A Comparision of Bootstrap Confidence Interval Methods

Justin Papagelis *
Department of Mathematics and Statistics, Amherst College

December 9, 2022

Abstract

This paper explores different bootstrap methods for creating confidence intervals and tests their performance against each other for different types of distributions. This paper will re-introduce the idea of bootstraps and confidence intervals, as well as the theory behind different methods of bootstrapped confidence intervals. Along with the theoretical details of each method, we will provide an example of each bootstrapped confidence interval method using a toy dataset. Then, we will run a simulation that evaluates the performance of a couple of the methods on a variety of sample sizes and types of distributions.

Keywords: simulation, percentile, bias-corrected, acceleration, studentized

*The authors gratefully acknowledge Profossor Wagaman

1 Introduction

Bootstrapping is an essential tool in Statistics, even more so now that there is access to higher computing power. We use bootstrapping to estimate the desired population parameter from a given sample without making assumptions about any underlying distributions of the sample. Different bootstrap techniques are developed, but we will focus on the non-parametric bootstrap for our purposes. One way to make a statistical inference is through confidence intervals which give a range of estimates for the unknown population parameter at a certain confidence level. We can use bootstrapping to create accurate approximate confidence intervals for our population parameter even without knowing its underlying distributions. There are many different ways to create intervals, and we will go through a couple of them: The Standard Method, The Percentile Method, The Bias-Corrected (BC) Method, The Bias-Corrected with Acceleration (BCa) Method, and finally, The Studentized Method. Additionally, we will go through an example of creating a bootstrapped confidence interval for each method. Then, we will perform a simulation to evaluate the performance of some of the bootstrapped confidence interval methods on different types of distributions.

2 Exposition

2.1 The Non-Parametric Bootstrap

Bootstrapping is a statistical method of resampling that allows the estimation of a test statistic from an unknown distribution. In particular, bootstrapping is a heavy computational method useful for many situations.

First, we introduce the non-parametric bootstrap. Suppose we have a random sample $X = (x_1, x_2, \ldots, x_n)$ from our unknown distribution, F and a statistic of interest, $\hat{\theta} = \hat{\theta}(X)$ (Efron & Hastie 2021). Ideally, the desired test statistic could be found by repeatedly sampling new reproductions of X from F. However, F is unknown, so this is not possible. The non-parametric bootstrap creates an estimate $\hat{F}$ from F using our sample, X, without making any parametric assumptions about F (such as its distribution type).

Therefore, the bootstrap sample could be represented as $X^* = (x_1^*, x_2^*, \ldots, x_n^*)$ where each x_i^* is sampled randomly with equal probability and with replacement from our original sample: $\{x_1, x_2, \ldots, x_n\}$. From this bootstrap sample, a bootstrap replication of the test statistic can be computed using $\hat{\theta}^* = \hat{\theta}(X^*)$. A large number, B, of bootstrap samples are drawn independently, and the corresponding bootstrap replication of the test statistic is calculated.

$$\hat{\theta}^{*b} = \hat{\theta}(X^{*b}) \text{ for } b = 1, 2, \ldots, B.$$

The test statistic's bootstrap estimate is the test statistic's empirical value from all of the $\hat{\theta}^{*b}$ replications. As B increases, $\hat{F}$ approaches F, which means that the test statistic of interest approaches its true value as well.

2.1.1 Using the Non-Parametric Bootstrap

We demonstrate performing a non-parametric bootstrap below using `SnowGR`, which gives the official snowfall dataset by the month for Grand Rapids, MI, starting in 1893 and is found in the `mosaic` package (Pruim et al. 2017). We will be using the `Dec` variable, which is the number of inches of snow that fell in December of each year. In particular, we are interested in the sample mean and later finding confidence intervals for the population mean of `Dec`. The distribution is shown (Figure 1).

```
data(SnowGR)
gf_histogram(~ Dec, data = SnowGR) +
  labs(x = "Annual Snowfall in December",
       title = "Distribution of Annual Snowfall in December", y = "Count",
       caption = "Fig. 1: Histogram of SnowGR")
```

