STATMED Lecture 6: Linear relationship

Matti Pirinen

20.8.2024

In this lecture, we study relationship between two or more variables. First, we establish how to quantify the strength of a 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 the observed variables.

What is a linear relationship?

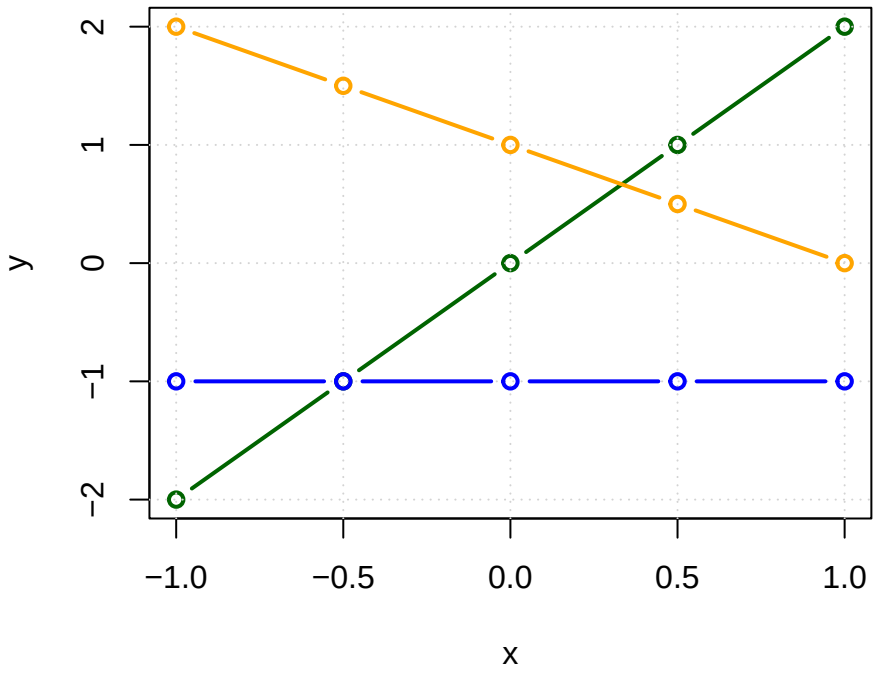
Mathematically, two variables X and Y are linearly related if there are some numbers a and b such that $Y = a + b \cdot X$. Here, b is the coefficient that links the changes in X to changes in Y: A change of one unit in variable X always corresponds to a change of b units in variable Y. Additionally,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

Let's plot three linear relationships with different parameters

- Green: $a = 0, b = 2$,
- Orange: $a = 1, b = -1$,
- Blue $a = -1, b = 0$.

Let's use five points to demonstrate these three lines when the x-coordinate takes values 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() #put a grid on the 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
```



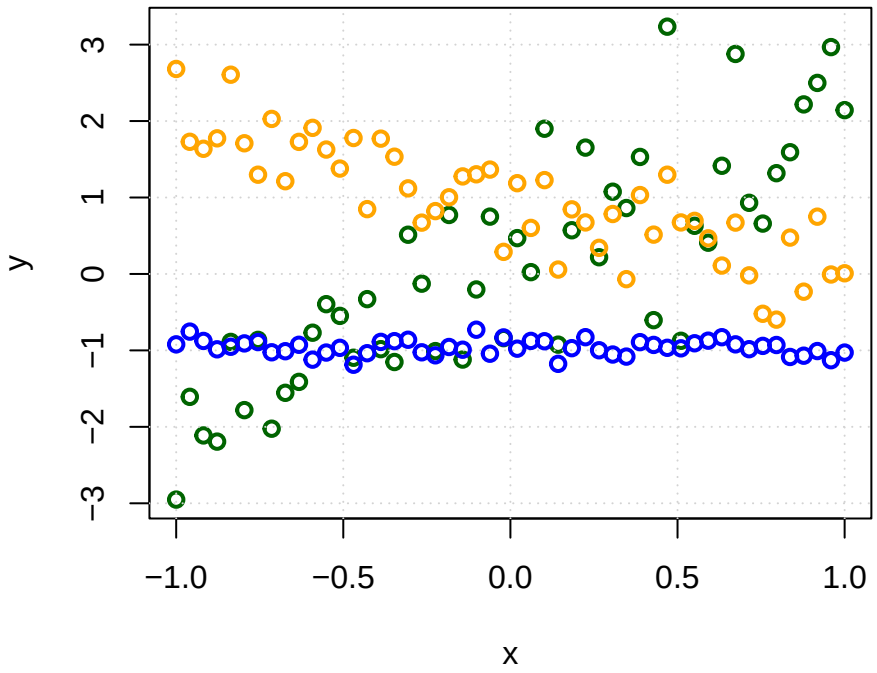
We see that depending on the sign of b, the line is either increasing ($b = 2$, green), decreasing ($b = -1$, orange), or flat ($b = 0$, blue). We call b the **slope** (kulmakerroin) and a the **intercept** (vakiotermi) of the linear relationship.

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 all are 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 the 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 (SD) 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 linear relationship. Here we use SDs of 0.8 (green), 0.4 (orange) and 0.1 (blue).

