Using R for Statistical Analyses of Quantitative Traits

Rasmus Bak Stephansen & Peter Sørensen

2022-03-04

Introduction

Quantitative traits such as size, obesity or longevity vary greatly among individuals. Their phenotypes are continuously distributed phenotypes and do not show simple Mendelian inheritance (i.e., their phenotypes are distributed in discrete categories determined by one or a few genes). Basic descriptive statistics such as mean, and variance can be used to describe each of these traits. Trait values are often assumed to follow a normal distribution. Furthermore relationships between traits can be characterized in terms of covariance and correlations and linear relationships. In this chapter we will use small data examples in R to illustrate some of the basic statistical concept used in quantitative genetics. We suggest to use R through a user-friendly interface called Rstudio.

Install R and Rstudio

R is a free software environment for statistical computing and graphics (https://www.r-project.org/). Because R is free and it is available for the most commonly used operating systems such as Windows, MacOSX and Linux, it has become very popular in statistics and in data science. Furthermore, R can be extended with user-contributed code and documentation (called R-packages) in a very easy and standardised way. The number of available R-packages is growing rapidly and has reached more than 18000 (https://cran.r-project.org/web/packages/).

RStudio (https://www.rstudio.com/) is a private company that among a large number of different products distributes the RStudio Integrated Development Environment (IDE) for R. A great number of different resources about R and RStudio IDE is available.

Install R from here: https://mirrors.dotsrc.org/cran/

Install Rstudio (free version) from here: https://www.rstudio.com/products/rstudio/download/

Further information and introduction to R and Rstudio can be found here:

https://cran.r-project.org/doc/manuals/r-release/R-intro.html

https://www.rstudio.com/resources/cheatsheets

https://www.rstudio.com/resources/webinars

Basic statistics used for characterising a quantitative trait

In this section we will introduce these basic statistical concepts using simple data examples in R.

Mean and Variance

The mean and variance of a quantitative traits is are descriptive statistics commonly used to summarize a set of observations.

Suppose that we have measured the body-mass-index (BMI) for $n = 5$ individuals and the values are

```
y = c(23, 31, 27, 22, 25) # vector of observations
n = length(y)             # how many elements in vector
y                         # show vector
```

```
## [1] 23 31 27 22 25
```

```
n                         # show value of 'n'
```

```
## [1] 5
```

The **mean** of these values is their sum divided by their count,

$$\text{mean} = \frac{1}{n}\sum_{i=1}^{n} y_i = \frac{23+31+27+22+25}{5} = 25.6.$$

Mean describes the center of data values around which the individual values deviate the least (in terms of variance). In that sense it is the single value that best describes where the mass center of the values is. Mean can be computed in R by `mean()` function.

```
mean(y)             # best way: use R
```

```
## [1] 25.6
```

```
sum(y)/length(y)    # other way: using sum
```

```
## [1] 25.6
```

```
(23+31+27+22+25)/5 # manual way: do NOT use this in R because it is prone to errors and wastes time
```

```
## [1] 25.6
```

To have an idea how much the values of BMI *vary* around their mean we can use **standard deviation** that is the square root of **variance**.

Variance is the average squared deviation from the mean. When variance is computed from a sample of size n, whose mean is first estimated from that same sample, then the denominator in variance calculation is $n - 1$ rather than n, so

$$
\begin{aligned}
\text{variance} &= \frac{1}{n-1}\sum_{i=1}^{n}(\text{y}_i - \text{mean})^2 \\
&= \frac{(23-25.6)^2 + (31-25.6)^2 + (27-25.6)^2 + (22-25.6)^2 + (25-25.6)^2}{4} \\
&= 12.8
\end{aligned}
\tag{1}
$$

The same in R:

```
var(y)                          # using var() function
```

```
## [1] 12.8
```

```
sum((y - mean(y))^2) / (n - 1) # in practice use var()
```

```
## [1] 12.8
```

Standard deviation is the square root of variance and is a measure of variability that is in the same units as the original measurement (variance is in squared unit of the original measurement). Square root is computed by `sqrt()`.

```
sd(y)          # using sd() function for sample standard deviation
```

```
## [1] 3.577709
```

```
sqrt(var(y))   # SD = sqrt of variance
```

```
## [1] 3.577709
```

Interpretation of SD is as an average deviation that the obervations show around the mean of the sample. In our example, on average, observed bmi values deviate about 3.6 units from the mean.

Normal Distribution

The most prevalent continuous distribution is the Normal distribution (also called the Gaussian distribution after German mathematician Carl F. Gauss). A reason for its wide applicability is that in very many settings the distribution of sums of independent random variables tend towards a Normal distribution, and hence many complex properties, such as height or susceptibility to coronary artery disease, that result from an interplay between a large number of genetic and environmental factors, follow approximately a Normal distribution in the population.

Normal distribution is defined by 2 parameters: mean (mu, μ) and standard deviation (sigma, σ). Often (outside R) variance (sigma^2, σ^2) is used in place of standard deviation to define the second parameter. Always pay attention to which one is in question since mixing up these parameters badly mixes up all the statistics! For now, remember that the basic R functions take in standard deviation `sigma`.

The standard normal distribution, N(0,1), has mean = 0 and sd = 1. Let's plot the density function `dnorm(,0, 1)` and the cumulative distribution function `pnorm(, 0, 1)` of N(0,1) next to each other. We use `par(mfrow=c(1,2))` to say that we set plotting parameter `mfrow` to split the plotting region into 1 row and 2 columns, whence the two plots show up next to each other.

