HUDM 5123 - Linear Models and Experimental Design
Lab 01 - Introduction to Linear Regression in R

1 Introduction

Today our goals are for you to become more familiar with how to

(a) Import data into R.

(b) Examine and understand a data frame.

(c) Run and interpret a linear regression analysis.

(d) Graph results.

2 Importing Data

The most straightforward ways to import data into R involve reading in .csv files or .txt files. Files that end in the sufficx .csv contain comma-separated values and may or may not contain a header. A header is present when the first row of the data gives the variable names. The `read.csv()` function is used to read in data that are in comma-separated format. Each R function will have an associated help file that describes the arguments, though sometimes the help files are hard to understand, especially for new users. In any case, let's look at the help file on `read.csv()` by running the following line.

```
?read.csv
```

Note that the first argument is called `file`. According to the Arguments section of the help file, this first argument is the name of the file which the data are to be read from. The second argument is `header`. If set to `TRUE`, the first row of the .csv file will be used as variable names; if set to `FALSE`, it will not. The third argument is called `sep` and is used to indicate which character is used to separate data on the same line. For `read.csv`, `sep = ","` by default.

Two options for setting the file argument are to either list the full path location of the file on your computer or to open a browser window. We will use the ECLS data for this example, which do have the variable names at the top of the file. To open with the specific file path, fill in the path on your machine similar to the following:

```
ecls <- read.csv(file = "/Users/keller/Teaching/2020 SPRING/ecls.csv",
                 header = TRUE)
```

To open the file via browser window instead, use

```
ecls <- read.csv(file = file.choose(),
                 header = TRUE)
```

Note that in both cases, we have assigned the name `ecls` to the imported data. Also note that when specifying the file location directly, you must use forward slashes like /, not backslashes like \. Another thing to note is that if your data has special codes that denote missing data such as, for example, -9999 and -8888, you can have them converted to NA, which is R's system missing value, on import by specifying the argument `na.strings = c("-9999", "-8888")`. Note that the file path location can be a url as well.

If your data are not in a .csv file but instead in a .txt file, the columns are likely separated by white space. In that case, you will use `read.table()` with argument `sep = " "` to import data. If your data are stored in another program's format (e.g., in SPSS as a .sav file), it is easy to open the data in that program and then save as or export as a .csv which can then be easily imported into R.

There are two other ways you might want to access data in R. The first is through an .Rdata file. A .Rdata suffix is used for data objects that have been saved from an R session. If you open a .Rdata file that has a data frame stored in it with RStudio, for example, you will directly add the data frame to your current workspace. Similarly, there are some data sets that are provided with R or in add-on packages. To see which data sets are available to you, type `data()`.

3 Examining Your Data

The data frame is a matrix-like data structure in R that has row names and column names. When you use `read.csv()` to import data it will be coerced to a data frame. Some functions that are useful for learning about the size and contents of a data frame include

- `dim()`; gives the dimensions: number of rows followed by number of columns
- `names()`; prints the column/variable names
- `head()`; prints the first six rows of the data frame
- `tail()`; prints the last six rows
- `str()`; gives the structure, including the type of each variable and the first few values
- `var()`; gives the variance/covariance matrix
- `cor()`; gives the correlation matrix
- Note that it is often convenient to round the output to a certain number of decimal places. This is achieved by using the `round()` function with argument `digits`. For example, the command `round(x = cor(ecls), digits = 2)` would produce the correlation matrix rounded to two decimal places.
- `plot()`; creates bivariate scatterplots of all variables

- Note that some functions such as `plot()`, `var()`, and `cor()` will not run unless all the variables in the data frame are numeric. If there are any factors in the data frame, they must be left out of the call. For example, suppose the 2nd and 4th columns/variables in `ecls` are coded as factors. Then `var(ecls[,-c(2,4)])` could be used to create the variance/covariance matrix for all variables except for the 2nd and 4th.

Task 1 *Use the functions above to explore the* `mtcars` *data set, which is available in base R (i.e., no package needs to be loaded to access it) and write up your results.*

The dollar sign $, if placed directly after the name of a data frame, may be used to access named objects of the data frame (i.e., the variables). Thus, because the `ecls` data frame contains a variable named `SES`, one can access that variable by running `ecls$SES`. The function, `str()`, mentioned above, can be used to determine which variables are coded as factors. In the `ecls` data, you will find that none of the variables are factors at the start. However, we might wish to code students' `GENDER` as a factor for subsequent regression analyses. There are, in general, two approaches to recoding a variable: you can replace the old variable with the new one, or add the new one without deleting the old. Here we will demonstrate both approaches.

