Assignment II: Advanced simulation techniques.

Computational Statistics
Instructor: Luiz Max de Carvalho
Student: Lucas Machado Moschen

December 8, 2021

Hand-in date: 08/12/2021.

Background

We have by now hopefully acquired a solid theoretical understanding of simulation techniques, including Markov chain Monte Carlo (MCMC). In this assignment, we shall re-visit some of the main techniques in the field of Simulation. The goal is to broaden your knowledge of the field by implementing one of the many variations on the general theme of simulation algorithms.

Each method/paper brings its own advantages and pitfalls, and each explores a slightly different aspect of Computational Statistics. You should pick **one** of the listed papers and answer the associated questions.

In what follows, ESS stands for effective sample size, and is similar to n_{eff} we have encountered before: it measures the number of effectively uncorrelated samples in a given collection of random variates.

Paper Bootstrap (Efron and Tibshirani, 1986)

In orthodox (frequentist) Statistics, it is common to want to ascertain long run (frequency) properties of estimators, including coverage of confidence intervals and standard errors. Unfortunately, for the models of interest in actual practice, constructing confidence intervals directly (exactly) is difficult. The bootstrap method is a re-sampling technique that allows for a simple yet theoretically grounded way of constructing confidence intervals and assessing standard errors in quite complex situations.

For this assigment, you are encouraged to consult the seminal 1986 review by stellar statisticians Bradley Efron and Robert Tibshirani (Efron and Tibshirani, 1986).

Hint: Brush off on your Normal theory before delving in. The book by Schervish (2012) – specially Chapter 5 – is a great resource.

1. Define and explain the bootstrap technique;
2. Define and explain the jackknife technique;
3. Implementation:

(a) Reproduce the results in Table I of Efron and Tibshirani (1986);

(b) Show what happens if one increases/decreases the value of B;

4. Why is it important to draw exactly n samples in each bootstrap iteration? Can this be relaxed?

5. (bonus) Propose an alternative bootstrap method to the one proposed in the paper and discuss the situations where the new method is expected to perform better.

Bootstrap

Bootstrap is a technique for measuring the accuracy of an estimator through approximations for standard errors and confidence intervals. The frequentist inference approach has several usages for bootstrap, but it can be interpreted in the Bayesian framework (Rubin, 1981). From samples of an experiment, the general idea is to generate new data points, and perform the estimation process for each one. This procedure creates an *empirical distribution* for the considered estimator, such as the *maximum likelihood estimator* (MLE).

The construction from Efron and Tibshirani (1986) supposes that $\boldsymbol{x} = (x_1, x_2, \dots, x_n)$ is an observation of the independent and identically distributed random variables

$$\boldsymbol{X} = (X_1, X_2, \dots, X_n), X_i \overset{iid}{\sim} F,$$

where F is an unknown probability distribution. This assumption can be relaxed for dependent data using block bootstrap strategies (Lahiri, 1999). Let $R(\boldsymbol{X}, F)$ be a random variable of interest, such as the statistic $\hat{\theta}(\boldsymbol{X})$. From sample $\boldsymbol{x}$, we build the empirical distribution $\hat{F}$, that is, if $X \sim \hat{F}$, we have $\Pr(X = x_i) = 1/n$.

The bootstrap technique offers a Monte Carlo estimator for some characteristic of the distribution of R, such as its mean $\mathbb{E}_{\hat{F}}[R]$. A common example of interest is the standard error

$$\sigma_F = \left(\operatorname{Var}_F(\hat{\theta}(\boldsymbol{X}))\right)^{1/2}$$

of a statistic $\hat{\theta}(\boldsymbol{X})$ which can be approximated[1] by

$$\hat{\sigma}_{\hat{F}} = \left(\operatorname{Var}(\hat{\theta}_{\hat{F}}(\boldsymbol{x}))\right)^{1/2}$$

the standard deviation at $F = \hat{F}$.

We know how to draw from $\hat{F}$: sampling with replacement from $\{x_1, \dots, x_n\}$. With that, we obtain B *bootstrap samples* $\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(B)}$ with fixed size n. As usual for Monte Carlo estimators, for each bootstrap sample $\boldsymbol{x}^{(i)}$, we evaluate $R^{(i)} = R(\boldsymbol{x}^{(i)}, \hat{F})$, obtaining samples $\{R^{(1)}, \dots, R^{(B)}\}$. Then, we calculate the quantity of interest, such as

$$\frac{1}{B}\sum_{i=1}^{B} R^{(i)} \xrightarrow{P} \mathbb{E}_{\hat{F}}[R],$$

[1]This is proved under some regularity conditions.

where P indicates convergence in probability as a result of the Strong Law of Large Numbers (SLLN).