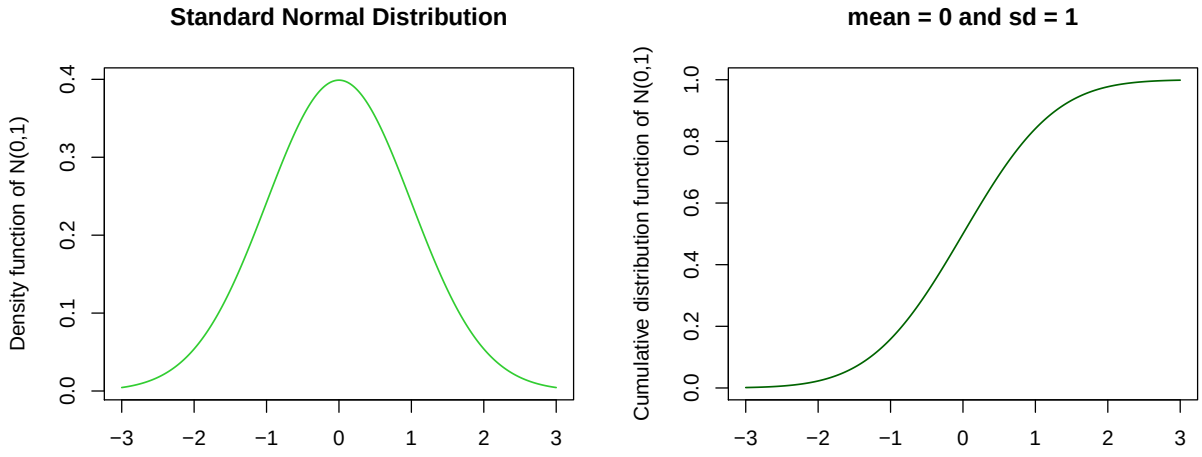
```
x = seq(-3, 3, 0.001) # range of x-values where we evaluate dnorm() and pnorm()

d = dnorm(x, 0, 1)    # values of density function of N(0,1)
p = pnorm(x, 0, 1)    # values of cumulative distribution function of N(0,1)

par(mfrow = c(1,2))   # plotting region split to 1 x 2 area to show two plots
                      # next to each other

plot(x, d, xlab = "", ylab = "Density function of N(0,1)",
     main = "Standard Normal Distribution",
     t = "l", lwd = 1.3, col = "limegreen")

plot(x, p, xlab = "", ylab = "Cumulative distribution function of N(0,1)",
     main = "mean = 0 and sd = 1",
     t = "l", lwd = 1.3, col = "darkgreen")
```



The density plot shows that the peak is at the mean of the distribution, and the density drops from there symmetrically making a bell-shaped curve characteristic to a Normal distribution.

The cumulative distribution plot shows how the probability mass accumulates as we move from small values (here starting from -3) to larger values (here up to 3). We know that 95% of the probability mass of the N(0,1) distribution is between values -1.96 and 1.96:

```
qnorm(c(0.025, 0.975), 0, 1) # 95% of mass of N(0,1) is between these two points
```

```
## [1] -1.959964  1.959964
```

Let's generate samples from a Normal distribution, say with mean=4 and sd=2, using `rnorm()` and plot the data using a histogram. Standard `hist()` shows on y-axis the counts of observations in each bin, but by setting `prob = TRUE` we can make the y-axis to scale to the values of a density function (making the total area of the histogram = 1). Then we can also show the theoretical density function `dnorm(,4,2)` in the same plot and compare the two.

```
n.samples = 5000  # samples from distribution
mu = 4            # mean
sigma = 2         # standard deviation

x = rnorm(n.samples, mu, sigma)        # random sample from normal distribution
data.frame(mean = mean(x), sd = sd(x)) # show empirical mean and sd of data
```

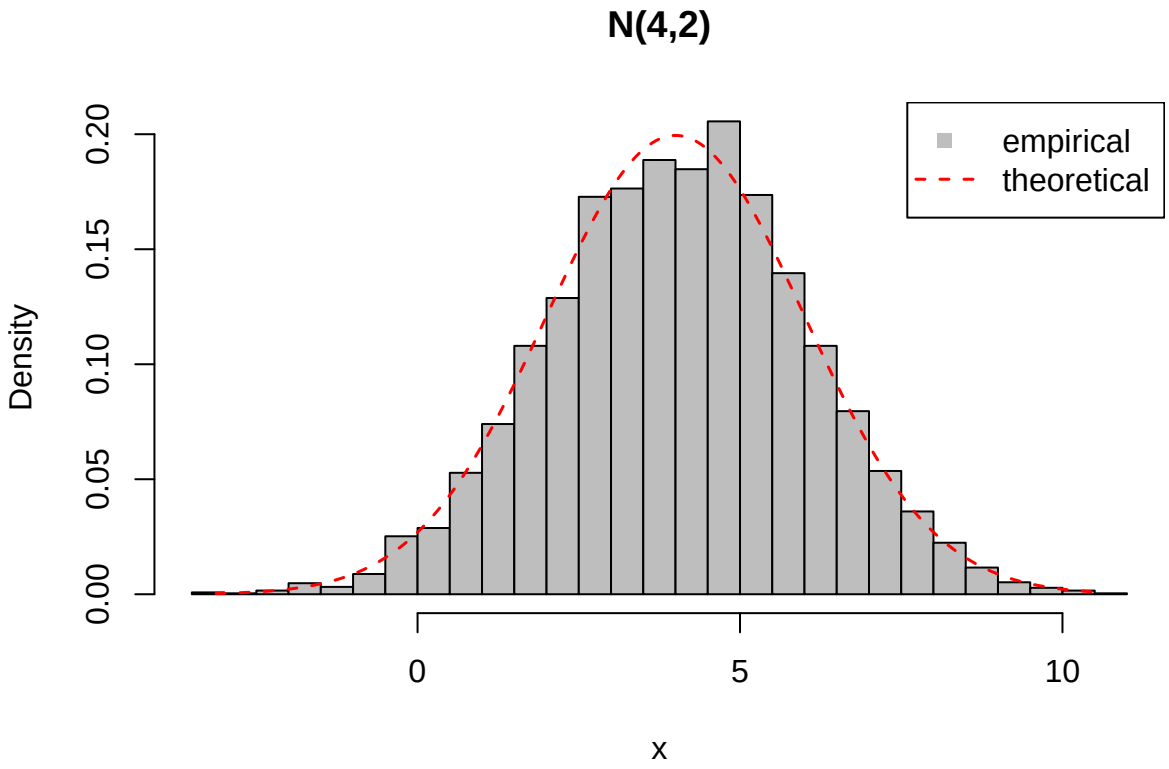
```
##       mean       sd
## 1 4.041594 1.981033
```

```
hist(x, breaks = 40, col = "gray", prob = TRUE, main = "N(4,2)")

# add grid of x-values to evaluate dnorm(, mu, sigma)
x.grid = seq(min(x), max(x), length.out = 1000)

# add dashed line to the current plot
lines(x.grid, dnorm(x.grid, mu, sigma), col = "red", lwd = 1.5, lty = 2)

# add a legend text to appear in the top right corner
# pch is plotting symbol (15 square; -1 no symbol)
# lty is line type (0 no line; 2 dashed line)
legend("topright", col = c("gray","red"), legend = c("empirical","theoretical"),
       lwd = 1.5, pch = c(15,-1), lty = c(0,2))
```



We see that the histogram of 5000 random samples from N(4,2) matches quite well with the theoretical density function of N(4,2), as it should. With more samples, the match would get even tighter.