1. To replace the `GENDER` variable with a factor version, we would need to reassign the name `GENDER` as follows

```
ecls$GENDER <- factor(x = ecls$GENDER,
                      levels = c(0,1),
                      labels = c("female", "male"))
```

2. To add a factor without deleting the original `GENDER` variable we would need to use a new name and add it to the data frame as an additional variable as follows

```
ecls$GENDER_fac <- factor(x = ecls$GENDER,
                          levels = c(0,1),
                          labels = c("female", "male"))
```

Task 2 *Create a new variable called* `am_f` *that is a factored version of the transmission-type indicator variable.*

3.1 Graphical Exploration

There are a number of plots that are built into R that can be helpful for examining your quantitative data.

- `hist()`; creates a histogram, the argument `breaks` controls (roughly) how many breaks there are in the domain of the plot

- `plot(density())`; creates a kernel density plot

- `plot(x = , y = )`; creates a bivariate scatterplot

For factor variables, tables are helpful. For example, `table(ecls$GENDER_fac)`. To show the relationship between a categorical factor and a quantitative variable, boxplots are useful. For example, `boxplot(ecls$MATH5   ecls$GENDER_fac)`. Most plotting functions allow the following arguments:

- `xlab`; quoted label for x-axis
- `ylab`; same for y
- `main`; main title

Task 3 *Create one plot of each of the above types using the variables in* `mtcars`*. Be sure to specify appropriate x and y labels and main titles.*

4 Linear Regression

Linear regression uses two main arguments: the formula and the data. Formulas are specified using a tilde, $\sim$, where the variable(s) on the left side of tilde are the outcome(s) and variables on the right side, separated by plus signs, $+$, are predictors. After a regression is run and saved as a named object, the `summary()` function is used to extract important information.

```
> lm1 <- lm(formula = MATH5 ~ SES,
+           data = ecls)
> summary(lm1)

Call:
lm(formula = MATH5 ~ SES, data = ecls)

Residuals:
    Min      1Q  Median      3Q     Max
-55.369 -11.685   4.129  14.632  40.625

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  123.777      2.132  58.052  < 2e-16 ***
SES           13.033      3.020   4.315 3.81e-05 ***
---

Residual standard error: 20.84 on 98 degrees of freedom
Multiple R-squared:  0.1597,Adjusted R-squared:  0.1511
F-statistic: 18.62 on 1 and 98 DF,  p-value: 3.814e-05
```

Task 4 *Run a simple regression of* `mpg` *on* `hp` *using the* `mtcars` *data. Interpret the slope, intercept, and R^2 value in context.*