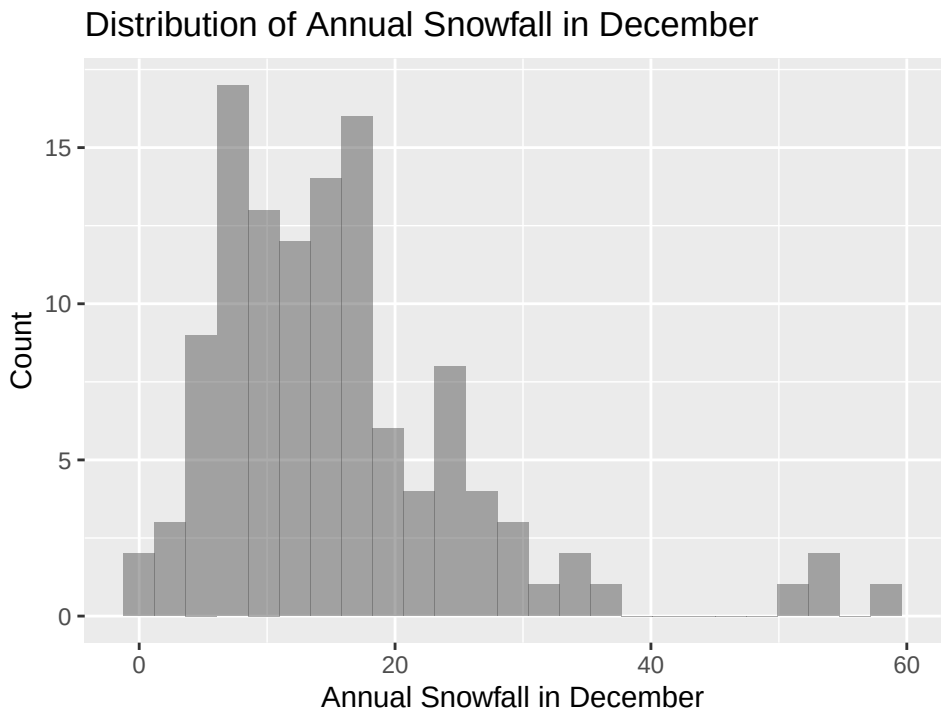

Fig. 1: Histogram of SnowGR

Next, we will perform the non-parametric bootstrap using 10,000 replications.

```
set.seed(495)
orig_mean <- mean(~ Dec, data = SnowGR) # theta hat
nboot <- 10000
# initialize storage vectors
mean <- rep(0,nboot)
sd <- rep(0,nboot)

for (i in 1:nboot) {
  resampled <- as.data.frame(mosaic::resample(SnowGR, replace = TRUE))
  mean[i] <- mean(~Dec, data = resampled)
  sd[i] <- sd(~Dec, data = resampled)
}
dec_means <- as.data.frame(mean)
```

Figure 2 shows the histogram of bootstrapped values. Since we performed a large num-

ber of bootstrap replications, the bootstrapped distribution appears to be approximately Normal.

```
gf_dhistogram(~ mean, data = dec_means) %>%
  gf_dens() %>%
  gf_labs(title = "Histogram of 10,000 Bootstrapped Mean Dec Values",
          caption = "Fig. 2: Bootstrap histogram of Dec values")
```

