MSG400-TMS150

Stochastic data processing and simulation 2024

Recap of topics in Linear Regression

Umberto Picchini

Welcome to the course! In this document we do a recapitulation of linear regression topics.

Some of you may have learned notions of linear regression, with pen-and-paper exercises, in previous courses (e.g. MSG110 Sannolikhetsteori, MVE302-MVE395 Sannolikhet och Statistik). However you may have not experimented much with statistical software.

The goal of this recap is to perform a recollection of some results and then show how to obtain those with the R software. After this is done, we will look at something quite interesting, that is: how do we make sure that we do not introduce "too little" or "too much" information in the model? That is we will touch upon the concept of model complexity and try to assess whether we should opt for a more complex or a simpler model.

Assignment A1: this first assignment is intentionally light; future ones will be a bit more demanding. To pass A1 you must solve all the exercises given in this document. Then do the following:
preferably, present your solutions to a teaching assistant (TA) during labs. This occasion must be booked, see the Canvas main page. At the presentation the TA will decide if what you did is enough or if you need more work. If you could not attend a presentation occasion, or if your work was deemed insufficient, (i) you will have to write a short document with your reasoning and answers and upload it to Canvas, either by the "recommended deadline" or by the final deadline. This short document does not have to be written in LaTeX (but it's ok if you do). (ii) **You will also have to upload all your R codes**.

A1 is graded as "passed/not passed". Notice the "recommended deadline" on the Canvas page. **Remember: this is individual, not group work. Plagiarism check will be carried out.**

1 A simple linear regression model: cars data

We are interested in understanding how some variable y varies, on average, in dependence of another variable x. We call y the **response variable** and x a **covariate**. This means that we are interested in the expected value $E(y|x)$, which we read as "the expected value of y conditional to x". We will need to specify a model

$$E(y|x) = f(x)$$

for some function f. This equation shows the relation between the *expected response* $E(y|x)$ and the "covariate" x, through a deterministic function $f(\cdot)$.

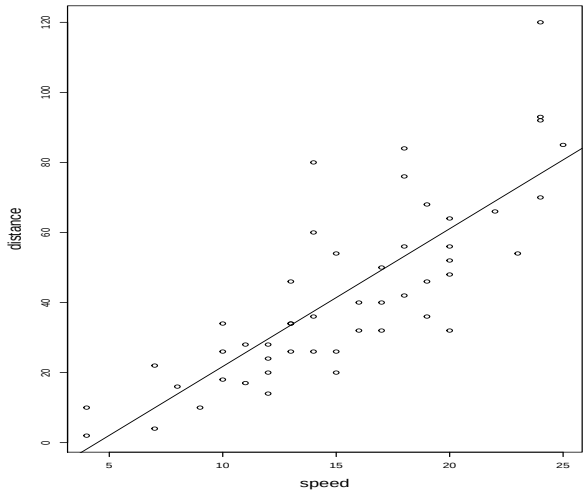

Figure 1: Linear fit.

We consider the "cars" dataset (included in the R software installation). The data includes the speed (mph) of 50 cars and the distances (ft) taken to stop the car running at the given speed. Note that the data was recorded in the 1920s.

In the scatterplot in Figure 1, we have distances vs speed. What do we see? Certainly a positive association. Shall we model the distance using a linear model based on the speed alone, or use nonlinear transformation of the speed, or...?

Example: suppose we wish to consider the model $E(dist|speed) = \beta_0 + \beta_1 speed$. This means we assume $f(x) = \beta_0 + \beta_1 x$ with $x = speed$ and $y = distance$.

Notice, in literature using $f(x) = \beta_0 + \beta_1 x$ is called *simple linear regression*, that is y is let depend on a single covariate x, instead of several covariates. We can obtain least-squares estimators for the intercept β_0 and slope β_1, so that our estimated expected distance is $\hat{E}(dist|speed) = -17.58 + 3.93 speed$. We use the "hat" $\hat{}$ to denote estimated quantities. In this case $\hat{\beta}_0 = -17.58$ and $\hat{\beta}_0 = 3.93$. This line is in Figure 1.