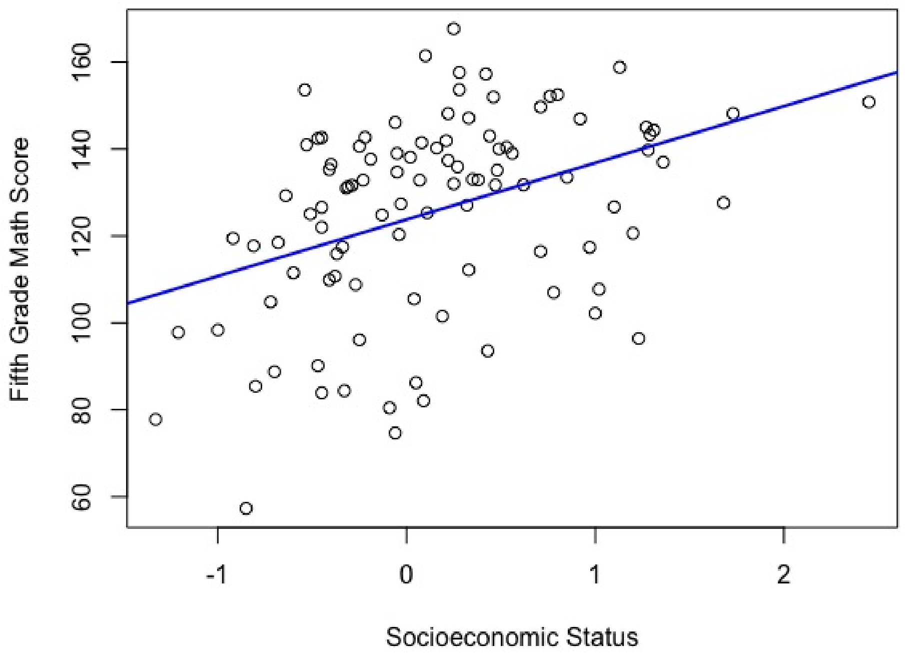

Figure 1: Simple linear regression

```
> lm2 <- lm(formula = MATH5 ~ SES + MATHK,
+            data = ecls)
> summary(lm2)

Call:
lm(formula = MATH5 ~ SES + MATHK, data = ecls)

Residuals:
    Min      1Q  Median      3Q     Max
-45.248 -10.374   1.209  10.422  37.526

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  75.5206     7.0311  10.741  < 2e-16 ***
SES           5.3757     2.6913   1.997   0.0486 *
MATHK         1.5223     0.2149   7.084 2.24e-10 ***
---

Residual standard error: 17 on 97 degrees of freedom
Multiple R-squared:  0.4462,Adjusted R-squared:  0.4347
F-statistic: 39.07 on 2 and 97 DF,  p-value: 3.582e-13
```

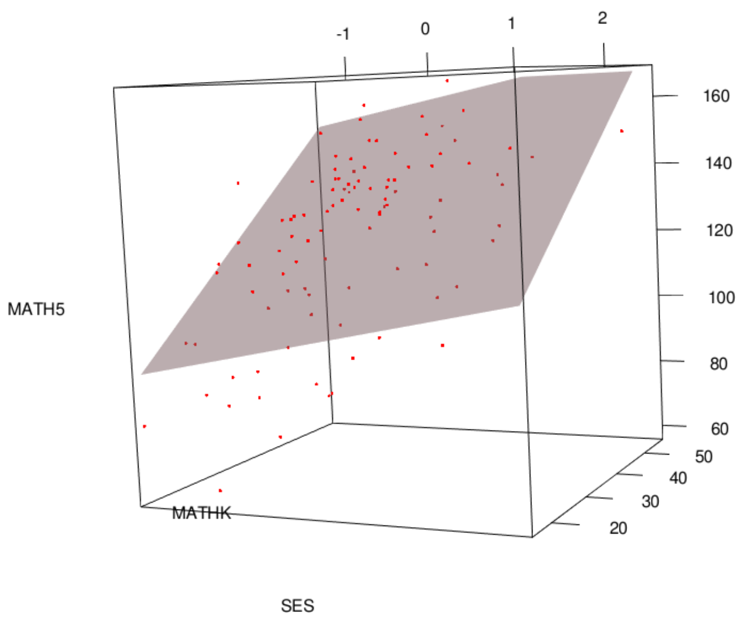

Figure 2: Multiple linear regression

Task 5 *Run a multiple regression of* `mpg` *on* `hp` *and* `wt` *using the* `mtcars` *data. Interpret the slope, intercept, and* R^2 *value in context.*

```
> lm3 <- lm(formula = MATH5 ~ GENDER_fac,
+           data = ecls)
> summary(lm3)

Call:
lm(formula = MATH5 ~ GENDER_fac, data = ecls)

Residuals:
    Min      1Q  Median      3Q     Max
-66.695 -14.422   7.135  15.790  40.081

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)     124.025      3.142  39.472   <2e-16 ***
GENDER_facmale    3.554      4.535   0.784    0.435
---

Residual standard error: 22.66 on 98 degrees of freedom
Multiple R-squared:  0.006226,Adjusted R-squared:  -0.003914
F-statistic: 0.614 on 1 and 98 DF,  p-value: 0.4352
```

Task 6 *Create a factor variable for* `am` *and then regress* `mpg` *on* `am_fac`*. Interpret the slope,*