Central Limit Theorem (CLT)

Assume dataset X contains n (independent) samples from some distribution with mean=μ and standard deviation=σ but X does not necessarily need to follow Normal, binomial or any other distribution we have ever heard about. CLT says that the distribution of the point estimate of the mean of X is approximately Normal(mean=μ,sd= $\sigma/\sqrt{n}$) in LARGE samples. This result can be used to derive confidence intervals for the estimated mean. For example, a 95%CI is the observed mean $\pm 1.96\, s/\sqrt{n}$, where s is the observed standard deviation of X. The importance of this result is its complete generality with respect to the shape of the underlying distribution. However, it requires a large sample to work in practice: Rather hundreds than a few dozens.

Let's try it out with Uniformly distributed values. Let's compute a mean of $n = 500$ U(0,1) distributed values, and repeat this computation 1000 times. We'll do that by making a big data matrix where rows are 1000 repetitions and columns are 500 observations. According to the theory (Uniform distribution, Variance of a uniform distribution), the mean of a uniform distribution U(0,1) is 0.5 and its standard deviation is $1/\sqrt{12} \approx 0.289$. Thus, we expect that the mean of 500 samples from U(0,1) has a mean of 0.5 and sd of $1/\sqrt{12 \cdot 500} \approx 0.0129$.. Our main interest is whether the distribution of the mean over the data sets is Normal as CLT claims.

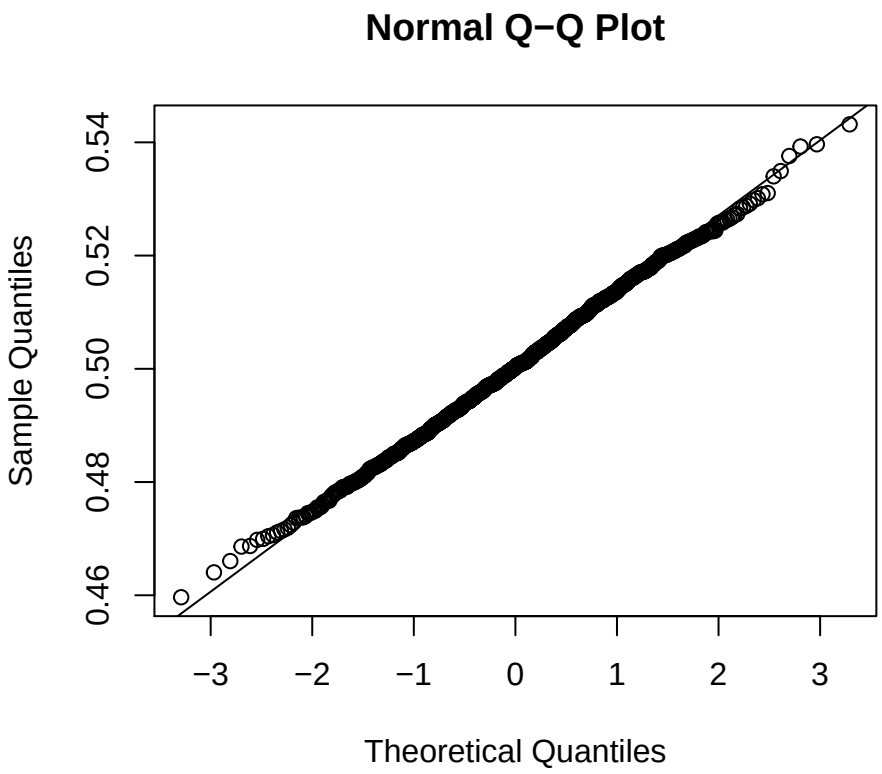
```
n = 500  # sample size in each repetition
m = 1000 # number of repetitions

X = matrix(runif(n*m), nrow = m, ncol = n) # data matrix
means = rowSums(X) / n                     # collect 100 means here

c(mean(means), sd(means)) # mean and sd of the estimates of the mean
```

```
## [1] 0.50047672 0.01297353
```

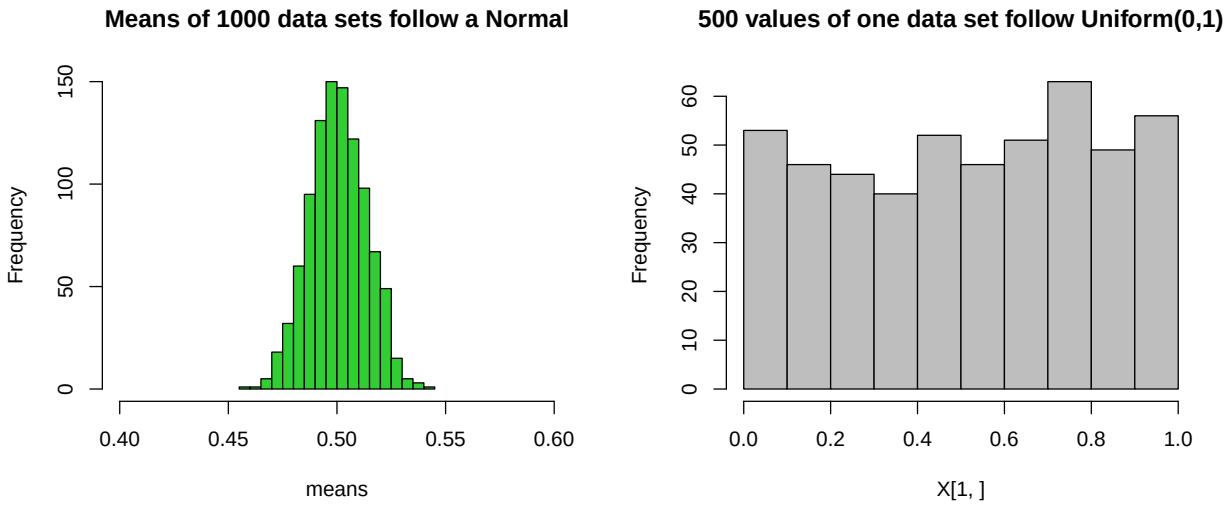
```
# See whether means seem Normally distributed
qqnorm(means)
qqline(means)
```



Indeed, the distributions of means look like Normal according to the QQ-plot. Let's see the histogram of the means, and let's also show a histogram of one of the individual data sets to see that it indeed looked like uniform (and not Normal).

```
par(mfrow = c(1,2))
hist(means, xlim = c(0.4,0.6), breaks = 15, col = "limegreen",
     main="Means of 1000 data sets follow a Normal")

# Plot the first data set just to make sure that the original data look like U(0,1)
hist(X[1,], breaks = 10, col = "gray",
     main="500 values of one data set follow Uniform(0,1)")
```



From the histograms we see how a single data set looks uniformly distributed on [0,1] (right) but how the means of 1000 such data sets is tightly concentrated around 0.5 and looks like Normally distributed. This is CLT in action.

