Introduction to R

Mark Gimbutas

24.11.2019

Table of Contents

[Frequency tables 1](#_Toc25484445)

[Chi-squared test 3](#_Toc25484446)

[Fisher test 3](#_Toc25484447)

[Correlation analysis 4](#_Toc25484448)

[Linear models 7](#_Toc25484449)

[Multiple linear regression 9](#_Toc25484450)

[Squared term in model 10](#_Toc25484451)

[Residuals 11](#_Toc25484452)

[ANOVA 15](#_Toc25484453)

[Type I, Type III SS 16](#_Toc25484454)

[Estimated differences 17](#_Toc25484455)

[Post-hoc tests: Tukey’s HSD 18](#_Toc25484456)

[Nested ANOVA 18](#_Toc25484457)

[Linear mixed models 19](#_Toc25484458)

[Fixed and random effects 21](#_Toc25484459)

[Repeated measures 23](#_Toc25484460)

[Logistic regression 25](#_Toc25484461)

[Poisson regression 28](#_Toc25484462)

[Principal component analysis 30](#_Toc25484463)

# Frequency tables

**Chi-squared test**

# Create the frequency table  
freqtab <- matrix(c(59, 155, 103, 28), ncol = 2)   
# Chi-squared test  
chisq.test(freqtab)

Frequency table should generally be represented as a matrix. Let’s create a frequency table by typing in the counts (the same as in lecture).

freqtab <- matrix(c(59, 155, 103, 28), # elements of the matrix  
 ncol = 2) # number of columns  
freqtab

## [,1] [,2]  
## [1,] 59 103  
## [2,] 155 28

Notice that (by default) the elements are written column-by-column (not row-by-row). For clarity, the rows and columns can be named.

colnames(freqtab) <- c("females", "males")  
rownames(freqtab) <- c("black", "white")  
freqtab

## females males  
## black 59 103  
## white 155 28

Marginal distributions (row and column sums) can be calculated like this:

colSums(freqtab)

## females males   
## 214 131

rowSums(freqtab)

## black white   
## 162 183

All sorts of percentages can be calculated from the frequency table.

prop.table(freqtab) # proportion out of total sum

## females males  
## black 0.1710145 0.29855072  
## white 0.4492754 0.08115942

# e.g. 17% are black females  
prop.table(freqtab, 1) # row percentages

## females males  
## black 0.3641975 0.6358025  
## white 0.8469945 0.1530055

# e.g. 36% of blacks are female  
prop.table(freqtab, 2) # column percentages

## females males  
## black 0.2757009 0.7862595  
## white 0.7242991 0.2137405

# e.g. 27% of females are black

## Chi-squared test

If the function chisq.test is provided a matrix, it performs the chi-squared test of independence.

chisq.test(freqtab, correct = F)

##   
## Pearson's Chi-squared test  
##   
## data: freqtab  
## X-squared = 85.041, df = 1, p-value < 2.2e-16

The option correct = F suppresses the continuity correction and enables to get exactly the same results as in the lecture notes. However, generally there is no need to suppress it.

We can get the expected cell counts like this:

test.result <- chisq.test(freqtab, correct = F)  
test.result$expected

## females males  
## black 100.487 61.51304  
## white 113.513 69.48696

Notice that the test result can be saved to a separate object, so the result is not lost at the moment it’s printed.

## Fisher test

Works like the chisq.test.

fisher.test(freqtab)

##   
## Fisher's Exact Test for Count Data  
##   
## data: freqtab  
## p-value < 2.2e-16  
## alternative hypothesis: true odds ratio is not equal to 1  
## 95 percent confidence interval:  
## 0.05959758 0.17795894  
## sample estimates:  
## odds ratio   
## 0.1043056

Notice that we’ll get the odds ratio with its 95% confidence interval for free.

# Correlation analysis

cor.test(weird.people$height, weird.people$weight, method = "pearson")

Let’s start by defining (fabricating) a dataset.

weird.people <- data.frame(  
 weight = c(76.0,77.5,77.0,83.0,69.7,76.0,79.5,73.0,71.4,69.1),  
 height = c(189.4,177.3,177.2,196.3,181.9,188.4,189.0,179.5,166.7,166.9),  
 age = c(11,12,13,14,15,16,17,18,19,20),  
 bmi = c(21.2,24.7,24.5,21.5,21.1,21.4,22.2,22.7,25.7,24.8)  
)

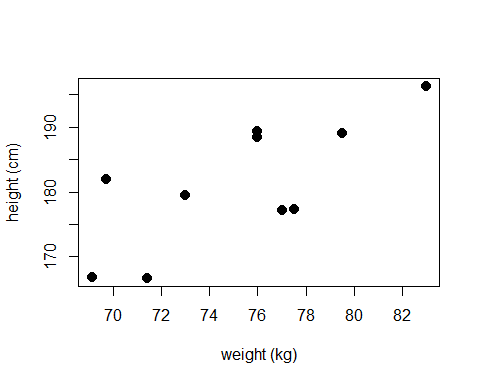
Correlation coefficients can be calculated with the function cor.

cor(weird.people$weight, weird.people$height)

## [1] 0.7460451

The relevant plot can be obtained like this:

plot(weird.people$weight, weird.people$height, # main part  
 xlab = "weight (kg)", ylab = "height (cm)", # axis labels  
 pch = 20, cex = 2) # formatting points



By default, this calculates the Pearson correlation coefficients. To perform statistical tests for correlation, cor.test can be used:

cor.test(weird.people$weight, weird.people$height)

##   
## Pearson's product-moment correlation  
##   
## data: weird.people$weight and weird.people$height  
## t = 3.1689, df = 8, p-value = 0.01322  
## alternative hypothesis: true correlation is not equal to 0  
## 95 percent confidence interval:  
## 0.219546 0.936003  
## sample estimates:  
## cor   
## 0.7460451

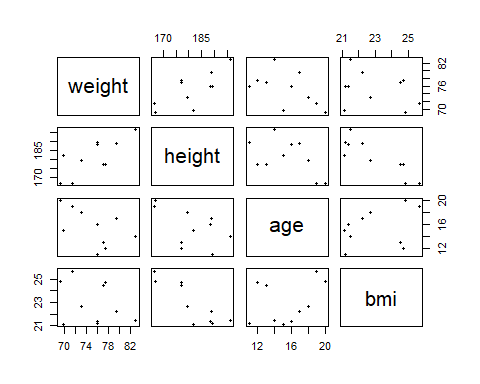
As is often the case with R, we can get the 95% confidence interval (of the correlation coefficient) along with the test. We can see that there is indeed a correlation between height and weight (r=0.75, p=0.013).