```
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)
grid()
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
```



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

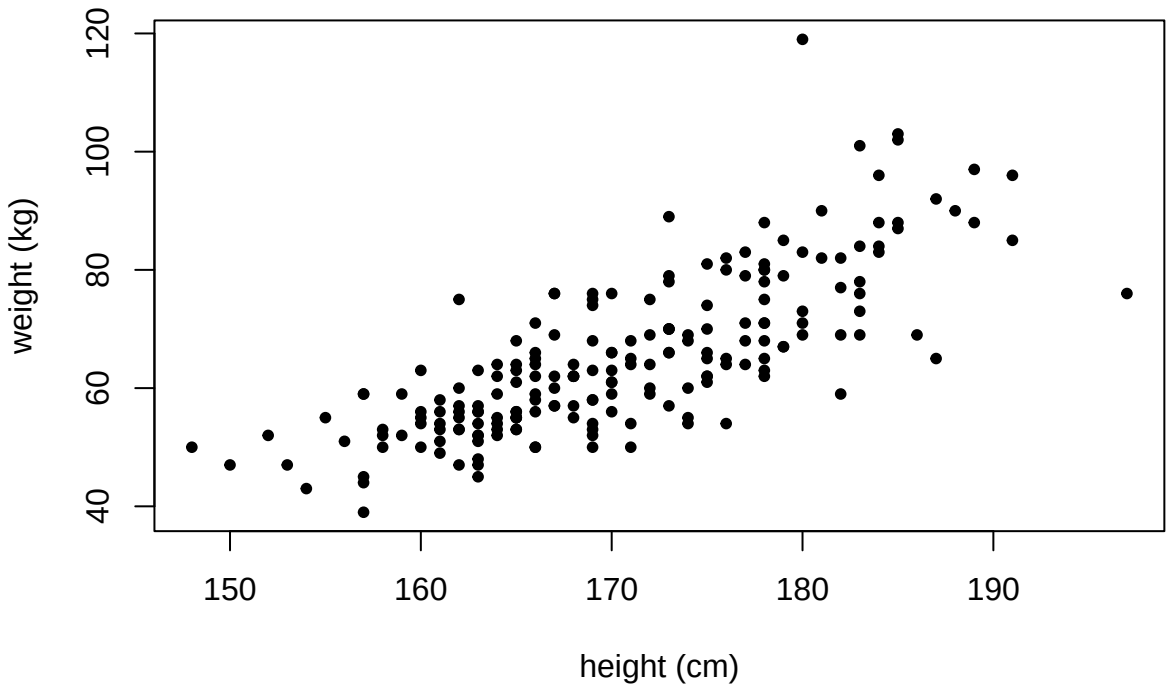
We read in a data set of heights and weights of 199 individuals (88 males and 111 females). This dataset originates from http://vincentarelbundock.github.io/Rdatasets/doc/carData/Davis.html. You can download it from the course webpage. We will call it `measures`.

```
measures = read.table("Davis_height_weight.txt", as.is = T, header = T) #using 'T' as a shorthand for '
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

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

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 the 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 from one variable to two variables.

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

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

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, on the other hand, 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) #correlation is symmetric cor(weight,height) = cor(height,wei
r
```

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

A correlation of ~0.8 is quite high. To get confidence intervals for the correlation coefficient we can use `r.con()` function from the `psych` package. (Install `psych` package with command `install.packages("psych")` first if Rstudio hasn't done it automatically for you.) Let's see what the 95%CI is.

```
n = nrow(measures) #sample size is the number of rows of y
library(psych) #DO FIRST: install.packages("psych")
r.res = r.con(r, n, p = 0.95, twotailed = TRUE) #returns lower and upper bounds of CI
c(r = r, low95CI = r.res[1], up95CI = r.res[2])
```

```
##         r   low95CI    up95CI
## 0.7707306 0.7074835 0.8217303
```

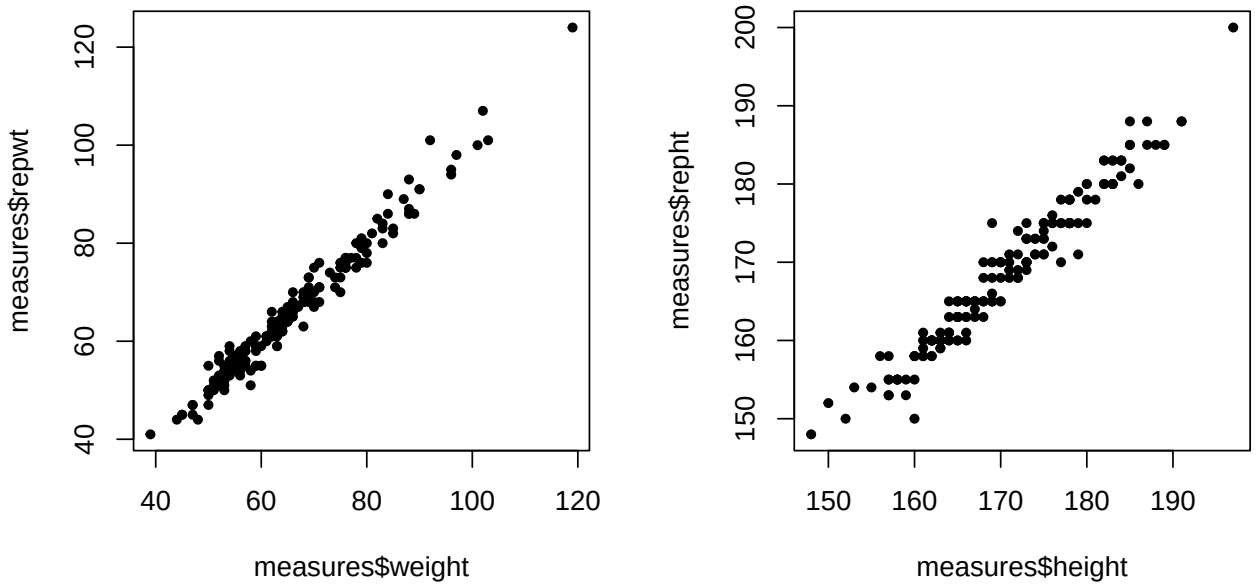
In this case, `r.con()` gives 95%CI as (0.707, 0.822), so we can conclude that the height-weight correlation is about 70-80% in this population.

Examples 6.1

1. Make a scatter plot of `weight` on x-axis and `repwt` on y-axis. Make a similar plot for `height` and `repht`. Do the reported values seem highly correlated with the measured values?

Let's plot them next to each other by splitting plotting area into 1 row and 2 columns by `par(mfrow = c(1,2))`.

