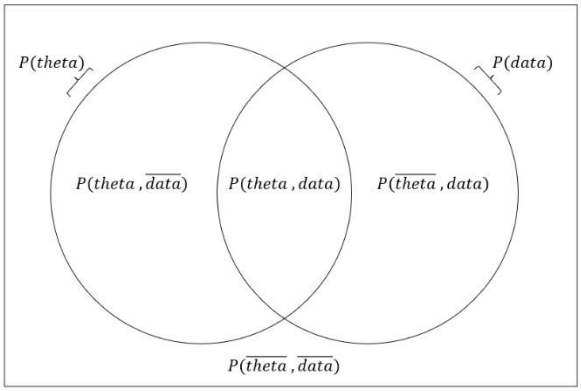
Inferential statistics II – Linear Regression via the Linear Least Square Method in R

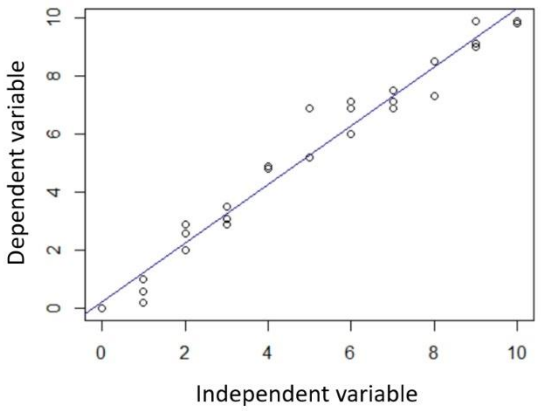
Beginner's Stat-o-Sphere

by Steffen Schwerdtfeger

Our second tutorial on inferential statistics focuses on simple linear (regression) models, obtained via the linear least square method. Below you will see a comparison between a probabilistic model and a linear model. Different to a probabilistic model, a linear model does not consist of probability values ranging from 0 to 1, but of values of scaled measurements.



Probabilistic model

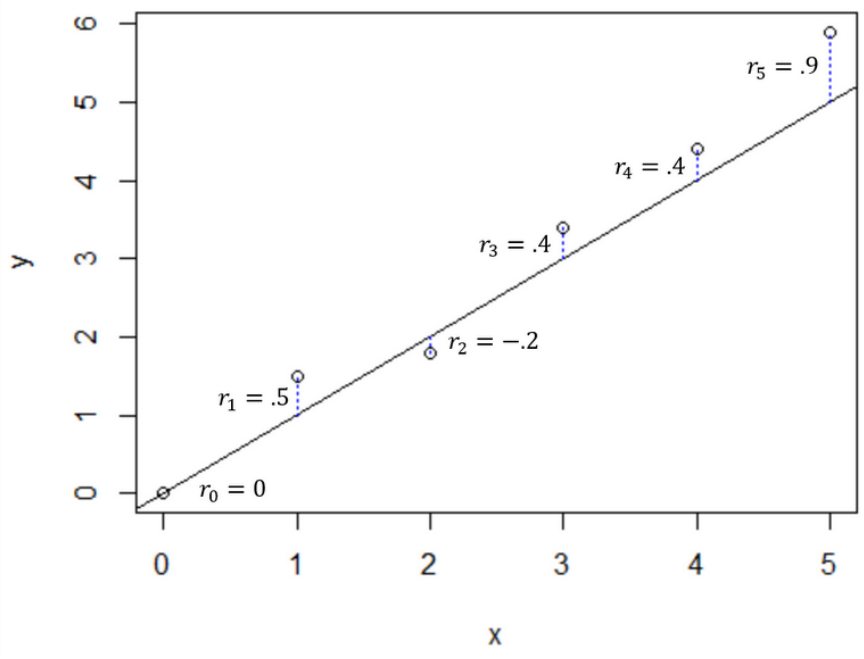


Linear model

In other words: different to a conditional probabilistic relation between a hypothesis and data, a linear model consists of the conditional linear relation between variables – the relation between an **independent** (**x-axis**) and a **dependent variable** (**y-axis**). The model is therefore literally the linear function above that was thoughtfully drawn through some data points.

