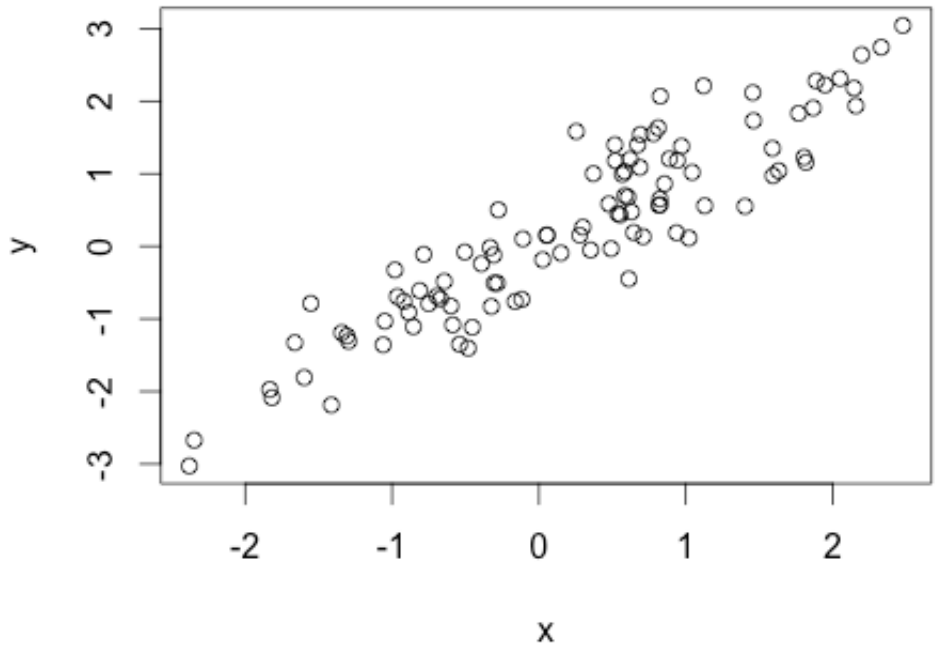
Lab05: Advanced linear regression

0) One more clarification: on p-values related to slope and intercept

```
set.seed(12345)
x = rnorm(100)
y = x + rnorm(100,0,.5)
cor(x,y)

# [1] 0.9184189

plot(x,y)
```




```
fit = lm(y~x)
summary(fit)
```

```
Call:
lm(formula = y ~ x)

Residuals:
     Min       1Q   Median       3Q      Max
-1.10174 -0.30139 -0.00557  0.30949  1.30485

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.01103    0.05176   0.213    0.832
x            1.04727    0.04557  22.982   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.5054 on 98 degrees of freedom
Multiple R-squared:  0.8435,    Adjusted R-squared:  0.8419
F-statistic: 528.2 on 1 and 98 DF,  p-value: < 2.2e-16
```

Q: Why is the p-value for the intercept so big (0.608) and the one for the slope is so small(<2e-16)? What is the meaning of the p-value for the intercept and for the slope?

A: After calculating our linear function's coefficients (intercept and slope in the simple linear regression case), we always come up with some values. This happens regardless of (1) how good our input data is (e.g. more or less noise) and also (2) whether our linear relationship assumption is valid or not.

To measure if and how much using those coefficients actually makes sense, we need to calculate some metrics which depend on the *Standard Error (SE)*. The first one is the *confidence interval* (see sections 3-4 of Lab03 material), that gives us a range around the estimates of our parameters where the true values of the parameters lie with a given probability. The second method we use to assess the validity of our model is the *t-statistic*: by calculating the parameter t (below calculated for beta1, but we can do the same for beta0)

$$t = \frac{\hat{\beta}_1 - 0}{\mathrm{SE}(\hat{\beta}_1)},$$

we aim to find the probability of the *null hypothesis* (beta1=0) to be true, by looking at the t-distribution (see table in the PDF: http://davide.eynard.it/teaching/2016_PAMI/ips6e_table-d.pdf) and verifying what is the probability of observing a value >=|t| for the given amount of degrees of freedom (n-2 in the simple linear regression case).

Now, what does a high or small p-value for the slope mean? If the value is large, it means that the association between predictor and response is mostly due to chance, and there is no real relationship between them (thus it makes sense to assume beta1=0, that is whatever your input is, the outputs are not affected by it). If the value is small, then there is a high probability that the null hypothesis is false, and it makes sense to assume that there is a relationship between the predictors and the responses.

What about the intercept? In this case the interpretation of the null hypothesis is the same: a low p-value means we should take the estimated intercept into account for our model, a high one

means that the intercept value is very likely to be not significant (so you might as well fit a model that does not take it into consideration). But the interpretation on the regression is different: as beta0 does not model a relationship between predictors and responses, we cannot conclude anything about "how good our model is", but we should just restrict our interpretation to "beta0 should be 0".

