continuation of Exercise 1.

(a) Write a function called `get_sd_est` that calculates an estimate of σ in one of two ways depending on input to the function. The function should take two arguments as input:

- `model_resid` - A vector of residual values from a fitted model.
- `mle` - A logical (`TRUE` / `FALSE`) variable which defaults to `FALSE`.

The function should return a single value:

- s_e if `mle` is set to `FALSE`.
- $\hat{\sigma}$ if `mle` is set to `TRUE`.

Solution

```
get_sd_est = function(model_resid, mle = FALSE) {
    SSE = sum(model_resid^2)
    if (mle == TRUE) {
        sqrt(SSE/(length(model_resid) - 2))
    } else {
        sqrt(SSE/length(model_resid))
    }
}
```

(b) Run the function `get_sd_est` on the residuals from the model in Exercise 1, with `mle` set to `FALSE`.

Solution

```
get_sd_est(faithful_model$residuals)
```

```
## [1] 0.4946842
```

(c) Run the function `get_sd_est` on the residuals from the model in Exercise 1, with `mle` set to `TRUE`.

Solution

```
get_sd_est(faithful_model$residuals, TRUE)
```

```
## [1] 0.4965129
```

(d) To check your work, output `summary(faithful_model)$sigma`. It should match at least one of **(b)** or **(c)**.

Solution

```
summary(faithful_model)$sigma
```

```
## [1] 0.4965129
```

Exercise 3 (Simulating SLR)

Consider the model

$$Y_i = 3 - 7x_i + \epsilon_i$$

with

$$\epsilon_i \sim N(\mu = 0, \sigma^2 = 4)$$

where $\beta_0 = 3$ and $\beta_1 = -7$.

Before answering the following parts, set a seed value equal to **your** birthday, as was done in the previous assignment.

```
birthday = 19920120
set.seed(birthday)
```

(a) Use `R` to simulate `n = 50` observations from the above model. For the remainder of this exercise, use the following "known" values of x.

```
x = runif(n = 50, 0, 10)
```

You may use the `sim_slr` function provided in the text. Store the data frame this function returns in a variable of your choice. Note that this function calls y `response` and x `predictor`.

Solution

```
sim_slr = function(n, x, beta_0, beta_1, sigma = 1) {
    epsilon = rnorm(n, mean = 0, sd = sigma)
    y = beta_0 + beta_1 * x + epsilon
    data.frame(predictor = x, response = y)
}
x = runif(n = 50, 0, 10)
sim_data = sim_slr(n = 50, x = x, beta_0 = 3, beta_1 = -7, sigma = 2)
head(sim_data)
```

```
##   predictor   response
## 1 7.2520587 -51.234127
## 2 9.5114285 -63.791913
## 3 5.9321426 -42.684890
## 4 3.4888550 -23.326774
## 5 0.8580868  -3.574250
## 6 1.8807353  -8.620173
```

(b) Fit a model to your simulated data. Report the estimated coefficients. Are they close to what you would expect? Briefly explain.

Solution

```
sim_fit = lm(response ~ predictor, data = sim_data)
coef(sim_fit)
```

