2.9 Homework

This is the first practical homework assignment. We'll be using R for data analysis, and we'll start by getting that all set up. Then we'll start analyzing two data sets! This sheet will walk you through the first, and then leave you to think a bit about the second. NOTE: These instructions assume you're using windows. Linux and Mac should be pretty similar, but there may be a few differences. All of the computers in the computer lab have R preinstalled and run windows!

2.9.1 Setting up R

If you're using a lab machine, it has R. Log into the machine using username `guest135` and password `SLs2018-8743`; it might take a minute or two to log in. Once the computer starts working, you can open R by finding "R" in the start menu. If you're using your own computer, go to `http://www.r-project.org/` and click the link to "download R." Once it's downloaded, open it up.

2.9.2 Using R

R can do basic math, so to get a feel for it, try typing 2+2 then hitting return. You should see:

```
> 2+2
[1] 4
```

While using R, it helps to have a specific folder on your computer to place your data. On the lab computers, I suggest creating a desktop folder and calling it StatModelling to use during this lab. To get the data, you can download it from `samgutekunst.com/mc2018stats/`. This first data set discusses the impact of LSD on performance on math exams. To load it into R:

- Download the LSD file and place it in the folder you created for this lab.
- In R, click on the `file` menu then click on `Change dir...`. Scroll through and select the folder where you saved lsd.txt.
- To read the data into R, type

  ```
  lsddat <- read.table("lsd2.txt", header=T)
  ```


We've now created an object lsddat that stores the information from the table, and the `header=T` part of the code is an option that tells R that the first row contains "headers" that can be used to identify the columns. To look at the object, type `lsddat`; your R Console should look like

```
 > lsddat
   LSD  Math
1 1.17 78.93
2 2.97 58.20
3 3.26 67.47
4 4.69 37.47
5 5.83 45.65
6 6.00 32.92
7 6.41 29.97
```

NOTE: The ">"is not something you need to type, here or later.

- Note that column with heading `LSD` lists the average tissue concentration of LSD in several volunteers at each of 7 time points. The second column lists their average score on a mathematics exam (when they had that concentration of LSD). Try typing `lsddat$LSD`.

2.9.3 Exploratory Data Analysis (read: looking at the data)

- First let's create a scatterplot of the data. To generate a graph of the Math test score versus LSD concentration, type

  ```
  > plot(lsddat$LSD, lsddat$Math)
  ```

  There are lots of ways to make this graph more pretty. You can type `help(plot)` to open up a window with lots of information about the arguments it can take. To add axis and a title, instead enter:

  ```
  > plot(lsddat$LSD, lsddat$Math, ylab="Average Math Score",
  + xlab="Average LSD Concentration", main="Effect of LSD on Math Ability")
  ```

  Note: to do this, hit enter after the comma on the first line, and the plus symbol will automatically appear.

Question 1 Look at the graph. What does it suggest about how LSD effects someone's ability to do math?

Question 2 Write the equation of a simple linear model for the effect of LSD concentration on math test scores. Don't forget to write down what the ϵ_i are, and to identify the variables you use.

2.9.4 Fitting a Linear Model

- To fit a linear model, enter

```
> MathModel.lm <- lm(lsddat$Math~lsddat$LSD)
```

R won't return any info, but so long as you don't get an error message, you've fit a model! To actually view it, try the following:

```
> MathModel.lm

Call:
lm(formula = lsddat$Math ~ lsddat$LSD)

Coefficients:
(Intercept)   lsddat$LSD
     89.124       -9.009
```

- The model we're using (which you should have written down in question 2!) is: for $i = 1, ..., 7$, with

$$y_i = b_0 + b_1 x_i + c_i, \quad c_i \overset{iid}{\sim} N(0, \sigma^2),$$

where y_i is the average score on a math test at the ith time and x_i is the LSD concentration at the ith time. What the first output is saying is that

$$\hat{b_0} = 89.1, \quad \hat{b_1} = -9.01.$$

Also, note that is not necessary to name your model with a ".lm" at the end (e.g., we could have typed `MathModel <- lm(...)`). The ".lm" suffix is just common practice in R to help keep track that the object `MathModel.lm` is a linear model.

2.9.5 Checking Linear Models

- There's a lot of information stored in the model. Look at

```
> objects(MathModel.lm)}
[1] "assign"        "call"          "coefficients"  "df.residual"
[5] "effects"       "fitted.values" "model"         "qr"
[9] "rank"          "residuals"     "terms"         "xlevels"
```

To look at any of these, type, e.g.,

```
> MathModel.lm$residuals
```


- To plot the fitted line on top of the graph, we can use the arbitrary line function and the fitted model:

```
> plot(lsddat$LSD, lsddat$Math, ylab="Average Math Score",
+ xlab="Average LSD Concentration", main="Effect of LSD on Math Ability")
> abline(MathModel.lm)
```

Question 3 Qualitatively, how does the line seem to fit?

2.9.6 Cleaning up and Using R

Here are a handful of useful things you might want to do in R.

- If you use the up and down arrow keys, you can pull up things that have already typed.
- The "#" is what you can use to type comments, as in the code below.
- To view a list of everything you've created, type

```
> objects() # See what you've created
```

- To save what you've already done in R (all the objects you've created, etc.) in case you want to pause, or to save your work when you're done for future reference, go to the file worksheet and click "Save Workspace." I save my files with a .Ri extension. To pull back up your work after you exit R, click load workspace and find your file! If you saved it as a .Ri file, you'll need to make sure you can view all file types. NOTE: If you are using the computer lab computers, you might want to email the .Ri file to yourself in case the computers delete it when you log off.