```
par(mfrow = c(1,2))
plot(measures$weight, measures$repwt, pch = 20) #pch = 20 means a solid round symbol
plot(measures$height, measures$repht, pch = 20)
```



The reported values seem highly correlated with the measured ones since the two are approximately on a line.

2. Compute correlation between `weight` and `repwt` (self-reported weight) and between `height` and `repht` (self-reported height).

```
#First try will give NA
cor(measures$weight, measures$repwt)
```

```
## [1] NA
```

We got NA, because there are some individuals who only have one of these measures available, and the other is NA. It then follows that the whole computation of correlation becomes NA. Let's only use the completely observed samples with respect to these two measures, by specifying `use = "complete.obs"`.

```
cor(measures$weight, measures$repwt, use = "complete.obs")
```

```
## [1] 0.9858579
```

```
cor(measures$height, measures$repht, use = "complete.obs")
```

```
## [1] 0.9757704
```

We see that the correlations are very near +1, so they are very high, as expected from the highly linear relationship in the Figure above.

3. Compute a **correlation matrix** of the four variables `weight`, `height`, `repwt` and `repht` by giving `cor()` function those four columns of the data matrix as input. Note that you need to use `use = "pairwise"` in `cor()` to get rid of NA values.

Let's pick the correct columns from `measures` by their names. If we would now `use = "complete.obs"`, then it would compute each correlation value by using only those samples that have all the four variables measured. For us, it is enough that it uses complete observations for each pair of variables when computing correlation for that pair. Hence we set `use = "pairwise"`.

```
M = measures[,c("weight","height","repwt","repht")]
cor(M, use = "pairwise")
```

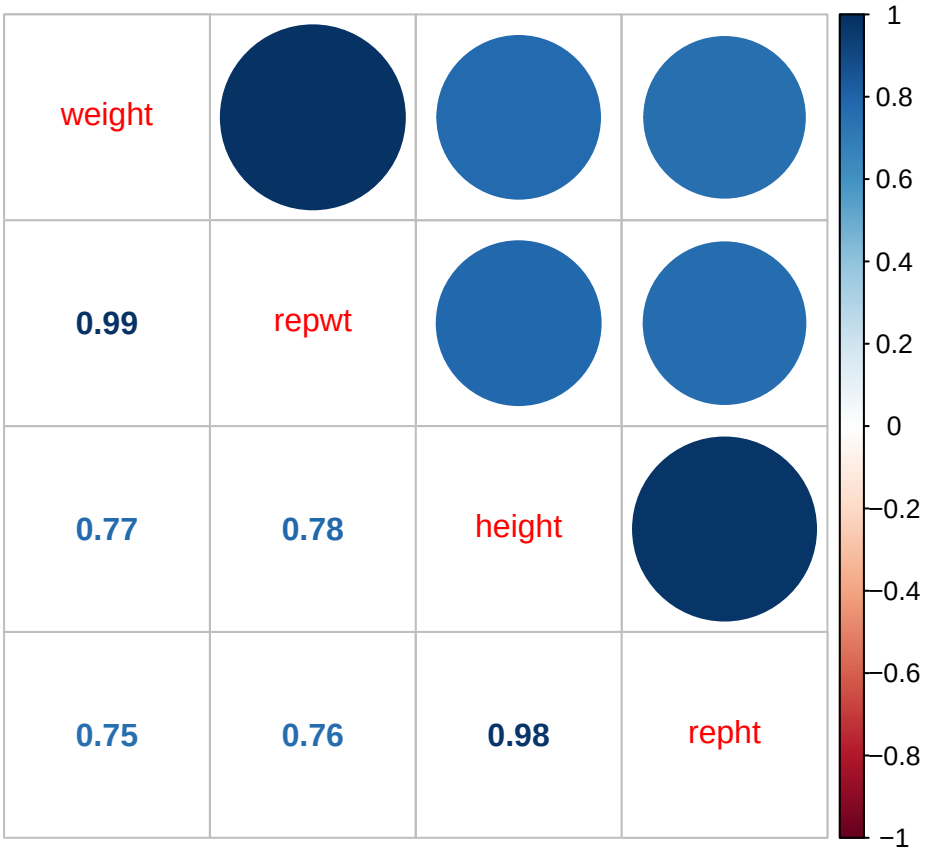
```
##           weight    height     repwt     repht
## weight 1.0000000 0.7707306 0.9858579 0.7515924
## height 0.7707306 1.0000000 0.7820177 0.9757704
## repwt  0.9858579 0.7820177 1.0000000 0.7613189
## repht  0.7515924 0.9757704 0.7613189 1.0000000
```

4. Visualize the correlation matrix by `corrplot` package. (Install it by `install.packages("corrplot")`.)

