MTH 3210 Lab 5

Due Thu., May 9

1 Part A: Scatterplots and Correlation

1.1 Yellowstone Lake Data Set

Yellowstone Lake is located in Yellowstone National Park and covers an area about 136 mi^2, depending on the level of water in the lake. The water level varies each year in response to differences in the winter's snowpack accumulation, spring precipitation, and air temperatures.

The file **yellowstone_lake.txt** contains data on the maximum daily **outflow** (ft^3/s) and maximum daily **elevation** (ft) for each of the **years** 1926-2001.

1. Save the file **yellowstone_lake.txt** onto your computer, then type:

```
my.file <- file.choose()
```

   and choose the file in the pop-up window. The file's name and location will be stored in `my.file`.

2. Now read the data from **yellowstone_lake.txt** into R using:

```
lake.data <- read.table(my.file, header = TRUE)
```

   (Specifying `header = TRUE` tells R that the first row of **yellowstone_lake.txt** has the column headers.)

3. Extract the columns from the `lake.data` *data frame* into three *vectors*:

```
Year <- lake.data$Year
Outflow <- lake.data$Outflow
Elevation <- lake.data$Elevation
```

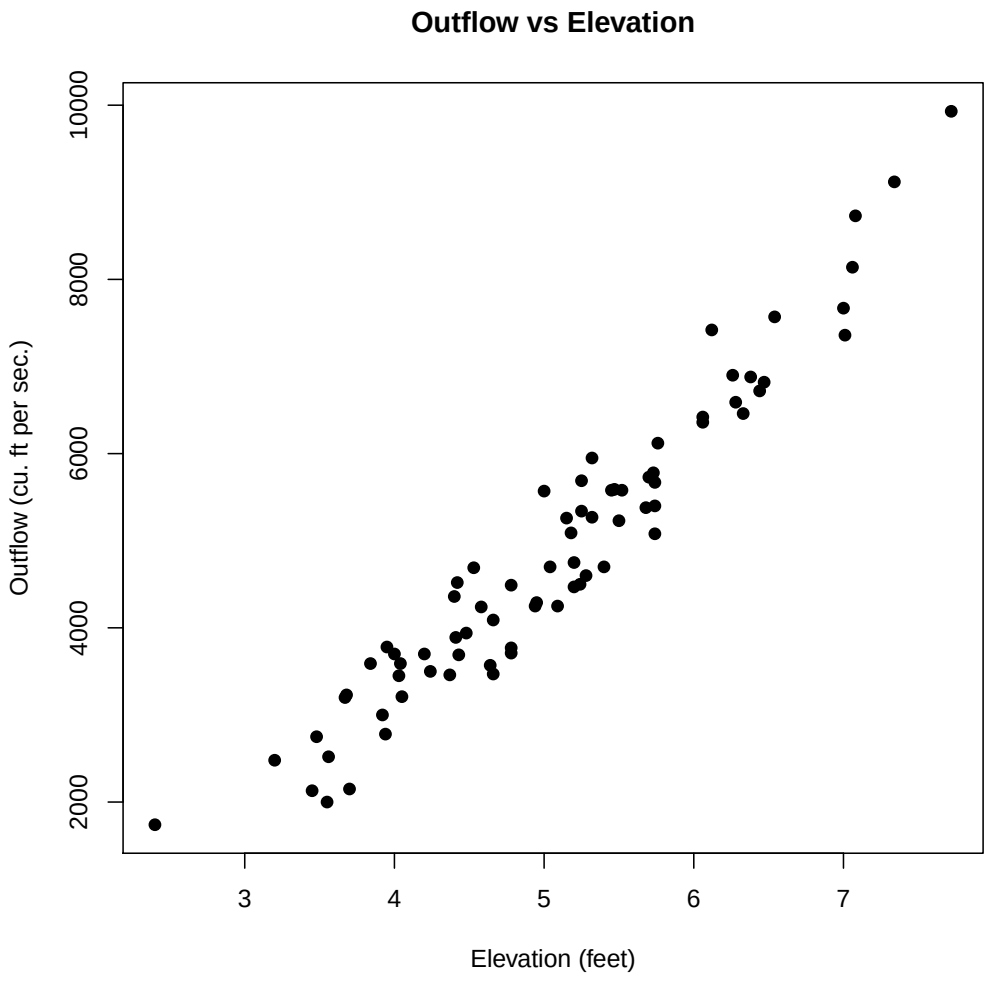
4. The function `plot()` will produce a ***scatterplot***. Among its arguments are

`x`	a data vector.
`y`	another data vector.
`pch`	an integer value specifying the plot character (symbol).
`type`	the type of plot that should be drawn ("p" for points, "l" for lines, "b" for both, etc., with default "p").
`xlab`	optional label for the x-axis (in quotes).
`ylab`	optional label for the y-axis (in quotes).
`main`	optional main title (in quotes).
`col`	optional color, e.g. "red", "blue", etc. (in quotes).

