Lab03 - Linear Regression basics

Notebook:	Didattica
Created:	November 3, 2014 9:31:22 AM
Updated:	November 3, 2014 7:26:39 PM
Author:	aittalam

1) Linear regression: simple exercise with only 8 points.

The exercise has ben solved step by step, using R only to help with calculations. First of all, let us estimate the parameters beta0 (b0) and beta1 (b1), i.e. the intercept and slope of the linear model.

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$$

```
# set up the predictor variables xi and the responses yi
# note that x and y have been generated as follows:
# x = rnorm(8)
# y = 2 * x + rnorm(8,5,.5)
# then they have been rounded to ease the calculations
x = c(0.75,-0.64,1.43,-0.61,0.23,0.43,-1.48,2.06)
y = c(6.60,4.31,7.51,3.48,5.21,5.74,1.65,9.76)
n = length(x)
```

First, calculate b1 (slope) and b0

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2},$$
$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x},$$

```
mean(x)
# [1] 0.27125
mean(y)
# [1] 5.5325

x - 0.27
# [1]  0.48 -0.91  1.16 -0.88 -0.04  0.16 -1.75  1.79
y - 5.53
# [1]  1.07 -1.22  1.98 -2.05 -0.32  0.21 -3.88  4.23

(x-0.27) * (y - 5.53)
# [1] 0.51 1.11 2.30 1.80 0.01 0.03 6.79 7.57
## sum((x-0.27) * (y - 5.53)) = 20.13

(x-0.27)^2
# [1] 0.23 0.83 1.35 0.77 0.00 0.03 3.06 3.20
## sum((x-0.27)^2) = 9.47
```

```
## b1 = slope coefficient = sum((x-0.27) * (y - 5.53))/sum((x-0.27)^2)
b1 = 20.13/9.47
# b1 = 2.12

## b0 = intercept
# b0 = mean(y) - b1 * mean(x)
b0 = 5.53 - 2.12 * 0.27
# b0 = 4.96

# given the parameters we calculated, the estimated yhat = 4.96 + 2.12 * x
# plot the points
plot(x,y)
# draw the estimated function
abline(b0,b1);
# draw the original function
abline(5,2,col="red")
```

---

2) Calculate the residuals

```
yhat = b0 + b1 * x
RSS=sum((y-yhat)^2)
# RSS = 1.059709
```

Note that this value of RSS is a minimum: changing values of b0 and b1 RSS will always be bigger

---

3) Calculate the standard error:

$$\mathrm{SE}(\hat{\beta}_0)^2 = \sigma^2 \left[\frac{1}{n} + \frac{\bar{x}^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \right], \quad \mathrm{SE}(\hat{\beta}_1)^2 = \frac{\sigma^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

We already have mean(x) = 0.27 and the SSE = 9.47. However, in practice we don't know sigma (we are usually not given the original distribution), so we need to estimate that. RSE (the Residual Standard Error) is a good estimate for it:

$$\mathrm{RSE} = \sqrt{\mathrm{RSS}/(n-2)}$$

```
RSE = sqrt(RSS/(n-2))
# RSE = 0.42

SEb0 = sqrt(.42^2 * (1/8 + (0.27^2 / 9.47)))
# SEb0 = 0.1529965
SEb1 = sqrt(.42^2 / 9.47)
# SEb1 = 0.1364817
```

---

4) Compute 95% confidence intervals

sample of data. For linear regression, the 95 % confidence interval for β_1 approximately takes the form

$$\hat{\beta}_1 \pm 2 \cdot \mathrm{SE}(\hat{\beta}_1). \tag{3.9}$$

```
c(b1-2*SEb1, b1+2*SEb1)
# [1] 2.088175 2.162684

c(b0-2*SEb0, b0+2*SEb0)
# [1] 4.909162 5.002793
```

NOTE that 2 is just an approximation (see note 3 at page 66). The next step will show how to calculate the proper interval to have 95% confidence.

---

5) Compute the t-statistic

t = (b1-0) / (SEb1) = 15.56769

For simple linear regression we use a t-distribution with n − 2 degrees of freedom: the sample size minus the number of estimated parameters.

Look up the table with the pre-computed probabilities for different degrees of freedom and values of t:

Table entry for p and C is the critical value t^* with probability p lying to its right and probability C lying between $-t^*$ and t^*.