```
#install.packages("corrplot") #Do this first
library(corrplot)
```

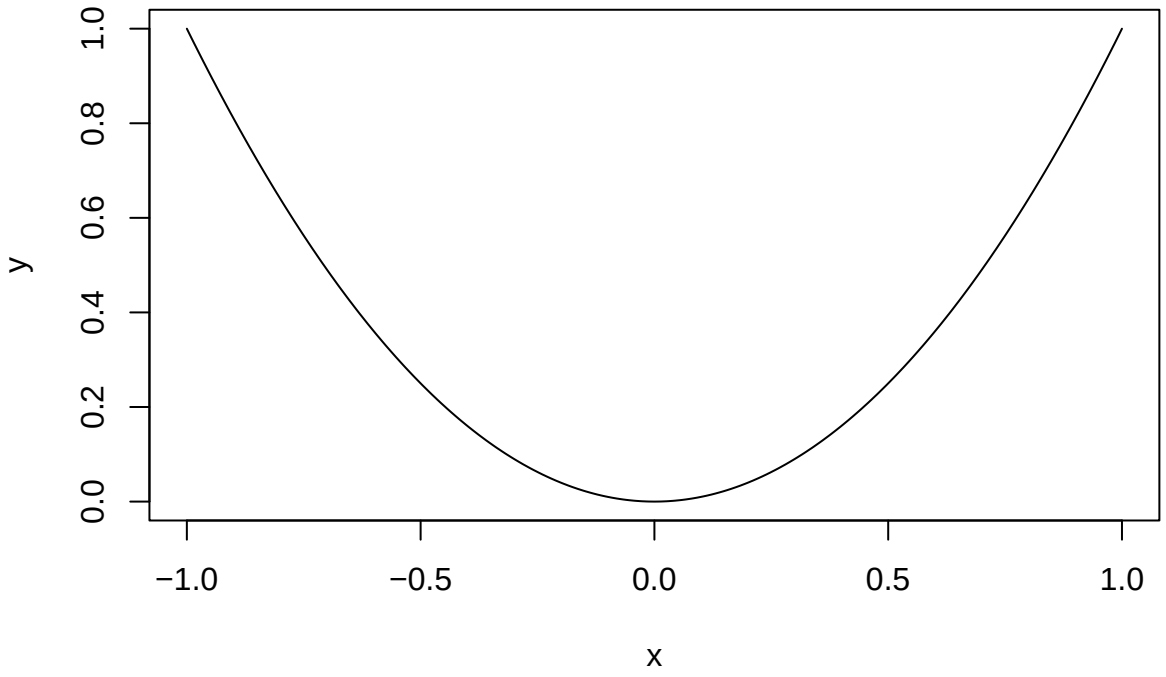
```
## corrplot 0.94 loaded
```

```
corrplot.mixed(cor(M, use = "pairwise"), order = "hclust")
```



5. Plot the curve $y = x^2$ in the range $x \in (-1, \ldots, 1)$ using 1000 equally spaced values for x. Guess what would be the correlation between x and y. Compute correlation.

```
x = seq(-1, 1, length = 1000)
y = x^2
plot(x, y, t = "l")
```



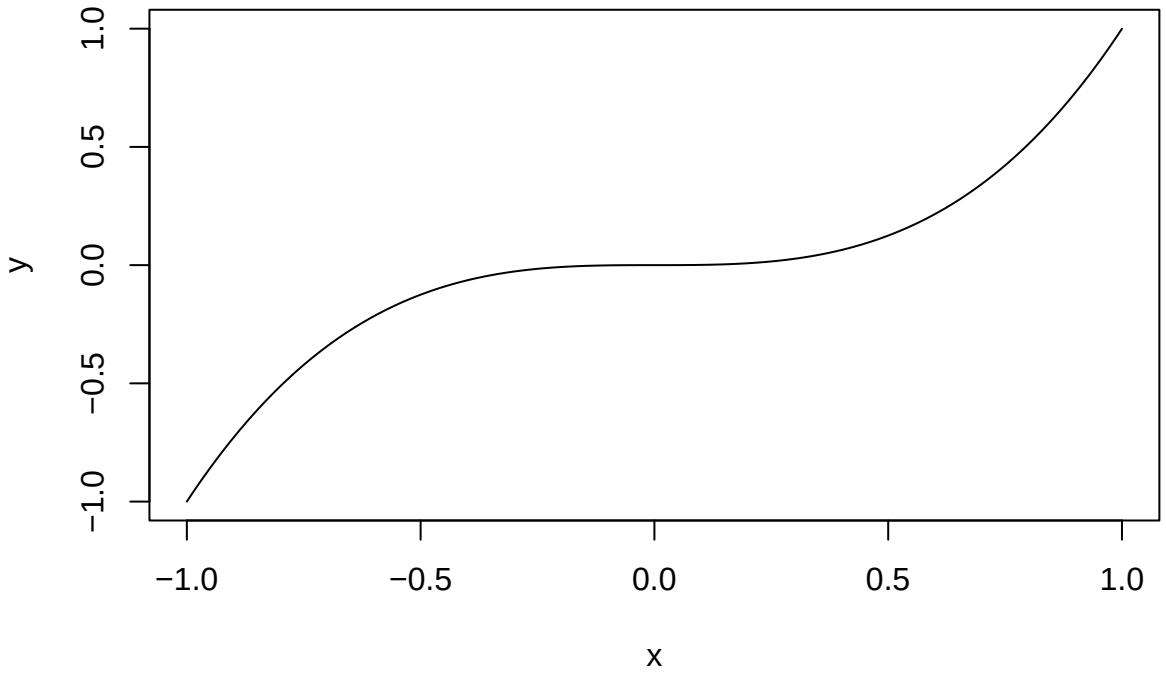