We might want to calculate all the (pairwise) correlation coefficients at once and visualize the data along with these. One option is to use the functions cor and plot. Generally, plot draws all kinds of plots, but if it is provided with a table of numeric variables, it tries to be clever and draw a matrix of scatterplots:

cor(weird.people) # all variables must be numeric!

## weight height age bmi  
## weight 1.0000000 0.7460451 -0.5354630 -0.3043058  
## height 0.7460451 1.0000000 -0.5275742 -0.8608247  
## age -0.5354630 -0.5275742 1.0000000 0.3408819  
## bmi -0.3043058 -0.8608247 0.3408819 1.0000000

plot(weird.people, pch = 20) # all variables must be numeric!



Another option is to use function chart.Correlation of the package PerformanceAnalytics.

#install.packages("PerformanceAnalytics") # only once per computer!  
library(PerformanceAnalytics) # once per session

## Warning: package 'PerformanceAnalytics' was built under R version 3.5.3

## Loading required package: xts

## Warning: package 'xts' was built under R version 3.5.3

## Loading required package: zoo

## Warning: package 'zoo' was built under R version 3.5.3

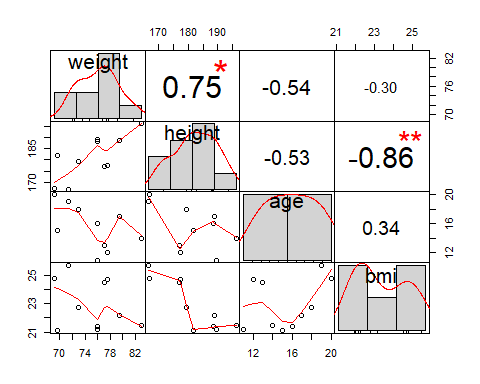
##   
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':  
##   
## as.Date, as.Date.numeric

##   
## Attaching package: 'PerformanceAnalytics'

## The following object is masked from 'package:graphics':  
##   
## legend

chart.Correlation(weird.people)



In the lower triangle, we see the pairwise scatterplots with a line trying to describe the relationship. In the diagonal, we see histograms. In the upper triangle, we see the correlation coefficients.

# Linear models

**Regression analysis with two variables, types I and III**

cor(weird.people$weight, weird.people$height)  
plot(weird.people$weight, weird.people$height, pch = 20,  
 xlab = "Weight", ylab = "Height", main = "Height vs weight")  
cor(weird.people$weight, weird.people$age)  
plot(weird.people$weight, weird.people$age)  
m1 <- lm(weight ~ height + age, data = weird.people)  
m2 <- lm(weight ~ age + height, data = weird.people)  
anova(m1) # type I ANOVA  
anova(m2)  
drop1(m1, test = "F") # type III ANOVA  
drop1(m2, test = "F")

**two-way ANOVA**

m1 <- lm(weight ~ color + sex, data = anova.data)  
m2 <- lm(weight ~ sex + color, data = anova.data)  
anova(m1) # type I ANOVA  
anova(m2)  
resids <- residuals(m1)  
hist(resids[anova.data$sex == "male"],  
 xlab = "", main = "Histogram of residuals\n(male)")  
hist(resids[anova.data$sex == "female"],   
 xlab = "", main = "Histogram of residuals\n(female)")

**Test of non-linearity**

m <- lm(weight ~ age + I(age\*\*2), data = weird.people)  
anova(m) # Type I ANOVA: age squared must be last  
plot(weight ~ age, data = weird.people, pch=20)  
x0 <- seq(0,20,length=100)  
y0 <- predict(m, newdata = data.frame(age=x0))  
lines(x0,y0, col="blue")  
coef(m) # weight = 41.7 + 5.3\*age - 0.20\*age\*\*2

**Interaction plots**

interaction.plot(anova.data$color, anova.data$sex, anova.data$weight,  
 ylab = "color", xlab = "mean weight")

ANOVA, ANCOVA, linear regression and multiple regression are all flavours of the same model: the linear model. In R, all of these analyses are performed by the function lm (**l**inear **m**odel).

## Multiple linear regression

As a first intro to the function, let’s do something we already know: (multiple) linear regression. It is wise to save the analysis result into a separate variable, e.g. m1.

m1 <- lm(weight ~ height + age + bmi, data = weird.people)

The first argument to the lm function is the *formula* of the model. Formula contains two parts: left-hand-side to the tilde (~, try Shift + (the key below Esc)), and right-hand-side. Left-hand-side contains (name of) the dependent variable. Right-hand-side contains (names of) all of the independent variables, separated by plus signs. Because the variables are in a data frame (not floating freely around), the dataframe has to be named also.

The main results of the multiple regression analysis can be obtained with summary.

summary(m1)

##   
## Call:  
## lm(formula = weight ~ height + age + bmi, data = weird.people)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.39684 -0.13500 0.06283 0.14680 0.28409   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -154.39308 6.17331 -25.010 2.69e-07 \*\*\*  
## height 0.85028 0.02073 41.017 1.40e-08 \*\*\*  
## age 0.01285 0.03617 0.355 0.734   
## bmi 3.27643 0.10354 31.643 6.62e-08 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.2694 on 6 degrees of freedom  
## Multiple R-squared: 0.9975, Adjusted R-squared: 0.9963   
## F-statistic: 806.1 on 3 and 6 DF, p-value: 3.314e-08

The first part (Call:) echoes the analysis you have just performed. The second part (Residuals:) shows the distribution of model residuals (minimum, lower quartile, median, upper quartile, maximum). The third part (Coefficients:) is the most important. It displays the table of all coefficients of the model, along with the tests whether the coefficient is zero or not. Based on this table, we can write that our model is:

weight = -154.39 + 0.85 \* height + 0.01 \* age + 3.28 \* bmi

Based on the Coefficients table we can also say that height and BMI are significantly related to weight (p<0.0001 for both), but not age (p=0.7). By the way, 1.40e-08 means 1.40 times 10 to the power of -8, in other words

1.40e-08 = 0.0000000140

The last part below the table of coefficients displays fit characteristics of the whole model: the standard error of residuals, R-squared values, and the F test of the whole model.

