# Report

*Drunken Master 2*

*Due 5 November 2018*

## Contents



### Introduction

Bootstrapping is an area of ststistics that is usually implimented using simple Monte Carlo simulations whereby a certain calculation is repeated a large number of times with random sampling (ref?). Repeating calculations a large number of times, say 1,000,000 times, can become slow to compute. Therefore, the use of efficient and fast code is essential.

This project aimed to improve and produce two fast and efficient bootstrap functions using R 3.5.1 (R, 2018) and SAS 9.4 (SAS Institute, Cary NC).

```
fitData <- read.csv("data/fitness.csv")
```

A short example analysis was given for each function. The fitness dataset from Rawlings (1998) contains measurements of the following seven variables obtained from 31 men: • Age: Age in years • Weight: Weight in kg • Oxygen: Oxygen intake rate, ml per kg body weight per minute • RunTime: time to run 1.5 miles in minutes • RestPulse: heart rate while resting • RunPulse: heart rate at end of run • MaxPulse: maximum heart rate recorded while running All alnalysis in the following report were carried out using R 3.5.1 software.

# R

### *The function lmBoot*

The function *lmBoot* (Appendix A.###) uses bootstrap sampling methods to calculate estimates for the means and confidence intervals of the slope and intercept parameters produced by a linear regression.

The function takes in two arguments:

- inputData: the dataset that will be used to for sampling, where the response variable is in the first column and the remainder of the columns contain the covariates of interest.
- nBoot: The number of bootstrap samples to compute.

The function outputs:

- BootResults: An array with the number of rows equivalent to the nBoot argument and as many columns as there are Beta coeficients; i.e. for the intercept and covariates.
- ConfidenceIntervals: A matrix containing 95% confidence intervals for each parameter estimate.

### *Changes made to lmBoot*

1. The original lmBoot function only produced bootstrap saples for one covariate. This was change so that lmBoot_2 produces bootstrap estimates for a multiple number of covariates and calculates ocnfidence intervals for each parameter estimate.
2. The use of the *lm* function was removed and the beta coefficients rather calculated using matrix clculations.

$$\beta = (X^T X)^{-1} X^T Y$$

3. *forloops* are known to be relatively slow and inefficient. Therefore, the *forloop* was replaced using *sapply* which applies a function to each element of a matrix. The function called *bootLM* was written to carry out the bootstrap algorithm.
4. Parallisation

Table ### illustrates differences in runtime for three different versions of the lmBoot function; the original, an improved version and a parallised version. Each function was timed on howlong it took to resaample 100, 1000, 10000 and 100000 samples.

Table 1: Changes in Runtime (in seconds) of lmBoot

| Samples | lmBoot | lmBoot Improved | lmBoot Parallised |
|---|---|---|---|
| 100 | 0.079 | 0.006 | 0.02 |
| 1000 | 0.941 | 0.052 | 0.026 |
| 10000 | 8.502 | 0.564 | 0.052 |
| 100000 | - | 5.538 | 0.37 |