For the standard error approximation of $\hat{\sigma}_{\hat{F}}$ we use

$$\hat{\sigma}_B = \left(\frac{1}{B-1} \sum_{i=1}^{B} (\hat{\theta}(\boldsymbol{x}^{(i)}) - \hat{\theta}^B)^2 \right)^{1/2},$$

where $\hat{\theta}(\boldsymbol{x}^{(i)})$ is the statistic of interest evaluated in $\boldsymbol{x}^{(i)}$ and

$$\hat{\theta}^B = B^{-1} \sum_{i=1}^{B} \hat{\theta}(\boldsymbol{x}^{(i)}).$$

In the same sense, by the SLLN,

$$\hat{\sigma}_B \xrightarrow{P} \hat{\sigma}_{\hat{F}}.$$

Notice that $R(\boldsymbol{X}, F)$ depends on n through $\boldsymbol{X}$. For instance, if $\hat{\theta}(\boldsymbol{X})$ is a strongly consistent estimator for θ, it gets close to θ with probability one. In that sense the standard error of the estimator vanishes when n tends to infinity. This shows that the distribution of $\hat{\theta}(\boldsymbol{X})$ changes with n with its limit being a punctual mass in θ. Since the distribution changes with n, we cannot get bootstrap samples with different sizes and use Monte Carlo estimator as we are current doing. In summary, it is important to draw bootstrap samples with size n to get samples from the same distribution $\hat{F}$ and, hence, $\hat{\theta}(\boldsymbol{x})$. Bickel and Freedman (1981) presents a situation where the bootstrap sample size m is different from n for approximating the finite mean μ of F.

We presented above the non-parametric form of bootstrap. If we believe that F belongs to a family, we can estimate it through its parametric MLE. For instance, if we believe that F comes from the normal distribution, its MLE is the normal distribution with mean $\bar{x}$ and variance s^2, where $\bar{x}$ is the sample mean and s^2 is the unbiased sample variance.

Bayesian Bootstrap

The *Bayesian bootstrap* (BB) generates a posterior distribution for each x_i such that not observed values have zero posterior probability. The posterior probability for each x_i is centered at $1/n$ with some variability. The ideia is to generate $u_1, \ldots, u_{n-1} \overset{iid}{\sim} \text{Unif}[0,1]$ and get the order statistics $u_{(1)}, \ldots, u_{(n-1)}$. At last, we define $g_i = u_{(i)} - u_{(i-1)}$, where $i = 1, \ldots, n$ and $u_{(0)} = 0, u_{(n)} = 1$. Then we attach probability g_i for x_i, which obtains a replication for the data.

For instance, suppose we are interested in the mean of the random variable X. Then, the posterior distribution of the posterior mean is $\sum_{i=1}^{n} g_i^{(l)} x_i$, such that $g_i^{(l)}$ is the i-th probability of the l-th Bayesian replication.

The distribution generated by Bayesian bootstrap is similar to the regular one, but Rubin (1981) argues that BB has an advantage regarding the interpretation since it generates probability statements about the parameters, rather than frequency of statistics.

We implemented this bootstrap, but the results were different from what we expected, since the standard error was less than the other two approaches. More work should be done to correct implementation bugs.

Jackknife technique

Jackknife is a re-sampling method to estimate bias and variance of estimators, which pre-dates bootstrap. The basic idea is to leave out one observation per time and calculate the estimated quantity for this reduced sample. The average of these calculations is the jackknife estimate. We denote $\boldsymbol{x}_{(i)}$ the sample except the observation i, i.e.,

$$\boldsymbol{x}_{(i)} = (x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n).$$

This is the *jackknife* sample i. An estimate for σ_F is

$$\hat{\sigma}_J = \sqrt{\frac{n-1}{n}\sum_{i=1}^{n}(\hat{\theta}(\boldsymbol{x}_{(i)}) - \hat{\theta}^J)^2},$$

where $\hat{\theta}^J = \sum_{i=1}^{n}\hat{\theta}(\boldsymbol{x}_i)/n$.