## Squared term in model

Inside the model formula, you can include mathematical functions of your numeric variables *using the function* I. For example, to test whether the association of age and weight is linear or curved, one can include the square of age age\*\*2 into the model formula inside I() and then test it with Type I test.

m2 <- lm(weight ~ age + I(age\*\*2), data = weird.people)  
anova(m2) # Type I test

## Analysis of Variance Table  
##   
## Response: weight  
## Df Sum Sq Mean Sq F value Pr(>F)  
## age 1 50.427 50.427 3.3605 0.1094  
## I(age^2) 1 20.406 20.406 1.3599 0.2818  
## Residuals 7 105.043 15.006

We can’t say that association of age and weight is curved (p=0.2818).

If the squared age is not enclosed inside I(), then R doesn’t understand it or understands it completely wrong.

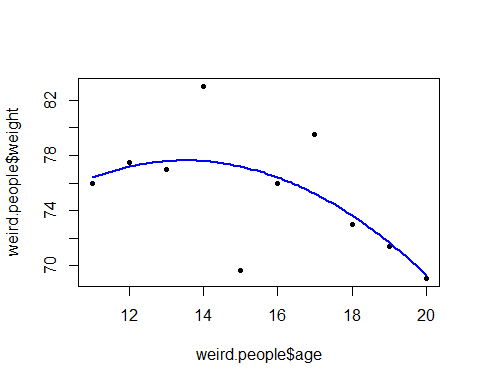
Of course, defining the squared age into a separate variable and then using that new variable is always an option:

weird.people$age.squared <- weird.people$age \*\* 2  
m2a <- lm(weight ~ age + age.squared, data = weird.people)  
anova(m2a)

## Analysis of Variance Table  
##   
## Response: weight  
## Df Sum Sq Mean Sq F value Pr(>F)  
## age 1 50.427 50.427 3.3605 0.1094  
## age.squared 1 20.406 20.406 1.3599 0.2818  
## Residuals 7 105.043 15.006

Plotting the regression line requires calculating the predicted values for some fine grid of pre-specified x-values, and then connecting the dots with a line:

# First, maybe it is good to plot the actual data to a scatterplot  
plot(weird.people$age, weird.people$weight, pch = 20)  
# Next, let's make a sequence of x-values to draw the line with  
x0 <- seq( # sequence  
 min(weird.people$age), # from minimal age  
 max(weird.people$age), # to maximal age  
 length = 100) # with length 100  
# make predictions from our model  
y0 <- predict( # predict  
 m2, # from the model m2  
 newdata = data.frame( # using this new data (COLUMN NAMES MUST MATCH!)  
 age = x0) # instead of the original age we use the sequence  
 )   
# Finally, add line to the plot  
lines(x0, y0,   
 lwd = 2, col = "blue") # modify line width and color



It is possible to extract only the coefficients of the model like this:

coef(m2)

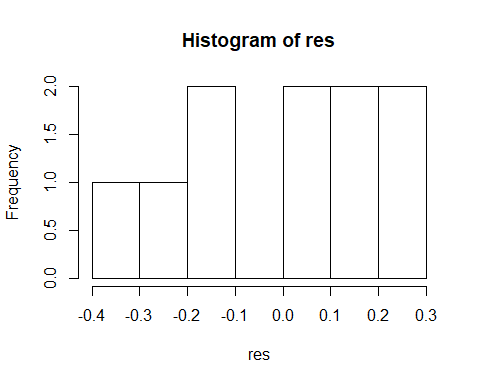
## (Intercept) age I(age^2)   
## 41.7290909 5.3125000 -0.1965909

From this, we can write down the formula: .

## Residuals

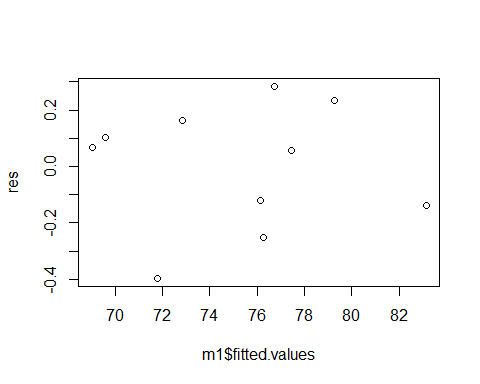
The significance tests for linear regression models and ANOVA models assume normal distribution of residuals. This can be examined in various ways. However, as our current model is based only on 10 observations, the diagnostic plots seem rather obscure and we probably can not definitely confirm or reject whether the normality assumption is satisfied.

res <- residuals(m1) # extract all residuals of the model  
hist(res) # histogram of residuals



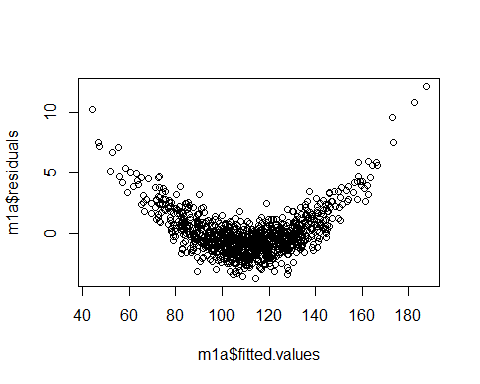
It might be useful to plot residuals against the fitted values. The residuals should show no trends nor differences in variance.

plot(m1$fitted.values, res)