We could also assume that the model we want to fit is quadratic: $E(dist|speed) = \beta_0 + \beta_1 speed + \beta_2 speed^2$, and the fit is in Figure 2.
And why not, try a polynomial of order 5: $E(dist|speed) = \beta_0 + \beta_1 speed + \beta_2 speed^2 + ... + \beta_5 speed^5$, see Figure 3.
Now, I can hear you wondering "but then polynomials of order larger than one are not linear models!!!".

So what do we mean with *linear model*?
What we care for is that the expected response $E(y|x)$ is *linear in the parameters*. And clearly so far we have seen models that do have such linearity. Even if we consider $E(y|x) = \beta_0 + \beta_1 \log x$, this is still a linear model, because it is linear in β_0 and β_1. So the polynomial models above are linear models.

The above was to clarify that a linear model does not need to look like a straight line, well, unless you decide to "recode" your variables. For example, we can always do something like $z = \log(x)$ and then write $E(y|z) = \beta_0 + \beta_1 \log x = \beta_0 + \beta_1 z$, so if we plot y vs z this results in a line, but then again, recoding the problem in terms of z is not necessary.

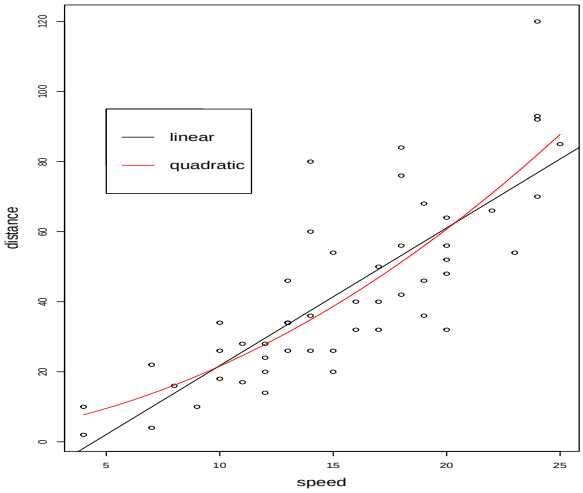

Figure 2: Linear and quadratic fit.

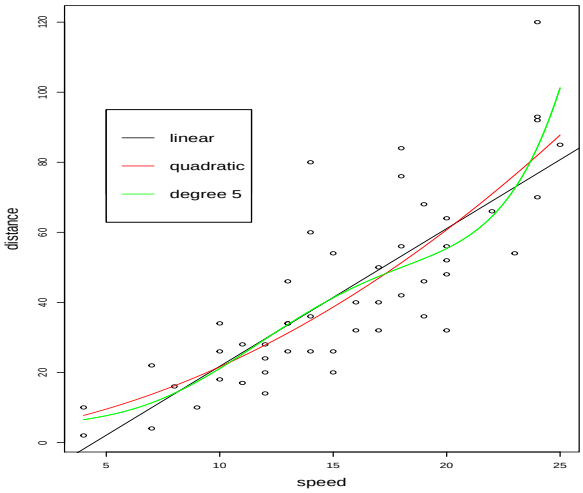

Figure 3: Linear, quadratic and a degree-five polynomial fit.

All the above is about modelling the expectation of the variable of interest y, that is writing $E(y|x)$. Then how do we express the actual data for y? Well, in the simplest type of linear regression analysis, so called OLS (ordinary least squares), our observed y is written as

$$y = E(y|x) + \epsilon = f(x) + \epsilon = \text{“}regression\text{”} + \text{“}error\text{”}$$

with some assumptions we are going to specify. Say we have collected n pairs of values $(x_i, y_i)_{i=1,...,n}$, then we can write

$$y_i = E(y_i|x_i) + \epsilon_i = f(x_i) + \epsilon_i, \qquad i = 1, ..., n$$

and in OLS we assume that each ϵ_i is a random variable having zero mean ($E(\epsilon_i) = 0$), constant variance ($Var(\epsilon_i) = \sigma^2$), and that errors are uncorrelated (zero covariance $Cov(\epsilon_i, \epsilon_j) = 0$ when $i \neq j$ and $Cov(\epsilon_i, \epsilon_j) = \sigma^2$ otherwise, $i, j = 1, ..., n$).

In turn, this implies that in OLS $E(y_i) = f(x_i)$, that $Var(y_i) = \sigma^2$ and that the y_i are uncorrelated. These assumptions imply that the y_i which we have observed (data) are nothing but a random sample “generated” by a probabilistic mechanism which has mean $f(x)$ and variance σ^2, both unknown.