Results in Table I

The experiment is as follows: set $n = 15$ and the statistic of interest $\hat{\theta}$ is the 25% trimmed mean. The 100α trimmed mean discard the α highest and α lowest in the sample and calculate the mean with the other $n(1-2\alpha)$ data points. Let $x_{(1)} \leq \ldots \leq x_{(n)}$ be the order statistic of $\boldsymbol{x}$. Let g be the integer and r the fraction part of $n\alpha$. The trimmed mean is

$$\bar{x}_\alpha = \frac{1}{n(1-2\alpha)}\left((1-r)(x_{(g+1)} + x_{(n-g)}) + \sum_{i=g+2}^{n-g-1} x_{(i)}\right).$$

In our case, $15 \cdot 0.25 = 3.75$, then

$$\bar{x}_{.25} = \frac{1}{7.5}\left(0.25(x_{(4)} + x_{(12)}) + \sum_{i=5}^{11} x_{(i)}\right).$$

Since we could not calculate the standard error of $\bar{x}_\alpha$ analytically, we propose a Monte Calor estimator. We have that,

$$\begin{aligned}\mathrm{Var}(\bar{x}_\alpha) = \frac{1}{n^2(1-2\alpha)^2}\Bigg(&\sum_{i,j=g+2}^{n-g-1}\mathrm{Cor}(X_{(i)}, X_{(j)}) \\ &+ 2(1-r)\sum_{i=g+2}^{n-g-1}\mathrm{Cor}(X_{(g+1)}, X_{(i)}) + \mathrm{Cor}(X_{(n-g)}, X_{(i)}) \\ &+ (1-r)^2\big(\mathrm{Cor}(X_{(g+1)}, X_{(g+1)}) + 2\,\mathrm{Cor}(X_{(g+1)}, X_{(n-g)}) \\ &\qquad + \mathrm{Cor}(X_{(n-g)}, X_{(n-g)})\big)\Bigg),\end{aligned}$$

and $\mathrm{Cor}(X_{(i)}, X_{(j)})$ can be calculated sampling $X_{(i)}$ and $X_{(j)}$ from distribution F. We consider two different probability distributions for F, (i) standard normal, and (ii) exponential with parameter $\lambda = 1$. Moreover, in order to provide a standard deviation for $\hat{\sigma}_B$ we analyse its variability for different data points $\boldsymbol{x}$ drawn from distribution F. We also calculate the coefficient of variation. Table 1 summarizes the results. The Jackknife is a little closer than bootstrap, but has a larger standard deviation. Both estimates are good for the standard deviation these simple situations. We also note that $B = 200$ already provide good estimates and $B = 10000$ is 50 times more expensive, but reduced the standard deviation very little.

Method	F standard normal			F exponential $\lambda = 1$		
	Average	SD	CV	Average	SD	CV
Bootstrap (200)	0.285	0.071	0.249	0.239	0.083	0.347
Bootstrap (10000)	0.285	0.07	0.246	0.241	0.082	0.34
Jackknife	0.277	0.081	0.292	0.243	0.093	0.383
True value (10^6)	0.28			0.236		

Table 1: Sampling experiment comparing bootstrap with $B = 200$ and $B = 10000$ samples and Jackknife estimates of standard error for the 25% trimmed mean with $n = 15$ for the standard normal and exponential distribution. We denote CV for the coefficient of variation and SD the standard deviation. The true value is calculated performing the estimates for 10^6 different samples from F and calculating the corresponding Monte Carlo estimate.

Figure 1 presents the standard deviation calculation for different values of B. It would be nice to perform this curve more times to observe the variability with respect to different values of B, but this would be too computationally expensive. Notice that bootstrap generates biased estimates as B grows.

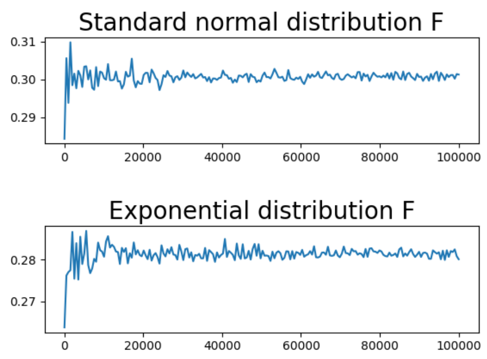

Figure 1: Convergence of the Bootstrap standard error estimator for the 25% trimmed mean for different values of B.

Finally, Figure 2 shows four situations: we consider two datasets, one generated drawn by the standard normal distribution, while the other by the standard t-Student with 5 degrees of freedom. The parametric approach is considered using the standard normal for both cases. This graphic suggests not much difference, but supposing a normal distribution instead of t-Student led to a more variable histogram, including values next to 1.