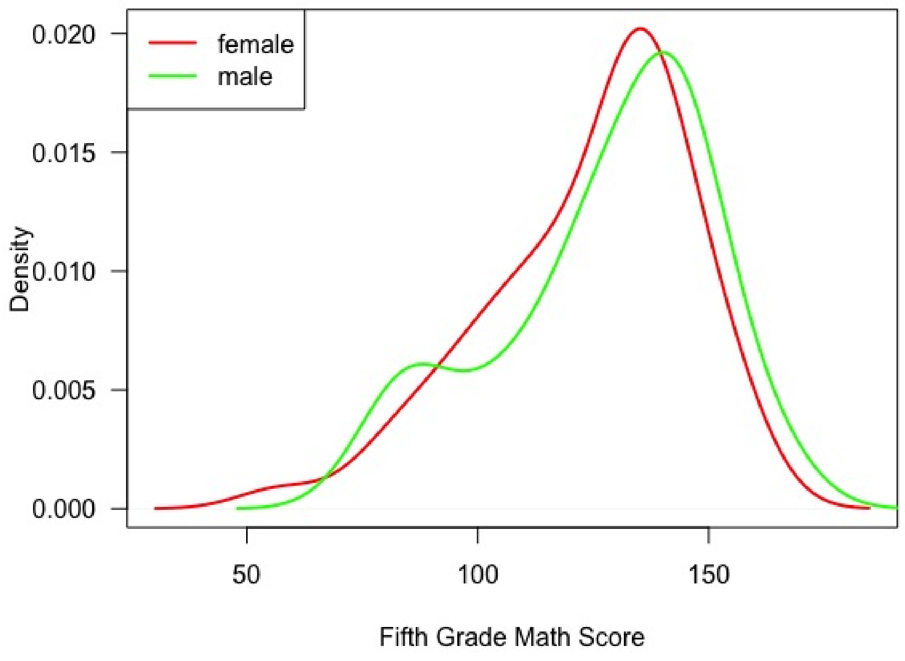

Figure 3: Kernel density plots of math score by gender

intercept, and R^2 value in context. Create a boxplot or kernel density plot with appropriate labels.

```
> lm4 <- lm(formula = MATH5 ~ GENDER_fac + MATHK + GENDER_fac:MATHK,
+           data = ecls)
> summary(lm4)

Call:
lm(formula = MATH5 ~ GENDER_fac + MATHK + GENDER_fac:MATHK, data = ecls)

Residuals:
    Min      1Q  Median      3Q     Max
-45.054 -11.421   0.703  11.249  40.983

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)           64.3784    10.2277   6.294 9.21e-09 ***
GENDER_facmale        11.5213    13.5703   0.849    0.398
MATHK                  1.8294     0.3050   5.999 3.50e-08 ***
GENDER_facmale:MATHK  -0.2289     0.4041  -0.566    0.572
---
```

```
Residual standard error: 17.28 on 96 degrees of freedom
Multiple R-squared:  0.4335,Adjusted R-squared:  0.4158
F-statistic: 24.49 on 3 and 96 DF,  p-value: 7.475e-12
```

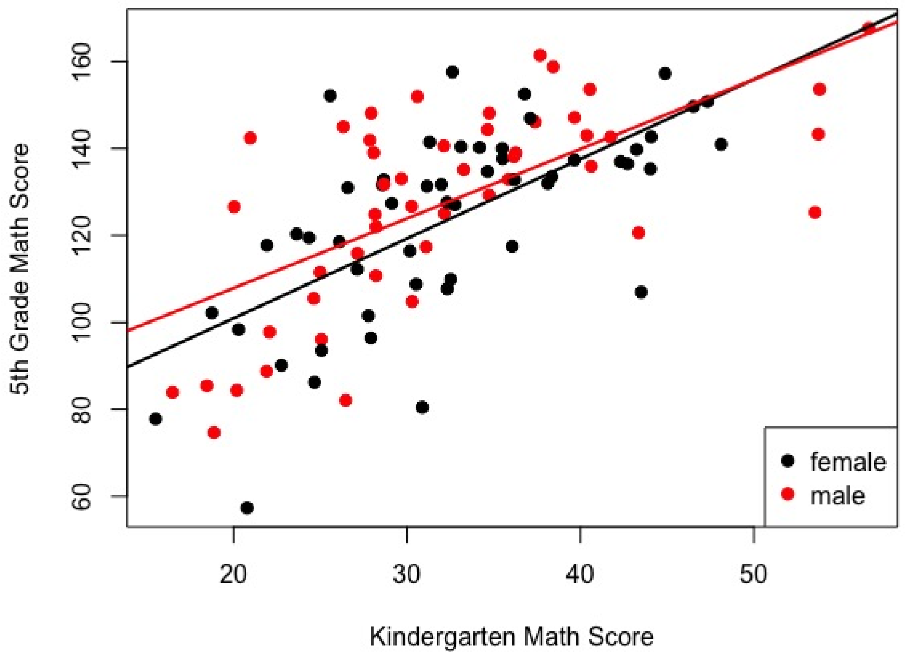

Figure 4: Multiple linear regression with interaction between categorical and quantitative predictors

Task 7 *Run a multiple regression of* `mpg` *on* `hp` *and* `am_fac` *including their interaction using the* `mtcars` *data. Interpret the slope, intercept, and R^2 value in context. Create an interaction plot with appropriate labels.*