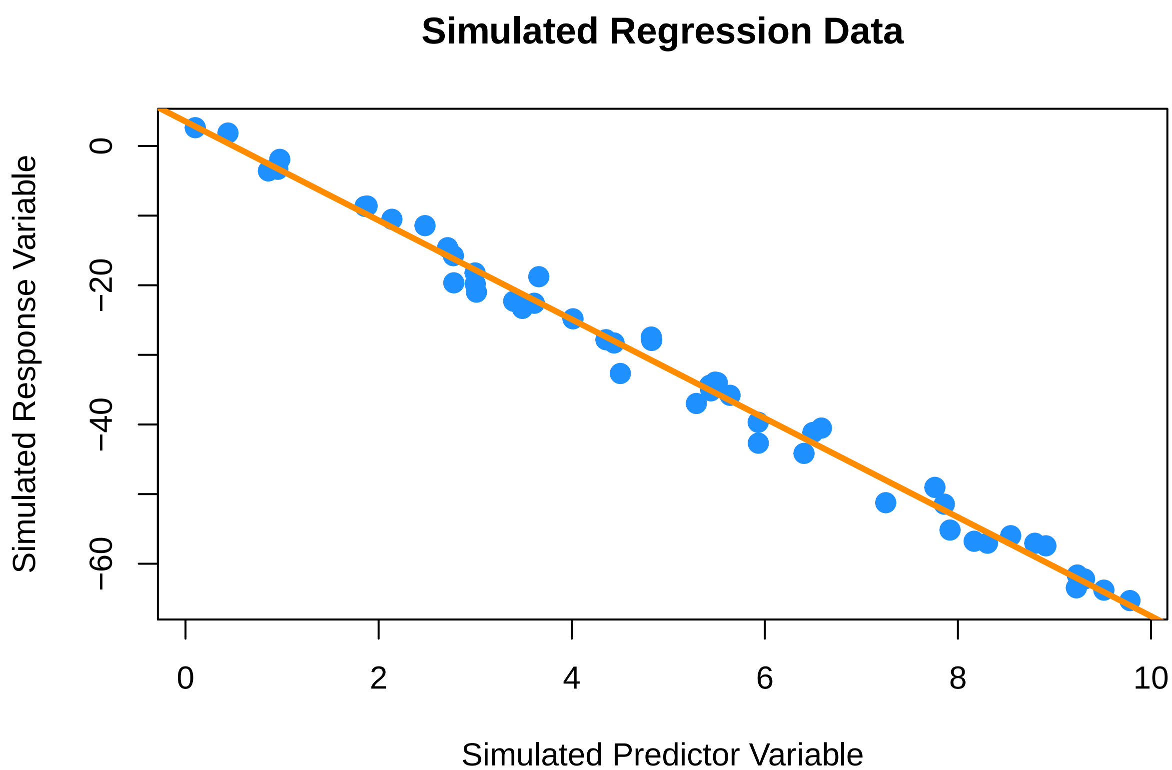
```
## (Intercept)   predictor
##    3.521404   -7.105498
```

They are pretty close to whats expected, negative slope and positive intercept of expected magnitudes.

(c) Plot the data you simulated in part **(a)**. Add the regression line from part **(b)**. Hint: Keep the two commands in the same chunk, so `R` knows what plot to add the line to when knitting your `.Rmd` file.

Solution

```
plot(response ~ predictor, data = sim_data, xlab = "Simulated Predictor Variable",
    ylab = "Simulated Response Variable", main = "Simulated Regression Data",
    pch = 20, cex = 2, col = "dodgerblue")
abline(sim_fit, lwd = 3, col = "darkorange")
```



(d) Use `R` to repeat the process of simulating `n = 50` observations from the above model 2000 times. Each time fit a SLR model to the data and store the value of $\hat{\beta}_1$ in a variable called `beta_hat_1`. Some hints:

- Use a `for` loop.
- Create `beta_hat_1` before writing the `for` loop. Make it a vector of length 2000 where each element is `0`.
- Inside the body of the `for` loop, simulate new y data each time. Use a variable to temporarily store this data together with the known x data as a data frame.
- After simulating the data, use `lm()` to fit a regression. Use a variable to temporarily store this output.
- Use the `coef()` function and `[]` to extract the correct estimated coefficient.
- Use `beta_hat_1[i]` to store in elements of `beta_hat_1`.

- See the notes on Distribution of a Sample Mean for some inspiration.

You can do this differently if you like. Use of these hints is not required.

Solution

We use `apply` in place of `for`.

```
beta_hat_1 = apply(data.frame(rep(0, 2000)), 1, function(y) coef(lm(response ~
    predictor, data = sim_slr(n = 50, x = x, beta_0 = 3, beta_1 = -7,
    sigma = 2)))["predictor"])
head(beta_hat_1)
```

```
## [1] -6.997438 -7.062953 -6.994328 -6.893919 -7.037497 -6.976572
```

(e) Report the mean and standard deviation of `beta_hat_1`. Do either of these look familiar?