Type something like this:

```
plot(x = Elevation, y = Outflow, pch = 19)
```

to plot Yellowstone Lake's **outflow** versus its **elevation**. You should end up with something similar to this:



5. The function `cor()` will compute a ***correlation r***. It takes arguments:

`x`	a data vector.
`y`	another data vector.

and returns the ***correlation*** between `x` and `y`.

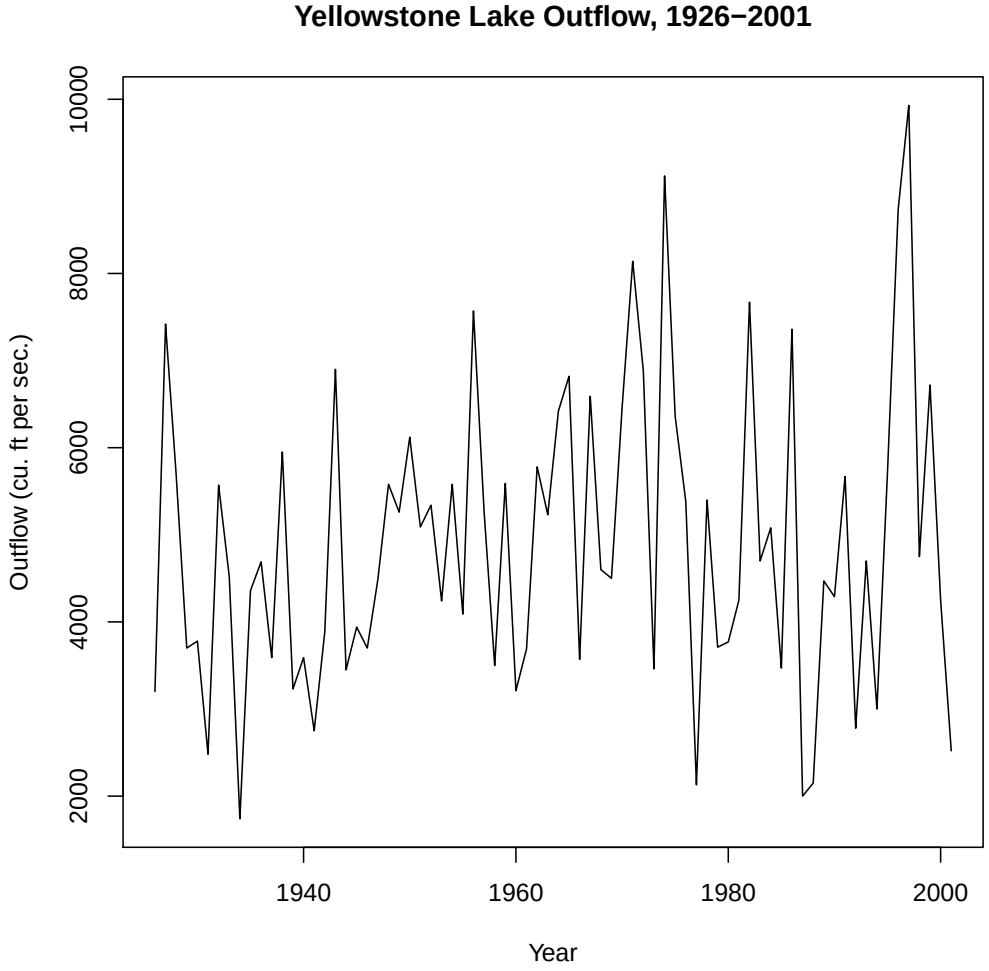
Use `cor()` to compute the ***correlation*** between Yellowstone Lake's **outflow** and its **elevation**.

2 Part B: Linear Regression

2.1 Yellowstone Lake Data Set (Cont'd)

1. Use `plot()`, with the optional argument `type = "l"` (for "line"), to create a so-called ***time series plot*** (line plot) of the Yellowstone Lake's **outflow** (Y) versus **year** (X).

   It should look something like this:



2. We want to know if there's an *increasing trend* in the lake's **outflow** and, if so, by how much it increases each **year**, on average.

   The function `lm()` (for "linear model") will carry out a ***linear regression analysis***. It takes arguments:

   `formula` a formula giving a symbolic description of the model to be

	fitted, stated as y ~ x, where x and y are a data vectors containing the explanatory and response variables, respectively.
`data`	an optional data frame containing the variables in the model.

Fit a ***regression line*** to the Yellowstone Lake data, with **outflow** as the response (Y) and **year** as the explanatory variable (X). Save the result in a so-called *lm* object called, say, `my.reg`, by typing:

```
my.reg <- lm(Outflow ~ Year)
```

3. To obtain the equation of the fitted line, look at a summary of the ***regression analysis*** using the `summary()` function:

```
summary(my.reg)
```

The y-intercept and slope are listed under the `Estimate` column of the output.