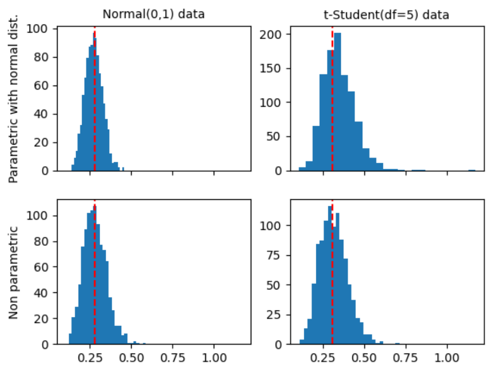

Figure 2: Histogram of the values of bootstrap standard errors with different datasets and different bootstrap approaches.

We finally present the difference between the Bayesian and regular bootstraps to present the implementation code we had. Figure 3 shows that although the the posterior mean of the trimmed mean distribution is close to the bootstrap mean, the variability is very difference, in a way that the Bayesian is underestimating the real accuracy of this statistic.

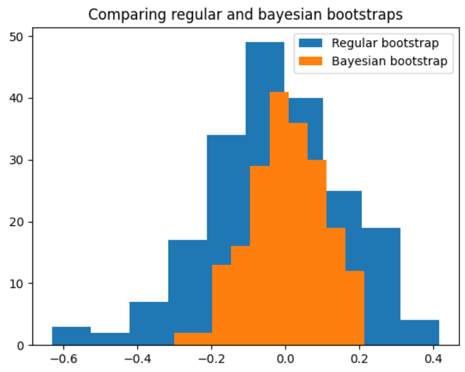

Figure 3: Histograms comparing the posterior distribution and the regular bootstrap distribution of the trimmed mean in the standard normal case.