The goal of a linear model is to thoughtfully make predictions of (new) data. As we see on the right graph above the data points are not all directly located on our model (the linear function in the graph). However, they appear to be at least rather close to the model estimation (the line in the graph), as we can see that our model follows the linear tendency of the data quite well. The ("vertical") deviation between the actual data and our model estimate is called **error** or **residual**.

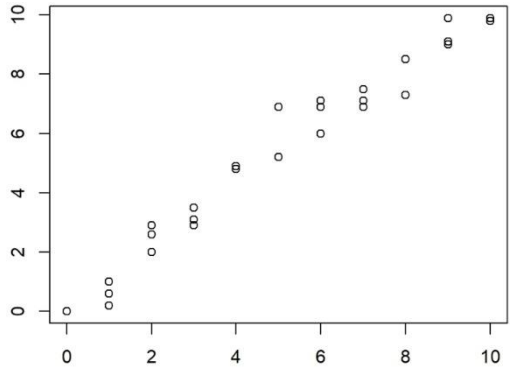
Below you will find a graph with some random data points and the evaluated residual value (see full tutorial for R code of the graph). The formula for evaluating the residual is: $y - f(x) = residual$. Outspoken the formula states: the residual / error is equivalent to the data value of the *dependent* variable minus the *estimated* model value of that dependent variable. The residual tells us how far of our model is from the actual data point. *Note* that both the data value y and the model estimate $f(x)$ refer to the same value of x. In other words: The residual difference only refers to "vertical" $y - axis$ so to speak (there are other methods as well, which we will not go into here).



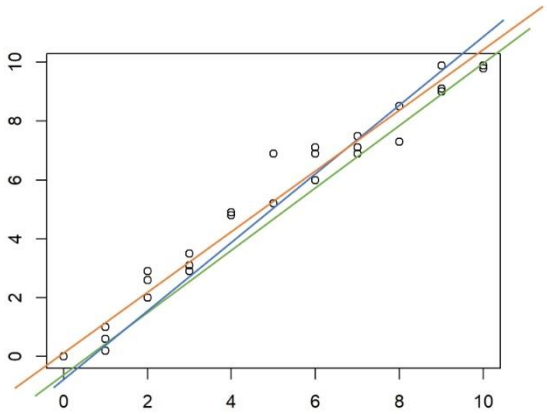
The goal is to find a model that makes the least errors in general, i.e., in account of all given data points (the least *sum* of alle errors of a possible linear model).

Deciding for a linear model that fits well to the given data is in general cast as "optimization problem". One of the simples being the **linear least square method.** As a solution for our optimization problem, it will help us make a transparent and reflected decision on where to optimally draw our line through the data. In other words: it tells us the optimal parameters for a and b of our function $f(x) = a + bx$.

So, intuitively finding a linear model is actually really simple and intuitive: it is *literally* just about drawing a line into a graph with data points in a way that our linear model doesn´t deviate too much from the actual data points, when the data points are seen as a whole of some kind (the sum or errors – we will get to that). Try yourself. Draw an optimal line through the given data points:



Raw data points



Different possible linear models

Dependencies of Variables

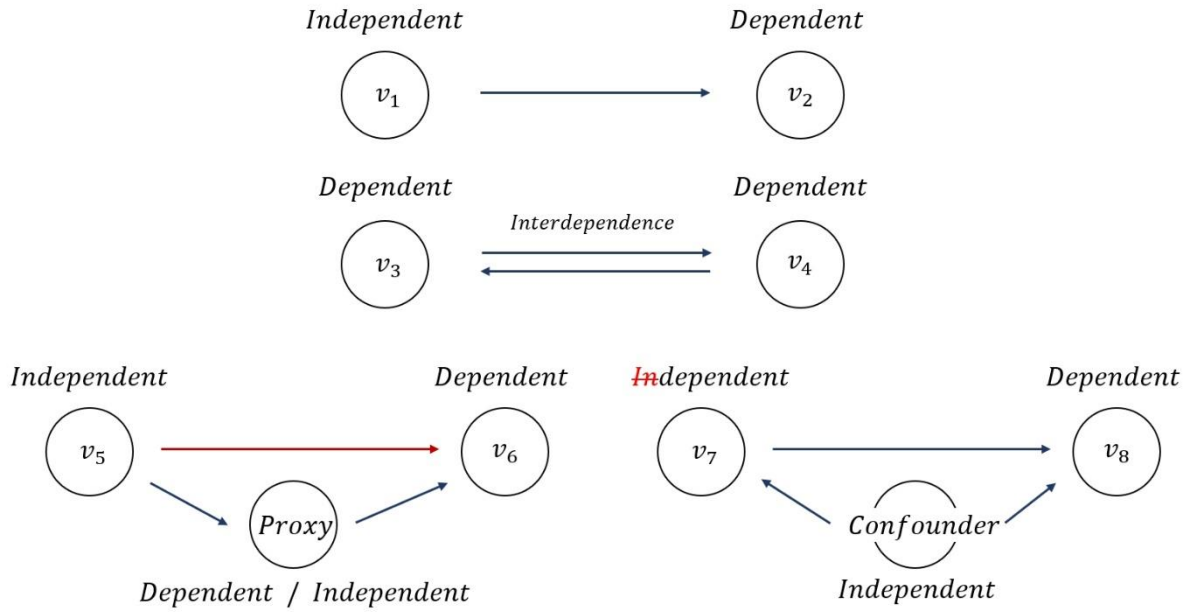
Again: Instead of a relation between $theta$ and $data$, a linear model consists of a (conditional linear) relation between an *independent* and *dependent variable*. In other words: a linear model consists of *estimated* measurement values, such as the amount of tea drank (dependent variable) in a certain amount of time (independent variable).

Evaluating the p(robability)-value, i.e., the significance of a linear model, will be done in part three of our tutorial series on inferential statistics and involves a t-test et cetera. In general, evaluating a linear model involves a method to look at a linear model from a probabilistic perspective, where $theta$ means *in nuce*: there is a linear relation that predicts the (new) data well enough (alternative hypothesis).

Independent variable (x): An independent variable is a variable that is not influenced by any other variable, but can influence another variable (the dependent variable). The independent variable is often "time", which is usually not influenced by other variables. Exceptions can be found in physics (relativity of time and space in relation to mass of an object), or in linguistics, e.g., when measuring the time needed to read an item (e.g., a small text). Changing the item may result in a change in the time needed to read the particular item. However, the item is not changing just because time passes (the text remains the same throughout time). Time is in this case the dependent variable, dependent on the particular text item.

Dependent variable ($f(x)$): A dependent variable is therefore a variable that is influenced by another variable, but does not change other variables – it is dependent on other (independent) variables. Another example for a dependent variable could look like this: An experimental setting is supposed to clarify if the mood of a dog changes when served different kinds of dog food. The dog food is considered the independent variable, which is not influenced by the dog, but is supposed to influences the dog's mood. The dog's mood on the other hand does not change which kind of dog food is served, as *we* decide which food is served and then check if there is a differentiable relation to the dog's mood. The dog's mood therefore represents the dependent variable – dependent on the dog food so to speak.

Handling the concept of the characteristic dependency of variables can be tricky for several reasons. We evaluated some of them in our full text tutorial and tried to establish algorithmic approaches for understanding and evaluating the dependencies of variables (e.g., in a paper).