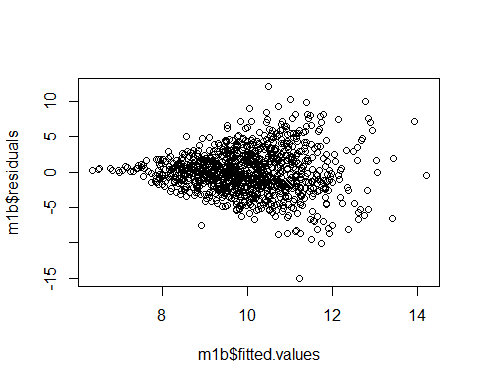


Currently we can’t say much, but for the following two toy-models, the plots would indicate a missing non-linear term and heteroscedascity.

toy.data <- data.frame(x = 10 + rnorm(1000)) # fabricate data  
toy.data$ya <- toy.data$x + toy.data$x\*\*2 + rnorm(1000)  
m1a <- lm(ya ~ x, toy.data)  
plot(m1a$fitted.values, m1a$residuals) # important predictor (x\*\*2) missing



toy.data$yb <- toy.data$x + (toy.data$x - 7) \* rnorm(1000) # generate data  
m1b <- lm(yb ~ x, toy.data)  
plot(m1b$fitted.values, m1b$residuals) # heteroscedascity



## ANOVA

ANOVA can be performed in exactly analogous way as the multiple regression analysis. Interactions can be specified with asterisk, i.e. sex \* color means sex + color + interaction of sex and color.

# read the data from file  
anova.data <- read.table("anova-data.csv", header = T, sep = ";", dec = ",")  
anova.data # look at the data

## color sex weight  
## 1 red male 10.1  
## 2 red male 9.4  
## 3 red male 10.0  
## 4 red male 10.4  
## 5 red female 13.0  
## 6 red female 13.1  
## 7 red female 13.1  
## 8 red female 12.9  
## 9 green male 11.0  
## 10 green male 11.4  
## 11 green male 10.8  
## 12 green male 11.2  
## 13 green female 14.0  
## 14 green female 14.1  
## 15 green female 14.2  
## 16 green female 14.3  
## 17 blue male 12.0  
## 18 blue male 11.9  
## 19 blue male 12.0  
## 20 blue male 12.0  
## 21 blue female 15.5  
## 22 blue female 15.0  
## 23 blue female 15.0  
## 24 blue female 15.0

# perform the analysis  
m2 <- lm(weight ~ sex \* color, data = anova.data)

The ANOVA table is obtained with function anova. Interaction term is indicated with colon :.

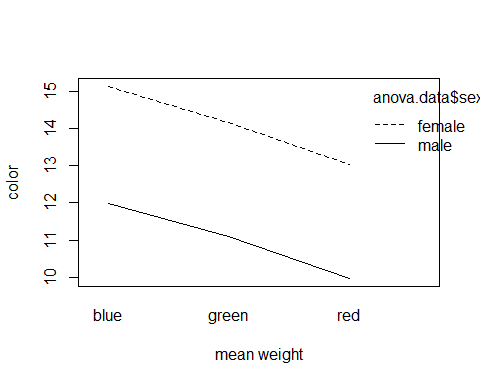
anova(m2)

## Analysis of Variance Table  
##   
## Response: weight  
## Df Sum Sq Mean Sq F value Pr(>F)   
## sex 1 57.042 57.042 1026.75 < 2.2e-16 \*\*\*  
## color 2 16.863 8.432 151.77 5.399e-12 \*\*\*  
## sex:color 2 0.013 0.007 0.12 0.8876   
## Residuals 18 1.000 0.056   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

As we can see, both sex and color are associated with weight, but effect of color does not differ between sexes (p=0.9).

The possible two-way interaction (or lack of it) can be illustrated with an interaction plot. This interaction plot displays the raw group means in the data (not the least-square means).

interaction.plot(anova.data$color, anova.data$sex, anova.data$weight,  
 ylab = "color", xlab = "mean weight")



## Type I, Type III SS

The function anova displays the Type I sums of squares and corresponding F tests. Type III tests can be obtained with the function drop1.

drop1(m2, test = "F") # Type III tests

## Single term deletions  
##   
## Model:  
## weight ~ sex \* color  
## Df Sum of Sq RSS AIC F value Pr(>F)  
## <none> 1.0000 -64.273   
## sex:color 2 0.013333 1.0133 -67.955 0.12 0.8876

As we can see, R only performed the Type III test for the interaction term. This is because R is very decent and refuses even to consider discarding the main effects while the interaction is included. After discarding the insignificant interaction, we can examine the Type III F-tests for the main effects.

m2.1 <- lm(weight ~ sex + color, data = anova.data)  
drop1(m2.1, test = "F")

## Single term deletions  
##   
## Model:  
## weight ~ sex + color  
## Df Sum of Sq RSS AIC F value Pr(>F)   
## <none> 1.013 -67.955   
## sex 1 57.042 58.055 27.200 1125.82 < 2.2e-16 \*\*\*  
## color 2 16.863 17.877 -3.069 166.41 3.425e-13 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

## Estimated differences

We have now established a model with only significant factors. Now its time to examine how the specific levels of the independent variables affect the dependent variable. As for the multiple regression model, this is achieved with the function summary.

summary(m2.1)

##   
## Call:  
## lm(formula = weight ~ sex + color, data = anova.data)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.55833 -0.09167 -0.00833 0.07292 0.44167   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 15.09167 0.09189 164.230 < 2e-16 \*\*\*  
## sexmale -3.08333 0.09189 -33.553 < 2e-16 \*\*\*  
## colorgreen -0.92500 0.11255 -8.219 7.66e-08 \*\*\*  
## colorred -2.05000 0.11255 -18.215 6.38e-14 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.2251 on 20 degrees of freedom  
## Multiple R-squared: 0.9865, Adjusted R-squared: 0.9844   
## F-statistic: 486.2 on 3 and 20 DF, p-value: < 2.2e-16

The output contains exactly the same parts with almost exactly the same meaning as for the multiple regression model. The only difference lies in the interpretation of the coefficients in the Coefficients table. The coefficients show the difference from the “base” level. By default, R uses the alphabetically first level as the “base”. For example, let’s consider the factor “sex”, which has levels “female” and “male”. Because “f” is before “m” in alphabet, “female” is taken as the “base”. The row sexmale -3.08333 means that the average for males is 3.08 units smaller than the average for females. In a similar fashion, colorgreen -0.92500 means that the average for greens is 0.925 units smaller than the average of blues.