Solution

```
mean(beta_hat_1)
```

```
## [1] -6.998588
```

```
sd(beta_hat_1)
```

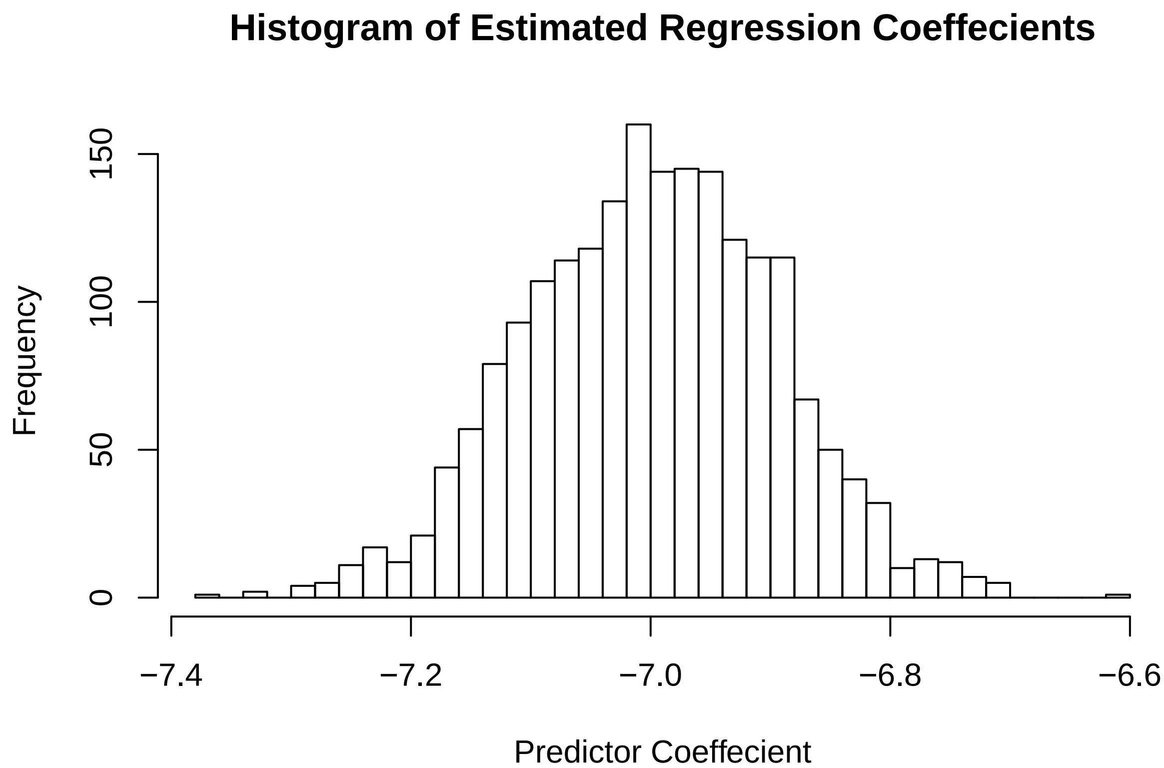
```
## [1] 0.1039814
```

Mean looks familiar, approximately equal to -7.

(f) Plot a histogram of `beta_hat_1`. Comment on the shape of this histogram.

Solution

```
hist(beta_hat_1, breaks = 50, main = "Histogram of Estimated Regression Coeffecients",
    xlab = "Predictor Coeffecient")
```



The shape verifies that the distribution of coeffecients is Normal.

Exercise 4 (Be a Skeptic)

Consider the model

$$Y_i = 10 + 0x_i + \epsilon_i$$

with

$$\epsilon_i \sim N(\mu = 0, \sigma^2 = 1)$$

where $\beta_0 = 10$ and $\beta_1 = 0$.

Before answering the following parts, set a seed value equal to **your** birthday, as was done in the previous assignment.

```
birthday = 19920120
set.seed(birthday)
```

(a) Use `R` to repeat the process of simulating `n = 25` observations from the above model 1500 times. For the remainder of this exercise, use the following "known" values of x.

```
x = runif(n = 25, 0, 10)
```

Each time fit a SLR model to the data and store the value of $\hat{\beta}_1$ in a variable called `beta_hat_1`. You may use the `sim_slr` function provided in the text. Hint: Yes $\beta_1 = 0$.