```
cor(x, y)
```

```
## [1] -3.261627e-17
```

We can observe that on the positive side of the x-axis the correlation is positive but on the negative side of the x-axis the correlation is the same in absolute value but would now be negative. Hence, in total correlation might be 0, and this is indeed the case. (1e-17 = 0.00000000000000001 is 0 in practice.)

6. Plot the curve $y = x^3$ in the range $x \in (-1, \ldots, 1)$ using 1000 equally spaced values for x. Guess what would be the correlation between x and y. Compute correlation.

```
x = seq(-1, 1, length = 1000)
y = x^3
plot(x, y, t = "l")
```



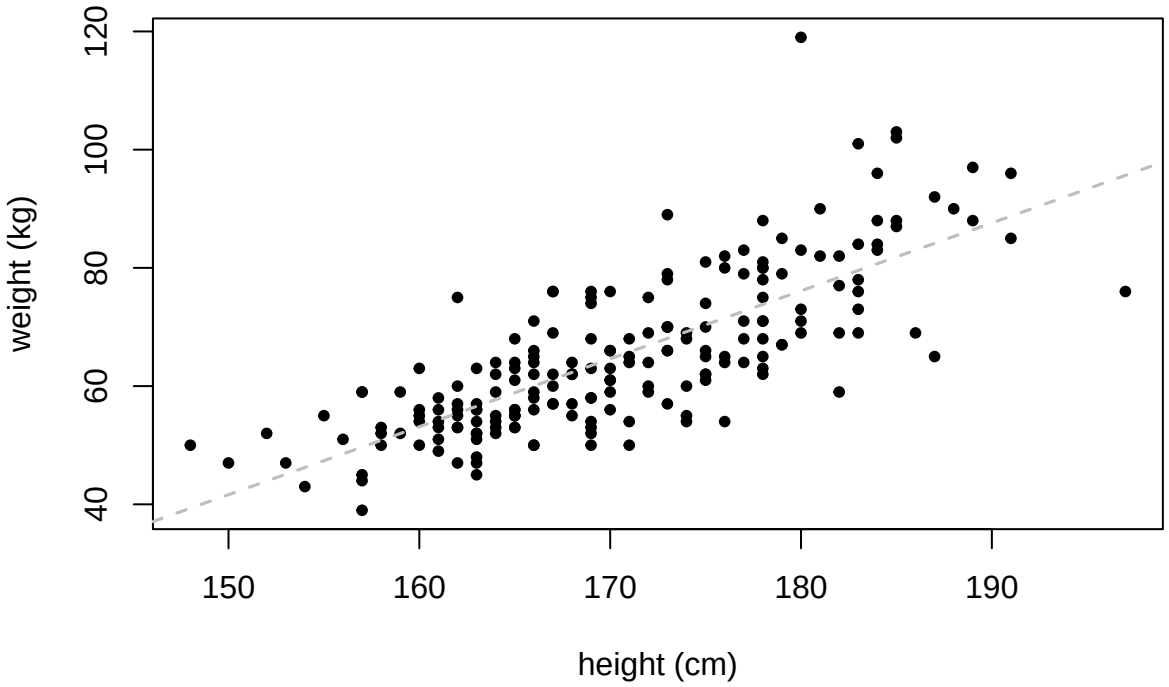

