*A markdown file (and whatever compiled variant) that explains what the two functions do and how to use them. This should also contain a short example analysis, including plots and interpretation. The data underpinning the example should also be available in the repository.*

Introduction to the problem

\*introduction talk bla bla, why we are doing what we doing\*

Bootstrapping allows us to calculate a 95% confidence bounds for our estimate without distributional assumptions, by resampling with replacement from the available data. By doing this in high repetition, we can get a good estimate with lower sampling uncertainty. However, to access this high repetition we must make the code fast to run and easy to use.

The optimised code is based on the provided sample, and was created by inspecting the code, and identifying the following parts that could be sped up:

* The *linear model* is a slow function to run, could rephrase as a matrix.
* The *rbind* function to be replaced with
* Avoid repetition by taking out the *for* loop and *if* statement
* There are no defined vector sizes, so calculations are repeated more than necessary.
* Similarly we wish to avoid using the function *nrow* multiple times.
* There is only space for a limited number of x covariates, and we wish to extend this.

SAS

R

What is the Maths behind it?

To optimise the linear model and consider multiple covariates, we rewrite the model into a matrix format, using the following theory. A linear model can be written as where is the error.

Writing this in a matrix form or simply .

**

*m* number of columns

Estimating two or more slope coefficients is straightforward using least squares. We find estimates that best fit the data by using .[[1]](#footnote-1)

How does it work?

The optimum function is given by *bestBadBootBrotherAnyCovars3(nBoot, yDat, …).* This function takes at least three variables:

* nBoot: the number of iterations,
* yDat: the response vector,
* … : one or more vectors that will form the columns of covariate design matrix.

First we set up the parallelization of tasks by creating clusters based on the computer cores. Then we take the input data and form the matrices *X* and *Y* following the model above. We use a helper function *parBootAnyCovars* that bootstraps the data and calculates the estimates for the best fit, and we return as a matrix.

*Note*: we have implemented two optimized functions, which give similar results on reruns. We considered it best to include both for the sake of scalability, as they are build differently and whilst one is initially faster, the other seems to scale in better time.

Analysis of Results

\*a short example analysis\*

\*need plot of times of how well each bootstrap function is doing\*

1. Taken from Chapter 3 of MT5753: Statistical Modelling notes from 2017. [↑](#footnote-ref-1)