Solution

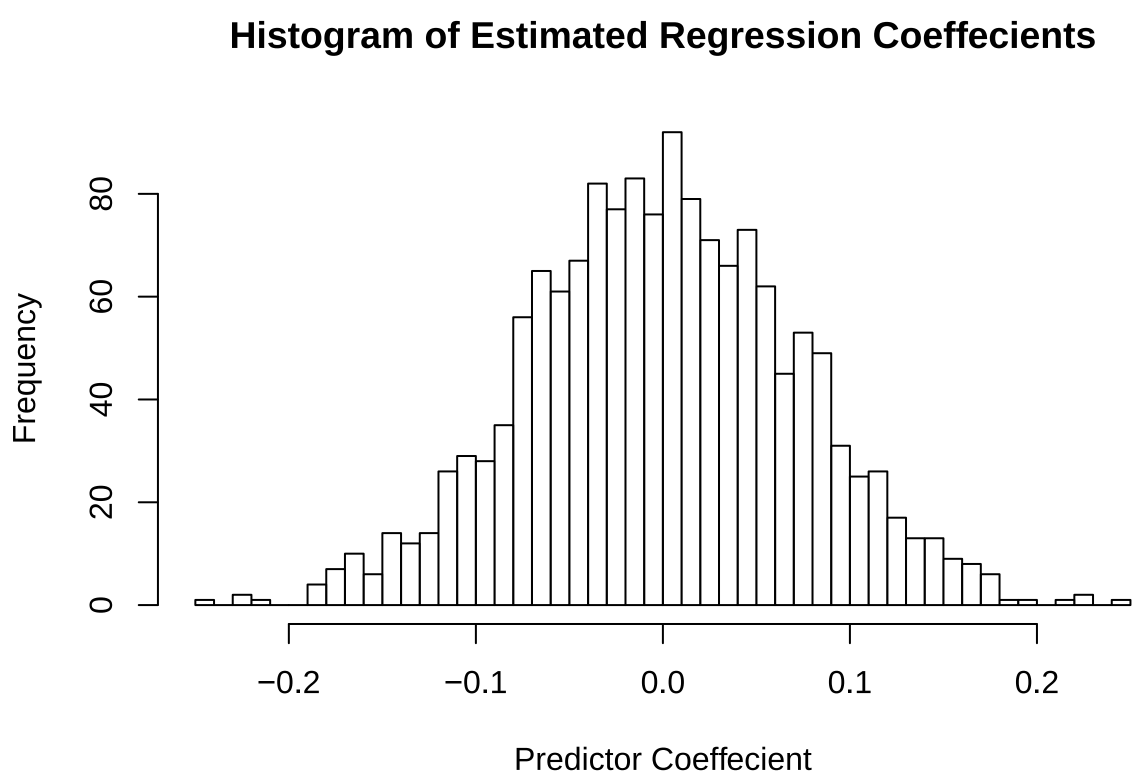
```
beta_hat_1 = apply(data.frame(rep(0, 1500)), 1, function(y) coef(lm(response ~
    predictor, data = sim_slr(n = 25, x = x, beta_0 = 10, beta_1 = 0,
    sigma = 1)))["predictor"])
head(beta_hat_1)
```

```
## [1] 0.0279182903 0.0690496708 0.0114883979 0.0001078901 0.0238075684
## [6] 0.0529488938
```

(b) Plot a histogram of `beta_hat_1`. Comment on the shape of this histogram.

Solution

```
hist(beta_hat_1, breaks = 50, main = "Histogram of Estimated Regression Coeffecients",
    xlab = "Predictor Coeffecient")
```



The shape verifies that the distribution of coeffecients is Normal.

(c) Import the data in `skeptic.csv` and fit a SLR model. The variable names in `skeptic.csv` follow the same convention as those returned by `sim_slr()`. Extract the fitted coefficient for β_1.