2 Parameters estimation and interpretation

Here we assume the following simple linear regression:

$$E(distance|speed) = \beta_0 + \beta_1 \cdot speed.$$

We now talk of parameter estimation. We first discuss a real data problem and then we will refresh your knowledge on how the computations are performed.

- The slope β_1 is the variation in the expected response when x increases by 1 unit. Therefore for the `cars` dataset, the estimate $\hat{\beta}_1 = 3.93$ means that we expect an increase in the stopping distance of 3.93 ft when the speed increases by 1 mph (mile per hour).
- the intercept β_0 is the value of the expected response when $x = 0$. In the cars dataset we have $\hat{\beta}_0 = -17.6$...an expected *negative* distance when speed is zero??! Yes this is because we do not have observations for speed close to zero, so our model is imprecise in that region (it would probably decrease nonlinearly towards zero if we were to take more measurements around the 0 abscissa). **Always be careful when extrapolating outside the observational interval!** In this case, be careful with making predictions with $speed < 4.5$ or $speed > 25$.

Do it with R!
It is useful to read this part while browsing the script `demo_cars.R`.

The main R function for linear regression is `lm()`. As with all R objects, just type `?lm` within R/Rstudio to know more.
Suppose you have a dataset named `mydata`, with values of a variable x stored in `x` and values of a variable y stored in `y`, then you can write in the Rstudio command window:

```
lm(y~x, data=mydata)
```

The above fits a linear regression model (lm=linear model). Interpret `y~x` as "the model is $E(y) = \beta_0 + \beta_1 x$".
However if you just do the above you won't see the result of your fit. Therefore, better to assign the result of the calculation to an *object*, by doing the following:

```
mymod <- lm(y~x)
```

The `<-` is pretty much the `=` in Matlab, it assigns the value of the right hand side to the left hand side `mymod` (shorthand for *my model*. You can choose a different name). We'll see what to do with `mymod` in a moment.

For *simple linear regression* (recall, this means we have just 1 covariate) the OLS estimators of the regression parameters are those minimizing the least squares criterion

$$\min_{\beta_0,\beta_1} \sum_{i=1}^{n} (y_i - E(y_i|x_i))^2 = \min_{\beta_0,\beta_1} \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2,$$

for given n pairs $(x_i, y_i)_{i=1,...,n}$. The solution to this problem is well known (this is assumed from previous courses, e.g. MSG110)

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

with $\bar{x} = \sum_{i=1}^{n} x_i/n$ and $\bar{y} = \sum_{i=1}^{n} y_i/n$.

You can access the least squares estimators in several ways, again check `demo_cars.R`. You can type the name of the model you just created

```
> mymod
```

and the estimated β_0 and β_1 will be displayed. Equivalently you can type

```
> mymod$coefficients
```

The above shows that `mymod` is not, say, a matrix or a vector. In R **everything is an *object*** (similarly to Matlab's structures), which can contain more than just data. From an object we can extract a number of features, using the operator $. For example, above we have extracted the estimated parameters.
If you want to see only the estimated intercept write `mymod$coefficients[1]` and for the estimated slope write `mymod$coefficients[2]`.

3 Expected responses $\hat{y}$

Suppose we wish to estimate the expected responses $E(y_i)$ at the already available data x_i. We define

$$\hat{y}_i = \hat{E}(y_i) = \hat{\beta}_0 + \hat{\beta}_1 x_i, \qquad i = 1, ..., n$$

then we can just do

```
> mymod$fitted
```

and it will print the sequence of all $\hat{y}_i$.
How about estimating $E(y)$ for values of x that are *not in our dataset*? We can use the function `predict`. For example obtain estimations for *speed* $= 5$ or *speed* $= 7.5$:

```
m1 <- lm(dist~speed)
x0 <- c(5, 7.5)  # vector of values for which we require predictions
yhat <- predict(m1, newdata = data.frame(speed = x0))
> yhat
        1         2
 2.082949 11.913971
```

Notice the syntax: we first tell the function `predict` which model we want to predict from (`m1` in this case), as a first argument, and then we pass as a second argument the vector of covariate values `x0` we want the prediction for. The result is that when speed=5 we have the expected distance $\hat{y} = 2.08$ and when speed=7.5 we have $\hat{y} = 11.91$.
See the provided file `demo_cars.R` for further examples with more than one covariate.

