The Bootstrap - Numerical Procedures and Applications

Seminar Paper submitted

to

Prof. Dr. Brenda López-Cabrera

Humboldt-Universität zu Berlin
School of Business and Economics
Climate, Weather and Energy Analysis

by

Erin Sprünken
(581608)

in partial fulfillment of the requirements
for the Seminar
Numerical Introductory Course
Berlin, March 3, 2021

Contents

List of Tables

List of Figures

1 Motivation

Since von Mises provided the statistical definition of probability in terms of relative frequency, the group of frequentist statistics has emerged and dominated the field for many years. However, frequentist statistics rely on large samples and asymptotic results or behavior. This raises the question of how well do such methods work in finite samples and especially small samples. For example, clinical researchers often have to deal with small samples in early studies, due to the missing information on toxicity and biokinetic behavior. Thus, classical frequentist statistics become unreliable. This motivates resampling techniques, to which bootstrap belongs. The idea is to overcome small sample problems by resampling from this sample and obtain artificial large samples by mimicking the distribution of the estimator. Although this paper will only present basic problems, further research will lead to more complex problems, which we will mention in the next section. Specifically, rank and pseudo-rank procedures are of current interest in this field as well as clustered data. For example, the COVID-19-pandemic leads to small clusters of data where multiple (statistical) test problems arise which might be solved by resampling techniques. Furthermore, since R is a popular programming language amongst statisticians we want to adress computational complexity and speed and compare it to lower-level languages, since speed becomes more and more of interest due to more complex problems (such as random forests) and since bootstrap requires a lot of computations.

2 Related Literature

In this section we provide an overview of literature relating to the topic of the bootstrap. Important to mention is the article of Efron (1979) who generalised the jackknife principle to what is the classical bootstrap today. Since then, many advancements, generalizations and applications were conducted by numerous authors. Kerns (2010) provides a general overview of resampling techniques in chapter 13 and Hastie et al. (2008) describe and relate bootstrap methods to Maximum Likelihood, Bayesian Inference and Machine Learning techniques in chapter 8.

Konietschke and Pauly (2014) find that bootstrap and permutation methods for matched pairs give valid asymptotic results even for different distributions which have no treatment effect under the null hypothesis. Their paper relates to estimation of t-type test statistics for paired samples and overcomes the problems of small sample sizes, heterogeneity and different distributions. Friedrich et al. (2017) find valid results applying the so-called wild bootstrap method to problems of repeated measurements. The nonparametric bootstrap approach delivers better results than WALD or ANOVA statistics, which are especially useful for small sample sizes. Umlauft et al. (2019) find that wild-bootstrapping ranked-based procedures overcome assumptions about normality or homogeneity in general factorial measure designs and deliver asymptotically correct multiple contrast tests. Furthermore, Bathke et al. (2018) find that parametric bootstrap can, under minimal assumptions, overcome the problems of normality assumptions or equal covariance matrices in multivariate factorial designs.

Prášková (2002) and Politis and Romano (1994) describe bootstrap methods for timeseries, which again are used in applied research. For example, Petukhina et al. (2018) refer to the latter one cominbing portfolios. Petukhina and Sprünken (2020) use a parametric bootstrap approach to numerically compute optimal CVaR (Conditional Value-at-Risk) portfolios.

3 Applications

As already mentioned before, various applications for the bootstrap method exist. For example, early clinical trials usually suffer from small sample sizes. For example, consider a very early stage for a new drug. Since it's effects are only considered theoretically, ethical standards would forbid testing such a drug on a large sample of patients. On the other hand, small samples are not reliable when it comes to statistical inference. However, valid inference is necessary to judge about approving or declining the public usage of this treatment. The solution lies in bootstrapping a small sample. For example, the clinical trial consists of 16 patients receiving the new treatment or the placebo, this leads to a two-sample of matched pairs design. Here, bootstrap solves this small sample problem and leads to reliable results, see Konietschke and Pauly (2014) and Friedrich et al. (2017).

In finance, especially portfolio management, parametric bootstrap and resampling in general can provide large samples when analytic solutions are not possible. Consider the following minimization problem (Condition Value-at-Risk):

$$\min_{x} \quad -\frac{1}{1-\alpha} \int_{x^\intercal \mu \leq -VaR_\alpha(x)} x^\intercal \mu f\left(x^\intercal \mu \mid x\right) \mathrm{d}x^\intercal \mu. \tag{1}$$

This has no analytic solutions. However, a portfolio manager wants to obtain the portfolio weights x. Petukhina and Sprünken (2020) solve this problem by sampling a vector of weights x from the uniform distribution n_b times and compute n_b (empirical) CVaRs and choose the vector x of weights which corresponds to the smallest CVaR of all.

Bagging, as mentioned and described in Hastie et al. (2008), uses bootstrapping to improve Machine Learning techniques. By bootstrapping and its property of drawing with replacement, one creates n_b samples on which an algorithm can be trained and estimated. The n_b models are then aggregated into one (for example by majority vote), see Breiman (1996). This parallel design is distinct to sequential designs such as Boosting. Random Forests are a special case of this, since it grows n_b trees instead of only one and thus significantly reduces the variance of the tree model, see Ho (1995) and Breiman (2001).

Generally, bootstrapping can be applied to many fields of statistical and applied sciences.

4 Methodology

This section will cover formal aspects of the bootstrapping and is divided into two parts. The first one is a short overview of advantages and disadvantages. The second part will briefly describe mathematical aspects of the different applications covered within this seminar paper.

4.1 Advantages and Disadvantages

One of the main advantages given by bootstrapping is the sample size. Usually, especially for frequentist statistics, small sample sizes are a huge problem since many applications or theorems rely on large samples or asymptotic behaviour. However, there are many situations where the researcher has to deal with a small sample size. In such a setting, classical statistics quickly become unreliable. Here, resampling provides a solution to overcome this issue.

However, if the original sample from which resampling is conducted is a bad sample, the resampling techniques will not help any further. For example, consider a sample size of four and two observations of this are outliers, then the probability of resampling the outliers is quite high, although they might be unlikely in general. So, the resampling (as the name implies) relies on the sample as well and this is the main disadvantage. Furthermore, bootstrapping can require a lot of computations and thus cost time. Thus, bootstrapping, although asymptotically consistent, cannot provide general finite sample guarantees for estimators as the results depend on the small sample.

4.2 Mathematical Aspects

This subsection deals with the formal aspects of the algorithm, specifically how and why it works.
The general idea of bootstrapping is the following Pseudocode (for $nboot$ number of resampling iterations):

1. Fix the data x
2. FOR i IN 1 TO $nboot$ DO
   (a) Sample $x\star$ from x
   (b) Compute the statistic of interest for $x\star$
   (c) Save the respective statistic at the i-th position of a vector $T\star$
3. Compute Mean (or Median, Confidence Intervals, or any other statistic) of the vector $T\star$.

Mathematically, the bootstrapping algorithm makes use of the strong law of large numbers and creates an empirical distribution of the statistic of interest to mimic the true but unknown distribution.

$$\lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} X_i \quad \overset{a.s.}{\to} \quad \mathbb{E}[X]$$
$$\lim_{n\to\infty} F_n(x) \quad \overset{a.s.}{\to} \quad F^X(x).$$

Note, that the first one (law of large numbers) relies on pairwise independence if $X \in \mathcal{L}^1$ or at least uncorrelatedness if $X \in \mathcal{L}^2$. The second equation is the convergence of the empirical distribution function $F_n(x)$ to the true distribution function $F^X(x)$. This follows immediately from the law of large numbers.

There are two versions of bootstrap incorporated in this paper: The nonparametric bootstrap and the wild bootstrap (Rademacher-version). The nonparametric bootstrap proceeds pretty much in the way described above, where in the resampling step the data is resampled with replacement and each observation has an equal probability of getting into the bootstrap sample. On the other hand, the wild bootstrap proceeds differently. The following Pseudocode demonstrates the Rademacher wild bootstrap:

1. Fix the data x
2. FOR i IN 1 TO $nboot$ DO
   (a) Sample vector z from $\{-1, 1\}$ with replacement and of same length as x
   (b) Create bootstrap sample $x\star = z \odot (x - \bar{x})$ (Hadamard product)
   (c) Compute the statistic of interest for $x\star$
   (d) Save the respective statistic at the i-th position of a vector $T\star$
3. Compute Mean (or Median, Confidence Intervals, or any other statistic) of the vector $T\star$.