Solution

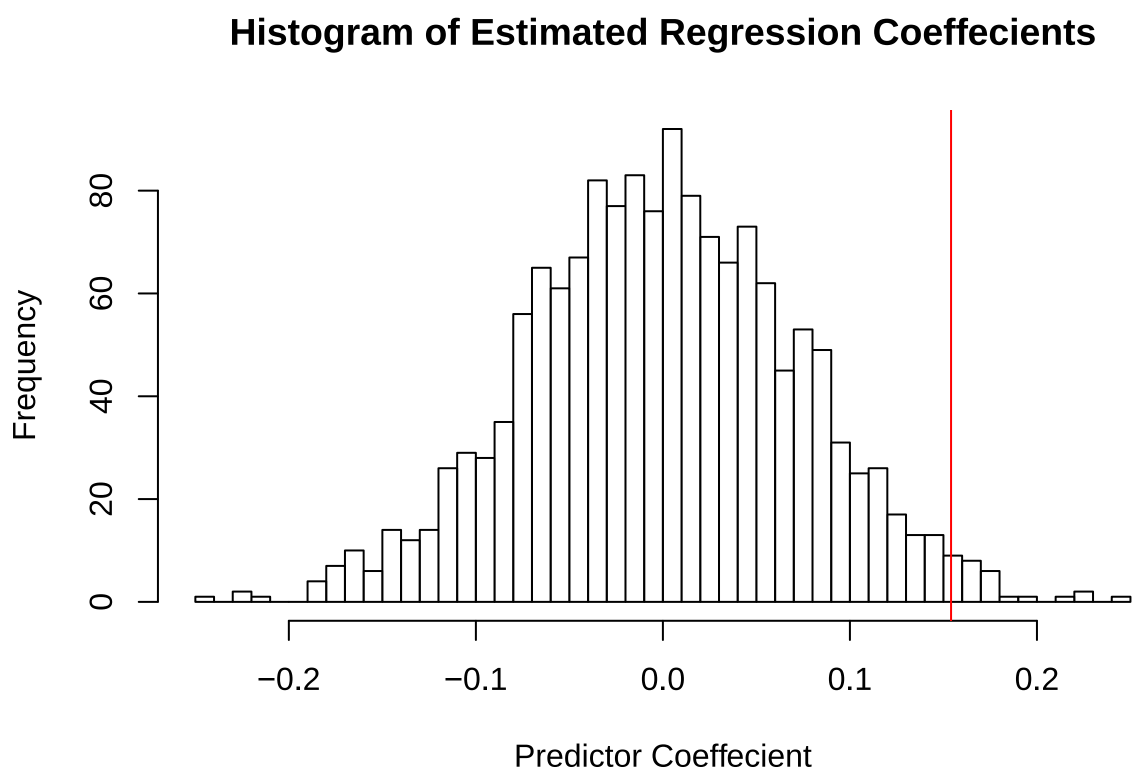
```r
skeptic <- read.csv("skeptic.csv")
beta_1_hat = coef(lm(response ~ predictor, data = skeptic))["predictor"]
beta_1_hat
```

```
## predictor
##  0.154081
```

(d) Re-plot the histogram from **(b)**. Now add a vertical red line at the value of $\hat{\beta}_1$ in part **(c)**. To do so, you'll need to use `abline(v = c, col = "red")` where `c` is your value.

Solution

```r
hist(beta_hat_1, breaks = 50, main = "Histogram of Estimated Regression Coeffecients",
    xlab = "Predictor Coeffecient")
abline(v = beta_1_hat, col = "red")
```



(e) Your value of $\hat{\beta}_1$ in **(c)** should be positive. What proportion of the `beta_hat_1` values are larger than your $\hat{\beta}_1$? Return this proportion, as well as this proportion multiplied by 2.

Solution

```r
prop = length(beta_hat_1[beta_hat_1 > beta_1_hat])/length(beta_hat_1)
prop
```

```
## [1] 0.016
```

```r
2 * prop
```

```
## [1] 0.032
```

(f) Based on your histogram and part **(e)**, do you think the `skeptic.csv` data could have been generated by the model given above? Briefly explain.

Solution

```r
within_95 = beta_1_hat < (mean(beta_hat_1) + 2*sd(beta_hat_1))
within_95
```

```
## predictor
##     FALSE
```

The data in `skeptic.csv` could not have been generated by model given above, as the estimated coeffecient doesnt lie in 95% CI.We are *Skeptic* about that.