## Post-hoc tests: Tukey’s HSD

The function summary displays the estimated differences between averages of factor levels. In the right-most column we even can see some p-values, however, these p-values test the difference only from the “base” level. They do not yield any information about the potential differences between the other levels. For example, we know that both green and red are significantly different from blue (p=7.66e-08 and p=6.38e-14, respectively), but we do *not* know whether green and red themselves are also different or not. Futhermore, the p-values in the summary table are not adjusted for multiple comparisons.

Tukey’s Honest Significant Difference (HSD) test comes in handy in these cases.

TukeyHSD(aov(m2.1))

## Tukey multiple comparisons of means  
## 95% family-wise confidence level  
##   
## Fit: aov(formula = m2.1)  
##   
## $sex  
## diff lwr upr p adj  
## male-female -3.083333 -3.27502 -2.891647 0  
##   
## $color  
## diff lwr upr p adj  
## green-blue -0.925 -1.20974 -0.6402599 2e-07  
## red-blue -2.050 -2.33474 -1.7652599 0e+00  
## red-green -1.125 -1.40974 -0.8402599 0e+00

We should look for the p adj column, which shows the adjusted p-values of all pairwise comparisons. Obviously the p-values are so small that the function does not handle them properly (0e+00 means p-value exactly zero). We would write p<0.0001 in this case.

## Nested ANOVA

If we have two factors such that one factor is essentially a sub-grouping of the other, we would say that the factors are nested and we need to take this into account in analyses. For example, we are interested in some health result of patients in ten hospitals, and in each hospital there are three wards with the same labels (“Ward 1”, “Ward 2” and “Ward 3”). *Obviously* the ward is nested in hospital, as there is no meaning for Ward 1, 2 or 3 out of the context of the hospital. It would be silly to consider “main effect” of ward, as Ward 1 of Hospital 1 has nothing to do with Ward 1 in any other hospital. However, examining differences between wards within hospitals would still be a valid task.

In the following example, we compare lengths of feathers of males and females of different species. It is conceivable that sex could be considered as nested within species. However, unlike ward labels, sexes most certainly do have meaning separately from species.

# fabricate data  
nested.data <- data.frame(  
 species = as.factor(rep(c("a", "b"), each = 10)),  
 sex = as.factor(rep(c("f", "m", "f", "m"), each = 5)),  
 feather = c(4,5,6,11,14,34,2,1,11,14,16,27,13,22,32,11,26,18,21,4)  
)  
# perform analysis  
m11 <- lm(feather ~ species/sex, # sex nested within species  
 data = nested.data)  
anova(m11)

## Analysis of Variance Table  
##   
## Response: feather  
## Df Sum Sq Mean Sq F value Pr(>F)   
## species 1 387.2 387.2 4.6820 0.04595 \*  
## species:sex 2 138.4 69.2 0.8368 0.45121   
## Residuals 16 1323.2 82.7   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

From the output we see that feather lengths of sexes (within species) do not differ (for the term species:sex, p=0.45). Feather lengths of species do differ (p=0.046).

As you might have noticed by now, nested ANOVA is somewhat similar to ANOVA with interaction: in both analyses, effect of one factor depends on the level of the other factor. The difference is that in the nested ANOVA, the “main effect” of the nested factor would be (completely and utterly) meaningless, but in the interaction case both main effects could have meaning.

# Linear mixed models

**Two random factors**

m1 <- lmer(para ~ trea + (1|nr) + (1|time))  
anova(m1) # ANOVA for fixed effects  
summary(m1) # variances of random effects, coefficients of fixed effects

**Repeated measurements ANOVA with autocorrelation in time**

# naive ANOVA (does not take into account that same individuals are measured repeatedly)  
m1 <- lm(para ~ trea \* time)  
anova(m1)  
# repeated measures, no autocorrelation  
m2 <- lme(para ~ trea \* time, random = ~1|nr)  
anova(m2)  
# repeated measures, autocorrelation in time  
m3 <- lme(para ~ trea\*time, random = ~1|nr, correlation = corAR1(,form = ~time|nr))  
anova(m3)

We will use two different packages with generally very similar functionality: package lme4 function lmer, and package nlme function lme. Both of these can be used almost exactly as the usual lm, and in the simplest case, lmer and lme are almost the same. However, using at least 2 random factors is easy with lme4 package but surprisingly obscure with nlme. Conversely, nlme enables using various autocorrelation structures quite easily, but defining it in lme4 is rather cumbersome.

So we will need to use and load both of them.

#install.packages("nlme") # once per computer  
library(nlme) # once per session  
#install.packages("lme4") # once per computer  
#install.packages("lmerTest") # once per computer  
library(lme4) # once per session

## Warning: package 'lme4' was built under R version 3.5.3

## Loading required package: Matrix

##   
## Attaching package: 'lme4'

## The following object is masked from 'package:nlme':  
##   
## lmList

library(lmerTest) # once per session. Must be loaded in this order: first lme4 then lmerTest!

## Warning: package 'lmerTest' was built under R version 3.5.3

##   
## Attaching package: 'lmerTest'

## The following object is masked from 'package:lme4':  
##   
## lmer

## The following object is masked from 'package:stats':  
##   
## step

In addition, we use the package lmerTest to get p-values into the output of lme4 models.

## Fixed and random effects

Let’s consider a toy data set about weights of nestlings from a few different broods fed with two different feeds. Compare to example 5.9 in [SAS](http://kodu.ut.ee/~tammarut/nai5-05.pdf) and [R](http://kodu.ut.ee/~tammarut/Rnaide5.txt).