From this result, we can derive the standard Normal approximation to the confidence interval for the mean of values from any distribution. Suppose that we have n values from a distribution D. The endpoints of the 95% confidence interval for the mean of D are

$$\overline{x} \pm 1.96 \times \widehat{s}/\sqrt{n},$$

where $\overline{x}$ is the empirical mean of the n values, $\widehat{s}$ is their empirical standard deviation and 1.96 `=qnorm(1-0.05/2)` is the quantile point from the standard Normal distribution below which probability mass $0.975 = 1 - 0.05/2$ lies.

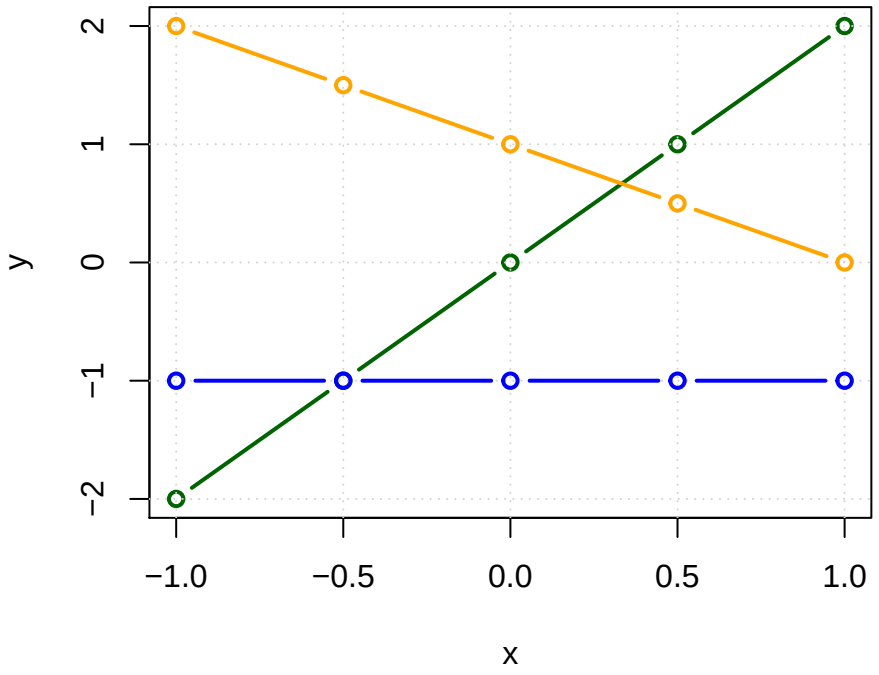
What is a linear relationship?

In this section, we study relationship between two or more variables. First, we establish how to quantify the strength of the linear relationship between continuous variables and then we learn how to use the relationship to predict value of an unknown variable given the values of observed variables.

Mathematically, two variables X and Y are linearly related if there are some numbers a and b such that $Y = a + b \cdot X$. Here, b is the coefficient that links the changes in X to changes in Y: A change of one unit in variable X always corresponds to a change of b units in variable Y. Additionally, a is the value that allows a shift between the ranges of X and Y.

Let's plot three linear relationships with parameters $a = 0, b = 2$ in green $a = 1, b = -1$ in orange and $a = -1, b = 0$ in blue. Let's use 5 points to demonstrate these two lines when x-coordinate is between -1 and 1.

```
n = 5
x = seq(-1, 1, length = n)
y = 0 + 2*x
plot(x, y, t = "b", col = "darkgreen", lwd = 2) #t="b" uses "b"oth lines and points
grid() #make a grid on background
y = 1 + (-1)*x
lines(x, y, t = "b", col = "orange", lwd = 2) # add line to the existing plot
y = -1 + 0*x
lines(x, y, t = "b", col = "blue", lwd = 2) # add line to the existing plot
```



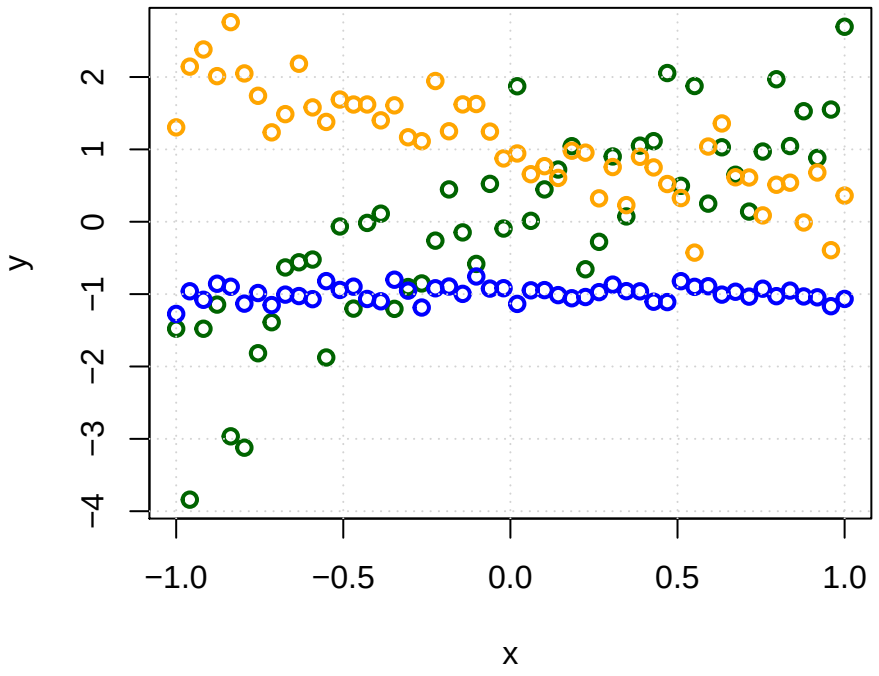
We see that depending on the sign of b, the line is either increasing ($b = 2$, green), decreasing ($b = -1$, orange), or flat ($b = 0$, blue). We call b as **slope** and a as **intercept** of the linear relationship.