Summary Statistics

As mentioned before, there are three types of statistics we are going to analyze in the empirical section of this paper. The first type are summary statistics, computing the following metrics: Minimum, 25%-quantile, Median, Mean, 75%-quantile, Maximum and Standard Deviation. Of course, the Minimum and the Maximum are not really able to be resampled, since the absolute Minimum (Maximum) that could be obtained in the resampling is the actual Minimum (Maximum) of the original sample. The quantiles (and the Median is the 50%-quantile) are computed in the default way of the respective R-function *quantile*, see R Core Team (2020).

The Standard Deviation is the squareroot of the corrected sample Variance. Since the original data is

centered for the wild bootstrap, finally the original's sample mean has to be added to the bootstrapped location parameters. Sloppily said, this procedure of estimating parameters can be thought of as a nonparametric maximum likelihood approach, although this is not quite precisely formulated.

t-type test statistics

Although simple, the t-type tests are a nice example for bootstrapping. However, this procedure can be extended to all possible test procedures one can think of (Lilliefors, Wilcoxon-Signed-Rank, Friedman, etc.). The idea is basically the same as before and the algorithm works in the same way. However, the statistic of interest is the t-statistic. For the one-sample case that is the following equation, see Toutenburg and Heumann (2008):

$$T(X) \quad = \quad \frac{\bar{X} - \mathbb{E}[\bar{X}]}{\hat{\sigma}}\sqrt{n}.$$

When bootstrapping this statistic, we aim to mimic it's distribution and finally refer to the quantiles of the bootstrap distribution to make a decision whether to reject or not reject the null hypothesis. However, in the resampling iteration we need to compute a slightly different statistic, since there is a setting of a bootstrap sample given the fixed data:

$$T(X\star \mid X) \quad = \quad \frac{\bar{X}\star - \mathbb{E}[\bar{X}\star \mid X]}{\hat{\sigma}}\sqrt{n},$$

where $\mathbb{E}[X\star \mid X] = \bar{X}$, so the conditional expectation of the bootstrap sample is the sample mean of the original data. For the two-sample t-test the statistic looks different as well, since not only the difference of means has to be taken into account, but the pooled standard deviation, too:

$$\begin{aligned} X'\star \quad &:= \quad X_2\star - X_1\star \\ T(X'\star \mid X) \quad &= \quad \frac{\bar{X'}\star - \mathbb{E}[\bar{X'}\star]}{\sqrt{\frac{\hat{\sigma_1^2}\star}{n_1} + \frac{\hat{\sigma_2^2}\star}{n_2}}}, \end{aligned}$$

where n_1 and n_2 refer to the sample sizes. Also, under the null hypothesis when testing for equality of means the expectation of mean difference is zero (i.e. $\mathbb{E}[\bar{X'}\star] = 0$). The resampling procedure for nonparametric and wild stays the same.

Regression Coefficients

Bootstrapping the regression is also an issue where bootstrap can provide significant improvements to estimation. Especially the wild bootstrap has its origins in the field of regression analysis, see Wu (1986). Furthermore, it is continuously studied, see Davidson and Flachaire (2008). As before,

bootstrap grants the possibility to obtain point- and set-estimates for the coefficients. According to Fox and Weisberg (2018), if the matrix of covariates X is random (or at least not fixed), one has to resample the whole pairs of $(Y_i, X_{i\cdot})$ and estimate the linear model for the new data. We apply such an approach in our nonparametric bootstrap version of the linear regression. However, if X is assumed to be deterministic/fixed, then it's enough to resample the residuals of the regression. This is, where we apply the wild bootstrap in the following way:

1. Fix the data (Y, X)
2. Estimate linear model for (Y, X) and predict fitted values $\hat{Y}$
3. Save residuals $\epsilon = Y - \hat{Y}$
4. FOR i IN 1 TO $nboot$ DO
   (a) Sample vector z from $\{-1, 1\}$ with replacement and of same length as ϵ
   (b) Create bootstrap sample $\epsilon\star = z \odot \epsilon$ (Hadamard product)
   (c) Create bootstrap sample $Y\star = \hat{Y} + \epsilon\star$
   (d) Estimate linear model $(Y\star, X)$ and save coefficients $\beta_i\star$ of i-th iteration
5. Compute Mean (or Median, Confidence Intervals, or any other statistic) of the array $\beta\star$.

5 Simulation Settings

In this section we present the simulation settings we use to demonstrate bootstrap techniques. For all different problems we adress, 25 combinations of bootstrapping iterations ($nboot$) and sample sizes (n) are computed: $nboot = \{10, 100, 500, 1000, 10000\} \times n = \{5, 10, 50, 100, 200\}$. When it comes to measuring the speed of the implemented algorithm, we use microbenchmarking, calling the function with specified parameters 100 times. Furthermore, the following problem-specific simulation settings are demonstrated within this paper.

5.1 Summary Statistics

Computing the summary statistics is mainly to demonstrate the accuracy of bootstrap techniques. This is, how precisely can we estimate paramters such as mean, median or standard deviation from a sample using bootstrap. To measure this, we compute the MSE and MAE of bootstrap estimates for randomly drawn samples from a $N(0, 1)$ population for each of the above named combinations. Recall the definitions of MSE and MAE:

$$\begin{aligned} MSE &:= \frac{1}{nboot} \sum_{i=1}^{nboot} (\hat{\vartheta}_i - \vartheta)^2 \\ MAE &:= \frac{1}{nboot} \sum_{i=1}^{nboot} |\hat{\vartheta}_i - \vartheta|, \end{aligned}$$

where ϑ is the parameter of interest. We acknowledge, that the choice of $N(0, 1)$ population is arbitrary and could be replaced by any other and is only motivated by it's simplicity and popularity.

5.2 Linear Regression

In order to demonstrate computational complexity of bootstrap techniques and advantages of lower-level languages such as C++ over high-level languages such as R we examine a simple linear regression problem with four variables from a pre-specified data generating process (DGP). Furthermore, we define two accuracy measures which will be computed for an arbitrary setting, but since accuracy is not the main interest of this part of the paper it will not be done for the whole set of combinations. Let $B = \{\beta_1, ..., \beta_i, ..., \beta_k\}$, then define the Compund MSE and MAE as:

$$\begin{aligned} cMSE &:= \frac{1}{|B|} \sum_{i=1}^{|B|} \frac{1}{nboot} \sum_{j=1}^{nboot} (\hat{\beta}_{ij} - \beta_i)^2 \\ cMAE &:= \frac{1}{|B|} \sum_{i=1}^{|B|} \frac{1}{nboot} \sum_{j=1}^{nboot} |(\hat{\beta}_{ij} - \beta_i)|. \end{aligned}$$

Further, we defined an arbitrary DGP with the following parameters and data: X_1: Discrete uniform from 30 to 100, X_2: U(50, 230), X_3: log(N(100, 7.5)), X_4: N(100, 7.5) and

$$\beta \quad = \quad (-16.046722, -9.140887, 10.661128, -11.014303).$$

Thus, we have: $Y = X\beta + \epsilon$, where $\epsilon_i \sim N(0, 1)$. Of course, specific bootstrapping techniques could work for more specific problems such as heteroskedastic errors or different data settings, but since we want to measure computational complexity and work on different problems in this paper we leave this for further research.

5.3 t-test

Finally, we want to adress the application of bootstrap techniques in statistical test problems. For this, we extensively simulate the t-test. We note, that the research on statistical tests is much larger, but due to its extensive application in empirical research the t-test works well to show properties of bootstrap. However, we definitely recommend further research on other test-problems. Our simulation uses both one- and two-sample problems and, as before, both nonparametric and wild bootstrap. To simulate the Type-I-Error we create four different populations and test against their expectation ($H_0 : \mu = \mu_0$) and ignore the required normality to gain some insight about how well the test reaches it's significance level, which is interesting for practioneers. Our actual populations are $N(0, 1), Pois(5), Exp(3)$ and $\chi^2(2)$. When it comes to Type-II-Error (or Power, which is $1 - \beta$ and thus directly related to it), we only use $N(\mu, \sigma)$ under the alternative and vary the true mean of the population. In the two-sample case we vary the difference in means for the power study.

