# Plotting a function

In this task we get to work with differentiation of functions in R and we will look at the difference between exact and numerical methods.

To begin, we need a sequence of $t$ values to which we can apply our mathematical func­tion.

**Task** Create a sequence of $t$ values between 0 and 10 with increments of 0.1, and call this sequence t.data . Use the built-in R function seq .$$

Once we have the $t$ data we can define a function. Let's start with the function $f(t)=at^{2}+b$.

**Task** Write a function test.function which has $t$ and the two parameters $a$ and $b$ as arguments in input, and which returns the above function. Allow default values 1 and 0 for the parameters $a$ and $b$, respectively.

Apply the function (with $a=1$ and $b=0$ ) to the sequence of $t$ values and name the resulting sequence test.data . To check if everything went well, we make a scatter plot of the obtained data set.

**Task** Use the built-in plot function to create a plot of t.data vs. test.data .

*Hint:* When everything goes well, the plot should look like the following diagram.



$$$$

We will now consider the function $f(t)=\frac{1}{1+e^{-t}}$. This is an example of a function that models restricted growth. Can you predict what happens to function values when $t$ becomes very large? Why is this restricted growth?

Write a new function called function1 which again accepts $t$ values as input and returns the function values of the above function as output. Then create a sequence of function values and name this sequence ft. Also construct a scatter plot in which you limit the display of $y$ values to the range from 0 to 1 via the option ylim.

*Hint:* When everything goes well, you should see the following.



# Exercises: Computing a derivative

## Question 1

Compute the derivative of $f(t)=\frac{1}{1+e^{-t}}$.

$f'(t)=\frac{df}{dt}=$ ............................................

# Plotting together the graphs of a function and its derivative

When all went well, you have derived on the previous page the derivative of $f(t)=\frac{1}{1+e^{-t}}$.

Again write a function that has $t$ values as input and return the function values of the deriva­tive as output.

Then use the lines command to plot the derivative together with the graph of the function itself. With the option col you can colour the lines differently.

Also add a legend that makes clear which functions are displayed.

When all goes well, the plot should resemble the following diagram. 

# Numerical derivatives via 2-point and 3-point difference formulas

So far, we have created a data set for the function $f(t)=\frac{1}{1+e^{-t}}$ where $t$ runs from $0$ up to and including $10$ s with increments of $0.1$ s. You have computed the derivative of this function, too.

However, in order to compute the derivative of an arbitrary data set on a computer, it is necessary to differentiate numerically. One approach is to use the ratio of the differences in the $f(t)$ direction and the corresponding differences in the $t$ direction.

The numerical derivative can be determined in several ways. Here we will focus on determining numerical derivatives via the so-called *two-point forward* and *3-point* *difference formula*.

$$$$

### The 2-point forward difference formula

When for a 'decent' function $f(t)$ and a certain step size $∆t$ the function values at $t\_{0}$ and $t\_{0}+∆t$ are known, then the derivative $f'(t\_{0})$ be cn approximated with the **forward finite difference.** This is given by:

$$f'(t\_{0})≈\frac{f(t\_{0}+∆t)-f(t\_{0})}{∆t}, for small positive value of ∆t$$

This can be read as the difference in the $f(t)$ direction divided by the difference in the $t$ direction. When the step $∆t$ is small enough, then this is indeed an approximation of the slope of the function at the point $t\_{0}$ and therefore an approximation of $f'(t\_{0})$ .

**Task** Create a script in R to compute the numerical derivative via the **2-point** **forward difference formula** for the data set used earlier and for which:

t <- seq(from=0, to=10, by=0.1)

 ft <- function1(t)

Plot the result where the graph meets the following requirements:

* The original dataset is a scatter plot. This graph is on top and is titled 'Original func­tion'.
* The result of the numerical derivative is below the graph of the original data set, and is plotted as a line without points and with thickness $3$. The title of this graph 'Derivative'.
* The values of the derivative calculated by you are plotted in the same graph as the result of the numerical derivative through a dashed line of thickness $5$.
* Add in the lower graph a legend to make the distinction between the curve of the derivative and the result calculated by the 2-point difference formula
* The $x$ axis of the two graphs is called $t$.
* The $y$ axis of the top graph is called $f(t)$, and of the bottom graph $f'(t)$

The resulting graphs are similar to:



*Hint 1:* Use a for loop.

*Hint 2:* Make sure in plotting that the sequences have the same length.

*Hint 3:* You can place multiple diagrams in a $m×n$ matrix using the following command (calling *before* you plot the graphs): par(mfrow=c(m,n)) .

*Hint 4:* The lines function enables you to add lines to an existing plot.

*Hint 5:* A legend can be added with the command legend .

$$$$

### The 3-point difference formula

After having determined the two-point forward derivative, you will determine the deriv­a­tive with the 3-point difference formula. The 3-point forward difference formula is a combination of the two-point *forward* method and the 2-point *backward* method (we have not dealt with the backward method, but this is similar to the forward method).

In the 3-point difference method, the derivative of a function at the point $t\_{0}$ can be ap­prox­i­mated by:

$$f'(t\_{0})≈\frac{f(t\_{0}+▵t)-f(t\_{0}-▵t)}{2▵t}, for small positive value of ▵t$$

**Task**

Extend your script in R with the computation of the derivative via the 3-point difference method and plot the result in the *same* graph as the result of the two-point difference method and the curve of the derivative calculated by you. Is there a difference?

* Don’t forget to adjust the legend.

*Hint 1:* Do the first and last time in the data set play a role in the computation of the derivative with the 3-point difference method?

*Hint 2:* How many points there are now superfluous in the original $t$ data?

# A noisy signal: 2-point and 3-point difference formulas

In previous assignments you have computed numerical derivatives via the two-point forward and the 3-point differences method for the function $f(t)=\frac{1}{1+e^{-t}}$. You have plotted the original function and you have plotted the results of various numerical derivatives together in one diagram beneath it. It is true that the given function $f(t)$ is a 'decent' function. A real dataset, however, often contains noise. In the tasks below you will investigate the effect of noise on the numerical derivatives.

**Task** Extend your script in R with a function that adds noise to the function $f(t)$ and plot the result together with the original curve of the function $f(t)$ in one diagram.

Please ensure that:

* you plot the noisy signal in a different colour than the signal without noise.
* the original form of the equation is still visible after addition of the noise. So make the signal not too noisy;
* you adjust the legend of the diagram;
* plotting of the 2nd graph is temporarily suppressed.

The final diagram will look like the following:



*Hint 1:* Use the jitter command to introduce noise.

*Hint 2:* Use the points command after the plot command to add a scatter plot to the original graph .

*Hint 3:* You can play with the values within the jitter command. Make sure that the original form of the function is still recognizable.

*Hint 4:* You can avoid the plotting of the second graph by transforming that piece of code into 'comment lines' by placing '#' at the beginning of the sentences. This you can also do by selecting a whole piece of code and then enter "ctrl + shift + c".

**Task** Apply now the 2-point and 3-point difference methods to compute the numeric derivative of the signal with noise. Does one of the methods give a better result?

* Make R functions of the 2-point and 3-point methods. Then you can call these R functions for any $t$ data and $f(t)$ data as input arguments.

*Hint 1:* Create *within* the function for computing a 2-point and 3-point based derivative an empty sequence for the storage of the results. Make sure you do this *outside* the for loop.

*Hint 2:* If the numerical derivatives are outside the $y$ range of the graph: Reduce the noise on the signal, then the results can be compared with each other more easily.

**Task** Investigate the effect of changing the step size in $t$ data and the increase/ decrease of the noise.