Chapter 1
Introduction

In this introduction, we discuss the basic idea of the bootstrap procedure using a simple example. Furthermore, the Statistical Software R and its use in the context of this manuscript is briefly covered. Readers who are familiar with this material can skip this chapter.

A short summary of the contents of this manuscript can be found in the Preface and is not listed here again.

1.1 Basic Idea of the Bootstrap

Typical statistical methods, such as constructing a confidence interval for the expected value of a random variable or determining critical values for a hypothesis test, require knowledge of the underlying distribution. However, this distribution is usually only partially known at most. The statistical method we use to perform the task depends on our knowledge of the underlying distribution.

Let us be more precise and assume that

$$X_1, \ldots, X_n \sim F$$

is a sequence of independent and identically distributed (i.i.d.) random variables with common distribution function (df.) F. Consider the statistic

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-73480-0_1) contains supplementary material, which is available to authorized users.

G. Dikta and M. Scheer, *Bootstrap Methods*,
https://doi.org/10.1007/978-3-030-73480-0_1

$$\bar{X}_n := \frac{1}{n}\sum_{i=1}^{n} X_i$$

to estimate the parameter $\mu_F = \mathbb{E}(X)$, that is, the expectation of X.

To construct a confidence interval for μ_F or to perform a hypothesis test on μ_F, we consider the df. of the studentized version of $\bar{X}_n$, that is,

$$\mathbb{P}_F\big(\sqrt{n}(\bar{X}_n - \mu_F)/s_n \le x\big), \qquad x \in \mathbb{R}, \tag{1.1}$$

where

$$s_n^2 := \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X}_n)^2$$

is the unbiased estimator of $\sigma^2 = \mathrm{VAR}(X)$, that is, the variance of X. Note that we write $\mathbb{P}_F$ here to indicate that F is the data generating df.

If we know that F comes from the class of normal distributions, then the df. under (1.1) belongs to a $t_{n-1}-$distribution, i.e., a Student's t distribution with $n-1$ degrees of freedom. Using the known quantiles of the $t_{n-1}-$ distribution exact confidence interval can be determined. For example, an exact 90% confidence interval for μ_F is given by

$$\big[\bar{X}_n - \frac{s_n\, q_{0.95}}{\sqrt{n}}\,,\ \bar{X}_n + \frac{s_n\, q_{0.95}}{\sqrt{n}}\big], \tag{1.2}$$

where $q_{0.95}$ is the 95% quantile of the t_{n-1} distribution.

But in most situations we are not able to specify a parametric distribution class for F. In such a case, we have to look for a suitable approximation for (1.1). If it is ensured that $\mathbb{E}(X^2) < \infty$, the central limit theorem (CLT) guarantees that

$$\sup_{x\in\mathbb{R}}\Big|\mathbb{P}_F\big(\sqrt{n}(\bar{X}_n - \mu_F)/s_n \le x\big) - \Phi(x)\Big| \longrightarrow 0, \quad \text{for } n \to \infty, \tag{1.3}$$

where Φ denotes the standard normal df. Based on the CLT, we can now construct an asymptotic confidence interval. For example, the 90% confidence interval under (1.2) has the same structure when we construct it using the CLT. However, $q_{0.95}$ now is the 95% quantile of the standard normal distribution. The interval constructed in this way is no longer an exact confidence interval. It can only be guaranteed that the confidence level of 90% is reached with $n \to \infty$. It should also be noted that for $q_{0.95}$ the 95% quantile of the $t_{n-1}-$ distribution can also be chosen, because for $n \to \infty$, the $t_{n-1}-$ df. converges to the standard normal df.

So far we have concentrated exclusively on the studentized mean. Let us generalize this to a statistic of the type

$$T_n(F) = T_n(X_1, \ldots, X_n; F),$$

where $X_1, \ldots, X_n \sim F$ are i.i.d. Then the question arises how to approximate the df.

$$\mathbb{P}_F\big(T_n(F) \le x\big), \qquad x \in \mathbb{R} \tag{1.4}$$

if F is unknown. This is where Efron's bootstrap enters the game. The basic idea of the bootstrap method is the assumption that the df. of T_n is about the same when the data generating distribution F is replaced by another data generating distribution $\hat{F}$ which is close to F and which is known to us. If we can find such a df. $\hat{F}$,

$$\mathbb{P}_{\hat{F}}(T_n(\hat{F}) \le x), \qquad x \in \mathbb{R} \tag{1.5}$$

may also be an approximation of Eq. (1.4). We call this df. for the moment a *bootstrap approximation* of the df. given under Eq. (1.4). However, this approach only makes sense if we can guarantee that

$$\sup_{x\in\mathbb{R}} \Big|\mathbb{P}_F\big(T_n(F) \le x\big) - \mathbb{P}_{\hat{F}}\big(T_n(\hat{F}) \le x\big)\Big| \longrightarrow 0, \quad \text{for} \quad n \to \infty. \tag{1.6}$$