# fabricate data  
feeding.data <- data.frame(  
 brood = as.factor(rep(letters[1:5],each=6)),  
 feed = rep(as.factor(rep(letters[1:2],each=3)),5),  
 weight = c(5,5,6,11,12,11,3,4,5,7,8,9,5,7,6,8,10,12,9,9,9,11,12,12,12,11,12,17,16,14)  
)  
# perform the analysis  
m3 <- lmer(weight ~ feed + (1|brood), data = feeding.data)

The random effects are included in the model formula along with the fixed effects. Each random effect is defined by a small “formula” in parentheses, (1|brood) in our example. The “formula” contains two parts: the left and right side of the vertical bar |. The right side contains the name of the random factor. Currently, brood. If we had more random factors, they would have to be included in the main formula in their own parentheses, e.g. weight ~ feed + (1|brood) + (1|randfact2). The left hand side of the vertical bar contains 1 in the simplest case. In random slopes model, there would be the variable with random slope instead of 1. If we have no random slopes, then the only random thing is often called the random intercept.

summary(m3)

## Linear mixed model fit by REML. t-tests use Satterthwaite's method [  
## lmerModLmerTest]  
## Formula: weight ~ feed + (1 | brood)  
## Data: feeding.data  
##   
## REML criterion at convergence: 105.8  
##   
## Scaled residuals:   
## Min 1Q Median 3Q Max   
## -1.88060 -0.75864 0.00965 0.68095 1.70507   
##   
## Random effects:  
## Groups Name Variance Std.Dev.  
## brood (Intercept) 8.203 2.864   
## Residual 1.244 1.116   
## Number of obs: 30, groups: brood, 5  
##   
## Fixed effects:  
## Estimate Std. Error df t value Pr(>|t|)   
## (Intercept) 7.2000 1.3129 4.1997 5.484 0.00467 \*\*   
## feedb 4.1333 0.4073 23.9999 10.147 3.69e-10 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Correlation of Fixed Effects:  
## (Intr)  
## feedb -0.155

The summary consists of a few sections, rather similar to that of lm.

The Random effects section is the main difference from the output of lm. Both standard deviations and variances are displayed for all random effects (currently we have only one, brood). In the simplest case, the “random effect” is in fact only the random intercept. From the output we can see that the mean levels of broods vary with standard deviation 2.86. The standard deviation of the residuals is also displayed in this section (1.12).

The Fixed effects table is very similar to the equivalent table in lm summary output. Without using the lmerTest package, the prominent difference would rest in the fact that p-values are not printed in the output (supposedly due to some esotheric-statistical correctness reason). However, currently we have those p-values.

The significance of fixed effects can be seen with the usual function anova:

anova(m3)

## Type III Analysis of Variance Table with Satterthwaite's method  
## Sum Sq Mean Sq NumDF DenDF F value Pr(>F)   
## feed 128.13 128.13 1 24 102.96 3.693e-10 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

The same model can easily be fitted with lme:

m3a <- lme(weight ~ feed, random = ~ 1|brood, data = feeding.data)  
summary(m3a)

## Linear mixed-effects model fit by REML  
## Data: feeding.data   
## AIC BIC logLik  
## 113.8105 119.1393 -52.90523  
##   
## Random effects:  
## Formula: ~1 | brood  
## (Intercept) Residual  
## StdDev: 2.864211 1.115547  
##   
## Fixed effects: weight ~ feed   
## Value Std.Error DF t-value p-value  
## (Intercept) 7.200000 1.3128990 24 5.484047 0  
## feedb 4.133333 0.4073401 24 10.147132 0  
## Correlation:   
## (Intr)  
## feedb -0.155  
##   
## Standardized Within-Group Residuals:  
## Min Q1 Med Q3 Max   
## -1.880603535 -0.758642545 0.009657616 0.680954001 1.705082293   
##   
## Number of Observations: 30  
## Number of Groups: 5

Matching the results is rather straightforward. However, expanding lme to two random factors is not as simple as you would think (if not impossible).

## Repeated measures

Effect of feed (two plants, a and b) on parasite load of birds is studied. Each individual bird is measured 4 times. We suspect that temporally closer measurements can be more similar than any measurements, therefore we should specify an autoregressive component in our model.

Since we are looking for some consistent trend throughout all the years (increase or decrease), we should include the time as numeric variable. An alternative would be to include it as a factor, which would require more observations, but enable to detect any sort of change pattern.

Compare to example 5.12 in [SAS](http://kodu.ut.ee/~tammarut/nai5-05.pdf) and [R](http://kodu.ut.ee/~tammarut/Rnaide5.txt).

# fabricate the data  
nr <- rep(1:6, each=4) # ID of the individual (bureaucratic necessity)  
trea <- factor(rep(c("a","b"),each=12)) # treatment (2 treatments)  
time <- rep(1:4,6) # time of measurement (each individual measured 4x)  
para <- c(1,2,3,4,2,2,3,4,2,3,4,5,1,4,4,4,2,5,5,4,2,6,5,5) # response variable  
# perform the analysis  
m16 <- lme(para ~ trea \* time, # time should be included as a numeric "fixed" effect, as in the naive approach  
 random = ~1|nr, # individual as a random factor  
 correlation = corAR1(,form = ~time|nr)) # This part specfies that measurements of an individual that are made closer togeter are more correlated  
anova(m16)

## numDF denDF F-value p-value  
## (Intercept) 1 16 210.15020 <.0001  
## trea 1 4 4.48672 0.1015  
## time 1 16 24.54783 0.0001  
## trea:time 1 16 0.15536 0.6987

Effect of feeding does not change in time (p=0.7). Furthermore, there is no difference between parasite loads of birds fed with different plants (p=0.10). However, the parasite load does change in time (p=0.0001). Notice that the time is a numeric variable! E.g. weeks after start of treatment or something like that.

As usually, the estimates of fixed effects are available with summary:

summary(m16)

## Linear mixed-effects model fit by REML  
## Data: NULL   
## AIC BIC logLik  
## 79.61521 86.58534 -32.80761  
##   
## Random effects:  
## Formula: ~1 | nr  
## (Intercept) Residual  
## StdDev: 0.3465368 0.9207629  
##   
## Correlation Structure: AR(1)  
## Formula: ~time | nr   
## Parameter estimate(s):  
## Phi   
## 0.003492675   
## Fixed effects: para ~ trea \* time   
## Value Std.Error DF t-value p-value  
## (Intercept) 0.6669579 0.6818471 16 0.978164 0.3426  
## treab 1.3298382 0.9642774 4 1.379103 0.2400  
## time 0.8999418 0.2379462 16 3.782123 0.0016  
## treab:time -0.1326350 0.3365068 16 -0.394153 0.6987  
## Correlation:   
## (Intr) treab time   
## treab -0.707   
## time -0.872 0.617   
## treab:time 0.617 -0.872 -0.707  
##   
## Standardized Within-Group Residuals:  
## Min Q1 Med Q3 Max   
## -1.6550419 -0.4445404 -0.1755253 0.5338711 2.4527209   
##   
## Number of Observations: 24  
## Number of Groups: 6

From the Random effects section we can see that individual birds’ mean parasite levels have standard deviation 0.35. Standard deviation that is not explained by the model (“error”) is 0.92.