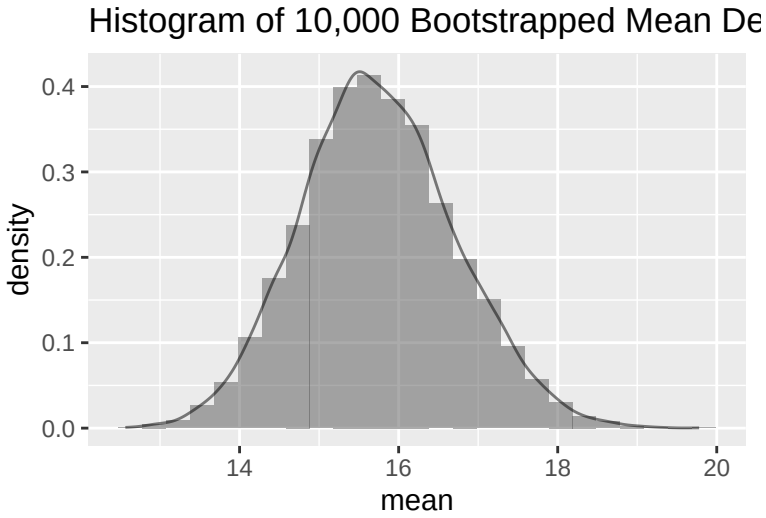

Fig. 2: Bootstrap histogram of Dec values

2.2 Standard Confidence Interval

Confidence intervals are tools that are used to estimate a parameter. Specifically, a confidence interval gives a range of values in which the true value of the parameter may lie. An α-level standard confidence interval is given by

$$\hat{\theta}_S[\alpha] = \hat{\theta} \pm z_\alpha \hat{\sigma},$$

where $\hat{\theta}$ is a point estimate of the parameter of interest θ, $\hat{\sigma}$ is the estimate of the standard deviation of $\hat{\theta}$ and z_α is the $(100{*}\alpha)$th percentile of the normal deviation (Efron & Tibshirani 1986). We say that the confidence interval constructed in this manner can capture the true parameter with a probability of α.

The standard confidence interval is built based on the assumption that the distribution we are sampling is Normal. The standard confidence interval is sometimes called the normal

confidence interval for a bootstrapped sample. This means that the standard confidence interval could present an incorrect range for an unknown skewed distribution. However, the same process can be used with bootstrap sampling to form the bootstrap percentile method. This way, an approximate bootstrap confidence interval will be created in the same automatic way that the standard confidence interval was created. For bootstrapped confidence intervals, the number of bootstrap replications B must be large (around 2,000) due to the nature of confidence intervals requiring greater accuracy (Efron & Tibshirani 1986).

2.2.1 Standard Confidence Interval in Use

Using our previous example to find a 95% confidence interval for the population mean of `Dec`, we would get the following confidence interval:

```
mean(~ Dec, data = SnowGR) + qnorm(c(0.025, 0.975)) *
  sd(~ Dec, data = SnowGR)/sqrt(nrow(SnowGR))
```

```
## [1] 13.85639 17.65957
```

Therefore, we are 95% confident that the true mean number of inches of snow that fall in December is between 13.86 inches and 17.66 inches. Our original histogram with the confidence interval is shown (Figure 3).

```
gf_histogram(~ Dec, data = SnowGR) +
  labs(x = "Annual Snowfall in December",
       title = "Distribution of Annual Snowfall in December
       with Standard Interval", y = "Count",
       caption = "Fig. 3: Histogram with Standard Interval Shown") +
  geom_vline(aes(xintercept= 13.85639)) +
  geom_vline(aes(xintercept= 17.65957))
```