4 R's `summary` function

An extremely useful function for linear models (and beyond) is `summary()`. Once you have fitted your model `mymod` you can type

```
> summary(mymod)
```

and it will print...well, a summary of the most useful results that you may typically want to know from a linear regression fit.

For example, consider the `cars` data from section 1. If we first fit the linear model `mymod<-lm(dist~speed)`, then we can obtain

```
> summary(mymod)

Call:
lm(formula = dist ~ speed)

Residuals:
    Min      1Q  Median      3Q     Max
-29.069  -9.525  -2.272   9.215  43.201
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -17.5791     6.7584  -2.601   0.0123 *
speed         3.9324     0.4155   9.464 1.49e-12 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 15.38 on 48 degrees of freedom
Multiple R-squared:  0.6511,Adjusted R-squared:  0.6438
F-statistic: 89.57 on 1 and 48 DF,  p-value: 1.49e-12
```

I am showing you this capability because, as you can see, that's just another way to obtain the OLS estimates. Plus a lot of more information that we will discuss in due time.

5 Estimating σ and the concept of residual

Our observed data are a sample from a theoretical "population" of cars, which happens to have an unknown[1] standard deviation σ, the square root of the variance of ϵ, that is $\sqrt{Var(\epsilon)}$. An estimate $\hat{\sigma}$, often denoted s, is given by

$$s = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n-q}}$$

where q is the number of parameters in a linear regression model. In this case (we are treating simple linear regression) we only have β_0 and β_1 so you can take $q = 2$ (but for more general models q can be larger than 2). The value of s is reported as `Residual standard error` when you type `summary(mymod)`. Therefore $s = 15.38$ for the `cars` data, and this value represents an empirical measure of variation for `dist`, regardless the speed, for the particular sample of data at hand.

The computation of s involves computing *residuals*: these are the n deviations $e_i = y_i - \hat{y}_i$ ($i = 1, ..., n$)), that is e_i is how much the predicted i-th response $\hat{y}_i$ deviates from the corresponding observed one. Ideally we wish to have residuals that are symmetrically scattered around zero, with all e_i "small" (close to zero), without any residual much larger than the others (the latter would denote that the model does not "fit" well all observations). For example, Figure 4 shows residuals scattered around 0 but some residuals are markedly larger than the others. It is up to you to decide, depending on the scale of the Y variable, whether a residual should be larger than, say, 3 or 30, or 300, to denote an outlier.

You can use `mymod$residuals` to obtain the vector of all residuals. And you can then plot them against the unit index i, using the following

```
> mymod <- lm(y~x)
> res <- mymod$residuals
> plot(res)
> abline(h=0)  # this adds an horizontal line at value 0, useful for display
```

[1] It is unknown because only if we could observe the whole theoretical population of cars we could have an idea of the exact probabilistic law that generated our specific sample data.

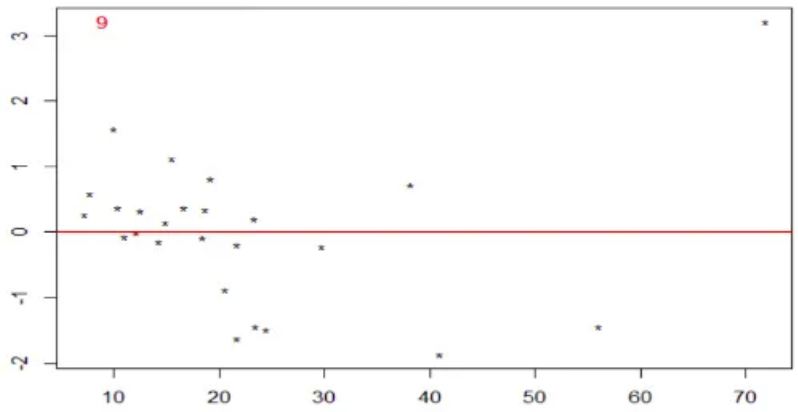

Figure 4: Residuals on y-axis and covariate value on the x-axis.