Now let us go back to construct a 90% confidence interval for μ_F based on the bootstrap approximation. For this, we take the studentized mean for T_n and assume that we have a data generating df. $\hat{F}$ that satisfies (1.6). Since $\hat{F}$ is known, we can now, at least theoretically, calculate the 5% and 95% quantiles of the df.

$$\mathbb{P}_{\hat{F}}\big(\sqrt{n}(\bar{X}_n - \mu_{\hat{F}})/s_n \le x\big),$$

which we denote by $q_{n,0.05}$ and $q_{n,0.95}$, respectively, to derive

$$\big[\bar{X}_n - \frac{s_n\, q_{n,0.95}}{\sqrt{n}}\,,\ \bar{X}_n - \frac{s_n\, q_{n,0.05}}{\sqrt{n}}\big], \tag{1.7}$$

an asymptotic 90% confidence interval for μ_F.

If we want to use such a bootstrap approach, we have

(A) to choose the data generating df. $\hat{F}$ such that the bootstrap approximation (1.6) holds,
(B) to calculate the df. of T_n, where the sample is generated under $\hat{F}$.

Certainly (A) is the more demanding part, in particular, the proof of the approximation (1.6). Fortunately, a lot of work has been done on this in the last decades. Also, the calculation of the df. under (B) may turn out to be very complex. However, this is of minor importance, because the bootstrap df. in Eq. (1.6) can be approximated very well by a Monte Carlo approach. It is precisely this opportunity to perform a Monte Carlo approximation, together with the rapid development of powerful PCs that has led to the great success of the bootstrap approach.

To demonstrate such a Monte Carlo approximation for the df. of Eq. (1.5), we proceed as follows:

(a) Construct m i.i.d. (bootstrap) samples independent of one another of the type

$$\begin{array}{ccc} X^*_{1;1} & \dots & X^*_{1;n} \\ \vdots & \vdots & \vdots \\ X^*_{m;1} & \dots & X^*_{m;n} \end{array}$$

with common df. $\hat{F}$.

(b) Calculate for each sample $k \in \{1, 2, \dots, m\}$

$$T^*_{k;n} := T_n(X^*_{k;1}, \dots, X^*_{k;n}; \hat{F})$$

to obtain $T^*_{1;n}, \dots, T^*_{m;n}$.

(c) Since the $T^*_{1;n}, \dots, T^*_{m;n}$ are i.i.d. , the Glivenko-Cantelli theorem (GC) guarantees

$$\sup_{x\in\mathbb{R}} \left| \mathbb{P}_{\hat{F}}\big(T_n(\hat{F}) \le x\big) - \frac{1}{m}\sum_{k=1}^{m} \mathrm{I}_{\{T^*_{k;n}\le x\}} \right| \longrightarrow 0, \quad \text{for } m \to \infty, \tag{1.8}$$

where $\mathrm{I}_{\{x\in A\}} \equiv \mathrm{I}_{\{A\}}(x)$ denotes the indicator function of the set A, that is,

$$\mathrm{I}_{\{x\in A\}} = \begin{cases} 1 & : x \in A \\ 0 & : x \notin A \end{cases}.$$

The choice of an appropriate $\hat{F}$ depends on the underlying problem, as we will see in the following chapters. In the context of this introduction, F_n, the *empirical df.* (edf.) of the sample $X_1, \dots, X_n$, defined by

$$F_n(x) := \frac{1}{n}\sum_{i=1}^{n} \mathrm{I}_{\{X_i\le x\}}, \qquad x \in \mathbb{R}, \tag{1.9}$$

is a good choice for $\hat{F}$ since, by the Glivenko-Cantelli theorem, we get with probability one (w.p.1)

$$\sup_{n\in\mathbb{R}} \left| F_n(x) - F(x) \right| \underset{n\to\infty}{\longrightarrow} 0.$$

If we choose F_n for $\hat{F}$ then we are talking about the *classical bootstrap* which was historically the first to be studied.

1.2 The R-Project for Statistical Computing

The programming language R, see R Core Team (2019), is a widely used open-source software tool for data analysis and graphics which runs on the commonly used operating systems. It can be downloaded from the R-project's website at www.r-project.org. The R Development Core Team also offers some documentation on this website:

- R installation and administration,
- An introduction to R,
- The R language definition,
- R data import/export, and
- The R reference index.

Additionally to this material, there is a large and strongly growing number of textbooks available covering the R programming language and the applications of R in different fields of data analysis, for instance, Beginning R or Advanced R.