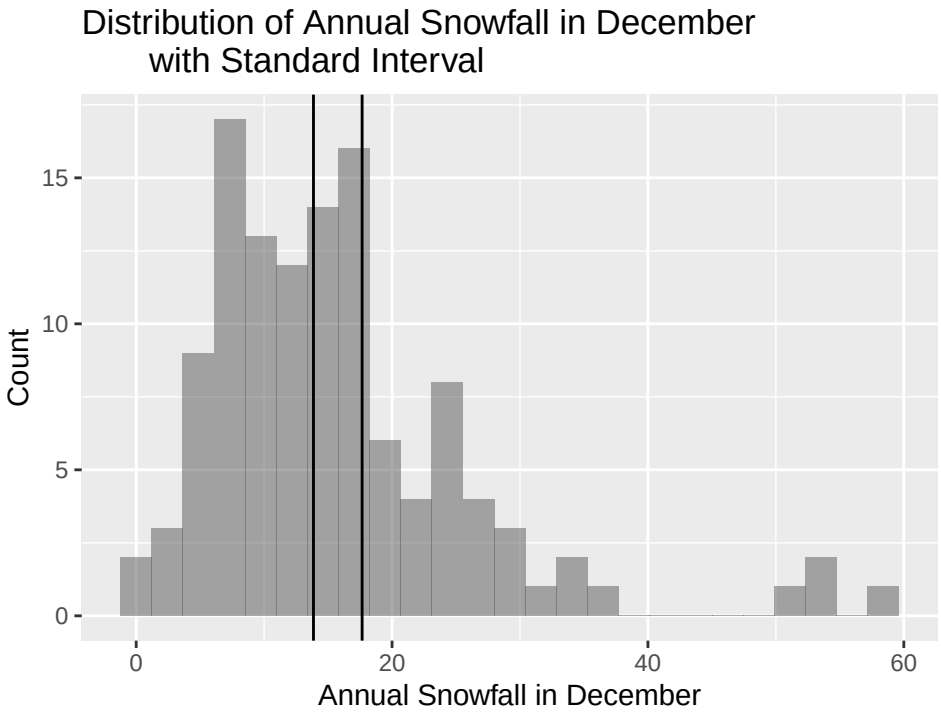

Fig. 3: Histogram with Standard Interval Shown

2.3 The Percentile Method

The percentile method interval is defined as the interval between the $100 * \alpha$ and the $100(1-\alpha)$ percentiles of the bootstrap distribution of $\hat{\theta}$. That is, the $(1-2\alpha)$ coverage interval can be defined as $[\hat{\theta}^*_\alpha, \hat{\theta}^*_{1-\alpha}]$ (Efron & Tibshirani 1986, Efron & Hastie (2021)). To go further, we can define $\hat{G}(t)$ as the bootstrap cdf, or the proportion of bootstrap samples less than t:

$$\hat{G}(t) = \frac{\#\{\hat{\theta}^{*b} \leq t\}}{B}.$$

Thus the αth percentile point of the distribution is given by

$$\hat{\theta}_p[\alpha] = \hat{\theta}^*_\alpha = \hat{G}^{-1}(\alpha).$$

It follows that the percentile interval can be represented as

$$\left[\hat{G}^{-1}(\alpha), \hat{G}^{-1}(1-\alpha)\right].$$

In the case that the bootstrap distribution of $\hat{\theta}^* \sim N(\hat{\theta}, \hat{\sigma}^2)$, the corresponding percentile interval would be equivalent to the standard interval. However, this is only sometimes the

case. When the bootstrap distribution is non-normal, we can suppose that there exists, for all θ,

$$\hat{\phi} \sim N(\phi, \tau^2),$$

for some monotone transformation $\hat{\phi} = g(\hat{\theta}), \phi = g(\theta)$, and τ is a constant. In other words, this transformation perfectly normalizes the distribution of $\hat{\theta}$. This transformation invariant can be applied to the bootstrap replications such that

$$\hat{\phi}^{*b} = g\left(\hat{\theta}^{*b}\right) \text{ for } b = 1, 2, \ldots, B.$$

The corresponding percentiles of the distribution transform similarly, $\hat{\phi}^*_\alpha = g\left(\hat{\theta}^*_\alpha\right)$. Or we can say that the $(1 - 2\alpha)$ percentile interval is $\hat{\phi} \pm \tau z_\alpha$ which can also be represented as $[\hat{\phi}^*_\alpha, \hat{\phi}^*_{1-\alpha}]$. This means that the interval on the θ scale can be defined as

$$\hat{\theta}^*_\alpha = g^{-1}(\hat{\phi} \pm \tau z_\alpha).$$

This also can be represented as an interval,

$$\left[g^{-1}(\hat{\phi} \pm \tau z_{1-\alpha}), g^{-1}(\hat{\phi} \pm \tau z_\alpha)\right].$$

Therefore, the percentile method produces a correct interval for ϕ and, due to the transformation invariance, also produces a correct percentile interval for θ. This method assumes the existence of some monotone normalizing mapping $\hat{\phi} = g(\hat{\theta}), \phi = g(\theta)$ and relies on that to create a correct interval. Since the process is automatic, we do not need to know the transformation itself, only that it exists. However, in some cases, no monotone normalizing mapping will exist (Efron & Tibshirani 1986).