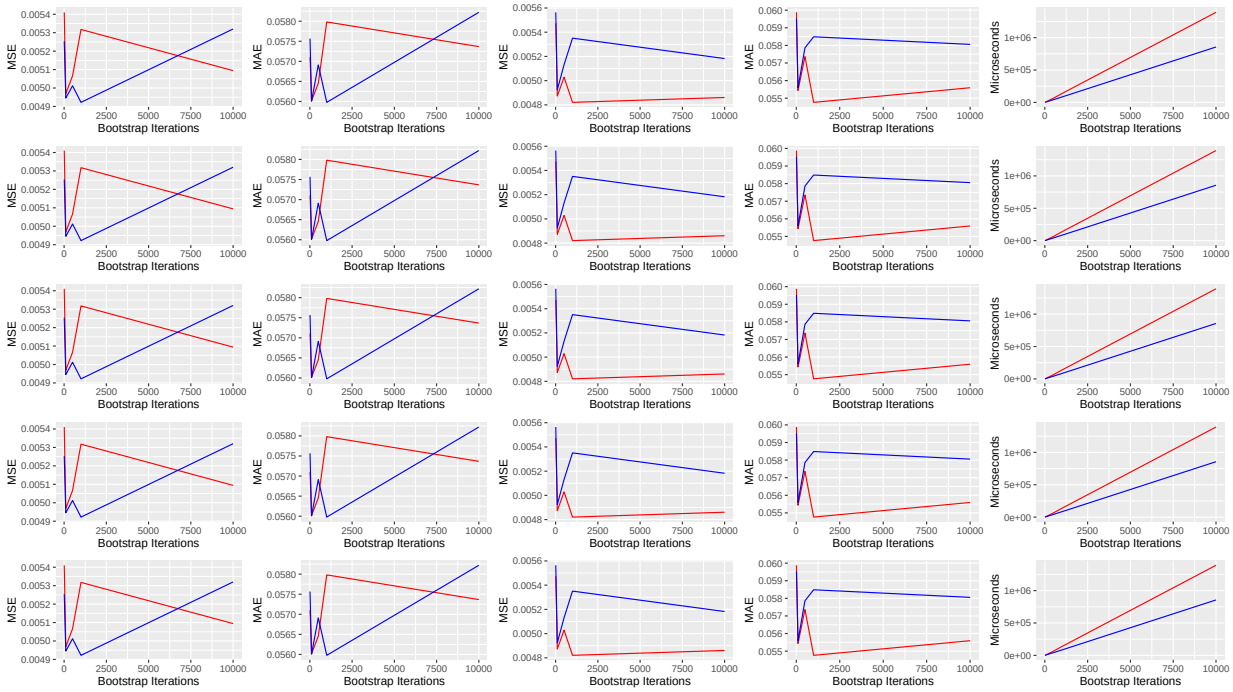

Figure 1: Red Line: Nonparametric Bootstrap, Blue Line: Wild Bootstrap. Each row corresponds to a different number of sample size from $(5, 10, 50, 100, 200)$, the first two columns correspond to the R-Version, whereas the third and fourth to the C++ version. The last column represents the median computation time (with red being the R and blue the C++ algorithm).

6 Empirical Analysis

This section covers the empirical results using the methods mentioned earlier. Our interest lies in the aspects of accuracy, computation time and complexity, and for tests the errors of type I and II.

6.1 Accuracy

The first point of interest is whether bootstrap techniques reach an acceptable level of accuracy. In this subsection we will assess this with the estimation of summary statistics (quantiles, mean and standard deviation). However, such a level of accuracy can be generalized easily without major drawbacks. Furthermore, we will adress the computation time of the bootstrap with respect to sample size and bootstrap iterations. Nonetheless, the computation time is not of much interest in such a setting and will be discussed more deeply in the following subsection.

As already mentioned, we measure accuracy with typical MSE and MAE. Figure 1 shows how MAE and MSE behave in estimating the mean of a sample using resampling methods. A complete table for summary statistics estimation can be found in Appendix A, see tables 1, 2, 3 and 4.

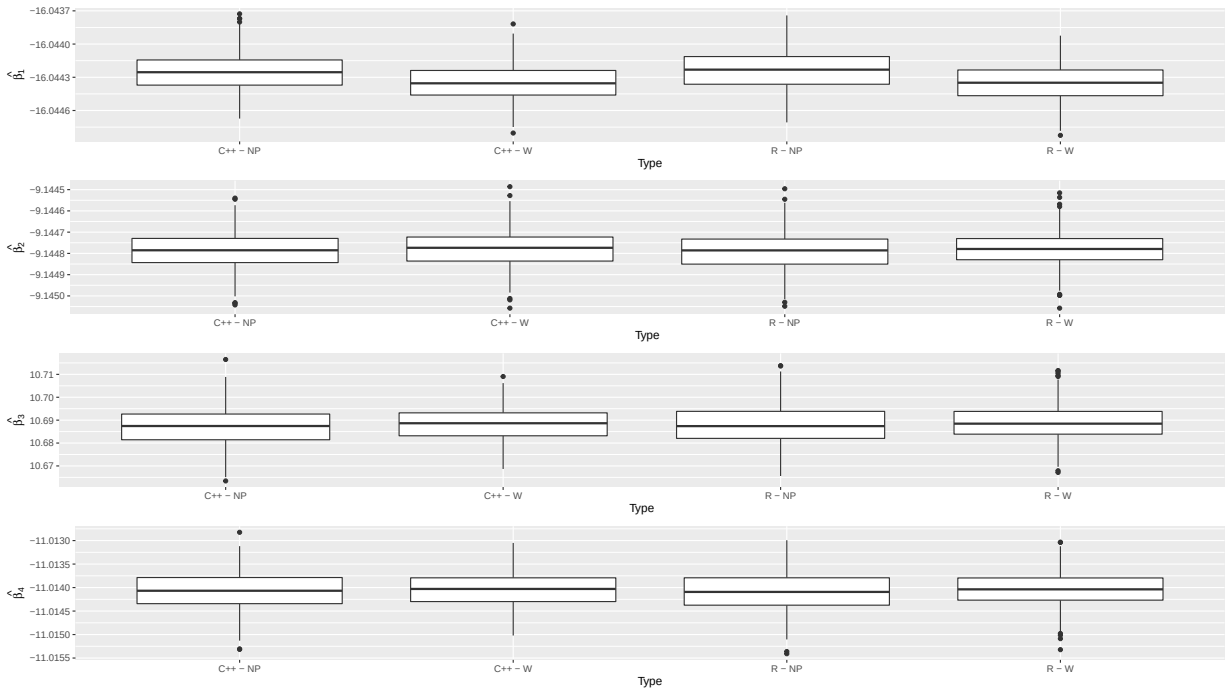

Figure 2: Boxplot for all coefficients with respect to the bootstrap technique used

6.2 Computational Complexity

In this subsection we want to adress one of the main advantages of C/C++. This is the speed of computations. Whereas R (or other languages such as python) are very high level languages which allow the user to do scientific analyses easily, lower level languages such as C and C++ offer a great advantage in speed for the cost of intuitiveness. To demonstrate such issues we present a simulated linear regression problem as mentioned in section 5. In the following, we also show a figure of accuracy for the coefficients, although we used a different accuracy measure, namely the compound MSE and compound MAE as mentioned in the same section 5. Since the accuracy is related to the accuracy of summary statistics and also dependent on the accuracy of the specific algorithm itself, there is not much additional error due to bootstrap (no systematic bias), see table 5 in Appendix B.

We see, that the estimated coefficients are quite close to the true coefficients. Also, we see that wild bootstrap provides slightly better results than nonparametric bootstrap. Since the origin of wild bootstrap is in regression analysis, this is not very surprising. However, the following plot shows how R and C++ differ in terms of computation time.

