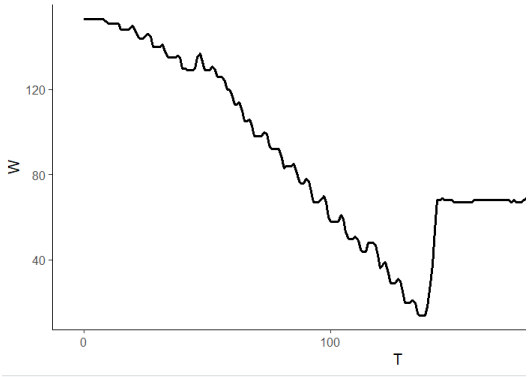
Sliding window lab

Below you will find instructions for creating an greedy algorithm using a sliding window for Lab I. You can use any sequential data you want, but make sure it has the proper file type for the read_xlsx() function (or read.csv()), i.e., .xlsx. You can also use the example file:



IMPORTANT! Make sure to try your code often to see if the last change worked. Also, use the help/documentation of functions as much as possible to understand the class, structure, and which arguments the functions expect.

To separate the code, we want to create at least two function types, i) functions that take a series of data (referred to as window) and returns a value, and ii) a function that goes iteratively through the time series and runs the previously created window function at each iteration. For this lab we will create a function that adds values to the beginning of the data frame if the difference between the minimum and maximum value of the window if higher than a certain threshold value and a window that returns the minimum value of the window.

Vectorised language – one of the benefits for this algorithm is that R is a vectorised language which allows us to add a single value to parts of the data frame (or vector) without another loop, which makes the current algorithm quicker. The reason is that for the window where we are checking the difference between min and max, if we find a value higher than the threshold we have decided on, we can add the maximum value to the entire loop up until this point: vector[1:a] <- vector[1:a] + x (pseudocode).

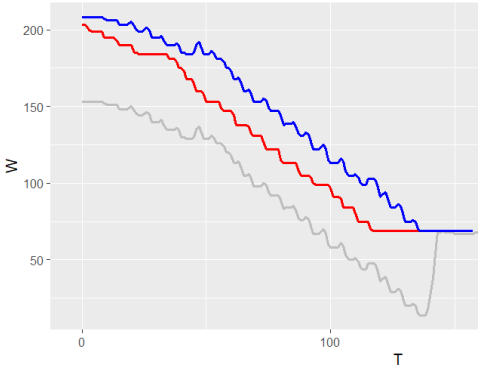
Important to note is that the object returned from the read_xlsx() function provides a data frame where one column is the seconds and one is the weight.

1. To make it easier to import an image to R we use the package “readxl” and to be able to display our data we use the package “ggplot2”.
2. We then use the read_xlsx() function with the appropriate file path to load a time series object to the environment.
3. We can then display the image in the R plot window by using the ggplot() function, with the data frame as first argument and aes(x = T), stringed together with a + followed by the function geom_line(aes(y = W)). This is not required, since we can also look at the data in out environment.
4. To create separation in our code, we then create window functions, one for returning the maximum and minimum difference in the window provided an one returning only the minimum value. Here you can use the already existing min() and max() functions inside the function.

5. We then create a function that iteratively works its way through the time series. The function should take the data frame, the window size, and the threshold size as arguments. The function should also contain a for loop that goes through each index of the weight column in the data frame. Make sure to account for the window size, so that you do not provide a higher index than there are rows in the weight series.
6. The easiest way to store the data is to create additional columns in the data frame where you add the return values from the window functions. One way of doing this is to write data[, c("rule", "min")] <- NA, if we want to add two columns named “rule” and “min” to our data frame with only NA values, that we can then populate.
7. Use the ggplot() function again to make sure you have successfully changes the look of the image. This time adding the + geom_line(aes(y = rule)) and + geom_line(aes(y = min))

When you are finished you should be able to create two different lines, depending on what window function you provide

The final pictures should look like this:



If you want to continue experimenting with the code, you can create a function that runs in the opposite direction or that adds values on top of each other if the min and max value difference exceed a certain value.