Python code of the implementation

```python
import numpy as np
from tqdm import tqdm
import matplotlib.pyplot as plt

def random_sample(data_points, format):
    """
    Imitates the function numpy.choice with replacement.
    """
    size = format[0] * format[1]
    n = len(data_points)
    unif = np.random.random(size=size)
    choices = np.zeros(size)
    for i in range(size):
        pos = 0
        while pos/n < unif[i]:
            pos += 1
        choices[i] = data_points[pos - 1]
    choices = np.reshape(choices, format)
    return choices

def bootstrap_nonparametric(data_points, B):
    """
    Denotes the non parametric version of bootstrap
    """
    n = len(data_points)
    bootstrap_samples = random_sample(data_points, format=(n, B))
    return bootstrap_samples

def bootstrap_parametric(data_points, B):
    """
    Denotes the parametric version of bootstrap
    """
    n = len(data_points)
    bootstrap_samples = np.random.normal(loc=np.mean(data_points),
                                         scale=np.std(data_points, ddof=1),
                                         size=(n, B))
    return bootstrap_samples

def bootstrap_bayesian(data_points, B):
    """
    Bayesian bootstrap.
    """
    n = len(data_points)
    unif = np.random.random(size=(n-1, B))
    unif_sort = np.zeros((n+1, B))
    unif_sort[1:-1, :] = np.sort(unif, axis=0)
    unif_sort[-1, :] = 1
    g = np.diff(unif_sort, axis=0)
    return g

def jackknife_method(data_points):
    """
    Denotes the jackknife method for resampling
    """
    n = len(data_points)
    jack_samples = np.zeros((n-1, n))
    for i in range(n):
        jack_samples[:i, i] = data_points[:i]
        jack_samples[i:, i] = data_points[i+1:]
    return jack_samples

def trimmed_mean_bayesian(data_points, alpha, B):
    """
    Calcultate samples from the trimmed mean posterior
    """
    n = len(data_points)
    g = int(n * alpha)
    r = n * alpha - g
    dp_sort = np.sort(data_points)
    bootstrap_samples = bootstrap_bayesian(data_points, B)
    trimmed_mean = (1-r) * (bootstrap_samples[g] * dp_sort[g] +
    bootstrap_samples[n-g-1] * dp_sort[n-g-1])
    trimmed_mean = trimmed_mean + (bootstrap_samples[g+1:n-g-1] *
    dp_sort[g+1:n-g-1].reshape(-1, 1)).sum(axis=0)
    trimmed_mean /= ((1-r) * (bootstrap_samples[g] +
    bootstrap_samples[n-g-1]) + bootstrap_samples[g+1:n-g-1].sum(
    axis=0))
    return trimmed_mean

def trimmed_mean(data_points, alpha, B, method='bootstrap'):
    """
    Calculate samples from the trimmed mean estimator.
    """
    n = len(data_points)
    g = int(n * alpha)
    r = n * alpha - g
    if method == 'bootstrap':
        bootstrap = bootstrap_nonparametric(data_points, B)
    elif method == 'bootstrap-parametric':
        bootstrap = bootstrap_parametric(data_points, B)
    elif method == 'bootstrap-bayesian':
        return trimmed_mean_bayesian(data_points, alpha, B)
    else:
        bootstrap = jackknife_method(data_points)
    dp_sort = np.sort(bootstrap, axis=0)
    trimmed_mean = (1-r) * (dp_sort[g] + dp_sort[n-g-1]) + dp_sort[
    g+1:n-g-1].sum(axis=0)
    trimmed_mean = trimmed_mean/(n * (1 - 2*alpha))
    return trimmed_mean

def true_standard_error(F_distribution, n=15, alpha=0.25):
    """
    Calculate an approximation for the true standard error.
    """
    if F_distribution == 'normal':
        samples = np.random.normal(size=(n, 1000000))
    elif F_distribution == 't':
        samples = np.random.standard_t(df=5, size=(n, 1000000))
    else:
        samples = np.random.exponential(scale=1, size=(n, 1000000))
    samples = np.sort(samples, axis=0)
    g = int(n * alpha)
    r = n * alpha - g
    cov_matrix = np.cov(samples, rowvar=True)
    var = cov_matrix[g+1:n-g-1, g+1:n-g-1].sum()
    var += 2*(1-r) * cov_matrix[g, g+1:n-g-1].sum()
    var += 2*(1-r) * cov_matrix[n-g-1, g+1:n-g-1].sum()
    var += (1-r)**2 * (cov_matrix[g, g] + 2 * cov_matrix[g, n-g-1]
    + cov_matrix[n-g-1, n-g-1])
    var /= (n*(1-2*alpha))**2
    standard_error = np.sqrt(var)
    return standard_error

def monte_carlo_estimates(n=15, alpha=0.25, B=200, B2=10000):

    m = 1000
    table = np.zeros((m, 8))
    for k in tqdm(range(m)):
        y_norm = np.random.normal(size=n)
        y_exp = np.random.exponential(scale=1, size=n)
        tm_samples_norm_boots = trimmed_mean(y_norm, alpha=alpha, B=B, method='bootstrap')
        tm_samples_exp_boots = trimmed_mean(y_exp, alpha=alpha, B=B, method='bootstrap')
        tm_samples_norm_boots_more = trimmed_mean(y_norm, alpha=alpha, B=B2, method='bootstrap')
        tm_samples_exp_boots_more = trimmed_mean(y_exp, alpha=alpha, B=B2, method='bootstrap')
        tm_samples_norm_jack = trimmed_mean(y_norm, alpha=alpha, B=B, method='jack')
        tm_samples_exp_jack = trimmed_mean(y_exp, alpha=alpha, B=B, method='jack')
        tm_samples_norm_boots_bayes = trimmed_mean(y_norm, alpha=alpha, B=B, method='bootstrap-bayesian')
        tm_samples_exp_boots_bayes = trimmed_mean(y_exp, alpha=alpha, B=B, method='bootstrap-bayesian')

        table[k, 0] = tm_samples_norm_boots.std(ddof=1)
        table[k, 1] = tm_samples_exp_boots.std(ddof=1)
        table[k, 2] = tm_samples_norm_boots_more.std(ddof=1)
        table[k, 3] = tm_samples_exp_boots_more.std(ddof=1)
        table[k, 4] = (n-1)/np.sqrt(n) * tm_samples_norm_jack.std(ddof=1)
        table[k, 5] = (n-1)/np.sqrt(n) * tm_samples_exp_jack.std(ddof=1)
        table[k, 6] = tm_samples_norm_boots_bayes.std(ddof=1)
        table[k, 7] = tm_samples_exp_boots_bayes.std(ddof=1)

    print(table.mean(axis=0))
    print(table.std(axis=0, ddof=1))
    return table
```

Bibliography

Bickel, P. J. and Freedman, D. A. (1981). Some asymptotic theory for the bootstrap. *The annals of statistics*, pages 1196–1217.

Efron, B. and Tibshirani, R. (1986). Bootstrap methods for standard errors, confidence intervals, and other measures of statistical accuracy. *Statistical science*, pages 54–75.

Lahiri, S. N. (1999). Theoretical comparisons of block bootstrap methods. *Annals of Statistics*, pages 386–404.

Rubin, D. B. (1981). The bayesian bootstrap. *The annals of statistics*, pages 130–134.

Schervish, M. J. (2012). *Theory of Statistics*. Springer Science & Business Media.