Although the scale is already in log(microseconds) there is a significant difference between those two. The data used for the plot is the median time for each function, for full summary statistics of the time see tables 6 and 7 in Appendix B. For both languages the time necessary increases with increased bootstrap iterations, which is only natural. However, even for 10000 bootstrap iterations C++ is

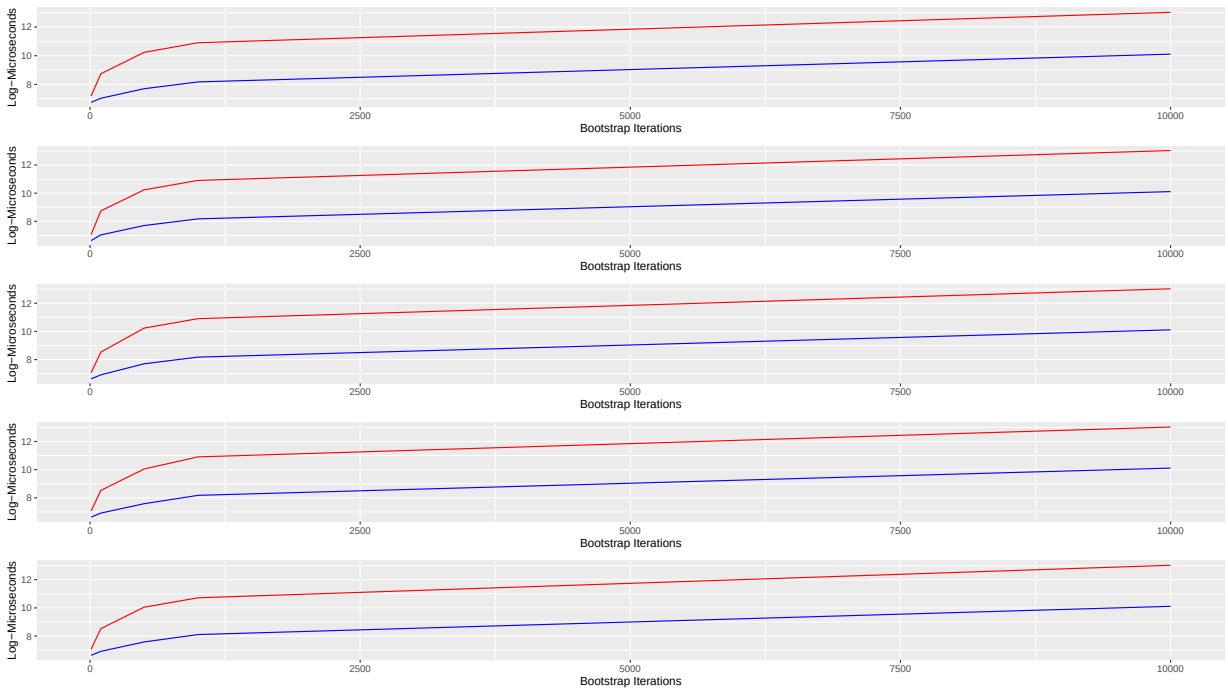

Figure 3: Median speed of linear regression: Blue Line: C++, Red Line: R. Each row corresponds to a different sample size from $(5, 10, 50, 100, 200)$.

roughly eight times faster than its R counterpart. Since the simulated problem is of quite simple nature this highlights a huge advantage of lower level languages. We expect that even for more complex linear regression problems (more variables) the computation time increases and that problems which are more complex in their nature themselves would have an even higher advantage in using low level languages (such as generalized linear models from exponential families, classifier problems, dimension reduction techniques) and thus recommend more research in that area.

6.3 Type-I and -II Errors

Finally, we want to demonstrate the utility of bootstrap in a specific area: statistical tests. We do this exemplary with the t-test (one and two samples), although we do not show how permutation tests compare to these, which work especially well for two-sample problems. It is noteworthy, that resampling is an important aspect in the field of statistical tests since there exist a large number of research fields where such tests are necessary but suffer from small sample sizes (e.g. translational studies in clinical research). Ideally, a resampled test should reach a Type-I-Error equal to the significance level and also maintain a high power, just as we expect this from a statistical test under large sample situations (or at least asymptotically in theory). Thus, we show Type-I-Error and power for one- and two-sample t-tests

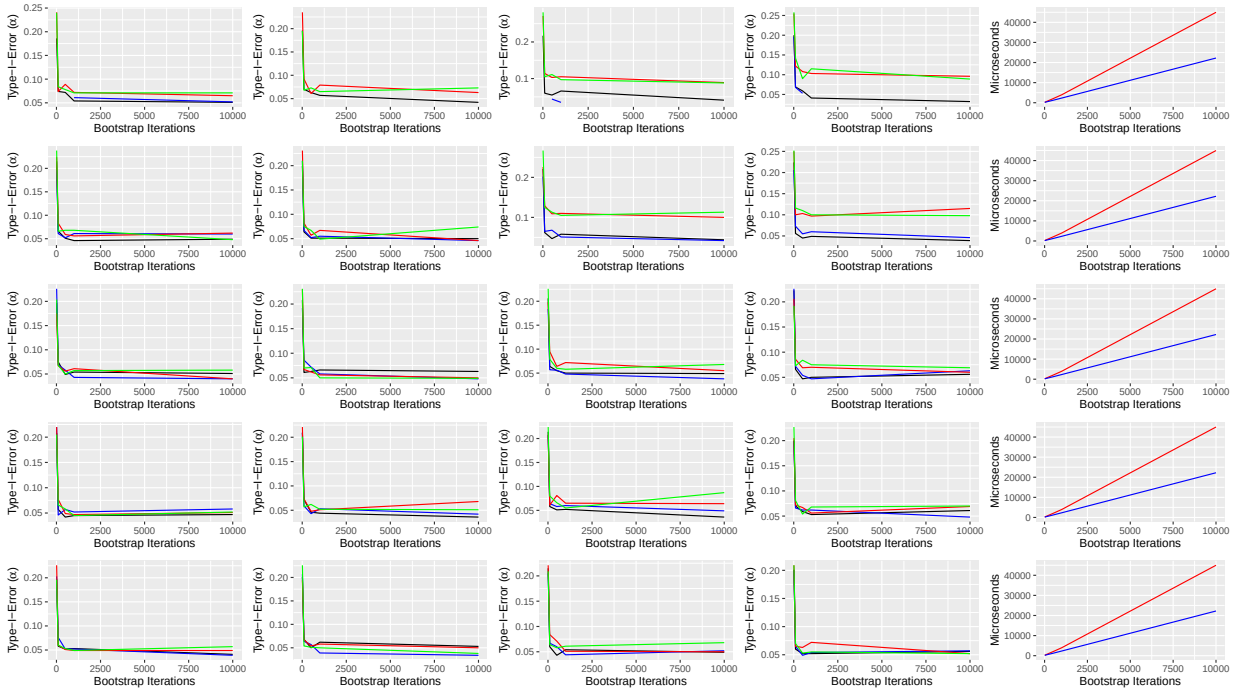

Figure 4: Type-1-Error for four different populations in a one-sample setting: Green: $\chi^2(2)$, Red: Exp(3), Blue: Pois(5), Black: N(0,1). Each row corresponds to a different sample size from $(5, 10, 50, 100, 200)$. The first two columns correspond to the nonparametric bootstrap in R, C++, whereas the third and fourth column correspond to the wild bootstrap in R, C++. The last column depicts the median computation time.

with respect to different sample sizes and different bootstrap iterations.

It can be seen in figures 4 and 5 that the χ^2 and exponential distribution have some struggle to reach the alpha-level, although with increasing sample size they also converge to the significance level. Furthermore, with increasing bootstrap iterations the test converges to the significance level for all samples. Also, the significance level is reached quite fast, raising the question about what should be an ideal number of maximum bootstrap iterations. The figures show that the wild bootstrap performs slightly worse in terms of convergence than the nonparametric bootstrap. It can be seen that in the one-sample case convergence for χ^2 and exponential distribution does not really occur for the first two sample sizes, but do in the two-sample case. A reason for this might be that in the two-sample case two estimators from the same distribution are compared rather than one sample mean against a fixed value. Finally, the computation time is growing linear for both R and C++, but C++ always beats R in the median computation time. As already mentioned before, this is not that important for such trivial problems but can be of high importance for more complex problems.