4. The function `abline()` adds a line to an existing plot. It takes arguments:

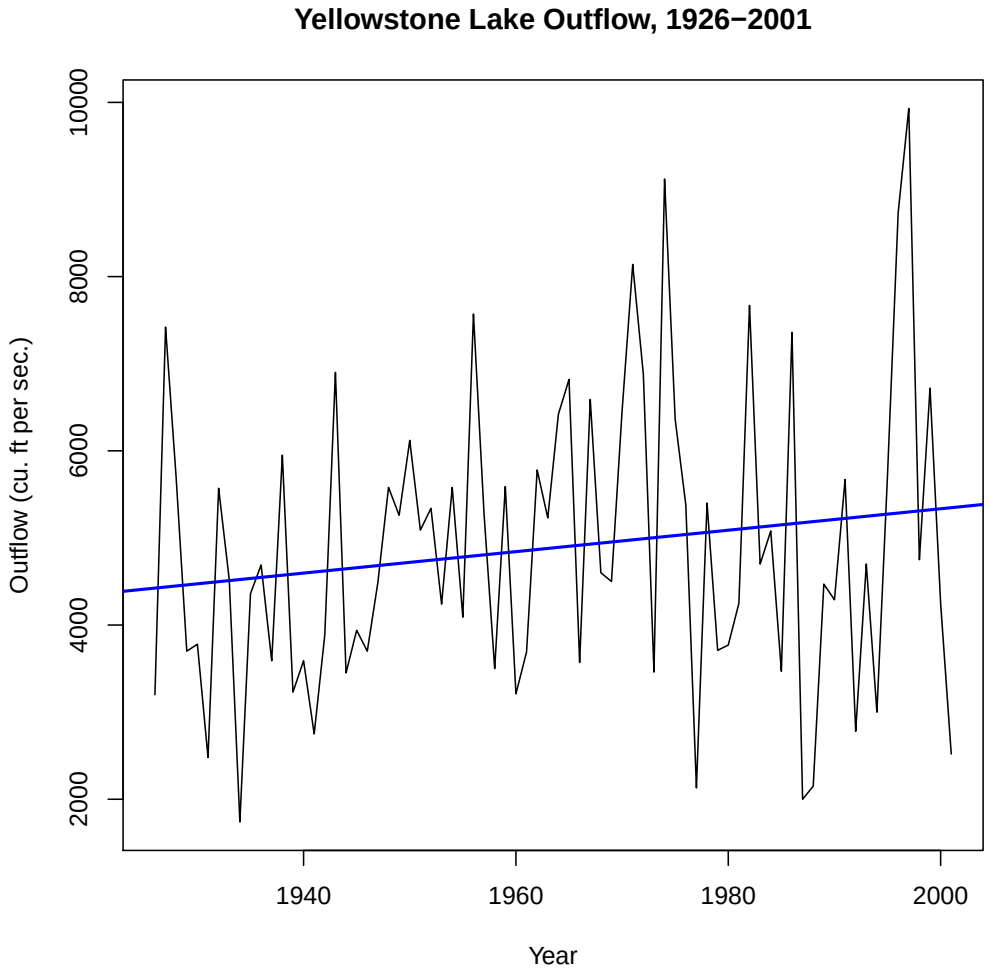
`a`	an optional y-intercept for the line.
`b`	an optional slope for the line.
`reg`	an optional lm object containing the coefficients of the line.
`col`	optional color, e.g. "red", "blue", etc. (in quotes).
`lwd`	optional line width, with default value 1.

For example, `abline(a = 3, b = 0.5)` would add a line whose equation is $y = 3 + 0.5x$.

We'll instead pass the *lm* object `my.reg` to `abline()`, in which case `abline()` will add the ***fitted regression line*** to the plot. Type:

```
abline(my.reg, col = "blue", lwd = 2)
```

Your plot should look something like this:



3 Part C: Regression (Cont'd)

3.1 Francis Galton's Heights Data

Francis Galton collected data on **heights** of **fathers** ($\boldsymbol{X}$) and **heights** of their **sons** ($\boldsymbol{Y}$), both in inches. The data are in the file **galtons_heights.txt**.

1. Save the file **galtons_heights.txt** onto your computer, then type:

```
my.file <- file.choose()
```

   and choose the file in the pop-up window. The file's name and location will be stored in `my.file`.

2. Now read the data from **galtons_heights.txt** into R using:

```
hts.data <- read.table(my.file, header = TRUE)
```

(Specifying `header = TRUE` tells R that the first row of **galtons_heights.txt** has the column headers.)

