Regression in R

Intro to Stats, Spring 2017

Prof. Gaston Sanchez

Learning Objectives

- How to run a regression analysis in R
- Getting to know the function `lm()`
- How add the regression line to a scatter diagram
- How to graph the plot of residuals

About

In this tutorial I will show you how to:

- perform a Regression analysis in R.
- interpret the output and main results.
- use R to obtain the results the way they are calcuated in the book.

Introduction

To make sure we are all on the same page, I should start with a brief discussion about the term "Regression". This term was coined by Sir Francis Galton and introduced in his 1886 paper *Regression Towards Mediocrity in Hereditary Stature*. He used the word "regression" to describe a phenomenon that he observed when analyzing height data of parents and their adult children. What he noticed was that exceptionally tall parents had children who were, on average, less tall; and exceptionally short parents had children who were, on average, less short. Because of the connotations of the word "mediocrity", statisticians later modified the regression statement as: *regression towards the mean*.

For better or for worse, the term regression has evolved and become broader over the years. Nowadays people use the word regression in a more loosely way. The most common terms that you will probably find are "regression analysis" and "regression model" or "regression modeling".

The core idea behind virtually all regression tools is to predict a (quantitative) response variable in terms of one or more predictor variables. In its simplest version, the regression method is used for explaining or modeling the relationship between a single variable Y, called the *response*, *outcome*, *output* or *dependent* variable; and one *predictor*, *input*, *independent* or *explanatory* variable X. This version is commonly referred to as **simple linear regression**, and it is actually the method presented in the FPP book (chapters 10-12)—although it is not explicitly called like that. The reasons why statisticians call it *simple linear regression* are:

- "simple" because there is only one Y and one X
- "linear" because the mathematical model is expressed with a line equation
- "regression" because X is used to predict Y

Note: Pretty much the word "regression" has become synonym for prediction. The word "prediction" implying that the response variable is quantitative. If the response variable to be predicted is of categorical nature, then we talk about "classification". The type of regression we will discuss is the one in which both X and Y are quantitative variables.

Data

To illustrate the regression ideas, we are going to use a data set collected by English mathematician and biostatistician Karl Pearson (1857-1936). Among other things, Pearson was Galton's protégé, he founded the world's first university statistics department at University College London in 1911, and he is considered one of the founding fathers of modern-day Statistics.

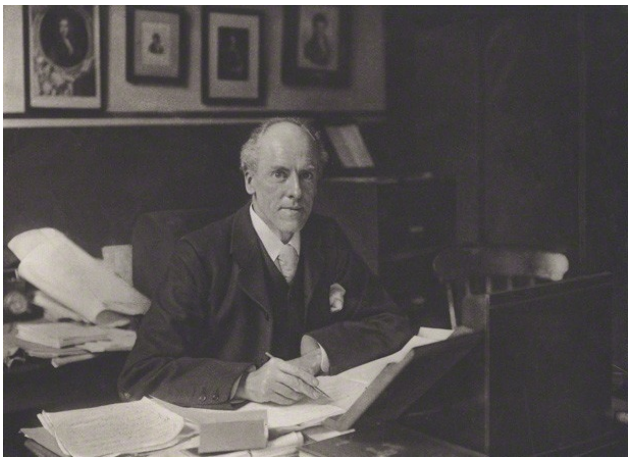


Figure 1: Karl Pearson (source: wikipedia)

The data set is in the csv file `pearson.csv` (in the github repository), and it contains the heights of 1078 fathers, and their adult sons.

```
# assembling the URL of the CSV file
# (otherwise it won't fit within the margins of this document)
repo = 'https://raw.githubusercontent.com/ucb-introstat/introstat-spring-2017/'
datafile = 'master/data/pearson.csv'
url = paste0(repo, datafile)

# read in data set
dat = read.csv(url)
```

If you are having problems importing the data from github, you can try this other source:

```
# another way to read in the data
dat = read.csv('http://www.math.uah.edu/stat/data/Pearson.csv')
```

Univariate Exploration

The data frame `dat` contains 1078 rows, and 2 columns.