2.3.1 The Percentile Method in Use

Finding a 95% confidence interval for the true population mean of `Dec` using the percentile method, we get:

```
qdata(~ mean, c(0.025, 0.975), data = dec_means)
```

```
##     2.5%    97.5%
## 13.91170 17.72353
```

Therefore, we can say that we are 95% confident that the true population mean of inches of snow that fall in December is between 13.91 inches and 17.72 inches. Our original histogram with the percentile confidence interval is shown (Figure 4).

```
gf_histogram(~ Dec, data = SnowGR) +
  labs(x = "Annual Snowfall in December",
       title = "Distribution of Annual Snowfall in December
       with Percentile Interval", y = "Count",
       caption = "Fig. 4: Histogram with Percentile Interval Shown") +
  geom_vline(aes(xintercept= 13.91170)) +
  geom_vline(aes(xintercept= 17.72353))
```

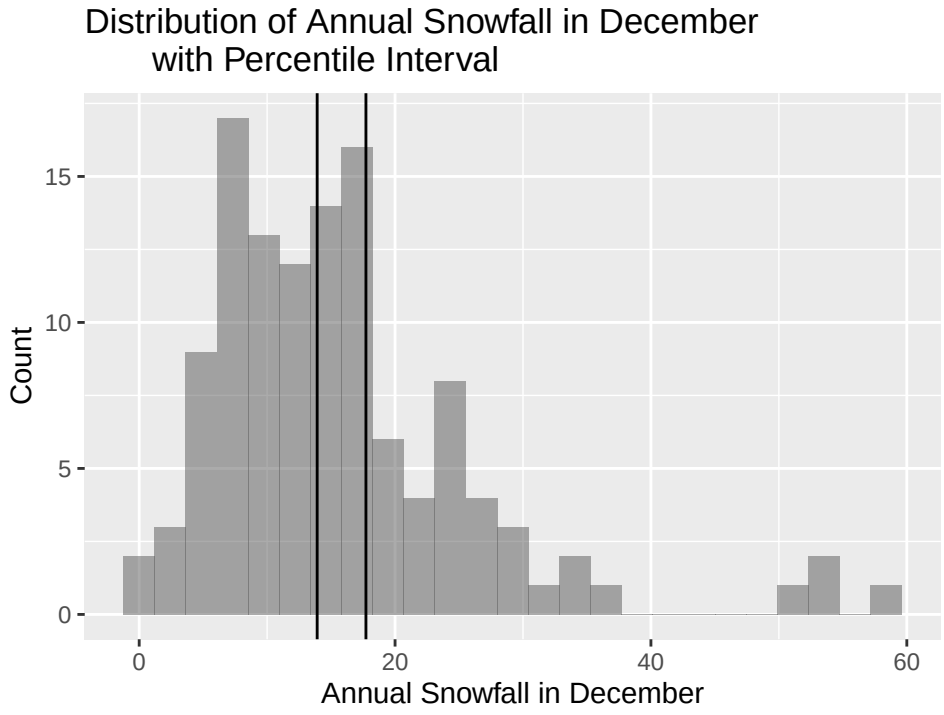

Fig. 4: Histogram with Percentile Interval Shown

This interval is similar to the one created using the standard method, but shifted slightly to the right.

2.4 The Bias-Corrected (BC) Method

The following method we will be looking at is the bias-corrected percentile method (BC method) which is an improvement upon the previous percentile method because we now consider the possibility of bias. It can be shown that $\hat{\theta}$ is biased upwards relative to θ, which means that the confidence intervals should be adjusted downwards (Efron & Tibshirani 1986, Efron & Hastie (2021)). From our simulated bootstrap replications $\hat{\theta}^{*1}, \hat{\theta}^{*2}, \ldots, \hat{\theta}^{*B}$, define

$$p_0 = \frac{\#\{\hat{\theta}^{*b} \leq \theta\}}{B},$$

and define the bias-correction value

$$z_0 = \Phi^{-1}(p_0),$$

where Φ^{-1} is the inverse function of the standard normal cdf. Thus, we define a transformation $\hat{\phi} = g(\hat{\theta}), \phi = g(\theta)$ such that for any θ,

$$\hat{\phi} \sim N(\phi - z_0\tau, \tau^2),$$

with z_0 and τ constants. This means that we can say the bias-corrected method has an α-level endpoint which can be represented as

$$\hat{\theta}_{BC}[\alpha] = \hat{G}^{-1}\left[\Phi\left(2z_0 + z_\alpha\right)\right].$$

If $\hat{G} = 0.50$, then half of the bootstrap distribution is less than $\hat{\theta}$ and our bias-correction value $z_0 = 0$. In this case, the confidence interval produced by BC would be the same interval that was produced by the percentile method.