Example. Friedewald's formula is an example of a linear relationship. It tells how to estimate LDL cholesterol values from total cholesterol, HDL cholesterol and triglycerides (when measured in mmol/l):

$$\text{LDL} \approx \text{TotalC} - \text{HDL} - 0.45 \cdot \text{TriGly}.$$

In practice, we never observe perfect linear relationships between measurements. Rather we observe relationships that are linear to some degree, and that are further diluted by noise in the measurements. We can model such imperfect linear relationships by adding some Normally distributed random variation on top of a perfect linear relationships of the previous figure. Let's add most noise to the green and least to the blue line. The amount of noise is determined by the standard deviation of the Normal variable that is added on top of the perfect linear relationship, where larger SD means that we are making a more noisy observation of the underlying line.

```
n = 50
x = seq(-1, 1, length = n)
y = 0 + 2*x + rnorm(n,0,0.8)
plot(x, y, t = "p", col = "darkgreen", lwd = 2) #t="b" uses "b"oth lineas and points
y = 1 + -1*x + rnorm(n,0,0.4)
lines(x, y, t = "p", col = "orange", lwd = 2) # add line to the existing plot
y = -1 + 0*x + rnorm(n,0,0.1)
lines(x, y, t = "p", col = "blue", lwd = 2) # add line to the existing plot
grid() #make a grid on background
```



In this Figure, the quality of the linear model as an explanation of the relationship between X and Y varies between the three cases (blue best, green worst). Next we want to quantify such differences.

Correlation

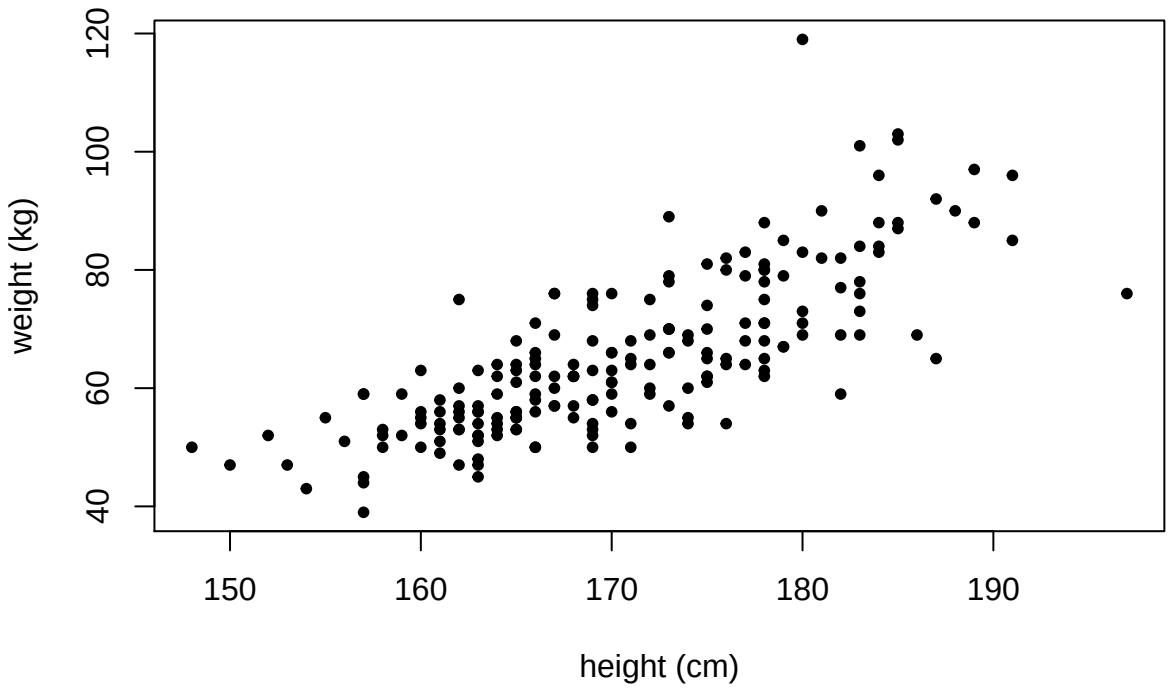
To illustrate the concept of correlation we will use a data set of heights and weights of 199 individuals (88 males and 111 females). This dataset originates from http://vincentarelbundock.github.io/Rdatasets/doc/carData/Davis.html. You can download it from the course webpage. We will call it `measures`. https://vincentarelbundock.github.io/Rdatasets/csv/carData/Davis.csv

```
#measures = read.table("Davis_height_weight.txt",as.is = TRUE, header = TRUE)
measures <- read.csv("https://vincentarelbundock.github.io/Rdatasets/csv/carData/Davis.csv")
measures <- measures[-12,]  # remove observation 12 which is an outlier
head(measures)
```

```
##   X sex weight height repwt repht
## 1 1   M     77    182    77   180
## 2 2   F     58    161    51   159
## 3 3   F     53    161    54   158
## 4 4   M     68    177    70   175
## 5 5   F     59    157    59   155
## 6 6   M     76    170    76   165
```

The last two columns are self-reported values of weight and height. Let's plot weight against height.

```
plot(measures$height, measures$weight, pch = 20, #pch = 20 means a solid round symbol
     ylab = "weight (kg)", xlab = "height (cm)")
```



Unsurprisingly, there is a clear pattern where taller individuals weight more. To quantify the (linear part of the) relationship, we compute a Pearson's **correlation coefficient** between the variables. This happens in two parts: compute the covariance between the variables and scale the covariance by the variability of both variables to get a dimensionless correlation coefficient.

Covariance measures the amount of variation in the variables that is linearly shared between the variables. Technically, it is the average product of deviations from the means of the variables, and from a sample it is computed as

$$\text{cov}(X, Y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y}), \text{ where } \overline{x}, \overline{y} \text{ are the means of } x_i \text{ and } y_i, \text{respectively.}$$

When both X and Y tend to be above their mean values simultaneously, then covariance is positive, whereas the covariance is negative if when X is above its mean, Y tends to be below its mean. In other words, covariance is positive when X and Y tend to increase together and decrease together; covariance is negative when they tend to go to opposite directions. Covariance of 0 says that there is no linear relationship between X and Y. Note that if we compute the covariance between the variable and itself, that is, $X = Y$ in the formula above, the result is simply the variance of that one variable. Thus, covariance is a generalization of the concept of variance to two variables.

Correlation coefficient results when covariance is normalized by the product of the standard deviations of the variables:

$$\text{cor}(X, Y) = \frac{\text{cov}(X, Y)}{\text{SD}(X)\text{SD}(Y)}.$$

Correlation is always between -1 and +1, and it denotes the strength of the linear relationship between the variables. If correlation is +1, then values of X and Y are on a line that has a positive slope and if correlation is -1, then X and Y are on a line that has a negative slope. When correlation is 0, there is no linear association between the variables. (See Figures from Wikipedia.) Note that if correlation between X and Y is 1 it does not necessarily mean that $Y = X$, but only that there is a perfect linear relationship of the form $Y = a + b \cdot X$ for some constants a and $b > 0$, and any such linear relationship leads to the correlation of 1.