- `Father`: height of the father (in inches)
- `Son`: height of the son (in inches)

As it is customary, the first thing when analyzing data (one variable at a time), is to obtain summary statistics and look at the distributions:

```
# basic summary statistics
summary(dat)
```

```
##      Father           Son
##  Min.   :59.00   Min.   :58.50
##  1st Qu.:65.80   1st Qu.:66.90
##  Median :67.80   Median :68.60
##  Mean   :67.69   Mean   :68.68
##  3rd Qu.:69.60   3rd Qu.:70.50
##  Max.   :75.40   Max.   :78.40
```

```
# number of rows
n = nrow(dat)

# SD of Father
sqrt((n-1)/n) * sd(dat$Father)
```

```
## [1] 2.744553
```

```
# SD of Son
sqrt((n-1)/n) * sd(dat$Son)
```

```
## [1] 2.814888
```

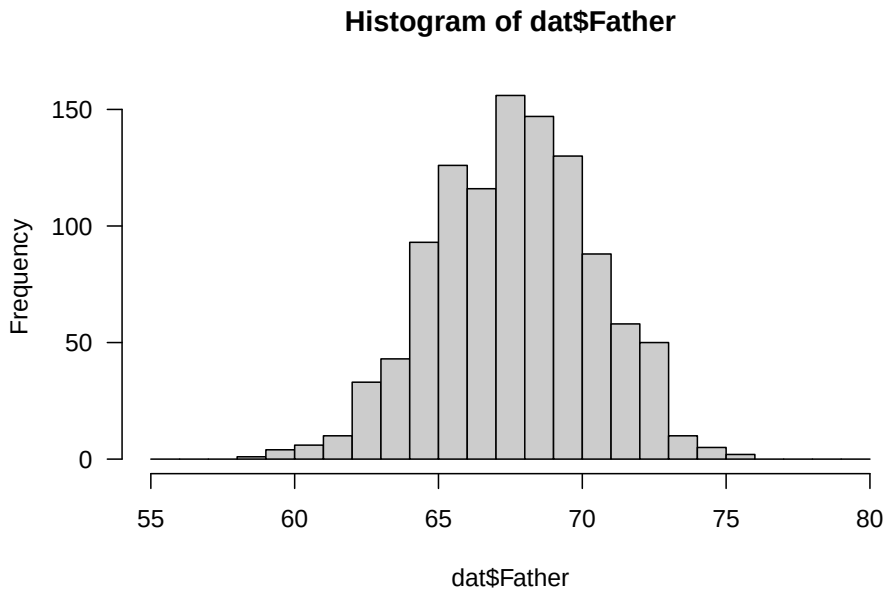
Histograms

For convenience purposes, to plot histograms for the `Father` and `Son` heights, we can define a vector of class intervals or bins for the argument `breaks` inside the `hist()` function. Looking at the output of `summary()`, we can take a minimum of 55, and a maximum of 80 to create a vector of `bins` with a numeric sequence like this:

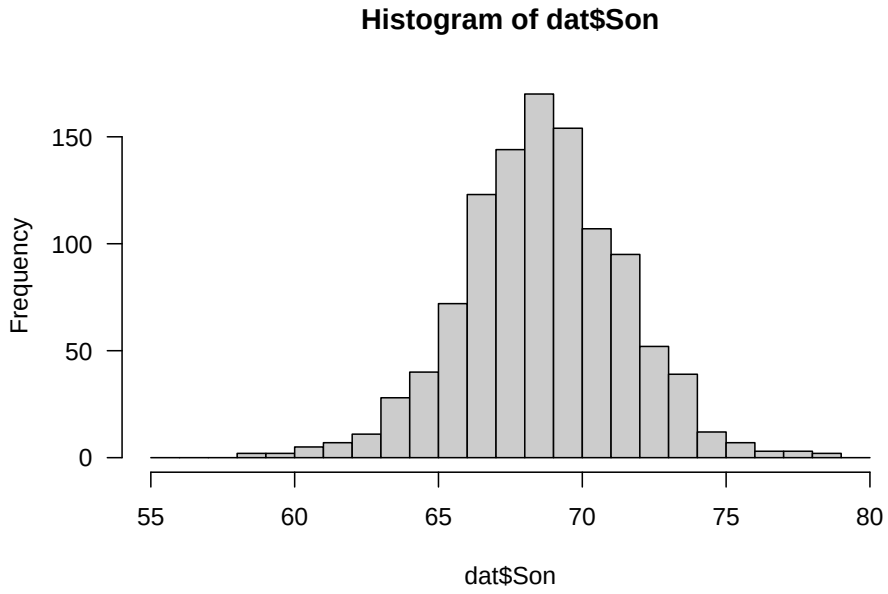
```
bins = seq(from = 55, to = 80, by = 1)
```