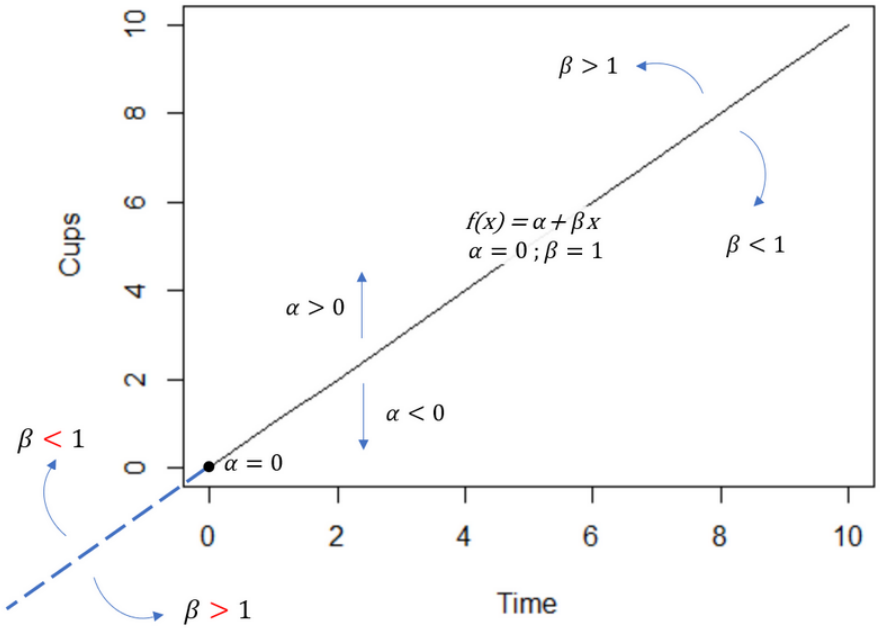


Above you see several general graphical relations between variables, including confounder and by proxy effects. **See full tutorial for more details (not essential for understanding the linear least squares method below).**

Recap Linear Functions:

Below you can see an idealized linear function with a slope $\beta = 1$ and a point where the linear function crosses the $y - axis$ at $P_\alpha(y = 0|x = 0)$.

Formally our function is written as $f(x) = 1 * x = x$. The point where the function crosses the y-axis can be evaluated by setting x within $f(x) = \alpha + \beta * x$ to zero and adding the value of β, such that $0 = \alpha + 1 * 0$, resulting in: $0 = \alpha$.



The graph above importantly indicates that a change in β does not indicate an indirect change of the value of α. Think of α more as an axis point on which our linear function rotates on when β is changed.

We can now reformulate our goal of drawing an optimal line by saying: We are looking for a particular α and β in a function $f(x) = \alpha + \beta * x$ that fits best to the given data points, i.e., which leads to the least "amount" of errors when making a prediction on (new) data.

In other words: the optimization we are about to perform concerns *minimizing an error*: We want to find the best model function amongst all possible model functions, where best is defined as an attribute of a function that makes a minimum of errors when doing predictions (on given or new data).

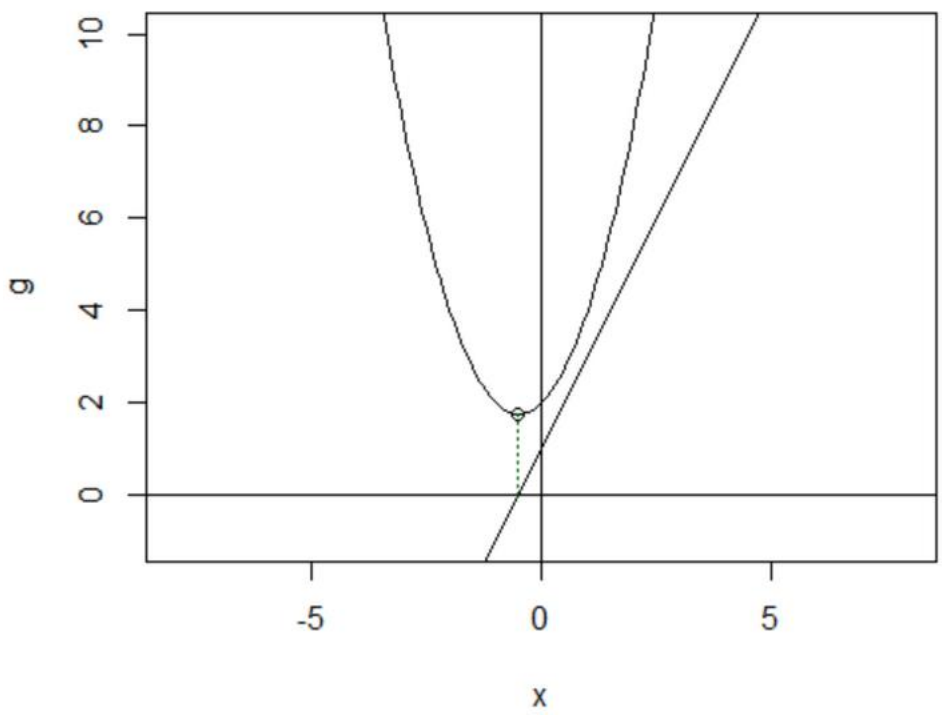
But why is it called the linear leas *squares* method? You may recall from school that particular "pond like" parabola functions can have a minimum and "mountain like" parabola functions can have a maximum. Comparing our goal of minimizing errors to the concept of a regular linear function, you might already anticipate that finding the minimum of a linear function does not seem to work, as it would lead to a residual value approaching minus infinity. In order to overcome such relational boundaries, we are going to *square* the errors, which turns our linear problem of finding a particular α and β in a function $f(x) = \alpha + \beta * x$ into a quadratic problem: finding a particular α and β that leads to the least or minimum of *the sum of squared errors* ($\sum$= sigma = sum sign; r = residual):