Besides the R software, one also should install an editor or an integrated development environment (IDE) to work with R conveniently. Several open-source products are available on the web, like

- RStudio, see RStudio Team (2020), at www.rstudio.org;
- RKWard, at http://rkward.sourceforge.net;
- Tinn-R, at http://www.sciviews.org/Tinn-R; and
- Eclipse based StatET, at http://www.walware.de/goto/statet.

1.3 Usage of R in This Book

Throughout the book we implement, for instance, different resampling schemes and simulation studies in R. Our implementations are free from any checking of function arguments. We provide R-code that focuses solely on an understandable implementation of a certain algorithm. Therefore, there is plenty of room to improve the implementations. Some of these improvements will be discussed within the exercises.

R is organized in packages. A new installation of R comes with some pre-installed packages. And the packages provided by the R-community makes this programming language really powerful. More than 15000 packages (as of 2020/Feb) are available (still growing). But especially for people starting with R this is also a problem. The CRAN Task View https://cran.r-project.org/web/views summarizes certain packages within categories like "Graphics", "MachineLearning", or "Survival". We decided to use only a handful of packages that are directly related to the main objective of this book, like the `boot`-package for bootstrapping, or (in the opinion of the authors) are too important and helpful to be ignored, like `ggplot2`, `dplyr`, and `tidyr`. In addition, we have often used the `simTool` package from Marsel Scheer to carry out simulations. This package is explained in the appendix. Furthermore,

we decided to use the pipe operator, i.e., `%>%`. There are a few critical voices about this operator, but the authors as the most R users find it very comfortable to work with the pipe operator. People familiar with Unix systems will recognize the concept and probably appreciate it. A small example will demonstrate how the pipe operator works. Suppose we want to apply a function `A` to the object `x` and the result of this operation should be processed further by the function `B`. Without the pipe operator one could use

```
B(A(x))
# or
tmp = A(x)
B(tmp)
```

With the pipe operator this becomes

```
A(x) %>%
     B
# or
x %>%
  A %>%
  B
```

Especially with longer chains of functions using pipes may help to obtain R-code that is easier to understand.

1.3.1 Further Non-Statistical R-Packages

There are a lot of packages that are worth to look at. Again the CRAN Task View may be a good starting point. The following list is focused on writing reports, developing R-packages, and increasing the speed of R-code itself. By far this list is not exhaustive:

- `knitr` for writing reports (this book was written with `knitr`);
- `readxl` for the import of excel files;
- `testthat` for creating automated unit tests. It is also helpful for checking function arguments;
- `covR` for assessing test coverage of the unit tests;
- `devtools` for creating/writing packages;
- `data.table` amazingly fast aggregation, joins, and various manipulations of large datasets;
- `roxygen2` for creating help pages within packages;
- `Rcpp` for a simple integration of C++ into R;
- `profvis` a profiling tool that assess at which line of code R spends its time;
- `checkpoint`, `renv` for package dependency.

Of course, further packages for importing datasets, connecting to databases, creating interactive graphs and user interfaces, and so on exist. Again, the packages provided by the R-community make this programming language really powerful.

Finally, we want to strongly recommend the R-package `drake`. According to the package-manual: *It analyzes your workflow, skips steps with up-to-date results, and orchestrates the rest with optional distributed computing.* We want to briefly describe how this works in principle. One defines a plan with steps one wants to perform:

```
plan <- drake::drake_plan(
  raw = import_data("/foo/bar/data.csv"),
  wrangled = preprocess(raw),
  model1 = fit1(wrangled),
  model2 = fit2(wrangled)
)
```

This plan can then be executed/processed by `drake`.

```
drake::make(plan)
```

This creates the four objects raw, wrangled, model1, and model2. Assume now that we change the underlying source code for one of the model-fitting functions, then there is, of course, no need to rerun the preprocess step. Since `drake` analyzed our defined plan it automatically skips the import and preprocessing for us. This can be extremely helpful if the preprocess step is computationally intensive. Or imagine the situation that we refactor the data-import function. If these changes do not modify the raw object created in the first step, then again there is no need to rerun the preprocess step or to refit the models. Again `drake` automatically detects that and skips the preprocessing. Furthermore, looking at the definition of model1 and model2, we see that there is no logical need to process them sequentially and with `drake` one can easily do the computation in parallel. The package does also a lot of other helpful things in the background, for instance, it measures the time used to perform a single step of the plan. Although we do not use `drake` in this book we encourage the reader to try out the package. A good starting point is the excellent user manual accessible under https://books.ropensci.org/drake.

References

R Core Team (2019) R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria, https://www.R-project.org/

RStudio Team (2020) RStudio: integrated development environment for R. RStudio, PBC, Boston, MA, http://www.rstudio.com/