Having defined `bins`, we can then graph the histograms:

```
hist(dat$Father, breaks = bins, las = 1, col = 'gray80')
```




```
hist(dat$Son, breaks = bins, las = 1, col = 'gray80')
```

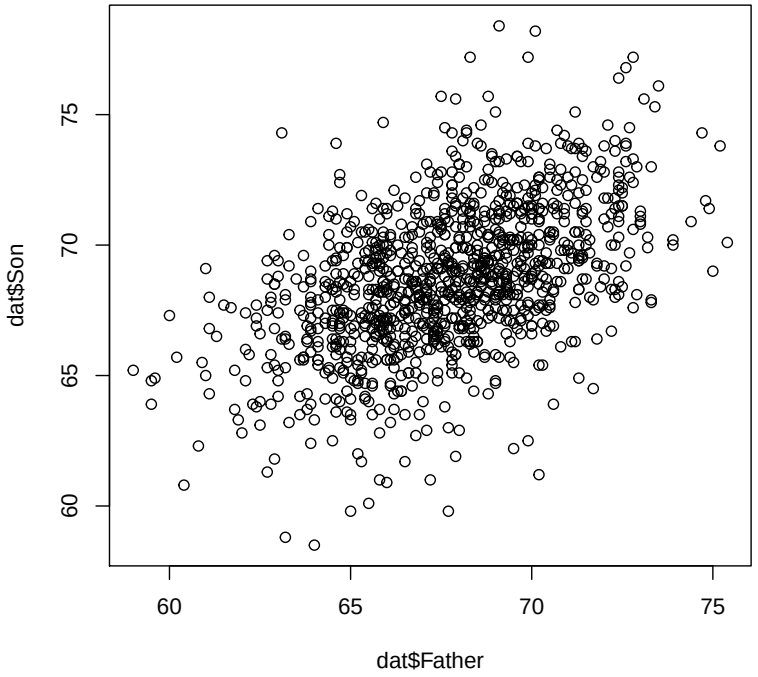


Both histograms have nice symmetric bell-shaped distributions. Keep in mind that these are classic textbook examples of variables that follow the normal curve.

Scatter Plot

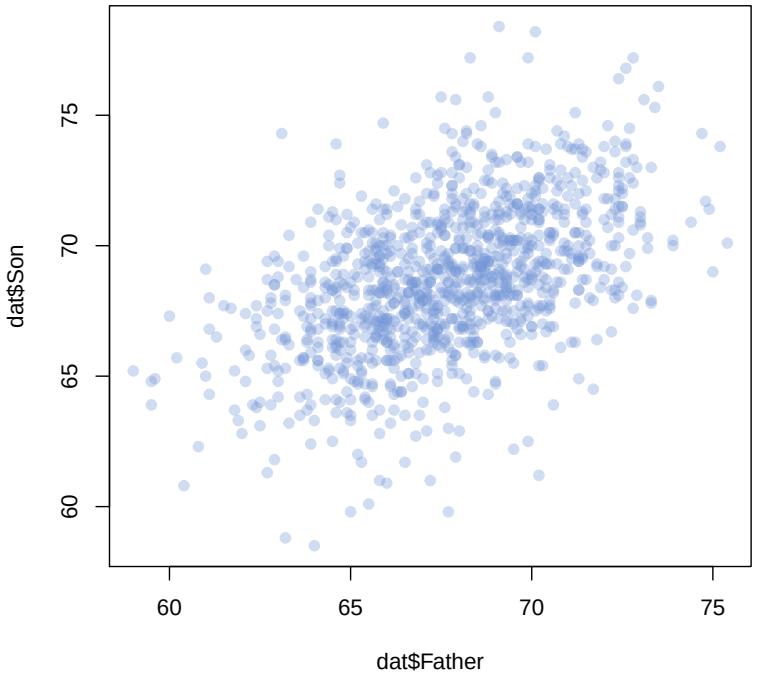
The next step is to get a picture of the relationship between the `Father` and `Son` heights. The quickest option is to create a scatter diagram with the function `plot()`:

```
plot(dat$Father, dat$Son)
```