Exercise 5 (Comparing Models)

For this exercise we will use the data stored in `goalies.csv`. It contains career data for all 716 players in the history of the National Hockey League to play goaltender through the 2014-2015 season. The variables in the dataset are:

- `Player` - NHL Player Name
- `First` - First year of NHL career
- `Last` - Last year of NHL career
- `GP` - Games Played
- `GS` - Games Started
- `W` - Wins
- `L` - Losses
- `TOL` - Ties/Overtime/Shootout Losses
- `GA` - Goals Against
- `SA` - Shots Against
- `SV` - Saves
- `SV_PCT` - Save Percentage
- `GAA` - Goals Against Average
- `SO` - Shutouts
- `MIN` - Minutes
- `G` - Goals (that the player recorded, not opponents)
- `A` - Assists (that the player recorded, not opponents)
- `PTS` - Points (that the player recorded, not opponents)
- `PIM` - Penalties in Minutes

For this exercise we will define the "Root Mean Square Error" of a model as

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}.$$

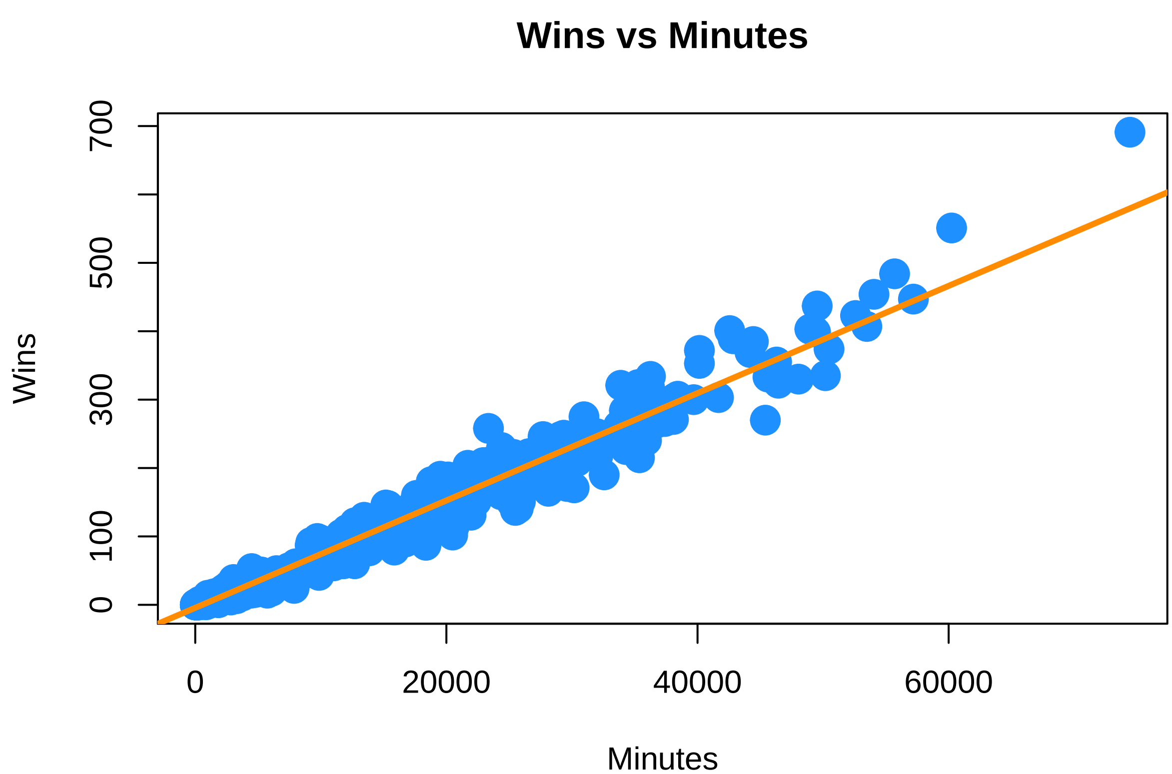
(a) Fit a model with "wins"" as the response and "minutes" as the predictor. Calculate the RMSE of this model. Also provide a scatterplot with the fitted regression line.

