# STAT167 Lab #6 - Spring 2025

Ethan Choi

2025/5/9

## Contents

**Discussion/Lab #6 instructions** 1

Install necessary package . . . 2

Load necessary packages . . . 2

Set the random seed . . . 2

**The `Boston` data set** 3

**Lecture Review - Simple linear regression** 3

**Exercise #1** . . . 4

**Exercise #2** . . . 5

**Exercise #3** . . . 5

---

# Discussion/Lab #6 instructions

This week, we will review example code for simple linear regression.

- First, download the `rmd` file from Canvas.
- Open this `rmd` file in RStudio and click `Knit -> Knit to PDF` to render it to PDF format. You need to have `LaTex` installed on the computer to render it to PDF format. If not, you can also render it to HTML format.
- Read this `rmd` file and the rendered `pdf/html` file side-by-side, to see how this document was generated!
- Be sure to play with this document! Change it. Break it. Fix it. The best way to learn R Markdown (or really almost anything) is to try, fail, then find out what you did wrong.
- Read over the example code and the output. If you have any questions about certain functions or parameters, it is the time to ask!
- There are some exercises through out this document. Replace **INSERT_YOUR_ANSWER** with your own answers. Knit the file, and check your results.

**Please comment your R code thoroughly, and follow the R coding style guideline (https://google.github.io/styleguide/Rguide.xml). Partial credit will be deducted for insufficient commenting or poor coding styles.**

**Lab submission guideline**

- After you completed all exercises, save your file to `FirstnameLastname-SID-lab6.rmd` and save the rendered pdf file to `FirstnameLastname-SID-lab6.pdf`. If you can not knit it to pdf, knit it to html first and then print/save it to pdf format.
- Submit **BOTH your source `rmd` file and the knitted `pdf` file** to **GradeScope**. Do NOT create a zip file.
- You can submit multiple times, you last submission will be graded.

---

## Install necessary package

Note that you only need to install each package once. Then you can comment out the following installation lines.

```
#install.packages("MASS")
```

## Load necessary packages

```
library(tidyverse) # for `ggplot2`, `dplyr`, `tibble`, and more
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.4
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error

library(MASS) # for the `Boston` data set
##
## Attaching package: 'MASS'
##
## The following object is masked from 'package:dplyr':
##
##     select
```

## Set the random seed

```
# set the random seed so that the analysis is reproducible
set.seed(167)
```

## The Boston data set

The Boston data set (included in the MASS library) contains housing values in 500+ suburbs of Boston.

```
?Boston # full documentation
## starting httpd help server ... done
glimpse(Boston)
## Rows: 506
## Columns: 14
## $ crim    <dbl> 0.00632, 0.02731, 0.02729, 0.03237, 0.06905, 0.02985, 0.08829,~
## $ zn      <dbl> 18.0, 0.0, 0.0, 0.0, 0.0, 0.0, 12.5, 12.5, 12.5, 12.5, 12.5, 1~
## $ indus   <dbl> 2.31, 7.07, 7.07, 2.18, 2.18, 2.18, 7.87, 7.87, 7.87, 7.87, 7.~
## $ chas    <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ nox     <dbl> 0.538, 0.469, 0.469, 0.458, 0.458, 0.458, 0.524, 0.524, 0.524,~
## $ rm      <dbl> 6.575, 6.421, 7.185, 6.998, 7.147, 6.430, 6.012, 6.172, 5.631,~
## $ age     <dbl> 65.2, 78.9, 61.1, 45.8, 54.2, 58.7, 66.6, 96.1, 100.0, 85.9, 9~
## $ dis     <dbl> 4.0900, 4.9671, 4.9671, 6.0622, 6.0622, 6.0622, 5.5605, 5.9505~
## $ rad     <int> 1, 2, 2, 3, 3, 3, 5, 5, 5, 5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4,~
## $ tax     <dbl> 296, 242, 242, 222, 222, 222, 311, 311, 311, 311, 311, 311, 31~
## $ ptratio <dbl> 15.3, 17.8, 17.8, 18.7, 18.7, 18.7, 15.2, 15.2, 15.2, 15.2, 15~
## $ black   <dbl> 396.90, 396.90, 392.83, 394.63, 396.90, 394.12, 395.60, 396.90~
## $ lstat   <dbl> 4.98, 9.14, 4.03, 2.94, 5.33, 5.21, 12.43, 19.15, 29.93, 17.10~
## $ medv    <dbl> 24.0, 21.6, 34.7, 33.4, 36.2, 28.7, 22.9, 27.1, 16.5, 18.9, 15~
```