```
cor(x, y)
```

```
## [1] 0.9165158
```

Now the correlation is clearly positive, about 0.917.

Linear regression

Correlation coefficient r describes the strength of a linear relationship between two variables. Both variables are treated symmetrically in the 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).

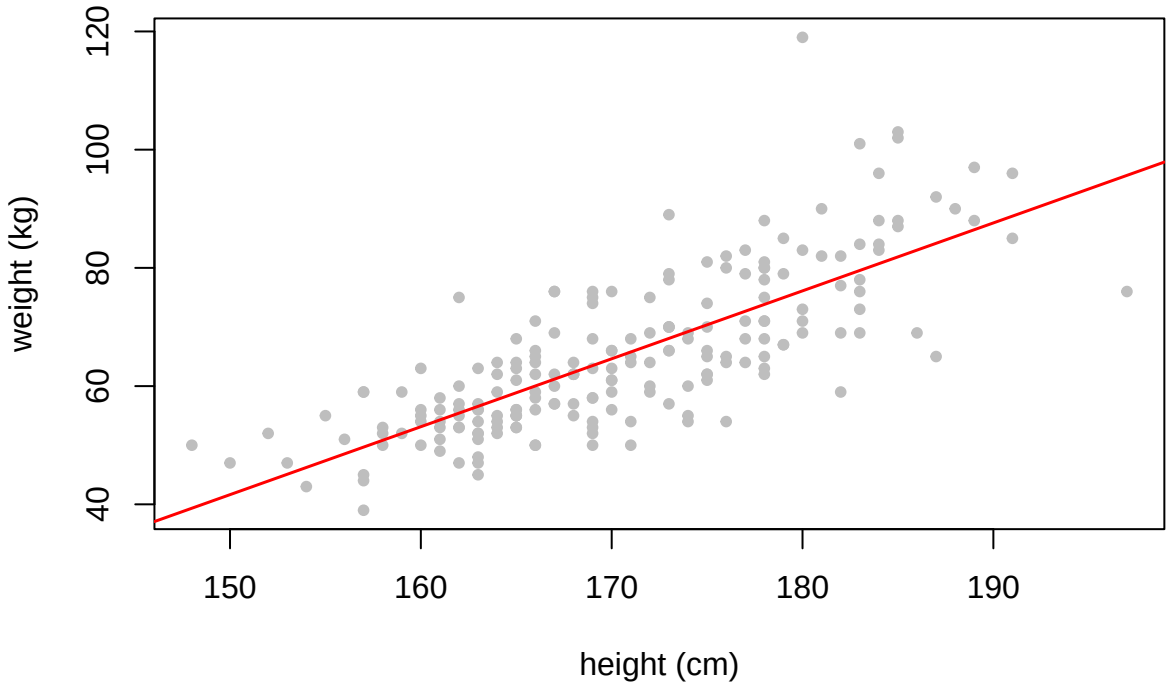
Any pair of parameters a and b define one line (namely $y = a + bx$) in the X-Y coordinates. Which of them fits the best to the data? In standard linear model, we choose the line that minimizes the sum of the squared residuals. That is, we choose a and b in such a way that residual sum of squares

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

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.

```
lm.1 = 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(lm.1, col = "red", lwd = 1.5) #add the regression line
```




```
summary(lm.1)
```

```
##
## 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 estimates the standard deviation of the error terms ε_i in the model, and in this simple case, it matches with the empirical standard deviation of the residuals.

- R-squareds tell how large proportion of the variance in Y is explained by the linear model compared to the total variance in Y. Typically, we should be looking at Adjusted R-squared as it is more reliable than the raw R-squared, 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), R-squared is exactly the square of the correlation between X and Y.

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

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

```
summary(lm.1)$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(lm.1)
```

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

We could get both the fitted values (`lm.1$fitted.values`) as well as residuals (`lm.1$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 'lm.1'
length(lm.1$residuals) == nrow(measures)
```

```
## [1] TRUE
```

```
ii = which( abs(lm.1$residuals) > 20) #returns indexes for which condition is TRUE
data.frame(measures[ii, 2:4], fitted = lm.1$fitted.values[ii], residual = lm.1$residuals[ii])
```

```
##     sex weight height   fitted residual
## 20    M    119    180 76.11303 42.88697
## 29    M    101    183 79.56070 21.43930
## 53    M    102    185 81.85914 20.14086
## 96    M    103    185 81.85914 21.14086
## 191   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 a single linear regression model would be optimal simultaneously 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). Technically, this is done by setting different intercept terms to the groups defined by the factor.

```
lm.2 = lm(weight ~ height + sex, data = measures)
summary(lm.2)
```

```
##
## 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 R-squared 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
lm.m = lm(weight ~ height, data = measures[males,]) #Use only male rows
lm.f = lm(weight ~ height, data = measures[!males,]) #Use only non-male = female rows
```

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

```
lm.m$coeff
```

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

```
lm.f$coeff
```

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

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

```
confint(lm.m, "height")
```

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

```
confint(lm.f, "height")
```

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

It seems like slope in females, 0.62 (0.43,0.82), might be considerably smaller than in males, 1.00 (0.66,1.33). Since the 95%CIs are quite wide, we cannot conclude very accurately 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:

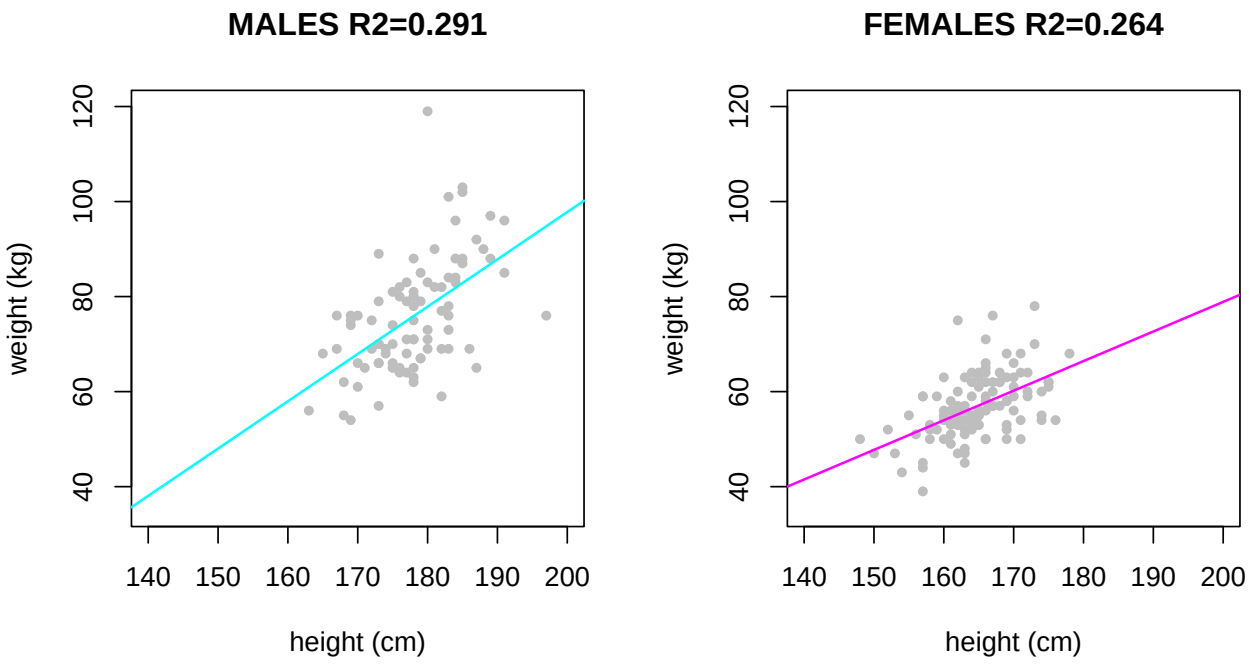
```
summary(lm.m)$coeff
```

```
##                 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
```

In the P-value computations, the null hypothesis has been that the value of the coefficient is 0. Thus, a small P-value says that we have a good reason to suspect that the value of the coefficient is not 0.

Finally, let's plot the fitted models and put the R-squared 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(lm.m, col = "cyan", lwd = 1.5)
title(paste0("MALES R2=", signif(summary(lm.m)$r.squared,3) ) )
plot(measures[!males,"height"], measures[!males,"weight"], pch = 20,
     ylim = c(35, 120), xlim = c(140, 200),
     ylab = "weight (kg)", xlab = "height (cm)", col="gray")
abline(lm.f, col = "magenta", lwd = 1.5)
title(paste0("FEMALES R2=",signif(summary(lm.f)$r.squared, 3) ) )
```



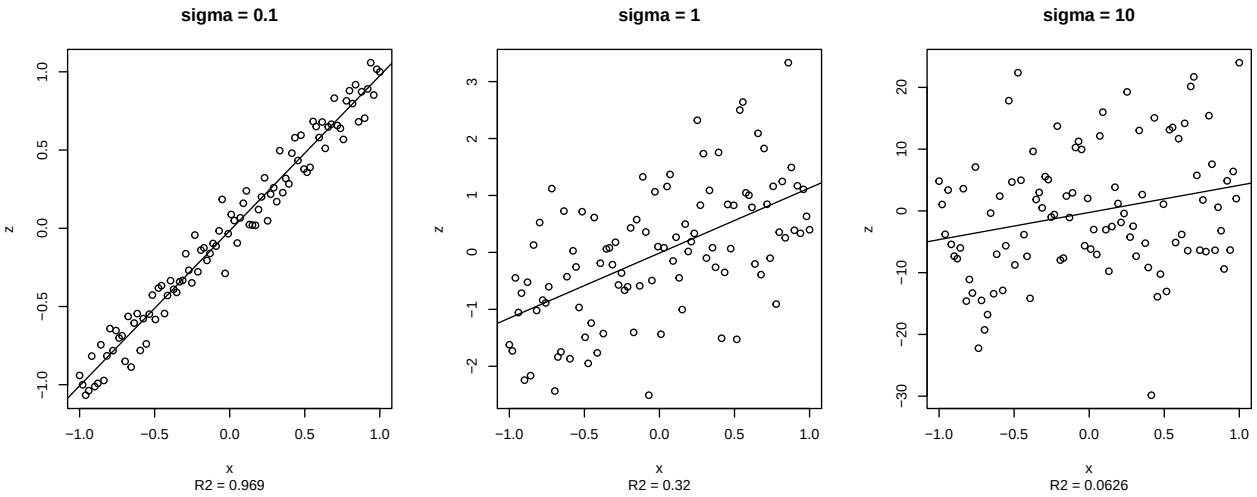
We see a dramatic decrease in variance explained after we have adjusted the analyses for sex. This is because there is a strong difference in distributions of both height and weight between males and females, and therefore a large part of the explanatory power of the orginal model was the effect of sex. After sex is accounted for, height explains "only" 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. We'll talk more about the effects of *covariate adjustment* in the next lecture.

Example 6.2

1. Generate 100 equally spaced values for x from interval $(-1, \ldots, 1)$ and generate corresponding values for z using `rnorm(100, x, sigma)` where `sigma` takes sequentially values of 0.1, 1 and 10. For each value of `sigma`, plot the points in X-Y coordinates, fit the linear model and add the fitted line to the figure. Use `summary()$r.squared` command to see how much X linearly explains of variation in Z in each data set. (E.g. print R-squared as a subtitle to each plot.)