### *Example analysis using lmBoot*

```
## Loading required package: foreach

## Loading required package: iterators

## Loading required package: parallel
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Table 2: 95% Confidence Intervals

| | 2.5% | 97.5% |
|---|---|---|
| Intercept | 80.997 | 118.090 |
| Age | -0.405 | -0.032 |
| Weight | -0.164 | 0.041 |
| RunTime | -3.240 | -1.864 |
| RestPulse | -0.164 | 0.097 |
| RunPulse | -0.579 | -0.106 |
| MaxPulse | -0.033 | 0.538 |

## SAS

### *The program SASBoot*

The macro program *SASBoot* (Appendix A.###) uses bootstrap sampling methods to calculate estimates for the means and confidence intervals of the slope and intercept parameters produced by a linear regression.

It takes in four agruments:

- NumberOfLoops: the number of bootstrap iterations.
- DataSet: A SAS dataset containing the response and covariate.
- XVariable: The covariate for our regression model (gen. continuous numeric).
- YVariable: The response variable for our regression model (gen. continuous numeric).

The program then outputs:

- ResultHolder: A SAS dataset with the number of rows equivalent to the NumberOfLoops argument and two columns; RandomIntercept and RandomSlope.
- output.rtf: An RTF file containing 95% confidence intervals for the mean, the mean estimate for each parameter and plots of the distributions of the bootstrap parameters.

The function makes use of

- MACRO statements to create a flexible program with input arguments.
- PROC SURVEYSELECT which allows the use of random sampling to generate random samples from a selected or inputed dataset.
- PROC REG to perform a linear regression.

### *Changes made to SASBoot*

The changes made to SASBoot were motivated, in part, by the work of Cassel (2018) in his paper "Don't Be Loopy: Re-Sampling and Simulation the SAS® Way".

1. The %do% loop was first removed and the following simple code was added to PROC SURVEYSELECT:
   *samprate = 1*
   *outhits*
   *rep = &NumberOfLoops*

which ensures that NumberOfLoops samples of the same size as the origional data set are produced recorded.

2. A linear regression using PROC REG was improved by introducing the by-variable REPLICATE. This varibale is automatically produced from PROC SURVEYSELECT to keep track of each new bootstrap sample, then ensuring that the linear regression is run on each sample. Thus, only the Result Holder Dataset was necessary, and there was no need to generate the Temp Dataset.

3. The SASFILE statement was included to upload the dataset to RAM rather than the hard drive before any sampling was carried out so that the dataset does not have to be readin every time a resample needs to be done.

(4. Replaced noprint and ODS listing close - still working on this)

The program was run six times and Table ### displays the runtime (in seconds) for 1000 loops. The code used to measure the run time of the SASBoot program can be found in Appendix A.### (H, 2012).

Table 3: Changes in Runtime of SASBoot

| RegBoot | SASBoot | SASBoot (with rtf output) |
|---|---|---|
| 183.360 | 0.297 | 6.068 |
| 189.973 | 6.087 | 6.087 |

| RegBoot | SASBoot | SASBoot (with rtf output) |
|---|---|---|
| 195.420 | 0.422 | 6.064 |
| 192.835 | 0.313 | 6.041 |
| 192.786 | 0.296 | 5.941 |
| 193.558 | 0.314 | 6.062 |

***Example analysis using SASBoot***

```
#Include plots and interpretation
```

An example analysis was conducted using the fitness datset. A linear model was set up with Oygen as the response and Weight as the covariate. The bootstrap 95% confidence intervals produced by SASBoot were then used to test the null hypothesis that there is no relationship between Oxygen and Weight, i.e. $\beta_i = 0$.

If the confidence interval contains 0, one then fails to reject the null hypothesis and if it does not contain 0 one can reject the null hypothesis.

The confidence interval for the intercept term (36.4824, 73.1590) does not contain 0 which suggests that the estimator is not signicant at the 5% level. The confidence interval for the intercept term (-0.32842, 0.13533) does contain 0 which suggests that the estimator is signicant at the 5% level.

## References

Cassell, D. (2018). Don't Be Loopy: Re-Sampling and Simulation the SAS® Way. [online]. Available at: http://www2.sas.com/proceedings/forum2007/183-2007.pdf [Accessed 26 Oct. 2018].

Donovan, C. (2018). MT5763 Project 2 - code collaboration and computer intensive inference. [Online].

H, J. (2012). To calculate SAS program run time. [online]. Available at: http://sashowto.blogspot.com/2012/06/to-calculate-sas-program-run-time.html [Accessed 26 Oct. 2018].

R Core Team (2018). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. Available at: https://www.R-project.org/.

SAS 9.4, SAS Institute Inc., Cary, NC, USA.

SAS Institute Inc. 2004. Proceedings of the Twenty-Ninth Annual SAS® Users Group International Conference. Cary, NC: SAS Institute Inc.

## Appendix

### A.### The lmBoot Function

```
#The final code used for lmBoot
```

### A.### Code to measure program runtime in SAS

```
%let _sdtm=%sysfunc(datetime());

  Program of interest to be timed

%let _edtm=%sysfunc(datetime());
%let _runtm=%sysfunc(putn(&_edtm - &_sdtm, 12.4));
%put It took &_runtm seconds to run the program;
```

### A.### The SASBoot Program

```
#The final code used for SASBoot
```