Lecture 6 R notes

Simple Linear Regression

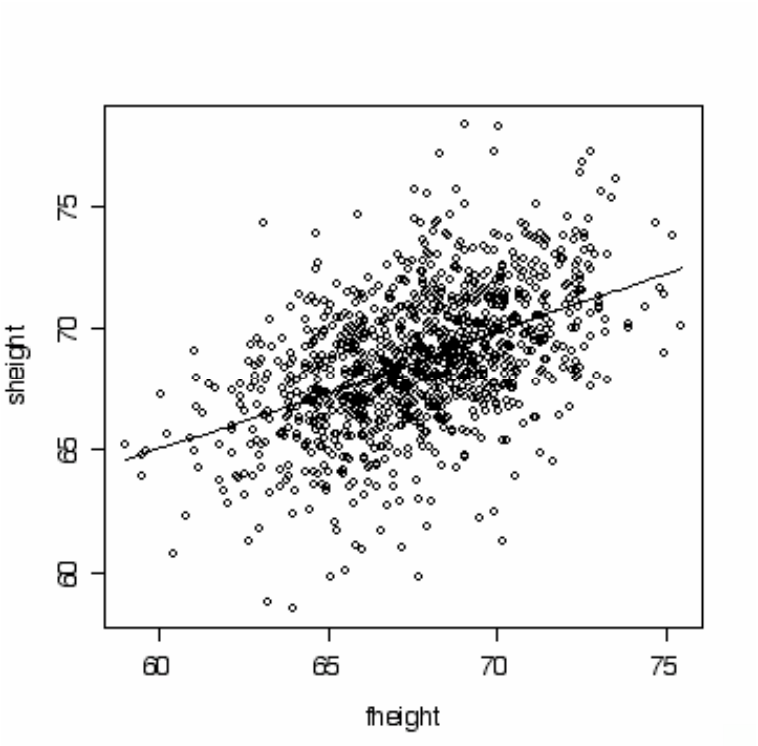
How to do it in R, using Galton's Father/Son data.

```
> source("galton.R")
> ls()
[1] "father.son"
> attach(father.son)
> names(father.son)
[1] "fheight" "sheight"
```

NOTE: why did we use "source" and not "read.table"? To see why, open this file in a text-editor. You'll see that it is not just a list of data. Instead, it is a file containing R commands (one of which is the read.table) command. The command "source(filename)" executes all of the commands in filename as if they were R commands.

```
> plot(fheight,sheight)
> lines(lowess(fheight,sheight))
```

Always start with a plot. Here the relation looks very linear.



```
> output <- lm(sheight ~ fheight)
```

The lm command looks like lm(formula, data=dataframe) but specifying the dataframe is optional if you've done the "attach" command so that R knows which variables you've identified in the formula.

Formulas look like this:
response variable ~ function of predictors

At the moment we're considering only simple linear regression, and so we have only one variable on the right hand side. R interprets the RHS to be a + b*fheight. R always assumes there is an intercept. If, for some reason, you do not want an intercept (you want to force it to be 0) you can type
y ~ x -1
and the "-1" tells R to remove the constant term. Strange, I know. But effective.

The lm command produces a list of various items which I've saved in the "output" variable. You can summarize this output either by simply typing "output" (or whatever you named your variable) or, even better, through the summary command:

```
> summary(output)

Call:
lm(formula = sheight ~ fheight)

Residuals:
      Min        1Q    Median        3Q       Max
-8.877151 -1.514415 -0.007896  1.628512  8.968479

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 33.88660    1.83235   18.49   <2e-16 ***
fheight      0.51409    0.02705   19.01   <2e-16 ***
---
```

```
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 `
' 1

Residual standard error: 2.437 on 1076 degrees of freedom
Multiple R-Squared: 0.2513,   Adjusted R-squared: 0.2506
F-statistic: 361.2 on 1 and 1076 DF,  p-value: < 2.2e-16
```

For now, the important part of this outupt is the values for the coefficients (33.8866 for interecept and 0.51409 for slope). You could also see these using the coefficients command:

```
> coefficients(output)
(Intercept)     fheight
  33.886604    0.514093
```

We'll talk about the rest of the output later.

It is also nice to note that the residuals do seem to be centered around 0 (according to the 5 number summary near the top). Note that "Multiple r-squared is .2513" which means we've explained only about 25% of the variation.

Suppose we want to use this line to make predictions. Given two fathers: 70" and 72", how tall do we predict their sons will be? Or, put differently, what are the average heights of sons whose fathers are 70 and 72?

R makes this sort of thing easy:

```
> newvalues <- data.frame(fheight=c(70,72))
> predictions <- predict(output, newvalues)
> length(predictions)
[1] 2
> predictions
       1        2
69.87312 70.90130
```

Or you could write a function:

```
predictions.fun <- function(x, lm){
a <- coefficients(lm)[1]
b <- coefficients(lm)[2]
a + b*x}
```

And then run it

```
> predictions.fun(c(72,70),output)
[1] 70.90130 69.87312
```

This function makes use of the "coefficients" command.

You can plot the regression line over the data:

```
> plot(fheight,sheight)
> abline(output)
```

And we can plot the residuals against the predictor and check their normality.

```
> resid <- residuals(output)
> par(mfrow=c(3,1))
> plot(fheight,resid)
> hist(resid)
> qqnorm(resid)
```

The par command puts all three graphs in one column.