Regarding the power (which can be directly converted to Type-II-Error), the figures 6 and 7 show the

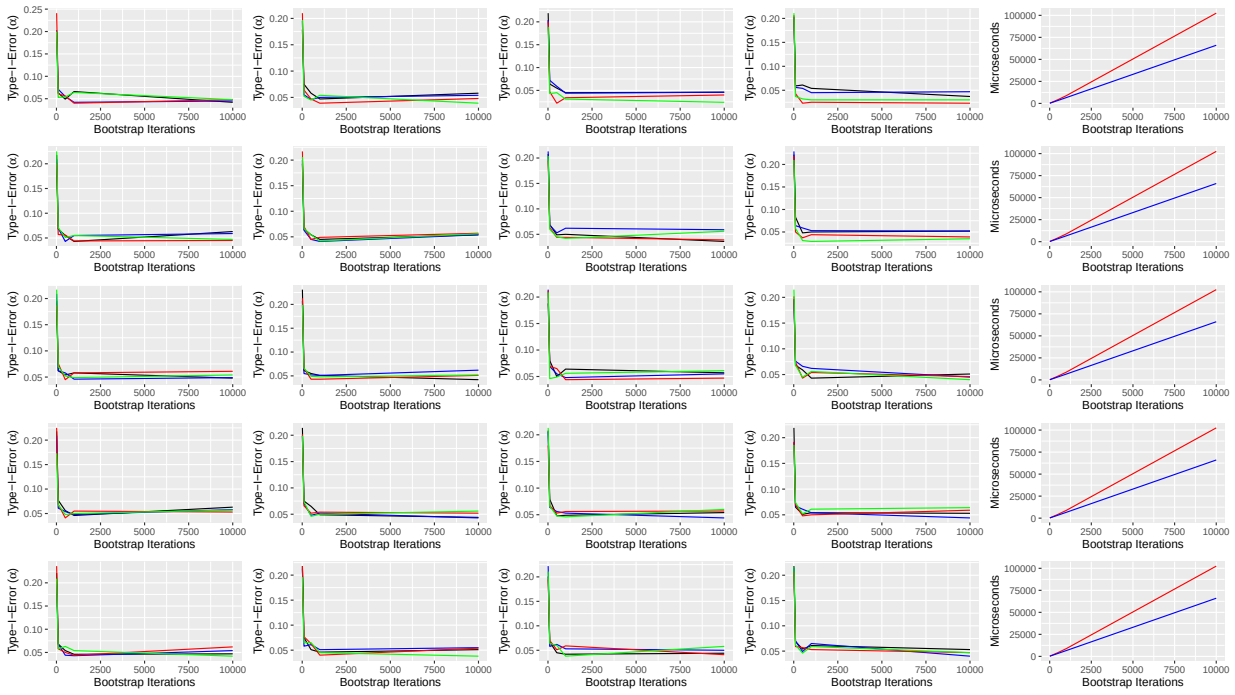

Figure 5: Type-1-Error for four different populations in a two-sample setting: Green: $\chi^2(2)$, Red: Exp(3), Blue: Pois(5), Black: N(0,1). Each row corresponds to a different sample size from $(5, 10, 50, 100, 200)$. The first two columns correspond to the nonparametric bootstrap in R, C++, whereas the third and fourth column correspond to the wild bootstrap in R, C++. The last column depicts the median computation time.

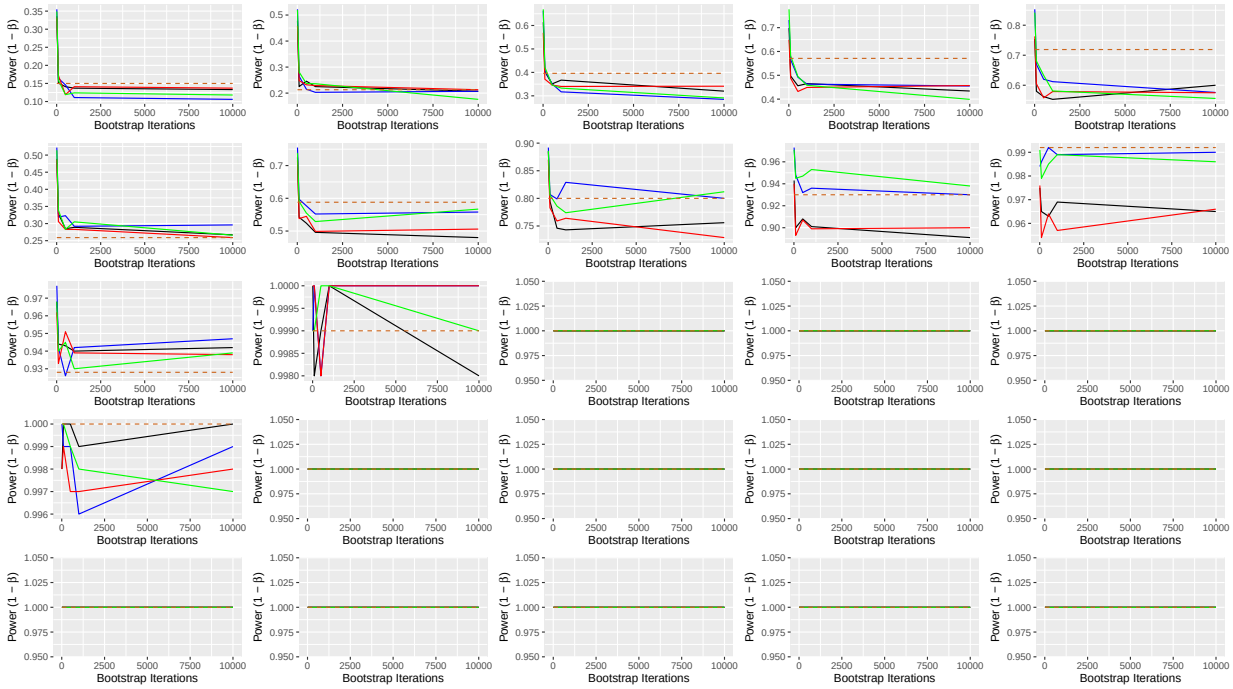

Figure 6: Power for five different population means in a one-sample setting: Black: Nonparametric (R), Blue: Wild (R), Red: Nonparametric (C++), Green: Wild (C++). Each row corresponds to a different sample size from $(5, 10, 50, 100, 200)$. Each column corresponds to a different population mean μ for a normal distribution $N(\mu; \sigma = 1)$, with means $\mu = (0.5, 0.75, 1, 1.25, 1.5)$ tested against a mean under H_0: $\mu_0 = 0$. The dashed brown line refers to a simple t-Test without bootstrap.

evolution of the power regarding effect size and sample size, all dependend on bootstrap iterations. From the left to right, the power increases, said in other words with an increasing effect size the power increases. This is only natural, since with increasing actual effect size it becomes more unlikely not to reject the null hypothesis. Furthermore, with increasing sample size the power increases very fast, with sample size of 50 already reporting a power of around 1 for the first effect size in the one-sample problem and at least 0.7 for the two-sample problem. However, it should be noted that usually, for such a simple test problem with a sample size of 50 does not require resampling techniques. There seems to be no systematic difference regarding the different bootstrap techniques or languages.

