Predictions and Errors in Regression

Intro to Stats, Spring 2017

Prof. Gaston Sanchez

Learning Objectives

- Calculating predicted values with the regression method
- Looking at the regression residuals
- Calculating r.m.s. error for regression

Introduction

In the previous script, you learned about the function `lm()` to obtain a simple lienar regression model. Specifically, we looked at the regression `coefficients`: the intercept and the slope. You also learned how to plot a scatter diagram with the regression line, via the `abline()` function, as well as how to "manually" calculate the intercept and slope with the formulas:

$$slope = r \times \frac{SD_y}{SD_x}$$

In turn, Chapter 12 presents the formula of the intercept as:

$$intercept = avg_y - slope \times avg_x$$

Regression with Height Data Set

To cotinue our discussion, we'll keep using the data set in the file csv file `pearson.csv` (in the github repository):

```
# 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)
```

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

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

Here's a reminder on how to use the function `lm()` to regress `Son` on `Father`:

```
# run regression analysis
reg = lm(Son ~ Father, data = dat)
reg
```

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

You can compare the coefficients given by `lm()` with your own calculated b_1 and b_0 according to the previous formulas. First let's get the main ingredients:

```
# number of values (to be used for correcting SD+)
n = nrow(dat)

# averages
avg_x = mean(dat$Father)
avg_y = mean(dat$Son)

# SD (corrected SD+)
sd_x = sqrt((n-1)/n) * sd(dat$Father)
sd_y = sqrt((n-1)/n) * sd(dat$Son)

# correlation coefficient
r = cor(dat$Father, dat$Son)
```

Now let's compute the slope and intercept, and compare them with `reg$coefficients`

```
# slope
b1 = r * (sd_y / sd_x)
b1
```

```
## [1] 0.5140059
```

```
# intercept
b0 = avg_y - (b1 * avg_x)
b0
```

```
## [1] 33.8928
```

```r
# compared with coeffs
reg$coefficients
```

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

Predicting Values

As I mentioned in the last tutorial, regression tools are mainly used for prediction purposes. This means that we can use the estimated regression line $\texttt{Son} \approx b_0 + b_1\texttt{Father}$, to predict the height of Son given a particular Father's height.

For example, if a father has a height of 71 inches, what is the predicted son's height?

Option a) One way to answer this question is with the regression method described in chapter 10 of FPP. The first step consists of converting x in standard units, then multiplying times r to get the predicted $\hat{y}$ in standard units, and finally rescaling the predicted value to the original units.

```r
# height of father in standard units
height = 71
height_su = (height - avg_x) / sd_x
height_su
```

```
## [1] 1.207181
```

```r
# predicted Son's height in standard units
prediction_su = r * height_su
prediction_su
```

```
## [1] 0.6049941
```

```r
# rescaled to original units
prediction = prediction_su * sd_y + avg_y
prediction
```

```
## [1] 70.38722
```

Option b) Another way to find the predicted son's height when the height of the father is 71 is by using the equation of the regression line:

```r
# predict height of son with a 71 in. tall father
b0 + b1 * 71
```

```
## [1] 70.38722
```

Option c) A third option is with the `predict()` function. The first argument must be an `"lm"` object; the second argument must be a data frame containing the values for `Fater`:

```
# new data (must be a data frame)
newdata = data.frame(Father = 71)

# predict son's height
predict(reg, newdata)
```

```
##        1
## 70.38722
```

If you want to know the predicted values based on several `Father`'s heights, then do something like this:

```
more_data = data.frame(Father = c(65, 66.7, 67, 68.5, 70.5, 71.3))

predict(reg, more_data)
```

```
##        1        2        3        4        5        6
## 67.30318 68.17699 68.33120 69.10221 70.13022 70.54142
```

R.M.S. Error for Regression

The predictions given by the regression line will tend to be off. There is usually some difference between the observed values y and the predicted values $\hat{y}$. This difference is called **residual**. The residuals are part of the `"lm"` object `reg`. You can take a peek at such residuals with `head()`

```
# first six residuals
head(reg$residuals)
```

```
##          1          2          3          4          5          6
## -7.5031849 -3.2293748 -4.0031849 -4.9143896 -0.9985618 -2.0751730
```

By how much the predicted values will be off? To find the answer, you need to calculate the *Root Mean Square* (RMS) error for regression. In other words, you need to take the residuals (i.e. difference between actual values and predicted values), and get the square root of the average of their squares.

```
# r.m.s. error for regression
rms = sqrt(mean(reg$residuals^2))
rms
```

```
## [1] 2.435872
```

The r.m.s. value tells you the typical size of the residuals. This means that the typical predicted heights of sons will be off by about 2.44 inches.

Are residuals homoscedastic?

As you know, the main assumption in a simple regression analysis is that X and Y are approximately linearly related. This means that we can use a line as a good summary for the cloud of points. For a line to able to do a good summarizing job, the amount of spread around the regression line should be fairly the same (i.e. constant). This requirement has a very specific—and rather ugly—name: **homoscedasticity**; which simply means "same scatter". Visually, homoscedascity comes in the form of the so-called football-shaped cloud of points. Or in a more geometric sense, cloud of points with a chiefly elliptical shape.