Let's compute an estimate $\widehat{r}$ of the correlation between height and weight based on our sample of $n = 199$ observations.

```
r = cor(measures$height, measures$weight)
r
```

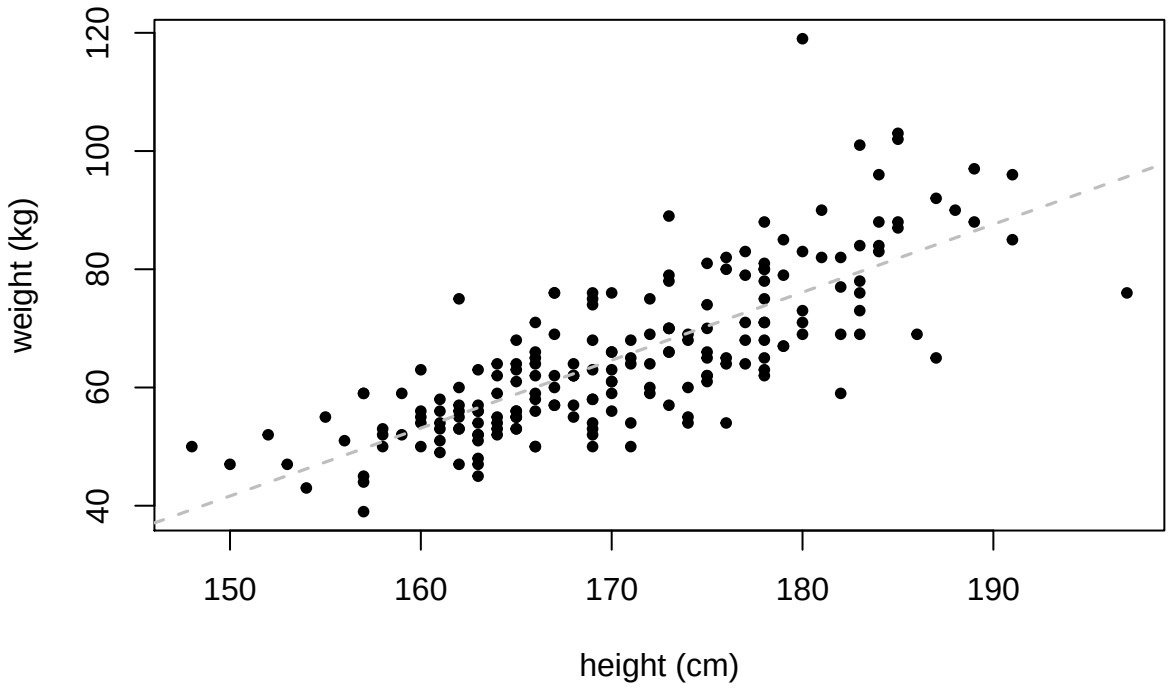
```
## [1] 0.7707306
```

A correlation of ~0.8 is quite high.

Linear regression

In statistics, linear regression is a linear approach for modelling the relationship between response variable and one or more explanatory variables (also known as dependent and independent variables). The case of one explanatory variable is called simple linear regression; for more than one, the process is called multiple linear regression. This term is distinct from multivariate linear regression, where multiple correlated dependent variables are predicted, rather than a single scalar variable (https://en.wikipedia.org/wiki/Linear_regression).

Correlation coefficient r describes the strength of linear relationship between two variables. Both variables are treated symmetrically in definition of r. However, we may also want to utilize the linear relationship to predict the value of Y given that we know the value of X. For example, we may use our observed population above to make a statistical prediction of weights of individuals who are exactly 170 cm tall. From the earlier Figure, that is redrawn below, we see that such individuals have weights roughly in the range from 50 kg to 80 kg.



To characterize the relationship more mathematically, we want to fit a line through the observed data that describes how Y depends on X. (This line is shown in gray in Figure above.)

The **linear regression** model assumes that

$$y_i = a + bx_i + \varepsilon_i,$$

where a (intercept, vakiotermi) and b (slope, kulmakerroin) are model parameters that define the line. With them, we can compute, for each observed value x_i, the value of Y predicted by the line as $\widehat{y}_i = a + bx_i$. We call $\widehat{y}_i$ as the predicted value, or prediction (for x_i). In the linear model, $\varepsilon_i = y_i - \widehat{y}_i$ is the difference between the observed value y_i and the value $\widehat{y}_i$ predicted by the model, and is called a **residual** (for observation i).

Any pair of parameters a and b define one line (namely $y = a + bx$) in X-Y coordinates. Which of them fits the best to the data? In standard linear model, we choose the line that minimizes the sum of the squared

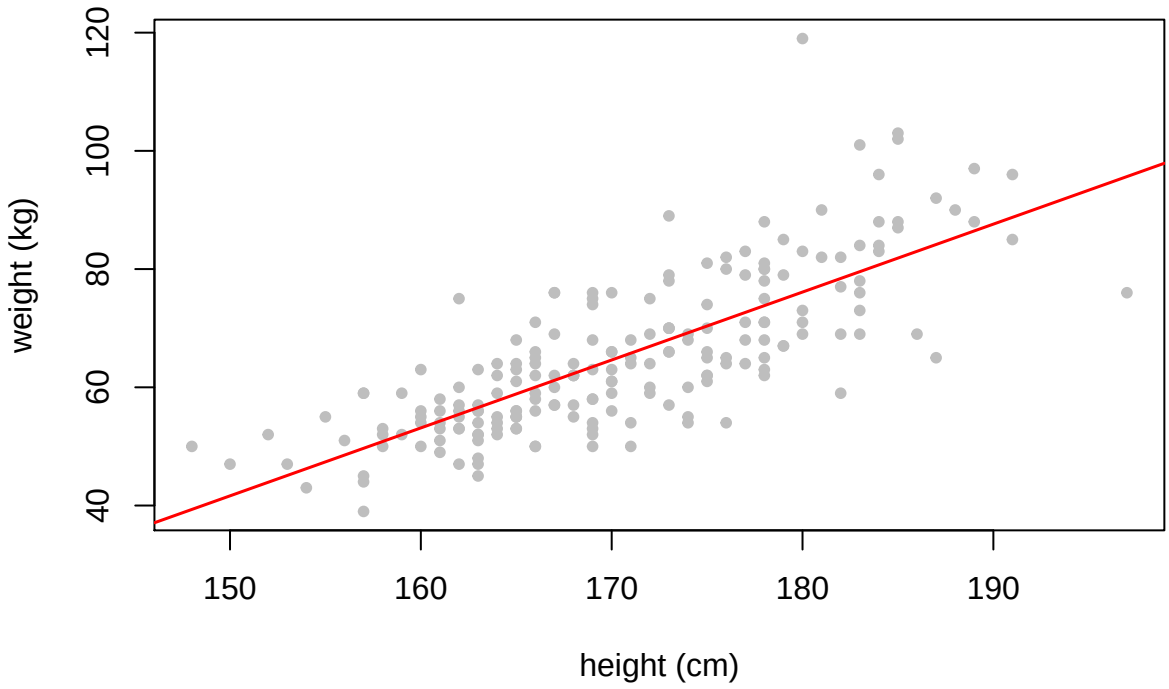
residuals. That is, we choose a and b in such a way that residual sum of squares