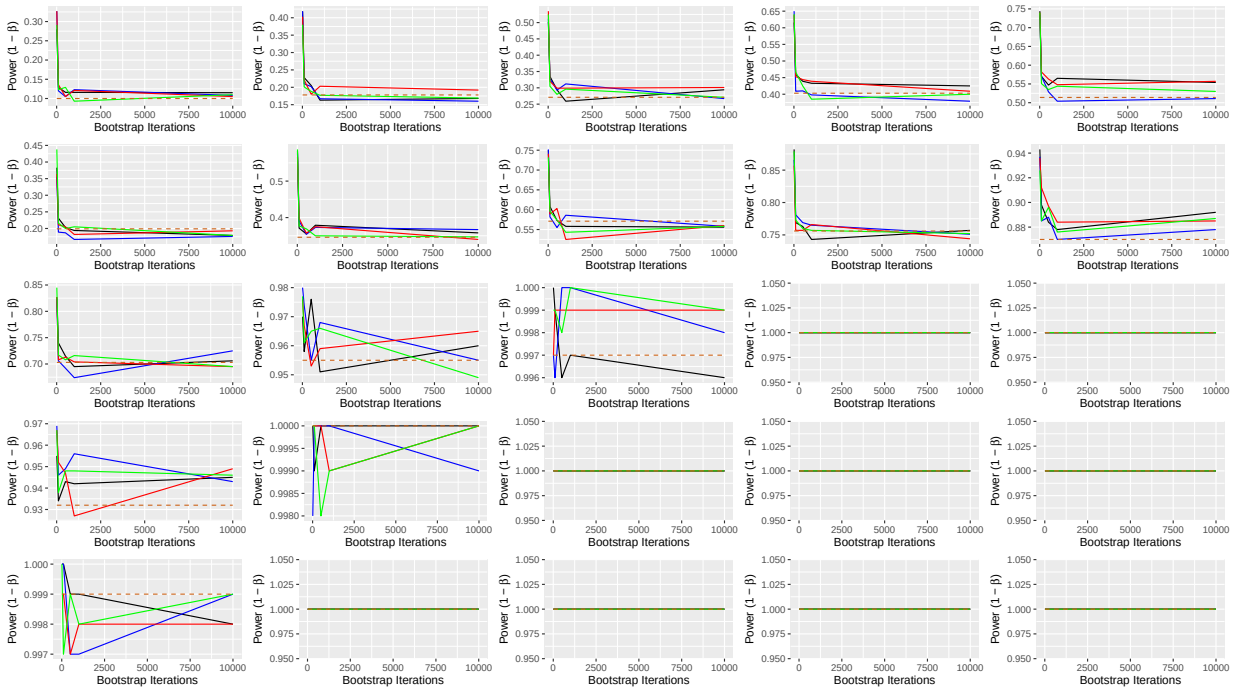

Figure 7: Power for five different population means in a one-sample setting: Black: Nonparametric (R), Blue: Wild (R), Red: Nonparametric (C++), Green: Wild (C++). Each row corresponds to a different sample size from $(5, 10, 50, 100, 200)$. Each column corresponds to a different population mean μ for a normal distribution $N(\mu; \sigma = 1)$, with means $\mu = (0.5, 0.75, 1, 1.25, 1.5)$ tested against a $N(0, 1)$ population. The dashed brown line refers to a simple t-Test without bootstrap.

7 Conclusion

Finally, we want to draw a conclusion. In our empirical section we found what was expected from a theoretical point of view. Bootstrap techniques can accurately estimate parameters and statistics even under small sample sizes. Furthermore, an extensive simulation of testing the mean showed that bootstrap-versions of the t-Test converge fast to the significance level in terms of Type-I-Error, although for some populations this took longer than for others, dependent on the sample sizes. This highlights one of the main drawbacks of bootstrap in general, namely that the sample itself has to be at least a little bit representative in order to obtain proper bootstrap results. Nonetheless, with increasing sample sizes and increasing bootstrap iterations the significance level is reached. Also, the power of the t-Test was roughly the same in its bootstrap versions as it was in its actual version for nearly all situations. Both aspects demonstrate that bootstrapping tests can be a proper method in practice. Also, bootstrap is more flexible in practice in the sense that one has not to rely on the theoretical distribution function of the test statistic, since this is created with an empirical distribution function (which is again, in theory, converging almost sure to the actual distribution). With a simple regression problem we demonstrated computational complexity of bootstrap and applied different language implementations. Even for such a simple problem the computation time was quite high and increased in this specific case exponentially with the number of bootstrap iterations. As expected, the implementation in a lower-level language, namely C++, provided a huge advantage against R, being roughly 8 times faster on median time. Although the time was still below a single second, this can be generalized since there exist many problems requiring much more computation time than a simple linear regression estimation. Since this paper only provided a basic overview, we highly recommend a deeper look on topics such as different versions of the wild bootstrap, since we only applied the Rademacher-version, as well as parametric bootstraps for specific problems. Also, many test-problems are of interest, for example multiple contrast tests, F-Test, Anderson-Darling, Kolmogorov-Smirnov or rank-based tests such as Mann-Whitney-U test and other effect-size tests. Even though there is a lot of research in that field, current problems in applied fields require new statistical methods as well. For example, the ongoing COVID-19-pandemic requires statistical tests for vaccines and treatments, as well as epidemiological test procedures for clusters of infected people. All of these begin with small samples (especially the first and second, until they reach phase II and III in the clinical trials). Thus, bootstrap is still a current issue of research and provides a lot of sub-topics to study.

References

Bathke, A., Friedrich, S., Pauly, M., Konietschke, F., Staffen, W., Strobl, N., and Höller, Y. (2018). Testing Mean Differences among Groups: Multivariate and Repeated Measures Analysis with Minimal Assumptions. *Multivariate Behavioral Research* 53, pp. 384 –359.

Breiman, L. (1996). Bagging Predictors. *Machine Learning* 24, pp. 123 –140.

— (2001). Random Forests. *Machine Learning* 45, pp. 5 –32.

Davidson, R. and Flachaire, E. (2008). The wild bootstrap, tamed at last. *Journal of Econometrics* 146, pp. 162 –169.

Efron, B. (1979). Bootstrap Methods: Another Look at the Jackknife. *The Annals of Statistics* 7, pp. 1 –26.

Fox, J. and Weisberg, S. (2018). An R Companion to Applied Regression.

Friedrich, S., Konietschke, F., and Pauly, M. (2017). A wild bootstrap approach for nonparametric repeated measurements. *Computational Statistics & Data Analysis* 113, pp. 38 –52.

Hastie, T., Tibshirani, R., and Friedman, J. (2008). The Elements of Statistical Learning.

Ho, T. K. (1995). Random decision forests. *Proceedings of 3rd International Conference on Document Analysis and Recognition* 1, pp. 278 –282.

Kerns, G. J. (2010). Introduction to Probability and Statistics Using R.

Konietschke, F. and Pauly, M. (2014). Bootstrapping and permuting paired t-test type statistics. *Statistics and Computing* 24, pp. 283 –296.

Petukhina, A. and Sprünken, E. (2020). Evaluation of Multi-Asset Investment Strategies with Digital Assets. Available at SSRN: `https://ssrn.com/abstract=3664219`.

Petukhina, A., Trimborn, S., Härdle, W. K., and Elendner, H. (2018). Investing with Cryptocurrencies - evaluating their potential for portfolio allocation strategies. Available at SSRN: `https://ssrn.com/abstract=3274193`.

Politis, D. N. and Romano, J. P. (1994). Large sample confidence regions based on Subsamples under minimal assumptions. *The Annals of Statistics* 22, pp. 2031 –2050.

Prášková, Z. (2002). Bootstrap in Nonstationary Autoregression. *Kybernetika* 38, pp. 389 –404.

R Core Team (2020). R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing. Vienna, Austria. URL: `https://www.R-project.org/`.

Toutenburg, H. and Heumann, C. (2008). Induktive Statistik - Eine Einführung mit R und SPSS.

Umlauft, M., Placzek, M., Konietschke, F., and Pauly, M. (2019). Wild bootstrapping rank-based procedures: Multiple testing in nonparametric factorial repeated measures designs. *Journal of Multivariate Analysis* 171, pp. 176 –192.

Wu, C. F. J. (1986). Jackknife, Bootstrap and Other Resampling Methods in Regression Analysis. *Annals of Statistics* 14, pp. 1261–1295.