Solution

```r
goalies <- read.csv("goalies.csv")
model = lm(W ~ MIN, data = goalies)
RMSE = sqrt(sum((model$residuals)^2)/length(goalies$W))
RMSE
```

```
## [1] 16.72244
```

```r
plot(W ~ MIN, data = goalies, xlab = "Minutes", ylab = "Wins",
    main = "Wins vs Minutes", pch = 20, cex = 3, col = "dodgerblue")
abline(model, lwd = 3, col = "darkorange")
```



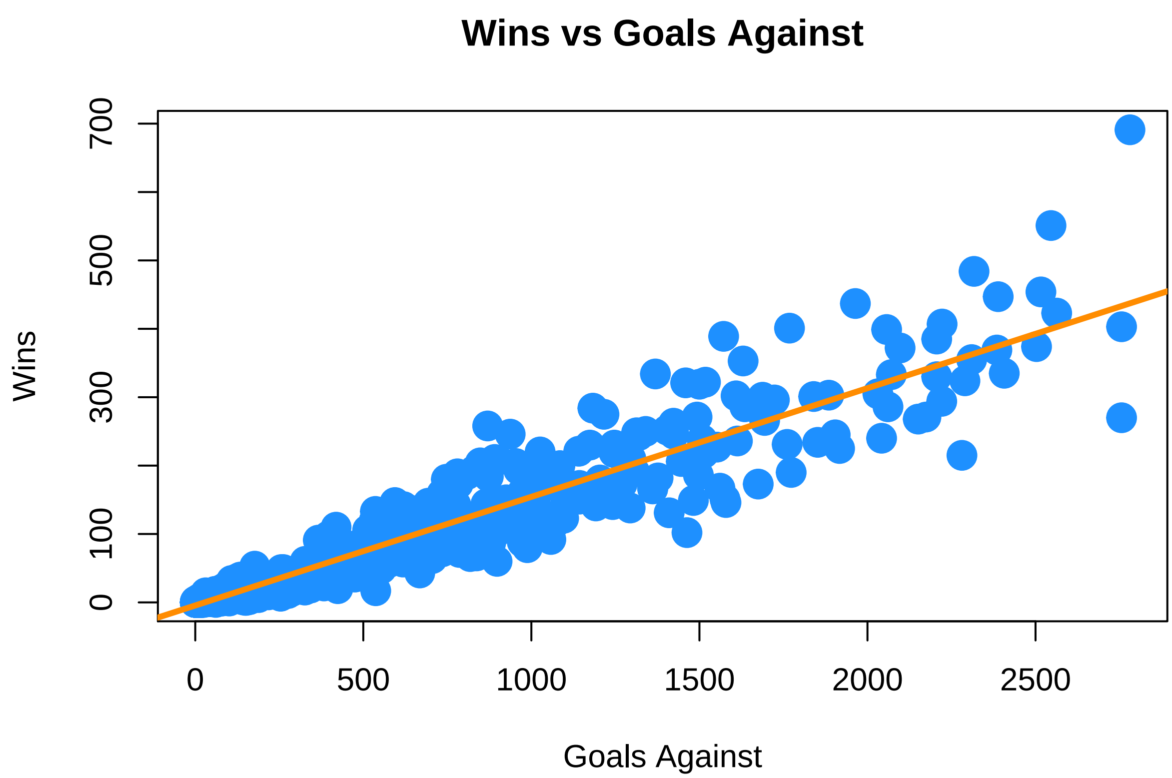
(b) Fit a model with "wins"" as the response and "goals against" as the predictor. Calculate the RMSE of this model. Also provide a scatterplot with the fitted regression line.