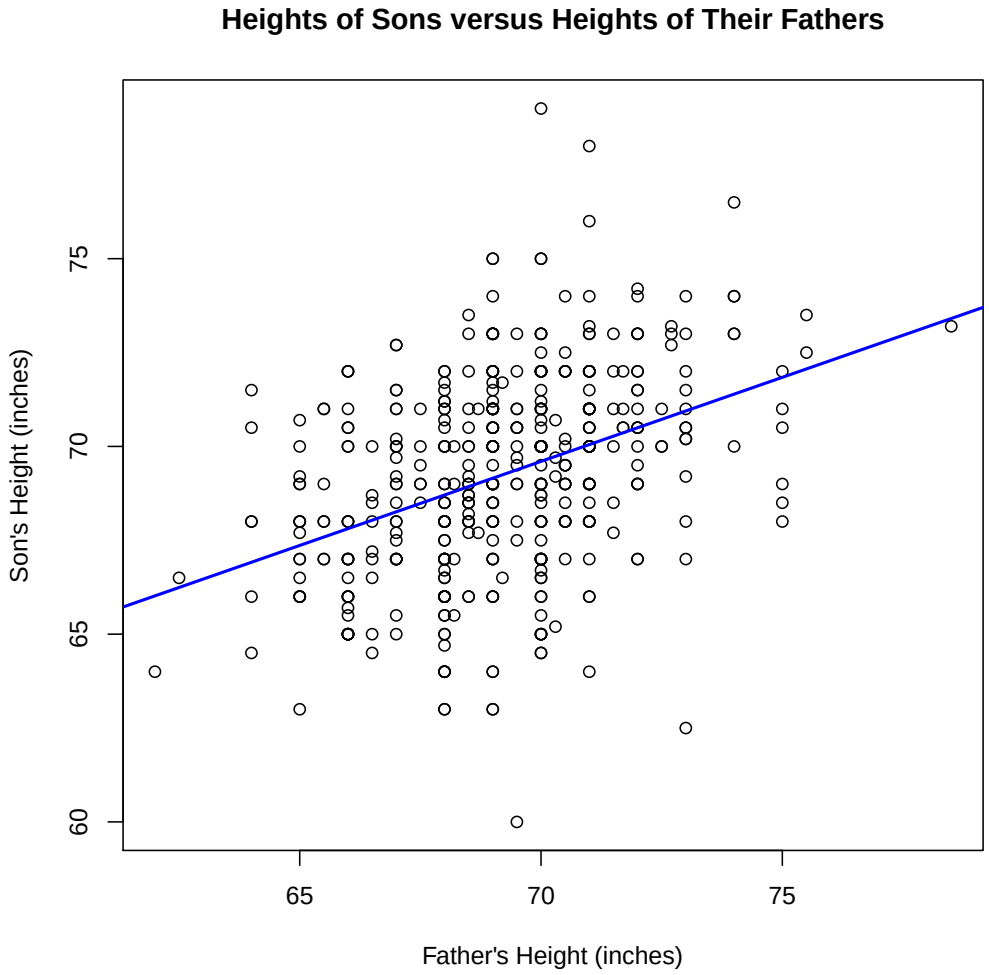
3. Extract the columns from the `hts.data` *data frame* into two *vectors*:

```
FathersHt <- hts.data$FathersHt
SonsHt <- hts.data$SonsHt
```

4. Use `plot()` to make a ***scatterplot*** of the **sons' heights** (Y) versus **fathers' heights** (X).
5. Use `cor()` to compute the ***correlation*** between **fathers'** and **sons' heights**.
6. Use `lm()` to carry out a ***linear regression analysis*** woth **sons' height** as the **response variable** (Y) and **fathers' height** as the **explanatory variable** (X).

   Then look at the results using `summary()`.

   Here's a ***scatterplot*** of the data, with ***fitted regression line***:



We'll use the **equation** of the ***fitted regression line*** to **predict** the **height** of a **son** whose **father** is **77** inches tall, and to **predict** the **height** of a **son** whose **father** is **63** inches tall.

7. We want to summarize how well the ***fitted regression line*** fits the **heights** data. Two ways to do this are the ***square root*** of the ***mean squared residual***

$$s = \sqrt{\frac{1}{n-2}\sum_{i=1}^{n} e_i^2},$$

where the $\boldsymbol{e_i}$'s are the ***residuals***, and the ***R-squared***

$$r^2 = 1 - \frac{\text{SSE}}{\text{SST}},$$

where $\text{SSE} = \sum e_i^2$ and $\text{SST} = \sum (y_i - \bar{y})^2$ are the ***error sum of squares*** and ***total sum of squares***, respectively.

In the output from `summary()` in Step 6, R calls s the `Residual standard error` and it calls r^2 the `Multiple R-squared`. Get the values of these two statistics.