Appendix A

nboots	n	NP 0.25 Q	NP 0.5 Q	NP Mean	NP 0.75 Q	NP Sd	W 0.25 Q	W 0.5 Q	W Mean	W 0.75 Q	W Sd
10	5	0.32	0.24	0.21	0.30	0.12	0.30	0.22	0.21	0.30	0.12
100	5	0.28	0.22	0.20	0.29	0.12	0.31	0.23	0.23	0.31	0.12
500	5	0.30	0.24	0.23	0.30	0.11	0.27	0.20	0.19	0.28	0.11
1000	5	0.29	0.22	0.20	0.28	0.12	0.29	0.19	0.19	0.26	0.12
10000	5	0.29	0.22	0.21	0.29	0.12	0.28	0.19	0.19	0.27	0.12
10	10	0.15	0.13	0.11	0.16	0.06	0.15	0.12	0.11	0.16	0.05
100	10	0.14	0.12	0.11	0.15	0.06	0.15	0.11	0.11	0.15	0.05
500	10	0.14	0.11	0.10	0.14	0.06	0.13	0.09	0.09	0.14	0.06
1000	10	0.15	0.12	0.10	0.14	0.06	0.14	0.10	0.10	0.13	0.05
10000	10	0.15	0.12	0.10	0.13	0.05	0.15	0.10	0.10	0.15	0.06
10	50	0.04	0.03	0.02	0.04	0.01	0.03	0.02	0.02	0.03	0.01
100	50	0.03	0.03	0.02	0.03	0.01	0.03	0.02	0.02	0.03	0.01
500	50	0.03	0.03	0.02	0.03	0.01	0.03	0.02	0.02	0.03	0.01
1000	50	0.03	0.03	0.02	0.03	0.01	0.03	0.02	0.02	0.03	0.01
10000	50	0.03	0.03	0.02	0.03	0.01	0.03	0.02	0.02	0.03	0.01
10	100	0.02	0.02	0.01	0.02	0.01	0.02	0.01	0.01	0.02	0.01
100	100	0.02	0.01	0.01	0.02	0.00	0.01	0.01	0.01	0.02	0.01
500	100	0.02	0.01	0.01	0.02	0.01	0.02	0.01	0.01	0.02	0.00
1000	100	0.02	0.01	0.01	0.02	0.01	0.02	0.01	0.01	0.01	0.01
10000	100	0.02	0.01	0.01	0.02	0.00	0.01	0.01	0.01	0.02	0.01
10	200	0.01	0.01	0.01	0.01	0.00	0.01	0.01	0.01	0.01	0.00
100	200	0.01	0.01	0.00	0.01	0.00	0.01	0.00	0.00	0.01	0.00
500	200	0.01	0.01	0.01	0.01	0.00	0.01	0.01	0.01	0.01	0.00
1000	200	0.01	0.01	0.01	0.01	0.00	0.01	0.00	0.00	0.01	0.00
10000	200	0.01	0.01	0.01	0.01	0.00	0.01	0.01	0.01	0.01	0.00

Table 1: MSE of Summary Statistics computed in R. NP stands for the nonparametric bootstrap and W for the wild version. Q stands for quantile, Sd for standard deviation.

nboots	n	NP 0.25 Q	NP 0.5 Q	NP Mean	NP 0.75 Q	NP Sd	W 0.25 Q	W 0.5 Q	W Mean	W 0.75 Q	W Sd
10	5	0.45	0.39	0.36	0.43	0.28	0.44	0.37	0.37	0.45	0.28
100	5	0.43	0.37	0.35	0.43	0.29	0.45	0.38	0.39	0.44	0.28
500	5	0.44	0.39	0.38	0.44	0.28	0.42	0.35	0.35	0.42	0.28
1000	5	0.43	0.37	0.36	0.42	0.29	0.43	0.35	0.35	0.41	0.29
10000	5	0.43	0.38	0.36	0.43	0.28	0.42	0.35	0.35	0.41	0.29
10	10	0.31	0.29	0.26	0.31	0.19	0.31	0.27	0.27	0.33	0.19
100	10	0.30	0.28	0.26	0.32	0.20	0.30	0.26	0.26	0.30	0.19
500	10	0.30	0.27	0.25	0.30	0.19	0.30	0.24	0.24	0.30	0.20
1000	10	0.31	0.28	0.26	0.30	0.19	0.30	0.25	0.25	0.29	0.18
10000	10	0.31	0.28	0.26	0.29	0.18	0.31	0.26	0.26	0.31	0.19
10	50	0.15	0.14	0.12	0.15	0.08	0.14	0.12	0.12	0.14	0.08
100	50	0.14	0.13	0.11	0.14	0.08	0.14	0.12	0.11	0.14	0.08
500	50	0.14	0.13	0.11	0.14	0.08	0.13	0.11	0.11	0.14	0.08
1000	50	0.15	0.13	0.12	0.14	0.08	0.14	0.12	0.12	0.14	0.08
10000	50	0.14	0.13	0.11	0.14	0.08	0.14	0.11	0.11	0.14	0.09
10	100	0.11	0.10	0.08	0.11	0.06	0.10	0.09	0.08	0.10	0.06
100	100	0.10	0.09	0.08	0.10	0.06	0.10	0.08	0.08	0.10	0.06
500	100	0.10	0.10	0.08	0.10	0.06	0.10	0.08	0.08	0.10	0.06
1000	100	0.10	0.09	0.08	0.10	0.06	0.10	0.08	0.08	0.09	0.06
10000	100	0.10	0.10	0.08	0.10	0.05	0.10	0.08	0.08	0.10	0.06
10	200	0.08	0.07	0.06	0.08	0.04	0.07	0.06	0.06	0.08	0.04
100	200	0.08	0.07	0.06	0.07	0.04	0.07	0.06	0.06	0.07	0.04
500	200	0.07	0.07	0.06	0.07	0.04	0.07	0.06	0.06	0.07	0.04
1000	200	0.08	0.07	0.06	0.07	0.04	0.07	0.06	0.06	0.07	0.04
10000	200	0.07	0.07	0.06	0.08	0.04	0.07	0.06	0.06	0.07	0.04

Table 2: MAE of Summary Statistics computed in R. NP stands for the nonparametric bootstrap and W for the wild version. Q stands for quantile, Sd for standard deviation.

nboots	n	NP 0.25 Q	NP 0.5 Q	NP Mean	NP 0.75 Q	NP Sd	W 0.25 Q	W 0.5 Q	W Mean	W 0.75 Q	W Sd
10	5	0.33	0.26	0.21	0.31	0.14	0.29	0.23	0.22	0.32	0.12
100	5	0.27	0.22	0.20	0.30	0.12	0.28	0.20	0.20	0.28	0.12
500	5	0.28	0.21	0.19	0.27	0.12	0.29	0.21	0.21	0.29	0.12
1000	5	0.29	0.21	0.20	0.26	0.12	0.26	0.20	0.20	0.27	0.11
10000	5	0.28	0.21	0.19	0.27	0.12	0.30	0.21	0.21	0.29	0.12
10	10	0.15	0.13	0.10	0.15	0.06	0.15	0.11	0.11	0.15	0.06
100	10	0.14	0.12	0.11	0.15	0.06	0.13	0.09	0.09	0.13	0.06
500	10	0.13	0.11	0.09	0.14	0.06	0.14	0.10	0.10	0.14	0.06
1000	10	0.14	0.11	0.10	0.13	0.05	0.14	0.10	0.10	0.14	0.06
10000	10	0.14	0.12	0.10	0.14	0.06	0.14	0.10	0.10	0.14	0.05
10	50	0.03	0.03	0.02	0.03	0.01	0.03	0.02	0.02	0.03	0.01
100	50	0.03	0.03	0.02	0.03	0.01	0.03	0.02	0.02	0.03	0.01
500	50	0.03	0.03	0.02	0.03	0.01	0.03	0.02	0.02	0.03	0.01
1000	50	0.03	0.03	0.02	0.03	0.01	0.03	0.02	0.02	0.03	0.01
10000	50	0.03	0.03	0.02	0.03	0.01	0.03	0.02	0.02	0.03	0.01
10	100	0.02	0.01	0.01	0.02	0.01	0.02	0.01	0.01	0.02	0.00
100	100	0.02	0.01	0.01	0.02	0.00	0.02	0.01	0.01	0.01	0.00
500	100	0.02	0.01	0.01	0.02	0.01	0.02	0.01	0.01	0.02	0.00
1000	100	0.02	0.01	0.01	0.02	0.01	0.02	0.01	0.01	0.02	0.01
10000	100	0.02	0.01	0.01	0.02	0.00	0.02	0.01	0.01	0.02	0.01
10	200	0.01	0.01	0.01	0.01	0.00	0.01	0.01	0.01	0.01	0.00
100	200	0.01	0.01	0.00	0.01	0.00	0.01	0.00	0.00	0.01	0.00
500	200	0.01	0.01	0.01	0.01	0.00	0.01	0.01	0.01	0.01	0.00
1000	200	0.01	0.01	0.00	0.01	0.00	0.01	0.01	0.01	0.01	0.00
10000	200	0.01	0.01	0.00	0.01	0.00	0.01	0.01	0.01	0.01	0.00