Section Correlation structure describes the strength of the autocorrelation. It is described by Phi. This is not a correlation coefficient but something a little bit more tricky (see [autoregressive models](https://en.wikipedia.org/wiki/Autoregressive_model) in Wikipedia), however, closer to zero means less autocorrelation.

From the Fixed effects section we read that one unit of time is related to 0.90 increase in parasite load (p=0.0016). Other effects are not reliably different from zero.

# Logistic regression

m <- glm(y ~ x1, family = binomial)  
drop1(m, test="LR")

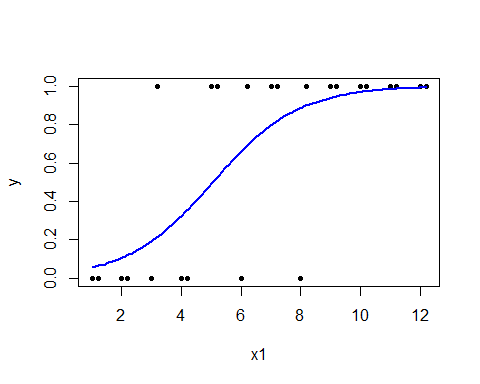
## Single term deletions  
##   
## Model:  
## y ~ x1  
## Df Deviance AIC LRT Pr(>Chi)   
## <none> 17.355 21.355   
## x1 1 31.755 33.755 14.4 0.0001478 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

coef(m)

## (Intercept) x1   
## -3.5242965 0.6985107

Logistilise kõvera võrrand on

plot(x1, y, pch=20)  
p <- function(x1) {  
 exp(-3.5243+0.6985\*x1) / (1 + exp(-3.5243+0.6985\*x1))  
}  
curve(p, add = T, col = "blue", lwd = 2)

 \*\*\*

Logistic regression, Poisson regression and many more types of analyses are specific members of the **g**eneralized **l**inear **m**odels, hence the name of the relevant R function, glm. This, too, can be used almost exactly as the function for linear models, lm. However, since a large variety of models are GLM-s, the specific *family* of models has to be specified in each analysis. For logistic regression, the family’s name is binomial. Follows the example 6.4 in [SAS](http://kodu.ut.ee/~tammarut/nai6-05.doc) and [R](http://kodu.ut.ee/~tammarut/Rnaide6.txt).

x1 <- c(c(1:12),c(1:12)+0.2) # fabricate data  
x2 <- factor(rep(c("a","b"), each=12))  
y <- c(0,0,0,0,1,0,1,0,1,1,1,1,0,0,1,0,1,1,1,1,1,1,1,1)  
# the model  
m5 <- glm(y ~ x1, # model formula  
 family = binomial) # family of models  
drop1(m5, test = "LRT") # Type III-ish likelihood ratio test for whole factors

## Single term deletions  
##   
## Model:  
## y ~ x1  
## Df Deviance AIC LRT Pr(>Chi)   
## <none> 17.355 21.355   
## x1 1 31.755 33.755 14.4 0.0001478 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

First we might want to look at an ANOVA-like table where we can see the significance tests for whole factors (as for ANOVA). Unlike ANOVA, however, we can’t use the F test to determine significance of a factor (or any independent variable). For generalized linear models, we can use the likelihood ratio test for a similar purpose. Likelihood is a number describing the fit of a model to our data. By comparing likelihoods of models with and without a particular factor, we can say whether that factor makes a difference or not. If a factor is pointless, then the likelihood ratio approximately follows chi-squared distribution.

Let’s look at the parameter estimate of that one continuous variable.

summary(m5) # Parameter estimates

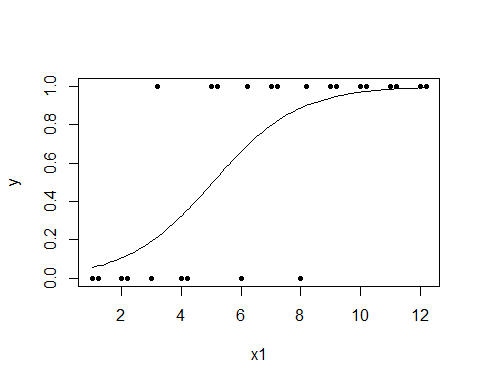
##   
## Call:  
## glm(formula = y ~ x1, family = binomial)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.0896 -0.4826 0.1702 0.5016 1.7507   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -3.5243 1.5574 -2.263 0.0236 \*  
## x1 0.6985 0.2727 2.561 0.0104 \*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 31.755 on 23 degrees of freedom  
## Residual deviance: 17.355 on 22 degrees of freedom  
## AIC: 21.355  
##   
## Number of Fisher Scoring iterations: 6

The model output looks somewhat similar to that of linear models, but the exact meaning of the coefficients is not so straightforward. In particular, the coefficients are *not* probabilities. Consult the lecture notes about this (lecture 9). Often it is sufficient to know that positive means “more” and negative means “less”.