$$\min_{a,b} \sum_{i=1}^{n} r^2$$

We are not going to use a parabola function to solve our new quadratic problem, but a partial differential equation (PDE) to do so. However, most of the mathematical steps of solving a PDE are equivalent to finding the minimum in a simple parabola function, so we will do short recap on this type of functions as well.

Recap Parabola Functions:

In a parabola function $g(x) = x^2$ the minimum/maximum value of a function $g(x)$ can be obtained by taking the *first derivative*, then setting it to zero, subsequently solving the equation for $x_{min/max}$. We can then obtain the corresponding value of y of that $x_{min/max}$ via plugging in $x_{min/max}$ into $g(x)$:



Above we see the parabola function $g(x) = x^2 + x + 2$.

The first derivative is the linear function $g'(x) = 2 * x + 1$, which we also see in the graph. Above we can see that when setting $g'(x)$, i.e., the y-coordinate to zero, we will obtain the value of x where that linear derivative function will cross with the x-axis. The important part is that the derivative function crosses the x-axis right at the minimum value of $g(x)$, i.e., the minimum y-coordinate value of the function.

We then precede and plug in that particular x_{min} into our parabola function $g(x)$, which will subsequently give us the respective y-coordinate of the minimum of the above parabola function $g(x)$.

The Linear Least Square Method in Detail:

In order to find a particular α and β that leads to the least *sum of squared* errors, we will operate with a function that contains two variables – α and β – instead of just one variable x. The rest of the process is the same: we will take the derivate of our PDE, set it to zero and will then solve the equations. That's right: equations. Taking the derivative of a partial differential equation will leave us with two equations, instead of one when setting the derivative of a parabola function to zero in order to evaluate its minimum.

Before taking the derivative, let us first look at our PDE in full, which essentially is just our residual / error formula squared and summed (∑= sigma = sum sign). Again: We need to sum the errors, as we are not only looking at the error of one data point in relation to our model, but all of them.

$$E(a;b) = \sum_{i=1}^{n} (y_i - f(x_i))^2 = \sum_{i=1}^{n} (y_i - a - bx_i)^2$$

Our PDE in the current form above is an error function E with respect to two variables, a **and** b (which is the same as α and β), which correspond to the given data values x_i and y_i, all together resulting in a value of the squared error (*not its minimum yet!!*).

Below you will find the two partial derivatives of our PDE. The fractions below are spoken "partial derivative of $E(a; b)$, regarding a or b, i.e., in respect of either a or b", which just means that either a or b is set constant (does not change; no difference), when the partial derivative with respect to either a or b is evaluated.

$$E'_a(a; b) = \frac{\partial E(a; b)}{\partial a} = \sum_{i=1}^{n} 2 * (y_i - a - bx_i) * (-1)$$

$$E'_b(a; b) = \frac{\partial E(a; b)}{\partial b} = \sum_{i=1}^{n} 2 * (y_i - a - bx_i) * (-x_i)$$