It is also ok to plot the e_i vs the x_i, as in

```
> mymod <- lm(y~x)
> res <- mymod$residuals
> plot(x,res)
> abline(h=0)  # this adds an horizontal line at value 0, useful for display
```

Active learning: let's experiment right away!

Just to get some practice with R's syntax, here you are asked to perform some simple calculations, using both (i) the R's built-in functions for linear regression, e.g. `lm` coupled with `$`, and (ii) "by hand", which means using R as a calculator but only via its basic arithmetic capabilities and using basic functions, but **without using** `lm`. Some basic commands are in `lab0.pdf`.

Consider the `sleeptab` dataset, which you can download from the Canvas course page. Make sure that sleeptab.dat is in your working-directory (see `lab0.pdf` for how to select a working directory). The dataset contains observations from 62 mammals[2], including brain and body weight, life span, gestation time, sleeping time, and predation and danger indices for 62 mammals. Detailed definitions of the variables in the dataset are in the last page of this document. Note: Missing values are denoted by `NA`.

```
# do the following AFTER you have placed sleeptab.dat in your favourite work-directory.
sleeptab <- read.table('sleeptab.dat',header=TRUE)  # loads the dataset
         # the option header = TRUE tells R that the first row
         # contains the variables names
attach(sleeptab)   # this way you can access the variables by name, without using $
```

[2]From Allison, T. and Cicchetti, D. (1976), "Sleep in Mammals: Ecological and Constitutional Correlates", Science, November 12, vol. 194, pp. 732-734.)

We will only use two variables, and perform simple linear regression with response variable y given by "brain weight" (`brwt`) and covariate x given by "body weight" (`bwt`). Some of the following questions are intentionally easy, just to get you started.

Exercise 1: For the `sleeptab` dataset

1. First, produce a plot of response (y axis) vs covariate (x axis). You notice three observations are a bit more extreme than the others. Then, identify which species these three belong to. (Tip: the function `which()` can help here though it is by no means necessary, for example for a vector `x`, `which(x)>0` returns *indeces* of the values in x that are positive).

2. Would it be appropriate to fit a linear model? To investigate this, once more plot response vs covariate, but this time only consider observations corresponding to brain weight smaller than 1000 grams (you can again use the function `which` to select the appropriate observations). Reflect on whether considering a linear model seems ok here.

3. Regardless of the previous question, assume a linear model $E(brwt) = \beta_0 + \beta_1 bwt$ to be fitted to all observations. Write an R code that computes the estimated slope and intercept "by hand" (as defined above) and then compare your results with what is returned by `lm`.

4. Interpret the value of $\hat{\beta}_1$. What can you say about it? And what does it help you conclude about predicting the average increase in brain weight for a 100 kg increase in body weight?

Exercise 2
In this exercise you may use the `predict` function to obtain the $\hat{y}_i$.

1. For the same model fitted in Exercise 1, produce the residuals plot. Do you observe any extreme observation? Extreme by how much?

2. Sometimes simple transformations can help the quality of our conclusions: log-transform (natural logarithm) both variables and plot them, then perform regression using the log-transformed variables, that is fit $E(\log(brwt)) = \beta_0 + \beta_1 \log(bwt)$. Then add the resulting fitted line to the plot with log-transformed variables. Better, no?

3. Using the original model, without transformed variables, estimate the value of brain weight when the body weight is 1000 kg, then do the same with the second model (for the second model remember to exponentiate the result, to obtain the estimated brain weight from the estimated log(brwt)). Give the estimated brain weights for the two models, and also the difference between them. Comment on whether you think this difference is substantial.

Note: It is normal (and expected) that the parameter estimates $(\hat{\beta}_0, \hat{\beta}_1)$ obtained with the log-transformed model are different from the corresponding estimates for the non-transformed model. After all, these estimates express the effect of a different (because transformed) covariate on a different (because transformed) response.