$$\text{RSS} = \sum_{i=1}^{n}(y_i - \widehat{y}_i)^2 = \sum_{i=1}^{n}(y_i - (a + bx_i))^2$$

is minimized. This line fits the observed data best in the "least squares" sense: the sum of squared deviations of the observed values from their predictions are minimized. We denote these **least squares estimates** of the parameter values as $\widehat{a}$ and $\widehat{b}$.

In R, the linear model is estimated using `lm(y ~ x)` function where formula `y ~ x` reads "regress y on x" or simply "y on x." When our data is in a data.frame object (like our current `measures`), we can specify the data.frame in the call of `lm()` by parameter `data =` and replace longer expressions such as `measures$height` in the model formula by variable name such as `height`. `lm()` returns a fitted model object that can be saved as a new variable. With that object, for example, the regression line can be added to a figure by `abline(lm.object)` and a summary of the estimated model parameters is given by `summary(lm.object)`. Let's try it out.

```
lm1 = lm( weight ~ height, data = measures) #fit linear regression of weight on height
plot(measures$height, measures$weight, pch = 20,
     ylab = "weight (kg)", xlab = "height (cm)", col = "gray")
abline(lm1, col = "red", lwd = 1.5) #add the regression line
```




```
summary(lm1)
```

```
##
```

```
## Call:
## lm(formula = weight ~ height, data = measures)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -19.650  -5.419  -0.576   4.857  42.887
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -130.74698   11.56271  -11.31   <2e-16 ***
## height         1.14922    0.06769   16.98   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.523 on 197 degrees of freedom
## Multiple R-squared:  0.594,  Adjusted R-squared:  0.592
## F-statistic: 288.3 on 1 and 197 DF,  p-value: < 2.2e-16
```

In output of `summary(lm)`:

- Residuals are the observed vertical differences between the line and the observed `weight` value and here we see a summary of their distribution.
- Coefficients show the least squares parameter estimates with their SEs and P-values (here highly significantly different from 0 with P-values $<$ 2e-16). The slope tells that each cm in `height` corresponds to an average increase of 1.15 kg in `weight`.
- Residual standard error is an estimate for standard deviation of errors (σ parameter in model)
- R-squareds tell how large proportion of the variance in Y the line explains compared to the total variance in Y. Typically, we should be looking at Adjusted Rsquared as it is more reliable than the raw Rsquared if there are many predictors in the model.

In this case, the linear model on height explains almost 60% of the variation in weight and therefore leaves about 40% of variation in weight as something that cannot be explained by (a linear effect of) height only. Note that in the case of simple linear regression with one predictor (here height), Rsquared is exactly the square of the correlation between X and Y.

```
cor(measures$weight, measures$height)^2
```

```
## [1] 0.5940256
```

```
summary(lm1)$r.squared
```

```
## [1] 0.5940256
```

This explains why we say that the correlation coefficient is a measure of strength of a linear association between variables: When $r = \pm 1$, then the linear regression model explains all variation in Y and hence the observations are on a single line in X-Y coordinates.

Let's see which variables we have in the `lm` object by `names()`:

```
names(lm1)
```

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

We could get both the fitted values (`lm1$fitted.values`) as well as residuals (`lm1$residuals`) for each individual from the regression model object. For example, let's find for which individuals the predicted weight is more than 20kg off. We use `abs()` to get absolute value of residuals and `which()` to get the indexes of the samples for which the absolute residual is > 20. NOTE: If there was NAs among the variables used in regression, then by default R removes those individuals from the regression, and therefore the fitted values and residuals would not be present for all individuals who were originally present in the data. If we want to use the regression results at individual level, it is important first to check that we either have the same number of individuals in the regression as we have in the data, or otherwise we need to play with the `na.action` parameter in `lm()`, as will be shown below at the last example of this document.

```
#Let's check that all inds were included in regression rather than had NAs.
#Otherwise, the indexing below would go wrong since 'measures' had more inds than 'lm1'
stopifnot(length(lm1$residuals) == nrow(measures)) #this would crash if condition was not true

ii = which( abs(lm1$residuals) > 20) #returns indexes for which condition is TRUE
cbind(measures[ii, 2:4], lm1$fitted.values[ii], lm1$residuals[ii])
```

```
##     sex weight height lm1$fitted.values[ii] lm1$residuals[ii]
## 21    M    119    180              76.11303          42.88697
## 30    M    101    183              79.56070          21.43930
## 54    M    102    185              81.85914          20.14086
## 97    M    103    185              81.85914          21.14086
## 192   M     89    173              68.06848          20.93152
```

We notice two things. First, there is one outlier case where weight is 119 kg whereas it is predicted to be 76 kg based on the height of 180 cm. Second, all of the worst predictions have happened for males. Indeed, it is unlikely that the same linear regression model would be appropriate for both males and females.