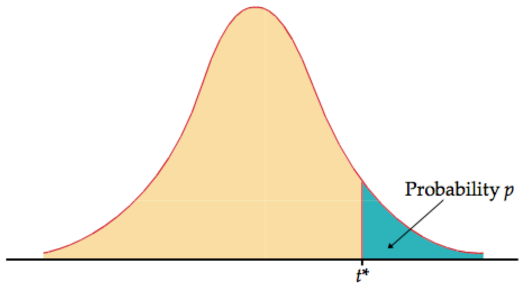

TABLE D

t **distribution critical values**

	Upper-tail probability p											
df	.25	.20	.15	.10	.05	.025	.02	.01	.005	.0025	.001	.0005
1	1.000	1.376	1.963	3.078	6.314	12.71	15.89	31.82	63.66	127.3	318.3	636.6
2	0.816	1.061	1.386	1.886	2.920	4.303	4.849	6.965	9.925	14.09	22.33	31.60
3	0.765	0.978	1.250	1.638	2.353	3.182	3.482	4.541	5.841	7.453	10.21	12.92
4	0.741	0.941	1.190	1.533	2.132	2.776	2.999	3.747	4.604	5.598	7.173	8.610
5	0.727	0.920	1.156	1.476	2.015	2.571	2.757	3.365	4.032	4.773	5.893	6.869
6	0.718	0.906	1.134	1.440	1.943	2.447	2.612	3.143	3.707	4.317	5.208	5.959
7	0.711	0.896	1.119	1.415	1.895	2.365	2.517	2.998	3.499	4.029	4.785	5.408
8	0.706	0.889	1.108	1.397	1.860	2.306	2.449	2.896	3.355	3.833	4.501	5.041
9	0.703	0.883	1.100	1.383	1.833	2.262	2.398	2.821	3.250	3.690	4.297	4.781
10	0.700	0.879	1.093	1.372	1.812	2.228	2.359	2.764	3.169	3.581	4.144	4.587
11	0.697	0.876	1.088	1.363	1.796	2.201	2.328	2.718	3.106	3.497	4.025	4.437
12	0.695	0.873	1.083	1.356	1.782	2.179	2.303	2.681	3.055	3.428	3.930	4.318

(original source: http://bcs.whfreeman.com/ips6e/content/cat_050/ips6e_table-d.pdf)

---

6) Recall RSE and compute R^2

RSE is an estimate of the *lack of fit:*

$$\mathrm{RSE} = \sqrt{\mathrm{RSS}/(n-2)}$$

```
% TSS = total sum of squares (similar to RSS but wrt the mean and not the yi)
TSS = sum((y-mean(y))^2)
% [1] 43.84995

% compute R^2
Rs = (TSS-RSS)/TSS
% [1] 0.9758408

% show relationship between R^2 and correlation in the univariate case
cor(x,y)^2
```

---

7) Show semi-automatic solution

The experiment above can be conducted in a faster way, just by making R do more calculations (instead of moving actual numbers from one formula to another - that was just to give a step-by-step introduction to linear regression). Here is the code:

```
# initialize variables
x = c(0.75,-0.64,1.43,-0.61,0.23,0.43,-1.48,2.06)
y = c(6.60,4.31,7.51,3.48,5.21,5.74,1.65,9.76)
n = length(x)

# find parameters
b1 = sum((x-mean(x)) * (y-mean(y))) / sum((x-mean(x))^2)
b0 = mean(y) - b1 * mean(x)

# calculate RSS and RSE
yhat = b0 + b1 * x
RSS=sum((y-yhat)^2)
RSE = sqrt(RSS/(n-2))

# calculate SEb0 and SEb1
SEb0 = sqrt(RSE^2 * (1/length(x) + mean(x)^2/sum((x-mean(x))^2)))
SEb1 = sqrt(RSE^2 /sum((x-mean(x))^2))

# compute t-statistics
t0 = (b0-0) / (SEb0)
t1 = (b1-0) / (SEb1)

# compute R^2
TSS = sum((y-mean(y))^2)
Rs = (TSS-RSS)/TSS
```

---

8) Redo everything automagically with R

```
help(lm)

lm.fit = lm(y~x)
plot(x,y); abline(lm.fit); abline(5,2,col="red")
```

```
# show that the values we find are consistent with the ones we calculated previously
summary(lm.fit)
coef(lm.fit)
confint(lm.fit)

# show that predictions can also be done
predict(lm.fit,data.frame(x = 4), interval="confidence")
```

---

9) Finally, show how estimates change with (1) number of points and (2) variance

```
# more points
x = rnorm(100)
y = 2 * x + rnorm(100, 5, .5)

lm.fit = lm(y~x)
plot(x,y); abline(lm.fit); abline(5,2,col="red")
summary(lm.fit)

# same points as in simple experiment, much more variance
x = rnorm(8)
y = 2 * x + rnorm(8, 5, 5)
```