6 Recap: confidence intervals

Everything we have done so far does not really include any assessment of variability, meaning that we have merely obtained point estimates $(\hat{\beta}_0, \hat{\beta}_1)$ of the parameters, or an estimate $\hat{y} = \hat{E}(y|x = x_0)$ of the expected response at some value x_0 of the covariate x. However, all these estimates are random variables, and not deterministic constants. This is because the estimates are based on the specific $\{x_i, y_i\}_{i=1}^n$ dataset of size n we have. Let's denote this dataset

with $\mathcal{D}_1$. This is a randomly extracted sample from a usually much larger "population" $\mathcal{P}$. Here $\mathcal{P}$ is a theoretical population of all possible datasets that we could possibly observe if we had the resources to execute an infinite amount of experiments. Clearly we do not have such infinite resources. But in the hypothesis that we had those resources, then we would obtain the exact values of (β_0, β_1), that is the characteristics of $\mathcal{P}$ that we cannot observe.
In reality we can only approximate the values of (β_0, β_1) by extracting a finite amount of (hopefully informative) samples. For example, if $\mathcal{D}_1$ is the random sample of 50 measurements consisting in the `cars` dataset, and $\mathcal{P}$ is the population of *all produced cars in year 1920*, it is not difficult to imagine another dataset $\mathcal{D}_2$, also consisting of another random sample of size 50 extracted from $\mathcal{P}$, containing similar measurements of cars, and then a $\mathcal{D}_3 \in \mathcal{P}$, a $\mathcal{D}_4$ etc. From each $\mathcal{D}_j$ we could obtain corresponding estimates $(\hat{\beta}_0^{(j)}, \hat{\beta}_1^{(j)})$. The several pairs $(\hat{\beta}_0^{(j)}, \hat{\beta}_1^{(j)})$ would all be different, because obtained on randomly sampled datasets. Then we would obtain a *distribution of estimates* of the unknown (β_0, β_1), and this reflects the fact that our estimates are based on data resulting from a random procedure, and as such this randomness propagates to functions of such data (i.e. the estimates).

In practice we *cannot* execute many experiments, because of limitations of time, financial resources and other difficulties. Say thay we have to be content with our limited amount of data $\mathcal{D} = \mathcal{D}_1$. Can we use this to produce a quantification of the uncertainty around the estimated $(\hat{\beta}_0, \hat{\beta}_1) = (\hat{\beta}_0^{(1)}, \hat{\beta}_1^{(1)})$? Yes we can and this is because for linear regression we can construct the distribution of the parameter estimates, *under assumptions*.

The assumption we need (in addition to the ones given for OLS at the end of section 1) is that the random errors are Gaussian, $\epsilon_i \sim N(0, \sigma^2)$. If that's the case then we have closed-form formulas to construct **confidence intervals**. Loosely speaking, in the (sometimes unrealistic) hypothesis of being able to repeat an experiment many times, the confidence intervals constructed from the many estimates $\hat{\beta}_1^{(j)}$ are such that **a fraction** $100(1-\alpha)\%$ **of those intervals will include the true (yet unknown) value** β_1. And similarly for confidence intervals for β_0. Here $1-\alpha \in (0, 1)$ is a probability value called "confidence level".

For example, the confidence intervals for β_0 and β_1 are (this should be known from previous courses)

$$I_{\beta_0} = \left(\hat{\beta}_0 \pm t_{\alpha/2,(n-2)} \cdot s \sqrt{\frac{1}{n} + \frac{\bar{x}^2}{\sum_{i=1}^n (x_i - \bar{x})^2}} \right)$$
$$I_{\beta_1} = \left(\hat{\beta}_1 \pm t_{\alpha/2,(n-2)} \cdot \frac{s}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2}} \right)$$

with

$$s = \sqrt{\frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n-2}}$$

and $t_{\alpha,\nu}$ the α-quantile from the Student's t distribution with ν degrees of freedom. This can be calculated using `qt`, see further below.

The easy route to obtain confidence intervals:

```
> confint(mymod)  # defaults to 1-alpha=0.95
```

or

```
> confint(mymod,level=0.99)  # here "level" is set to 1-alpha = 0.99
```

for example the simple linear regression for cars data gives

```
> confint(mymod,level=0.99)
                 0.5 %    99.5 %
(Intercept) -35.706610 0.5484205   # interval for beta0 when 1-alpha=0.99
speed         2.817919 5.0468988   # interval for beta1 when 1-alpha=0.99
```

From the above we write $I_{\beta_0} = [-35.71, 0.55]$ and $I_{\beta_1} = [2.82, 5.05]$, when $1 - \alpha = 0.99$. These intervals include the true values of the corresponding parameters with 99% **confidence** (if the model is appropriate for the data and if the OLS assumptions are satisfied). As an example, theoretically, we are confident that if we were to repeat the fitting over 100 different datasets, then the interval [2.82,5.05] would contain the true β_1 99 times (that is 99% of the times).