In the above example, you can see a very small p-value for the slope: that makes sense, as the value we estimated is really close to the real one and we know we have that relationship between x and y. The p-value for the intercept, instead, is much bigger: this just means that it would be better for us to just take beta0=0 (which is actually the "true" value of the intercept), by fitting a new model without intercept (i.e. lm(y ~ 0 + x)). Note that, as the parameters were calculated to minimize RSS, the new model fitted using beta0=0 will likely have a bigger RSS. However, if you also look at R-squared values you will see that the new model will usually fit the data better (especially when the fitted intercept was very far from 0).

```
Call:
lm(formula = y ~ 0 + x)

Residuals:
     Min       1Q   Median       3Q      Max
-1.09199 -0.29209  0.00717  0.32221  1.31534

Coefficients:
  Estimate Std. Error t value Pr(>|t|)
x  1.04936    0.04428    23.7   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.503 on 99 degrees of freedom
Multiple R-squared:  0.8501,    Adjusted R-squared:  0.8486
F-statistic: 561.6 on 1 and 99 DF,  p-value: < 2.2e-16
```

Note that this does not have much to do with the noise we have in the data: even if we take a set with much less noise:

```
set.seed(12345)
x = rnorm(100)
y = x + rnorm(100,0,.001)
cor(x,y)

#[1] 0.9999996

fit = lm(y~x)
summary(fit)

Coefficients:
             Estimate Std. Error   t value Pr(>|t|)
(Intercept) 2.205e-05  1.035e-04     0.213    0.832
x           1.000e+00  9.114e-05 10973.385   <2e-16 ***
```

On the contrary, even if we have a function with some noise, but which actually has an intercept:

```
set.seed(12345)
x = rnorm(100)
y = x + rnorm(100,5,1)
cor(x,y)

# [1] 0.7716394

fit = lm(y~x)
summary(fit)
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  5.02205    0.10353   48.51   <2e-16 ***
x            1.09454    0.09114   12.01   <2e-16 ***
```

When does the slope "break" instead? Let us try to generate data where there is no relationship between x and y at all:

```
set.seed(12345)
x = rnorm(100)
y = rnorm(100)
cor(x,y)

# [1] 0.1042097

fit = lm(y ~ x)
summary(fit)
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.02205    0.10353   0.213    0.832
x            0.09454    0.09114   1.037    0.302
```

1) Feature selection

```
# load the dataset from the previous class and look into it
Credit = read.csv("~/Downloads/Credit.csv")
attach(Credit)
pairs(Credit)

# see what's in credit and take away what we are not interested in (i.e. X)
Credit
C = Credit[,2:7]
fit = lm(Balance ~ ., C)
summary(fit)

# comment on the results: what is interesting and what is not?

# perform feature selection: recall regsubsets function and experiment
```

```
# with the different values of the "method" parameter
library(leaps)
C = Credit[,c(2,3,4,5,6,7)]
fit = regsubsets(Balance~., C, method='exhaustive')
summary(fit)

fit = regsubsets(Balance~., C, method='forward')
summary(fit)

fit = regsubsets(Balance~., C, method='backward')
summary(fit)

fit = regsubsets(Balance~., C, method='seqrep')
summary(fit)

# show few results with anova, to avoid manual calculation of RSS
```

2) Extensions of linear regression

 # discrete inputs - that's already built in the model

$$\texttt{balance}_i \approx \beta_0 + \beta_1 \times \texttt{income}_i + \begin{cases} \beta_2 & \text{if } i\text{th person is a student} \\ 0 & \text{if } i\text{th person is not a student} \end{cases}$$

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \epsilon_i = \begin{cases} \beta_0 + \beta_1 + \epsilon_i & \text{if } i\text{th person is Asian} \\ \beta_0 + \beta_2 + \epsilon_i & \text{if } i\text{th person is Caucasian} \\ \beta_0 + \epsilon_i & \text{if } i\text{th person is African American.} \end{cases}$$

```
Credit
C = Credit[,2:11]
fit = lm(Balance ~ ., C)
summary(fit)

# NOTE HOW THE VARIABLES HAVE BEEN AUTOMATICALLY SPLIT!
# (There will always be one fewer dummy variable than the number of levels. The levels you don't
see --e.g. Gender:Male, Student:No, Ethnicity:African American-- are called the baselines)
```

```
GenderFemale         -10.65325    9.91400  -1.075   0.2832
StudentYes           425.74736   16.72258  25.459  < 2e-16 ***
MarriedYes            -8.53390   10.36287  -0.824   0.4107
EthnicityAsian        16.80418   14.11906   1.190   0.2347
EthnicityCaucasian    10.10703   12.20992   0.828   0.4083
```

 # accounting for non-linear relationships

```
# note the use of I(.) below: as the "^" operand has its own semantics within the lm function call,
# we need to surround its usage with the I(.) function, whose purpose is to inhibit its
interpretation
fit2 = lm(Balance ~ . + I(Rating^2), C)
```

summary(fit2)