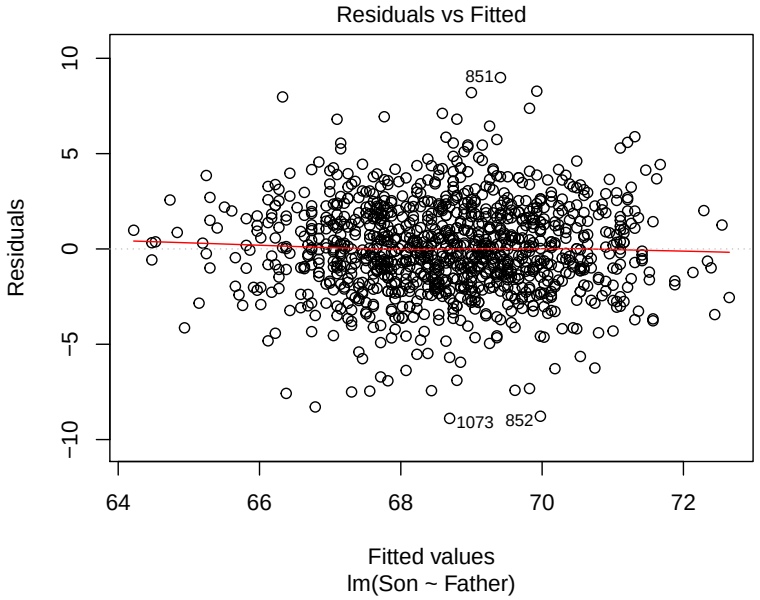
The `"lm"` object `reg` contains the vector of redisuals (see `reg$residuals`). The residuals from the regression line must average out to 0. To confirm this, let's get their average:

```
mean(reg$residuals)
```

```
## [1] 2.050079e-16
```

You can take a look at the *residual plot* by running this command:

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

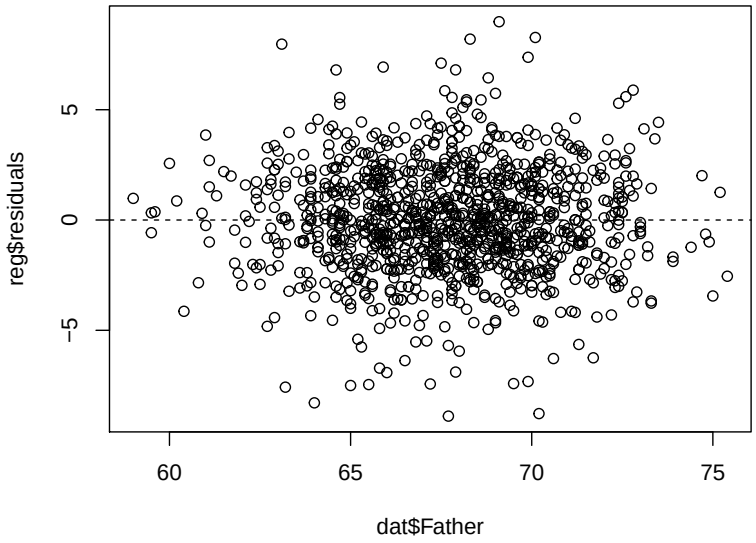


which is equivalent to this other command:

```
# equivalently
plot(reg$fitted.values, reg$residuals)
```

This residual plot is not exactly the same that the book describes (pages 187-188). To plot the residuals like the book does, you would need to use the `Father` variable in the x-axis:

```r
# residuals plot (as in FPP)
plot(dat$Father, reg$residuals)
abline(h = 0, lty = 2)  # horizontal dashed line
```



The difference is only in the scale of the horizontal axis. But the important part in both plots is the shape of the cloud. As you look across the residual plot, there is no systematic tendency for the points to drift up or down. The red line displayed by `plot(reg, which = 1)`, is a regression line for the residuals. When residuals are homoscedastic, this line is basically a horizontal line. This is what you want to see when inspecting the residual plot. Why? Because it supports the appropriate use of the regression line.

Summary output

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

```r
# summarized linear model
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
```

The information displayed by `summary()` is the typical output that most statistical programs provide about a simple linear regression model. There are four major parts:

- `Call`: the command used when invoking `lm()`.
- `Residuals`: summary indicators of the residuals.
- `Coefficients`: table of regression coefficients.
- Additional statistics: more diagnostics toosl.

In the same way that `lm()` produces `"lm"` objects, `summary()` of `"lm"` objects produce `"summary.lm"` objects. This type of objects also contain more information than what is displayed by default. To see the list of all the components in `sum_reg`, you can use again the function `names()`:

```
names(sum_reg)
```

```
##  [1] "call"          "terms"         "residuals"     "coefficients"
##  [5] "aliased"       "sigma"         "df"            "r.squared"
##  [9] "adj.r.squared" "fstatistic"    "cov.unscaled"
```