Table 3: MSE of Summary Statistics computed in C++. NP stands for the nonparametric bootstrap and W for the wild version. Q stands for quantile, Sd for standard deviation.

nboots	n	NP 0.25 Q	NP 0.5 Q	NP Mean	NP 0.75 Q	NP Sd	W 0.25 Q	W 0.5 Q	W Mean	W 0.75 Q	W Sd
10	5	0.47	0.41	0.37	0.45	0.31	0.43	0.38	0.38	0.46	0.28
100	5	0.42	0.37	0.36	0.44	0.29	0.43	0.36	0.36	0.43	0.29
500	5	0.42	0.37	0.35	0.41	0.29	0.43	0.36	0.36	0.43	0.28
1000	5	0.44	0.37	0.35	0.41	0.29	0.41	0.35	0.35	0.42	0.27
10000	5	0.43	0.37	0.35	0.41	0.29	0.44	0.37	0.37	0.43	0.29
10	10	0.31	0.29	0.26	0.31	0.21	0.32	0.27	0.27	0.31	0.19
100	10	0.30	0.28	0.26	0.31	0.19	0.29	0.23	0.23	0.29	0.19
500	10	0.29	0.26	0.25	0.29	0.19	0.30	0.25	0.25	0.29	0.19
1000	10	0.29	0.27	0.25	0.29	0.19	0.30	0.25	0.25	0.30	0.20
10000	10	0.30	0.27	0.26	0.31	0.20	0.30	0.25	0.25	0.30	0.19
10	50	0.14	0.14	0.12	0.15	0.08	0.14	0.12	0.11	0.14	0.08
100	50	0.14	0.13	0.11	0.14	0.08	0.15	0.12	0.12	0.14	0.08
500	50	0.15	0.13	0.12	0.14	0.08	0.14	0.11	0.11	0.14	0.08
1000	50	0.14	0.13	0.11	0.14	0.08	0.14	0.11	0.11	0.14	0.08
10000	50	0.15	0.13	0.11	0.14	0.08	0.14	0.12	0.12	0.14	0.08
10	100	0.11	0.10	0.08	0.11	0.06	0.10	0.08	0.08	0.10	0.05
100	100	0.10	0.10	0.08	0.10	0.05	0.10	0.08	0.08	0.10	0.06
500	100	0.10	0.09	0.07	0.10	0.06	0.10	0.08	0.08	0.10	0.06
1000	100	0.10	0.09	0.08	0.10	0.06	0.10	0.08	0.08	0.10	0.06
10000	100	0.10	0.10	0.08	0.10	0.06	0.10	0.08	0.08	0.10	0.06
10	200	0.08	0.07	0.06	0.08	0.04	0.08	0.06	0.06	0.07	0.04
100	200	0.08	0.07	0.06	0.07	0.04	0.07	0.06	0.06	0.07	0.04
500	200	0.07	0.07	0.06	0.07	0.04	0.07	0.06	0.06	0.07	0.04
1000	200	0.07	0.07	0.05	0.07	0.04	0.07	0.06	0.06	0.07	0.04
10000	200	0.07	0.07	0.06	0.07	0.04	0.07	0.06	0.06	0.07	0.04

Table 4: MAE of Summary Statistics computed in C++. NP stands for the nonparametric bootstrap and W for the wild version. Q stands for quantile, Sd for standard deviation.

Appendix B

	Compound MSE	Compound MAE
R - NP	0.00	0.01
R - W	0.00	0.01
C++ - NP	0.00	0.01
C++ - W	0.00	0.01

Table 5: Compound MSE and MAE for the linear Regression for all four algorithms.

nboots	n	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
10	5	1331.20	1565.88	1689.55	2481.18	1850.95	76240.10
100	5	6256.00	7021.27	7449.05	8058.75	8029.12	30743.90
500	5	27830.70	29000.10	30483.55	33732.64	38423.83	76362.70
1000	5	54494.50	55846.30	56406.45	59518.25	59817.97	93074.50
10000	5	453763.30	465940.88	467240.45	467885.53	468781.67	644922.00
10	10	1171.20	1204.57	1237.25	1275.78	1300.12	2125.40
100	10	5045.10	5184.77	5474.65	5655.83	5675.13	15215.70
500	10	23053.10	23596.28	23877.45	24777.96	24229.17	35129.80
1000	10	45152.40	45896.90	46325.00	49994.62	46965.70	233025.30
10000	10	460325.30	472598.40	475007.85	475332.75	476048.03	650120.40
10	50	1240.70	1255.25	1282.00	1331.12	1376.85	1702.40
100	50	5411.80	5719.05	5863.65	6212.04	6141.08	19286.40
500	50	25526.80	26015.95	26223.70	27471.37	26545.60	40862.70
1000	50	50168.50	50860.60	51238.15	53860.03	52015.25	72144.20
10000	50	512266.80	529350.48	531721.40	534446.96	534304.85	708729.50
10	100	1298.30	1326.75	1367.75	1416.12	1463.22	1872.20
100	100	6127.90	6318.28	6519.30	6800.21	6785.77	19191.80
500	100	28288.60	28724.00	29188.95	30692.39	29676.78	42266.20
1000	100	55434.80	56377.42	57037.45	60116.17	68241.80	74805.80
10000	100	581734.70	597094.50	609912.00	628853.86	628675.30	896184.80
10	200	1421.90	1465.48	1516.90	1568.52	1595.78	2008.00
100	200	7323.60	7652.15	8060.65	8383.12	8264.52	18409.70
500	200	34360.50	37696.12	42090.60	43812.40	47135.65	91459.10
1000	200	67994.80	69910.00	77643.85	78606.74	79627.85	259755.60
10000	200	719382.70	730148.57	738648.15	776793.42	756859.88	1895933.80

Table 6: Time of Regression Problem in R, scaled in microseconds.

nboots	n	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
10	5	855.70	1019.78	1119.25	1468.89	1256.48	30726.30
100	5	1138.10	1280.58	1394.70	1429.82	1497.75	2623.40
500	5	2209.50	2352.47	2446.90	2546.59	2655.85	3700.00
1000	5	3555.90	3748.15	3892.85	3972.69	4138.40	5023.40
10000	5	24589.20	25181.88	25310.30	25527.78	25650.42	31383.70
10	10	760.00	774.75	796.85	851.04	846.15	3525.00
100	10	1008.30	1031.68	1057.40	1086.30	1111.70	1425.00
500	10	1960.80	2021.70	2061.20	2121.34	2172.68	3216.50
1000	10	3319.30	3447.30	3524.45	3584.32	3636.28	4391.40
10000	10	27363.10	27812.45	28033.90	28187.71	28225.17	33683.30
10	50	816.50	833.10	859.90	883.24	897.88	1488.50
100	50	1330.60	1358.97	1393.90	1435.95	1498.15	1919.60
500	50	3594.00	3721.72	3853.25	3880.83	3997.47	4436.40
1000	50	6487.20	6651.55	6878.50	6877.55	7085.70	7811.70
10000	50	59482.50	60213.60	60576.50	61789.40	60933.88	86034.40
10	100	847.10	857.77	882.35	907.78	939.65	1212.00
100	100	1544.00	1581.55	1626.50	1668.27	1726.05	2597.70
500	100	4739.40	4858.40	4985.50	5062.35	5246.55	6145.40
1000	100	8689.00	9042.95	9214.15	9378.99	9381.10	25799.00
10000	100	82855.50	83849.97	84241.10	86397.85	85002.30	103707.60
10	200	923.00	973.23	992.65	1054.57	1048.45	1896.10
100	200	2271.80	2448.25	2605.55	2914.81	3046.05	6440.00
500	200	7542.00	7851.38	8095.70	8415.89	8501.77	20882.10
1000	200	14665.80	15098.05	15389.50	16144.47	16582.92	29964.60
10000	200	139417.30	141460.07	142299.45	147494.37	154295.97	191436.40

Table 7: Time of Regression Problem in C++, scaled in microseconds.