As we know, the next step involves setting the above equations to zero. From there the goal is to rearrange the equations to form a *system of equations*, in order to solve for a and b, e.g., via substitution methods. **We thoroughly go through each step in our full tutorial. Here we are going to jump ahead a few steps, such that our system of equations will look like this:**

$$a * n + b \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i$$

$$a \sum_{i=1}^{n} x_i + b \sum_{i=1}^{n} {x_i}^2 = \sum_{i=1}^{n} x_i y_i$$

It is important to recall that x and y are values that are given to us. So, the above system of equations entails two equations *that can be though of as looking something like* $a + b = 0$ and the other equation $2 * a + 3b = 5$. **Solving a system of equations via substitution means** that we can use the first equation, rearrange it as $a = -b$, plug it into the a of the second equation and are then able to solve for b (subsequently doing the same for a).

The above system of equations of our linear model is a generalization and can now be used for any values of x and y. In R the system of equations would formally look like this:

```
#   sum(a*length(x)) + b*sum(x)    = sum(y)
#      a*sum(x)      + b*sum(x^2)  = sum(x*y)
```

In the third part of our tutorial series, we will also introduce alternative methods to obtain the optimal a and b for $f(x) = a + bx$, e.g., via covariance and variance. The way we introduce it in our full tutorial relates especially well to those with a minimum background in mathematics, having at least a basic visual idea what a linear and parabola function is, in order to build up the rest of the knowledge needed to perform linear modelling from there.

Replicating the Math in R

In our full text tutorial we replicated the complete mathematics of the built in **`lm()`** function in R. Our function will also automatically plot the results.

Here is the code that at least works for basic linear models in the same way as the function **`lm()`** does. Below the **`solve()`** function was used in order to solve the system of equations (via substitution). It also entails the use of the **`cat()`** function to produce nice looking output.

```
# Go-go-gadgeto linear_least_square!!!!
# Replication of the lm() function:

linear_least_square = function (x,y){ # Start of function
    # "Guess function"
    lmRandom = function(x) (x)

    # Setting up system of equations:
    left_a_r1c1 = sum(a*length(x)); left_b_r1c2 = b*sum(x)
    left_a_r2c1 = a*sum(x)        ; left_b_r2c2 = b*sum(x^2)
    right_r1c1 = sum(y)
    right_r2c1 = sum(x*y)

    # Now we will set up the above as matrix (left) and
    # vector (right) objects:
    left = matrix(c(left_a_r1c1 , left_b_r1c2,
                    left_a_r2c1 , left_b_r2c2),
                          ncol=2, nrow=2, byrow = TRUE)
    right = c(right_r1c1, right_r2c1)

    # Now we can solve our system of equations via:
    Result = solve(left,right)

    # Fitted function:
    lmFitted = function(x) (Result[1]+Result[2]*x)
    SumError2fitted = sum((y-lmFitted(x))^2)

    # Code for nice console output
    cat(" Linear least square method in R","\n","\n",
    "Independent variable:", "\t", deparse(substitute(x)),"\n",
    "Dependent variable:", "\t", deparse(substitute(y)),"\n","\n",
    "alpha", "\t",Result[1],"\n",
    "beta","\t",Result[2], "\t","SumR2", "\t", SumError2fitted)

    # Plot Results:
    plot(x=x,y=y, ylab = deparse(substitute(y)),
                  xlab = deparse(substitute(x)))
    abline(a=Result[1], b=Result[2], col = "darkblue")
} # End of function
```

Below you find the synthetic data set used for the above graphs of a linear model:

```
# Cups of Tea = Dependent Variable
cups = c(0, 1, 2, 3.5, 4.8, 5.2, 6, 6.9, 8.5, 9.1, 9.9,
         0, .6, 2.6, 3.1, 4.8, 6.9, 7.1, 7.5, 8.5, 9.9, 9.9,
         0, .2, 2.9, 2.9, 4.9, 5.2, 6.9, 7.1, 7.3, 9, 9.8)

# Time passed for each run of measurements (cups), abbreviated:
#             = Independent Variable
time = c(0:10,0:10,0:10)
```

Execute the above code of the function and subsequently use it on the above synthetic data set in order to replicate the respective graphs above:

```
linear_least_square(time, cups)
```