# we then compare the two models fit and fit2 using the anova function. Anova performs a hypothesis
# test comparing the two models: the *null hypothesis* is that the two models fit the data equally well,
# while the alternative hypothesis is that the second model is superior (or inferior).
anova(fit,fit2)

```
Analysis of Variance Table

Model 1: Balance ~ Income + Limit + Rating + Cards + Age + Education +
    Gender + Student + Married + Ethnicity
Model 2: Balance ~ Income + Limit + Rating + Cards + Age + Education +
    Gender + Student + Married + Ethnicity + I(Rating^2)
  Res.Df     RSS Df Sum of Sq      F    Pr(>F)
1    388 3786730
2    387 2845913  1    940817 127.94 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

# NOTE the "Sum of sq": this is the RSS(1)-RSS(2). If positive then 2 fits better, and the corresponding
# p-value tells how likely it is for this model to be better. Note that if we run:
anova(fit2,fit)

```
Analysis of Variance Table

Model 1: Balance ~ Income + Limit + Rating + Cards + Age + Education +
    Gender + Student + Married + Ethnicity + I(Rating^2)
Model 2: Balance ~ Income + Limit + Rating + Cards + Age + Education +
    Gender + Student + Married + Ethnicity
  Res.Df     RSS Df Sum of Sq      F    Pr(>F)
1    387 2845913
2    388 3786730 -1   -940817 127.94 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

we wills still have a very small p-value, but that does not mean that the model 2 (the one without
quadrating rating) performs better. The RSS is bigger, thus we should interpret the result as saying
that model 2 is worse then model 1 in this case.

# you can also try with log:
fit2 = lm(Balance ~ . +log(Rating), C)
summary(fit2)
anova(fit,fit2)

# also try with poly
fit2 = lm(Balance ~ . + poly(Rating,5), C)
summary(fit2)

anova(fit,fit2)

discuss - if we had training and test sets what would happen?
check http://davide.eynard.it/2015/01/05/statistical-learning-with-r-part-1-overfitting/

# play with synthetic data:
x = rnorm(100)
y = 5 * x^3 - 2 * x^2 + rnorm(100, 12, 5)
plot(x,y)
fit = lm(y ~ x)
fit = lm(y ~ I(x^2) + x)
fit = lm(y ~ I(x^3) + I(x^2) + x)
fit = lm(y ~ poly(x,3,raw=TRUE))

 # accounting for interactions

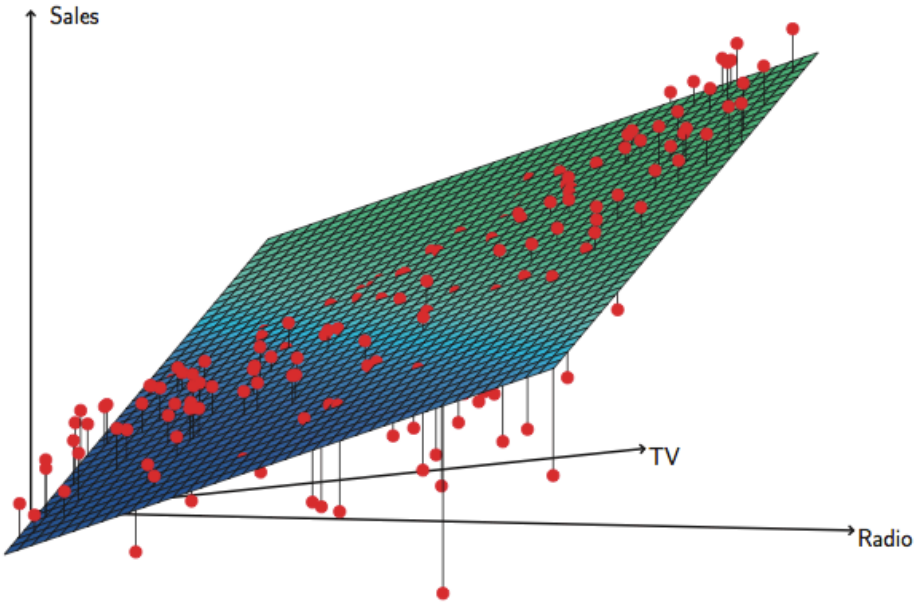


FIGURE 3.5. *For the* `Advertising` *data, a linear regression fit to* `sales` *using* `TV` *and* `radio` *as predictors. From the pattern of the residuals, we can see that there is a pronounced non-linear relationship in the data. The positive residuals (those visible above the surface), tend to lie along the 45-degree line, where TV and Radio budgets are split evenly. The negative residuals (most not visible), tend to lie away from this line, where budgets are more lopsided.*

Ads = read.csv("~/Downloads/Advertising.csv")
Ads = Ads[,2:5]
attach(Ads)
fit = lm(Sales ~ ., Ads)
summary(fit)

# actually we can just include TV and Radio, as Newspapers contribution is negligible
fit = lm(Sales ~ TV + Radio, Ads)
summary(fit)

```
fit2 = lm(Sales ~ TV + Radio + TV*Radio)
summary(fit2)
anova(fit, fit2)
```