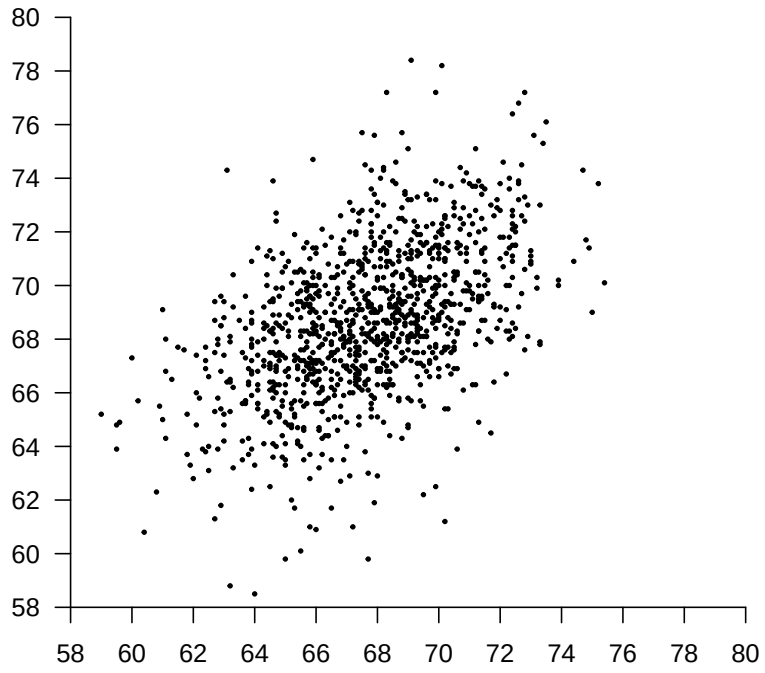
To obtain a better visual display, you can modify the point character `pch` value, and add a color with some transparency using hexadecimal notation:

```
plot(dat$Father, dat$Son, pch = 19, col = '#7396D655')
```



If you want to get a scatter diagram like the one displayed in the textbook (see page 120), then run the following lines of code:

```
plot.new()
plot.window(xlim = c(58, 80), ylim = c(58, 80))
points(dat$Father, dat$Son, pch = 20, cex = 0.5)
axis(side = 1, pos = 58, at = seq(58, 80, 2))
axis(side = 2, las = 1, pos = 58, at = seq(58, 80, 2))
```



The way in which this plot is constructed is a bit special. This approach to create graphs in R is by using only low-level functions that give you total control of the visual appearance and aspect of graphical elements:

- `plot.new()` starts a new plot frame
- `plot.window()` is used to set up the coordinates of the axes
- `points()` is used to actually plot the dots
- `axis()` is used to plot both the x-axis and the y-axis.

The most important thing to pay attention to when inspecting the scatter diagram is to check whether the cloud of points follows a linear pattern. When this is the case, it makes sense to use of a line to summarize the relationship between the analyzed variables.

Regression Method

The function `lm()` (i.e. linear model) allows to perform a regression analysis in R. To get the results using the Pearson data set:

```r
# regression analysis (and it's default output)
reg = lm(Son ~ Father, data = dat)
reg
```

```
##
## Call:
## lm(formula = Son ~ Father, data = dat)
##
## Coefficients:
## (Intercept)       Father
##      33.893        0.514
```

The first part simply tells you the command used to run the analysis, in this case `lm(formula = Son ~ Father, data = dat)`.