In our current analysis, the probability of success () depends on like this:

We can plot this association:

plot(x1,y, pch=20)  
# Let's define the association as R function  
p <- function(x1) {  
 exp(-3.5243+0.6985\*x1) / (1 + exp(-3.5243+0.6985\*x1))  
}  
curve(p, add = T)



# Poisson regression

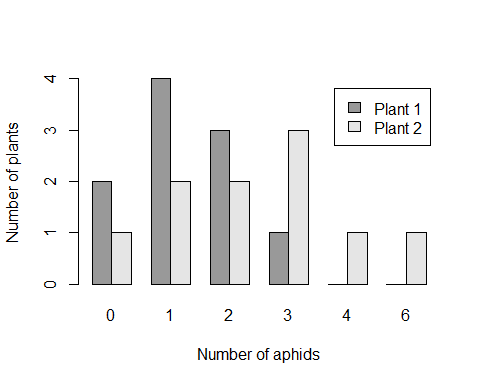
m <- glm(aphids ~ plant, family = poisson)  
drop1(m, test = "LRT")

Follows the example 6.5 in [SAS](http://kodu.ut.ee/~tammarut/nai6-05.doc) and [R](http://kodu.ut.ee/~tammarut/Rnaide6.txt).

# fabricate data  
plant <- factor(rep(c("Plant 1","Plant 2"), each=10))  
aphids <- c(0,0,1,1,1,1,2,2,2,3,0,1,1,2,2,3,3,3,4,6)

Let’s examine the data graphically:

barplot(table(plant, aphids), beside = T, # define the data  
 xlab = "Number of aphids", ylab = "Number of plants", # axis labels  
 legend.text = T, col = c("gray60", "gray90")) # formatting



Poisson regression is another example of a generalized linear model. So we can use glm again, only we need to specify that this time we want to have a poisson model.

m6 <- glm(aphids ~ plant, family = poisson)  
drop1(m6, test = "LRT")

## Single term deletions  
##   
## Model:  
## aphids ~ plant  
## Df Deviance AIC LRT Pr(>Chi)   
## <none> 20.185 68.144   
## plant 1 24.040 69.999 3.8551 0.04959 \*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Plant is borderline significant (likelihood ratio test p=0.04959), therefore we can say that there are differences between plants in aphid loadings. We can reach a similar conclusion when we compare the rates for Plant 2 and Plant 1.

summary(m6)

##   
## Call:  
## glm(formula = aphids ~ plant, family = poisson)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.2361 -0.5159 -0.2744 0.5685 1.8723   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 0.2624 0.2774 0.946 0.3442   
## plantPlant 2 0.6539 0.3419 1.912 0.0558 .  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for poisson family taken to be 1)  
##   
## Null deviance: 24.040 on 19 degrees of freedom  
## Residual deviance: 20.185 on 18 degrees of freedom  
## AIC: 68.144  
##   
## Number of Fisher Scoring iterations: 5

We can see that aphid loading for Plant 2 is times higher than that of Plant 1 (z=1.912, p=0.0558), however, this is a borderline significant finding. Notice that “1.92 times higher” also means “92% higher”. Due to the wonders of statistics, the parameter estimates for generalized linear models are approximately normally distributed for large (infinite) sample sizes, and that’s why we can use the z-statistic here.

# Principal component analysis

pca <- prcomp(cbind(wear,tear,date,size))  
cbind(wear, tear, date, size, pca$x[,1])  
cor.test(eggs, pca$x[,1])

Follows example 7.1 in [R](http://kodu.ut.ee/~tammarut/Rnaide7.txt) and [SAS](http://kodu.ut.ee/~tammarut/nai7-05.doc).

# fabricate data  
wear <- c(5,6,8,9,10,8,9,11,13,15)  
tear <- c(4,2,6,7,8,9,9,9,8,12)  
date <- c(1,1,5,5,6,7,6,8,7,9)  
size <- c(3,2,3,2,1,2,3,4,3,2)  
eggs <- c(100,90,85,88,67,50,45,40,30,20)  
# perform the analysis  
pca <- prcomp(cbind(wear,tear,date,size))

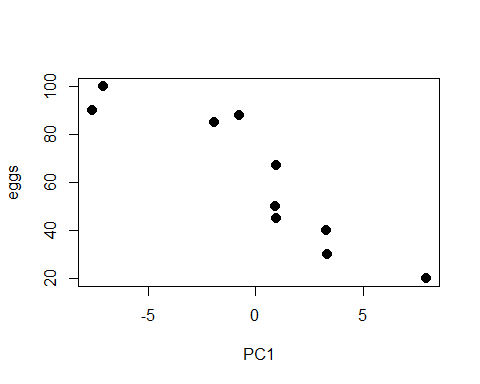
All principal component values of all the objects can be obtained as a matrix. R will calculate all possible principal components, but of course the intention is to use only first 1-2 of them (pca$x[,1:2]).

pca$x # All principal components

## PC1 PC2 PC3 PC4  
## [1,] -7.0930552 -0.32834563 0.2410251 0.97090863  
## [2,] -7.6480933 1.58653697 -0.3010269 -0.34489417  
## [3,] -1.9238876 -0.22524609 0.6654823 -0.47439852  
## [4,] -0.7497843 0.07318845 -0.4993887 -0.05547887  
## [5,] 0.9769312 0.07627476 -1.4526854 -0.38664292  
## [6,] 0.9062149 -2.38251488 -0.4892436 -0.53201333  
## [7,] 0.9598520 -1.34966755 0.2520284 0.64308890  
## [8,] 3.2733115 -0.41549006 1.6258143 -0.29921131  
## [9,] 3.3440279 2.04329959 0.6623725 -0.15384090  
## [10,] 7.9544830 0.92196444 -0.7043780 0.63248247

Now, if we want to analyse the association of the first principal component and the number of eggs eggs, we should extract the first column of the pca$x matrix:

plot(pca$x[,1], eggs, pch = 20, cex = 2,  
 xlab = "PC1", ylab = "eggs")



cor.test(pca$x[,1], eggs)

##   
## Pearson's product-moment correlation  
##   
## data: pca$x[, 1] and eggs  
## t = -5.9356, df = 8, p-value = 0.0003475  
## alternative hypothesis: true correlation is not equal to 0  
## 95 percent confidence interval:  
## -0.9770334 -0.6327942  
## sample estimates:  
## cor   
## -0.9027452

We see that the first principal component is strongly correlated with the number of eggs (r=-0.90, p=0.0003). Remember, however, that the direction (plus or minus) of the principal components is arbitrary. There is no inherent reason why the correlation is negative, not positive. Therefore the direction of the correlation shouldn’t be interpreted, but only the magnitude.