2.4.1 The BC Method in Use

Using this method to create a 95% confidence interval has a couple more steps because we need to find the bias-correction value (Efron & Hastie 2021). First, we calculate the sample mean:

```
sample_mean <- mean(~Dec, data = SnowGR); sample_mean
```

```
## [1] 15.75798
```

Then we find the proportion of bootstrap replications that have a sample mean less than the original sample mean.

```
less_than_sample_mean <- sum(ifelse(dec_means <= sample_mean, 1, 0))/10000
less_than_sample_mean
```

```
## [1] 0.5208
```

From this, we can calculate the bias-correction value, z_0:

```
z0 <- qnorm(less_than_sample_mean); z0
```

```
## [1] 0.05216151
```

Then we can find the modified percentiles and get the corresponding confidence interval:

```
alpha_lower <- 0.025; alpha_upper <- 0.975

new_lower <- pnorm(2*z0 + qnorm(alpha_lower));
new_upper <- pnorm(2*z0 + qnorm(alpha_upper));

qdata(~ mean, c(new_lower, new_upper), data = dec_means)
```

```
## 3.175238% 98.05047%
##  14.01172  17.82017
```

Therefore, we are 95% confident that the true mean inches of snow that fall in December in Grand Rapids, Michigan is between 14.01 inches and 17.82 inches. Our histogram (Figure 5) is shown.

```
gf_histogram(~ Dec, data = SnowGR) +
  labs(x = "Annual Snowfall in December",
       title = "Distribution of Annual Snowfall in December
       with BC interval", y = "Count",
       caption = "Fig. 5: Histogram with BC Interval Shown") +
  geom_vline(aes(xintercept= 14.01172)) +
  geom_vline(aes(xintercept= 17.82017))
```