2.9.7 Your Turn!

Now it's your turn look at a more complicated model! On the course website, download the Baseball.txt file into your R directory. This data comes from `http://lib.stat.cmu.edu/DASL/Stories/baberuth.html` and includes the Extra-Base-Power (EBP) and On-Base-Average (OBA) for several top hitters from baseball history. (These are both metrics that measure the performance of a player). You can ignore the sum column. There are two questions of interest here: how well can OBA be used to predict EBP? Are there any individual data points for which the model really does not seem to fit well, which may be indicative of an exceptional or a weak player (i.e., an individual player for which the model does not work well)?

First let's load up the data:

```
> bballDat <- read.table("Baseball.txt", header=T)
> head(bballDat) # To look at the first few entries
```

Question 4 Write the equation of a simple linear model for this scenario. Try fitting and looking at a plot with the data and fitted lines. Does the model seem to fit well?

Question 5 In the previous question, you may have found evidence that linear regression is not ideal. We'll now try a slightly more complicated linear model, a *quadratic model*. This means that we also want to include an x_i^2 term. Write a linear model describing this situation. Can you write it in matrix form?

To fit this model, you'll want to create a new variable to store the values of x_i^2. To do this:

```
> bballDat$OBA2<-bballDat$OBA^2
> head(bballDat) # the command head means: just show the first entries
          PLAYER OBA EBP SUM   OBA2
1      Babe.Ruth 481 271 752 231361
2   Ted.Williams 484 218 702 234256
3     Lou.Gehrig 447 219 666 199809
4     Jimmy.Foxx 429 213 642 184041
5 Rogers.Hornsby 449 185 634 201601
6 Hank.Greenberg 412 219 631 169744
> 481^2
[1] 231361
```

Note that we have a new column with what we want!

Now, to fit the model, the syntax is to use

```
quadMod.lm<-lm(bballDat$EBP~bballDat$OBA+bballDat$OBA2)
```

Plotting the actual curve is a little bit more complicated, and the abline command won't work. If you want to see the curve, you can use the following:

```
> quadPoly<-function(z) quadMod.lm$coefficients[3]*z^2+
+ quadMod.lm$coefficients[2]*z+quadMod.lm$coefficients[1]
# the second plus is just R's symbol for a continuation of the first line
> plot(bballDat$OBA, bballDat$EBP)
> curve(quadPoly, add=T)
```


The first line creates a new function, $\hat{b}_2 z^2 + \hat{b}_1 z + \hat{b}_2$, called `quadPoly`. The last line adds a plot of this polynomial to the graph of EBP vs OBA (which is what the "`add=T`" command tells us).

Question 6 How does this model fit? Does it fit better than the linear model? Do there appear to be any outliers? If so, who? What does this mean in the context of the real world? (hint: you can use R to identify points on a graph. For example, type:

```
>  plot(lsddat$LSD, lsddat$Math)
> identify(lsddat)
```

then go to the graph and click on a point. The number gives you the row that point corresponds to. To stop, right click and then hit "stop").

2.9.8 Writing Code in R

The following shows a full R function designed for a (very) toy simulation based on the model

$$Y_i = 100 + 5X_i + \epsilon_i,$$

where $\epsilon_i \sim N(0.64)$. Calling `generate(z)` generates z independent realizations of this model, where each realization uses $x_1 = 68, x_2 = 70, x_3 = 72, ..., x_{12} = 90$. For each realization, it fits a linear model G.lm and appends the fitted values of the y-intercept and slope, $\hat{b_0}$ and $\hat{b_1}$, to v1 and v2 respectively. It then computes the average value of $\hat{b_0}$ and $\hat{b_1}$ over the z realizations.

Try copying and pasting the code into R (you can do it in one giant batch and include comments). Run `generate(1)` a couple of times: the slope and y-intercept change, as does the plot. If you run `generate(100)` or `generate(1000)` (you may want to comment out the plot), and notice that the average estimate of $\hat{b_0}$ and $\hat{b_1}$ approach 100 and 5, respectively.

```
# Create a function generate that takes as input z,
# where z is the number of independent realizations to compute
generate <- function(z) {
# Create a list of the x_i: x=(68, 70, 72, ..., 90)
x=seq(68, 90, by=2)
# Create two empty lists
v1<- c()
v2 <- c()
# For i=1, ..., z
 for (i in 1:z) {
 # Generate a list of 12 realizations of a N(0, 64) random variable
Er<- rnorm(12, 0, 8)
```

```
# The observed data is the 100+5x+error
OD<-100+(5*x)+Er
# Fit a linear model, regressing the observed data on x
G.lm <- lm(OD~x)
# Add the intercept to v1 and slope to v2
 v1<- c(v1, G.lm$coefficients[1])
v2<- c(v2, G.lm$coefficients[2])
plot(x, OD)
}
# Output the average values of the intercept and slope
print(mean(v1))
print(mean(v2))
}
```

Question 7 Modify the code so that `generate(n)` creates n plots, each plot showing the true line $(100 + 5x)$ in red and the fitted line in blue. To change parameters like color in the `abline` command, note that it can be used with options `abline(LINETOPLOT, col="blue", lwd=5)` to set the color (col) and line width (lwd).