```
n = 100
x = seq(-1, 1, length = n)
par(mfrow = c(1,3))
for(sigma in c(0.1, 1, 10)){
```

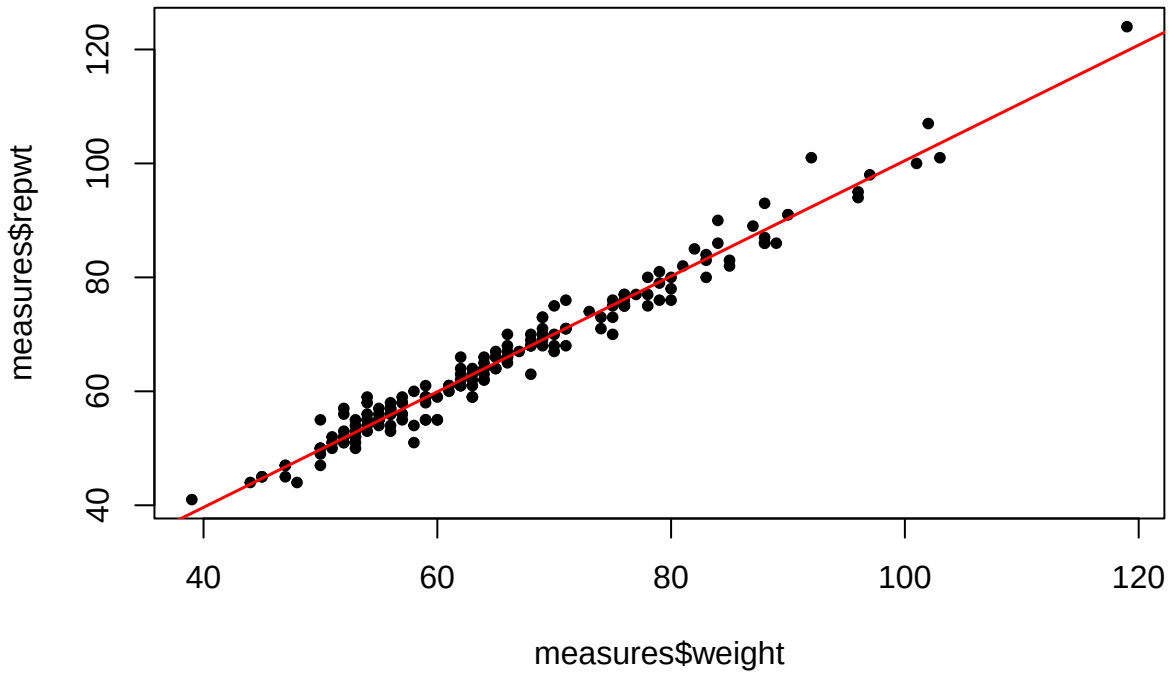
```
  z = rnorm(n, x, sigma)
  lm.1 = lm(z ~ x)
  plot(x, z, main = paste("sigma =",sigma),
       sub = paste("R2 =",signif(summary(lm.1)$r.squared,3))) #signif(,3) = 3 significant digits
  abline(lm.1)
}
```



This example shows that when Z becomes a more noisy measurement of X, then the R-squared value drops towards 0 because X can explain less and less of the total variation of Z. (Note that the Y-axis scales are also very different in the three plots because the range of Z values grow with the amount of added noise.)

2. Make a scatter plot of `weight` on x-axis and `repwt` on y-axis. Fit a linear model regressing `repwt` on `weight`. Add the regression line to the Figure. What is the slope of the model? What is the value of `repwt` for individual 199 and what is the fitted value of `repwt` based on the model for individual 199? What is the residual error for individual 199? Note that `repwt` has some missing values and we will use `na.action = "na.exclude"` in `lm()` to account for that.

```
plot(measures$weight, measures$repwt, pch = 20)
lm.wt = lm(repwt ~ weight, data = measures, na.action = "na.exclude") #Note 'na.action'
abline(lm.wt, col = "red", lwd = 1.5)
```




```
summary(lm.wt)$coeff[2,1] #slope
```

```
## [1] 1.01413
```

```
c(length(lm.wt$residuals), nrow(measures)) #how many individuals were included out of 199?
```

```
## [1] 182 199
```

Now the residuals are present only for 182/199 individuals so we would go wrong if we applied index 199 directly on `lm.wt$residuals`. Instead, thanks to `na.action = "na.exclude"`, we can get correctly indexed residuals and predicted values by functions `residuals()` and `fitted()` that add NAs to appropriate individuals.

```
measures$repwt[199] #value of repwt for last individual 199
```

```
## [1] 81
```

```
fitted(lm.wt)[199] #predicted value for 199
```

```
##      199
## 79.18826
```

```
residuals(lm.wt)[199] #residual for 199
```

```
##      199
## 1.811735
```