The second part shows information about the regression coefficients. In this case the intercept is 33.8928 and the slope is 0.514.

The object `reg` contains various components. To see a list of the different results in `reg`, check the `names()`

```r
names(reg)
```

```
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"
```

As you can tell, `reg` contains many more things than just the `coefficients`. Here's a short description of each of the output elements:

- `coefficients`: a named vector of coefficients.
- `residuals`: the residuals, that is response minus fitted values.
- `fitted.values`: the fitted mean values.
- `rank`: the numeric rank of the fitted linear model.
- `df.residual`: the residual degrees of freedom.
- `call`: the matched call.
- `terms`: the terms object used.
- `model`: if requested (the default), the model frame used.

For this course, the important output elements are `coefficients`, `residuals`, and `fitted.values`.

To inspect what's in each component, type the name of the regression object, `reg`, followed by the `$` dollar operator, followed by the name of the component. For example, to inspect the `coefficients` run this:

```r
# regression coefficients
reg$coefficients
```

```
## (Intercept)      Father
##  33.8928005   0.5140059
```

```r
# observed values
head(dat$Son, n = 5)
```

```
## [1] 59.8 63.2 63.3 62.8 64.3
```

```r
# predicted values
head(reg$fitted.values, n = 5)
```

```
##        1        2        3        4        5
## 67.30318 66.42937 67.30318 67.71439 65.29856
```

Summary output

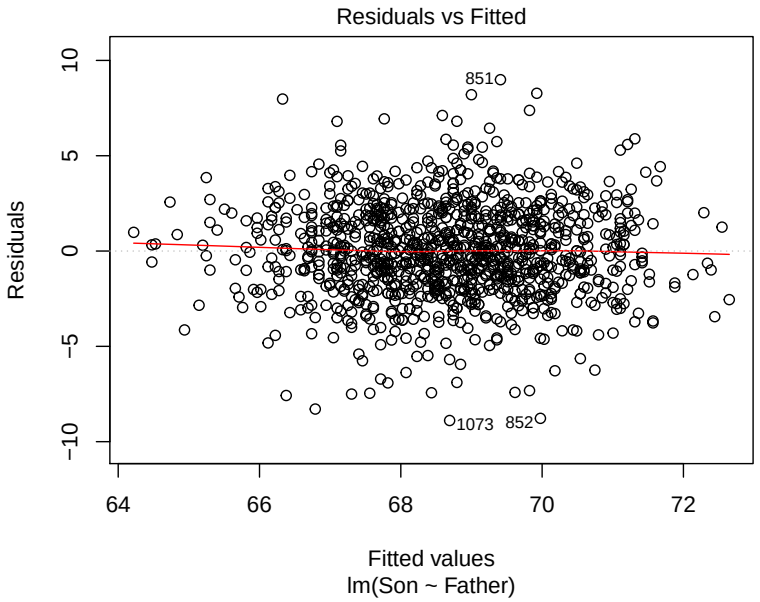
The object `reg` is a special type of object. More precisely, `reg` is an object of class `"lm"`—linear model. For this type of R objects, you can use the `summary()` function to get additional information and diagnostics:

```r
sum_reg = summary(reg)
sum_reg
```

```
##
## Call:
## lm(formula = Son ~ Father, data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.8910 -1.5361 -0.0092  1.6359  8.9894
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 33.89280    1.83289   18.49   <2e-16 ***
## Father       0.51401    0.02706   19.00   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.438 on 1076 degrees of freedom
## Multiple R-squared:  0.2512, Adjusted R-squared:  0.2505
## F-statistic: 360.9 on 1 and 1076 DF,  p-value: < 2.2e-16
```

Residual Plot

```
# residuals plot
plot(reg, which = 1)
```



equivalent to this:

```
# residuals plot
plot(dat$Father, reg$residuals)
abline(h = 0)
```