Now, knowing the results above, solve the assignment below by knowing that quantiles for the Student's distribution can be obtained using the function `qt` (shorthand for quantile of a t-distribution). Recall, **an α-quantile q_α is such that α% of a distribution lies on its left side, and $100(1-\alpha)$% on its right side.**

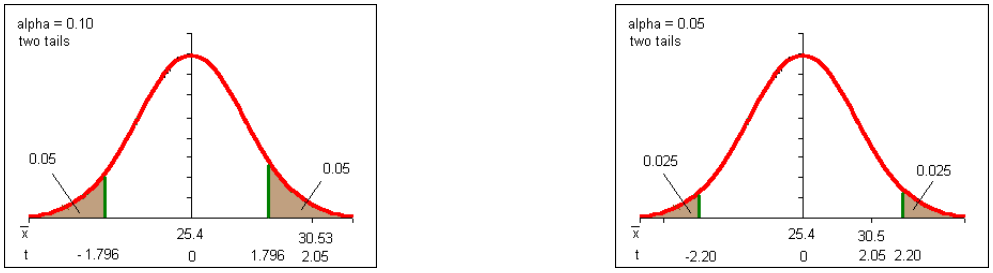

Figure 5: Student's t: examples of quantiles when $\nu = 11$. The plot on the left uses $\alpha = 0.10$ and the one on the right uses $\alpha = 0.05$.

While some probability distributions are asymmetric, the Student's one is symmetric so $q_\alpha = -q_{1-\alpha}$. In Figure 5 we have a Student's distribution with $\nu = 11$ degrees of freedom, and in the left panel we consider $\alpha = 0.10$. Therefore, in the left panel the brown areas sum up to 0.10 so the remaining white area equals $1 - \alpha = 0.90$. The value -1.796 separates a probability 0.05 on its left side from the remaining 0.95 on its right side. Therefore $t_{0.05,11} = -1.796$ and $t_{0.95,11} = 1.796$. In Figure 5 (right) the brown areas sum up to 0.05 so the white area equals 0.95. Therefore $t_{0.025,11} = -2.20$ and $t_{0.975,11} = 2.20$.

Suppose $1 - \alpha = 0.95$ and $\nu = 11$ degrees of freedom. We have

```
> qt(1-0.05/2,11)
[1] 2.200985
> qt(0.05/2,11)
[1] -2.200985
```

As usual, use `?qt` at R/Rstudio's command line to learn more.

Exercise 3: For the `sleeptab` data, here and in the following we keep using the linear model with untransformed variables, even if it fit worse than the transformed one, just because the theory for confidence intervals gets a bit complicated for the purpose of this course (this will be fixed in MVE190-MSG500).

1. For the linear model with untransformed variables, use R to calculate "by hand" the value of s (estimate of σ). This is useful for the next question.

2. For the linear model with untransformed variables, use R to calculate "by hand" the confidence intervals for β_0 and β_1 with confidence level 0.90, i.e. without using `confint`. Then of course you can check your results with the ones given by `confint`. Add your concrete interpretation of such intervals.

   [Curious of how to build confidence intervals for models with transformed variables? See you at the MSG500-MVE190 course.]

A common misunderstanding: it is not unusual to read interpretations of a confidence interval claiming that *"the true parameter value is enclosed in such interval with probability* $1-\alpha$*"*. This is incorrect. For example, β_1 is a fixed (unknown) quantity that is either inside the confidence intervals our outside of it with 100% certainty. Since confidence intervals are **random intervals** (they vary between datasets randomly extracted from the same population, as previously explained), we can say that these contain a population parameter (e.g. β_1) with some specified probability $1-\alpha$. This claim looks similar to the one in italic, however it is very different: the one in italic basically implies that the parameter is random (which is not, in our treatment). In the underlined definition what we have is instead that the intervals are random, and enclose the (non random) parameter with some probability.

You are now asked to verify that, indeed, confidence intervals are able to include true parameter values (separately for β_0 and β_1) with some pre-fixed probability. In order to do this, note how to produce Gaussian draws ϵ using R (this is also covered in the preliminary read `lab0.pdf`).