Let's improve the model and add `sex` as an another predictor to the model to allow different mean weights in males and females. Since `sex` is of type "character" with two values ("M" and "F"), R automatically treats it as *factor* in regression model. For factor variables, regression model will essentially give different mean values for different levels of the factor (here the two levels are males and females).

```
lm2 = lm(weight ~ height + sex, data = measures)
summary(lm2)
```

```
##
## Call:
## lm(formula = weight ~ height + sex, data = measures)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -20.131  -4.889  -0.404   5.205  41.490
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -76.63620   15.75543  -4.864 2.36e-06 ***
## height        0.81072    0.09555   8.485 5.24e-15 ***
## sexM          8.21621    1.71726   4.784 3.37e-06 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.086 on 196 degrees of freedom
## Multiple R-squared:  0.6365, Adjusted R-squared:  0.6328
## F-statistic: 171.6 on 2 and 196 DF,  p-value: < 2.2e-16
```

Now our model says that after we account for difference in average weight between males and females, each cm in height corresponds to 0.81 kg in height (it was 1.15 kg when sex wasn't accounted for). Additionally we see that for the same height, a male weights on average 8.2 kg more than a female. This model explains more of the variation in weight than the previous model as the adjusted Rsquared has increased to 63% from 59%. Numerically, the linear predictions are now sex-dependent for the intercept term but not for the slope:

$$\begin{aligned} \text{weight} &= 0.81 \cdot \text{height} - 76.6, \text{ for females,} \\ \text{weight} &= 0.81 \cdot \text{height} - 68.4, \text{ for males} \end{aligned}$$

But are there also differences in the height-weight slope between the sexes? Let's simply fit separate linear models in males and in females and check the slopes.

```
males = (measures$sex == "M") #TRUE/FALSE vector that is TRUE for males and FALSE for females
lmM = lm(weight ~ height, data = measures[males,]) #Use only male rows
lmF = lm(weight ~ height, data = measures[!males,]) #Use only non-male = female rows
```

We can get the estimated parameter values as `lm.object$coeff`.

```
lmM$coeff
```

```
##  (Intercept)       height
## -101.3300503    0.9955981
```

```
lmF$coeff
```

```
## (Intercept)      height
## -45.7084157   0.6229425
```

and confidence intervals by `confint(lm.object, parm = , level = 0.95)`.

```
confint(lmM, "height")
```

```
##            2.5 %   97.5 %
## height 0.6623346 1.328862
```

```
confint(lmF, "height")
```

```
##            2.5 %    97.5 %
## height 0.4254921 0.8203929
```

It seems like slope in females might be considerably smaller than in males. Since the 95%CIs are quite wide, we cannot conclude exactly how large the difference is from these data alone.

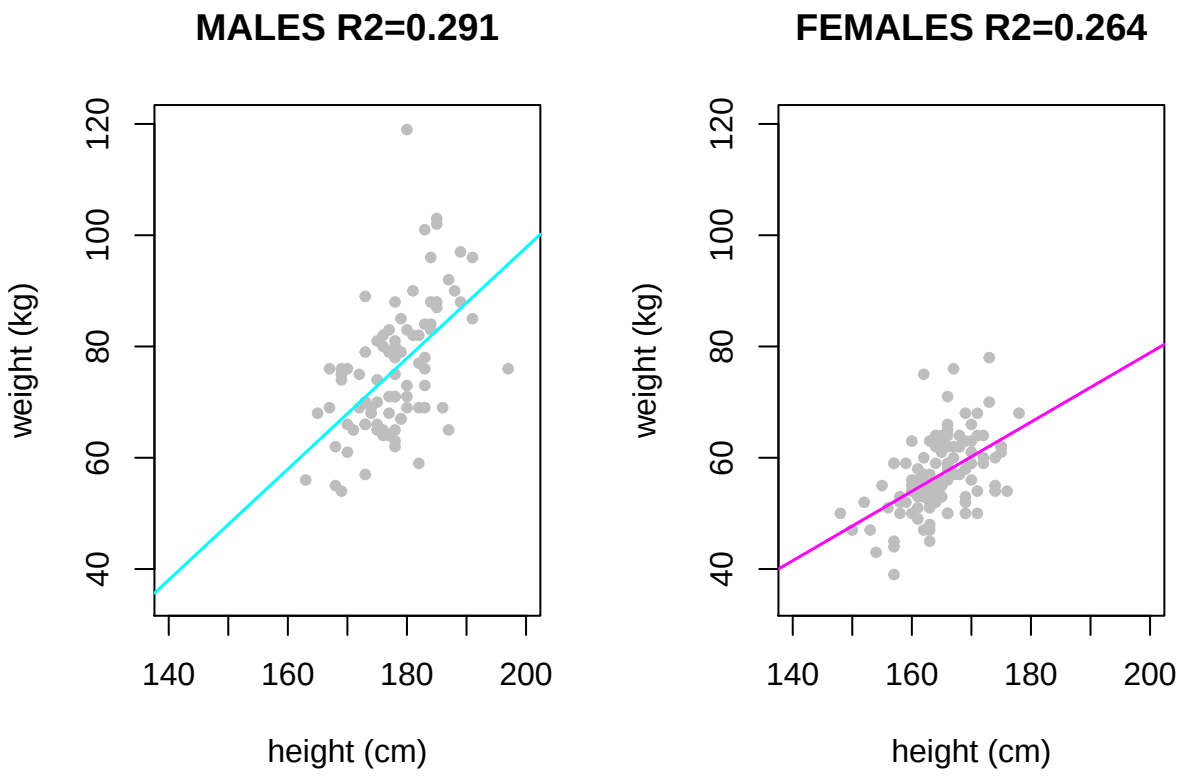
If we wanted to have also SEs and P-values for the coefficients, we could get them from `summary(lm.object)$coeff`, for example:

```
summary(lmM)$coeff
```

```
##                 Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) -101.3300503 29.8616911 -3.393313 1.046009e-03
## height         0.9955981  0.1676432  5.938794 5.921663e-08
```

Finally, let's plot the model fits and put Rsquared values in the titles of the plots to see how much height linearly explains of weight in each sex.

```
par( mfrow = c(1,2) )
plot(measures[males,"height"], measures[males,"weight"], pch = 20,
     ylim = c(35, 120), xlim = c(140, 200),
     ylab = "weight (kg)", xlab = "height (cm)", col="gray")
abline(lmM, col = "cyan", lwd = 1.5)
title(paste0("MALES R2=", signif(summary(lmM)$r.squared,3) ) )
plot(measures[!males,"height"], measures[!males,"weight"], pch = 20,
     ylim = c(35, 120), xlim = c(140, 200),
     ylab = "weight (kg)", xlab = "height (cm)", col="gray")
abline(lmF, col = "magenta", lwd = 1.5)
title(paste0("FEMALES R2=",signif(summary(lmF)$r.squared, 3) ) )
```



We see a dramatic decrease in variance explained after we have adjusted the analyses for sex. This is because there is a strong difference in distributions of both height and weight between males and females, and therefore a large part of the explanatory power of the orginal model was the effect of sex. After sex is accounted for, height explains anymore about 25-30% of variation in weight. This shows how important the additional variables, also called covariates, can be for the interpretation of the regression model parameters.