Solution

```r
goalies <- read.csv("goalies.csv")
model = lm(W ~ GA, data = goalies)
RMSE = sqrt(sum((model$residuals)^2)/length(goalies$W))
RMSE
```

```
## [1] 31.01641
```

```r
plot(W ~ GA, data = goalies, xlab = "Goals Against", ylab = "Wins",
    main = "Wins vs Goals Against", pch = 20, cex = 3, col = "dodgerblue")
abline(model, lwd = 3, col = "darkorange")
```



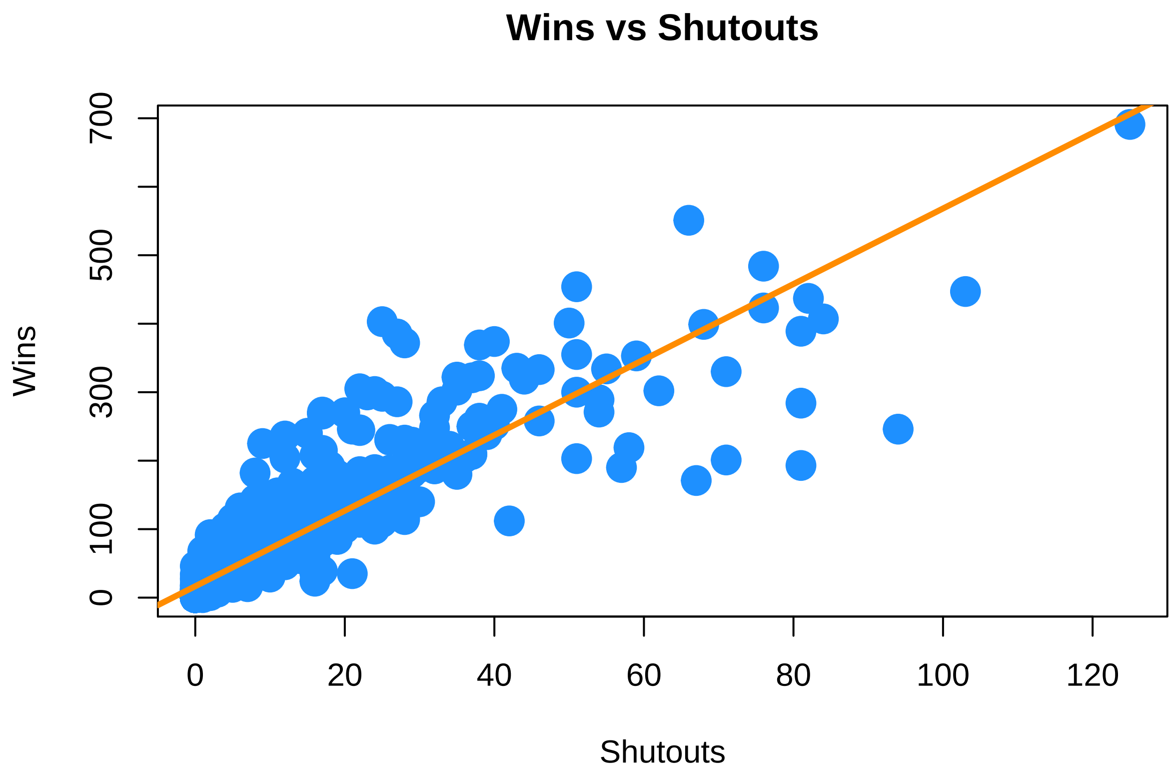
(c) Fit a model with "wins"" as the response and "shutouts" as the predictor. Calculate the RMSE of this model. Also provide a scatterplot with the fitted regression line.

Solution

```r
goalies <- read.csv("goalies.csv")
model = lm(W ~ SO, data = goalies)
RMSE = sqrt(sum((model$residuals)^2)/length(goalies$W))
RMSE
```

```
## [1] 44.74434
```

```r
plot(W ~ SO, data = goalies, xlab = "Shutouts", ylab = "Wins",
    main = "Wins vs Shutouts", pch = 20, cex = 3, col = "dodgerblue")
abline(model, lwd = 3, col = "darkorange")
```



(d) Based on the previous three models, which of the three predictors used is most helpful for predicting wins? Briefly explain.

Solution

Based on the previous three models, "minutes" is the best predictor for "wins" as it has least RMSE.

(e) *This question is not graded. You may delete it if you like.* How many games did Ed Belfour win? Why is he David Dalpiaz's favorite goalie on this list?

Solution

```
goalies[goalies$Player == "Ed Belfour*", "W"]
```

```
## [1] 484
```

He is ranked 3rd in number of Wins.