The function `rnorm` produces Gaussian pseudo-random numbers.

```
# type ?rnorm to learn more
> rnorm(1)     # returns a single draw from N(0,1), the standard Gaussian.
[1] 1.459218   # you will probably obtain a different number. That's ok.
> rnorm(2)     # returns two draws from N(0,1)
[1] -0.2152097  0.8462947
> rnorm(3,mean=3,sd=2)    # returns three draws from N(mu=3,sd=2)
[1] 4.664346 2.018924 3.094830
```

Common mistake: `rnorm` does not expect you to pass the value of a variance. It wants the value of a standard deviation in the `sd` argument. You are probably familiar with Gaussian distributions written as $N(\mu, \sigma^2)$, i.e. parametrised via mean and variance. However scientific software typically expects the user to provide a standard deviation σ, not a variance. Be careful and **always read a function's documentation!** In this case, type `?rnorm`.

Exercise 4: here we want to verify empirically the theoretical result that the true value of a parameter is included into confidence intervals with some specified probability. For the `sleeptab` dataset, now denote with (β_0^*, β_1^*) the already obtained least squares estimates. Let's pretend that (β_0^*, β_1^*) are the *true* parameter values. Write an R code using `for loops` to produce 2000 sets of parameter estimates and confidence intervals according to the following reasoning:

1. Plug the (β_0^*, β_1^*) in the following linear model and use it to produce simulated observations $\text{brwt}_i^{new} = \beta_0^* + \beta_1^* \text{bwt}_i + \epsilon_i^{new}$, for $i = 1, ..., n$ using $n = 62$ (i.e. the same size as the `sleeptab` dataset) where the bwt_i are the same values as in the given dataset. Use the same s value as from exercise 3 with `rnorm` to simulate the n values of the $\epsilon_i^{new} \sim N(0, s^2)$. So now we have a simulated dataset $\mathcal{D}_1 = (\text{bwt}_i, \text{brwt}_i^{new})_{i=1,...,n}$. Fit $\mathcal{D}_1$ via linear regression and denote the parameter estimates $(\hat{\beta}_0^{(1)}, \hat{\beta}_1^{(1)})$.

2. Repeat the procedure: use again the original (β_0^*, β_1^*) to produce new simulated observations $\text{brwt}_i^{new} = \beta_0^* + \beta_1^* \text{bwt}_i + \epsilon_i^{new}$, where the ϵ_i^{new} **are generated anew via** `rnorm` (they are not the same ϵ_i^{new} as in the previous step). Call the new dataset $\mathcal{D}_2 = (\text{bwt}_i, \text{brwt}_i^{new})_{i=1,...,n}$. Fit $\mathcal{D}_2$ via linear regression and denote the parameter estimates $(\hat{\beta}_0^{(2)}, \hat{\beta}_1^{(2)})$.

Repeat the two steps above until you have obtained 2000 sets of estimates $(\hat{\beta}_0^{(j)}, \hat{\beta}_1^{(j)})$, $j = 1, ..., 2000$. Construct confidence intervals (choose $1 - \alpha = 0.90$) from each set of estimates, so in the end you have obtained 2000 confidence intervals for β_0^* and 2000 intervals for β_1^*. At this point, you can finally compute the proportion of intervals that include the original value β_0^*. Then do the same for β_1^*. Show that both proportions are very close to $1 - \alpha$, as expected from the theory.

What we have done with this exercise is to use simulations to verify theoretical claims. Nice, isn't it?!

Here follow definitions of the variable contained in the `sleeptab.dat` dataset.

- species of the animal
- body weight in kg
- brain weight in g
- slow wave ("nondreaming") sleep (hrs/day)
- paradoxical ("dreaming") sleep (hrs/day)
- total sleep (hrs/day) (sum of slow wave and paradoxical sleep)
- maximum life span (years)
- gestation time (days)
- predation index (1-5) 1 = minimum (least likely to be preyed upon), 5 = maximum (most likely to be preyed upon)
- sleep exposure index (1-5), 1 = least exposed (e.g. animal sleeps in well-protected den), 5 = most exposed
- overall danger index (1-5) (based on the above two indices and other information) 1 = least danger (from other animals) 5 = most danger (from other animals)

Note: Missing values are denoted by NA.