---

## Lecture Review - Simple linear regression

First, let's run a simple linear regression to predict median housing values medv using only one predictor lstat – percent of lower status population.

```
lm.fit <- lm(medv ~ lstat, Boston)
summary(lm.fit)
##
## Call:
## lm(formula = medv ~ lstat, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.168  -3.990  -1.318   2.034  24.500
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 34.55384    0.56263   61.41   <2e-16 ***
## lstat       -0.95005    0.03873  -24.53   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.216 on 504 degrees of freedom
## Multiple R-squared:  0.5441, Adjusted R-squared:  0.5432
## F-statistic: 601.6 on 1 and 504 DF,  p-value: < 2.2e-16
```

When calling `lm(medv ~ lstat, Boston)`, the `lm` function builds the following linear model to fit the data:

$$Y = f(X) + \epsilon = \beta_0 + \beta_1 X + \epsilon,$$

where $Y$=`Boston$medv`, $X$=`Boston$lstat`, and $\epsilon$ is the measurement noise/error.

In the lecture, we have learned that the loss function for the least squares regression is the **residual sum of squares (RSS)**.

$$\begin{aligned}\text{RSS} = L(\beta_0, \beta_1) &= \sum_{i=1}^{n} r_i^2 \\ &= \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2\end{aligned}$$

And the analytical solution that **minimizes RSS** is

$$\begin{aligned}\hat{\beta}_1 &= \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \\ \hat{\beta}_0 &= \bar{y} - \hat{\beta}_1 \bar{x}\end{aligned}$$

The estimates $\hat{\beta}_0$ and $\hat{\beta}_1$ is saved in `lm.fit$coefficients`.

```
names(lm.fit)
##  [1] "coefficients" "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"
lm.fit$coefficients
## (Intercept)       lstat
##  34.5538409  -0.9500494
```

**Exercise #1**

```
lm.fit2 <- lm(medv ~ rm, data = Boston)
summary(lm.fit2)
```

```
##
## Call:
## lm(formula = medv ~ rm, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -23.346  -2.547   0.090   2.986  39.433
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -34.671      2.650  -13.08   <2e-16 ***
## rm             9.102      0.419   21.72   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.616 on 504 degrees of freedom
```

```
## Multiple R-squared:  0.4835, Adjusted R-squared:  0.4825
## F-statistic: 471.8 on 1 and 504 DF,  p-value: < 2.2e-16
```

Do you think there is a strong relationship between `medv` and `lstat`?
Explain your answer. Which statistic(s) did you use to draw your conclusion?

Yes, I do think that there is a strong relationship between medv and lstat. I think that the relationship is strongly negative. The coefficient for lstat is -0.95005, meaning for each miniscule increase in lstat, the median home value decreases by about 0.95005. Because the p-value for the lstat coefficient is small, this indicates that the relationship is statistically significant.

---

In the lecture, we learned that one measurement of the regression model accuracy is the $R^2$ statistic, which is the proportion of variance in the response variable that can be explained by the regression fit.

**Exercise #2**

(a) Look at the `summary(lm.fit)` output in the previous code chunk, what is $R^2$ statistic of your model?

The R squared statistic is 0.5441

(b) For simple linear regression, $R^2$ statistic equals $r^2$, where $r$ is the correlation coefficient between $X$ and $Y$.

Use the `cor` function to calculate $r$, compare $r^2$ with $R^2$. Do you get the same value?

```
r <- cor(Boston$lstat, Boston$medv)
r_squared <- r^2
r_squared
```

```
## [1] 0.5441463
```

Yes, I get the same value.

---

**Exercise #3**

Choose another predictor other than `lstat` and build a simple linear regression model to predict `medv` using your new predictor. Compare your model with the `lm.fit` (`medv` ~ `lstat`) model. Which model is better? Explain your answer. Which statistic(s) did you use to draw your conclusion?

```
lm.rm <- lm(medv ~ rm, data = Boston)
summary(lm.rm)
```

```
##
## Call:
## lm(formula = medv ~ rm, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -23.346  -2.547   0.090   2.986  39.433
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -34.671      2.650  -13.08   <2e-16 ***
## rm             9.102      0.419   21.72   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.616 on 504 degrees of freedom
## Multiple R-squared:  0.4835, Adjusted R-squared:  0.4825
## F-statistic: 471.8 on 1 and 504 DF,  p-value: < 2.2e-16
```

The R squared value for the rm model that I chose is roughly 0.483. This is lower than the R squared value of 0.5441 from the original model. This implies that the model using lstat explains more variance in housing prices than the rm model . So, this means that the lstat model is superior in terms of predictive power. ***