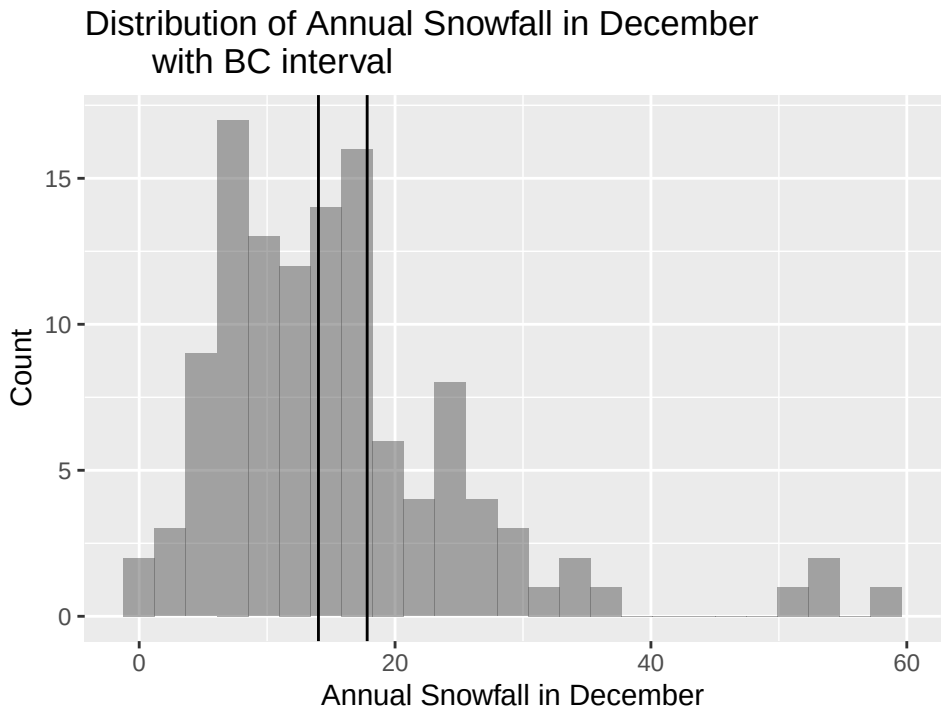

Fig. 5: Histogram with BC Interval Shown

2.5 The Bias-Corrected and Accelerated (BCa) Method

The bias-corrected and accelerated method (BCa) further modifies the BC interval. This method does not assume that the standard error, τ, is constant as we did in the BC interval (Efron & Tibshirani 1986, Efron & Hastie (2021)). Rather, we assume the existence of a monotone transformation $\hat{\phi} = g(\hat{\theta}), \phi = g(\theta)$ such that for any θ,

$$\hat{\phi} \sim N(\phi - z_0\tau_\phi, \tau_\phi^2) \text{ where } \tau_\phi = 1 + a\phi.$$

The a is known as the acceleration and is a constant that describes how the standard deviation of $\hat{\phi}$ varies with ϕ. In other words, a is proportional to the skewness of the bootstrap distribution. For example, $a = z_0$ for one-parameter exponential families. However, there are many different algorithms to compute and estimate a (Flowers-Cano et al. 2018). Now, our α-level endpoint for a BCa confidence interval is

$$\hat{\theta}_{BCa}[\alpha] = \hat{G}^{-1}\left[\Phi\left(z_0 + \frac{z_0 + z_\alpha}{1 - a(z_0 + z_a)}\right)\right].$$

We can observe that if $a = 0$, then $\hat{\theta}_{BCa}[\alpha] = \hat{\theta}_{BC}[\alpha]$. When calculating a BCa confidence interval, the acceleration value, a, is not a function of the bootstrap distribution and must be calculated separately. However, the process is algorithmic and can be calculated without too much work. As we saw, each of the three previous methods (percentile, BC, and BCa) all build upon each other and have less restrictive assumptions, but computation increases as we loosen assumptions.

2.5.1 The BCa Confidence Interval in Use

In practice, we can use the jackknife procedure to estimate the acceleration value for this method (Efron & Hastie 2021). The jackknife procedure is a different resampling technique often used to estimate the bias and variance of a large population. Thus, we will run the jackknife procedure:

```
theta <- function(x){mean(x)}
jackknife_results <- bootstrap::jackknife(SnowGR$Dec, theta)
```

We find the mean of the jackknife values below:

```
jack_mean <- mean(~ jackknife_results$jack.values); jack_mean
```

```
## [1] 15.75798
```

Then we can compute an approximation for a.

```
estimated_a <- (1/6)*sum((jackknife_results$jack.values - jack_mean)^3)/
  (sum((jackknife_results$jack.values - jack_mean)^2))^(1.5); estimated_a
```

```
## [1] -0.02674911
```

Now, we can find the adjusted percentiles and the bias-corrected with acceleration confidence interval.

```
a_new_lower <- pnorm(z0 + (z0 + qnorm(alpha_lower))/
                     (1-estimated_a*(z0 + qnorm(alpha_lower))));
a_new_upper <- pnorm(z0 + (z0 + qnorm(alpha_upper))/
                      (1-estimated_a*(z0 + qnorm(alpha_upper))));
qdata(~ mean, c(a_new_lower, a_new_upper), data = dec_means)
```

```
## 2.510119% 97.50908%
##  13.91591  17.72353
```

We are 95% confident that the true mean number of inches of snow that fall in December is between 13.92 inches and 17.72 inches. Figure 6 shows our confidence interval.

```
gf_histogram(~ Dec, data = SnowGR) +
  labs(x = "Annual Snowfall in December",
       title = "Distribution of Annual Snowfall in December
       with BCa interval", y = "Count",
       caption = "Fig. 6: Histogram with BCa Interval Shown") +
  geom_vline(aes(xintercept= 13.91591)) +
  geom_vline(aes(xintercept= 17.72353))
```

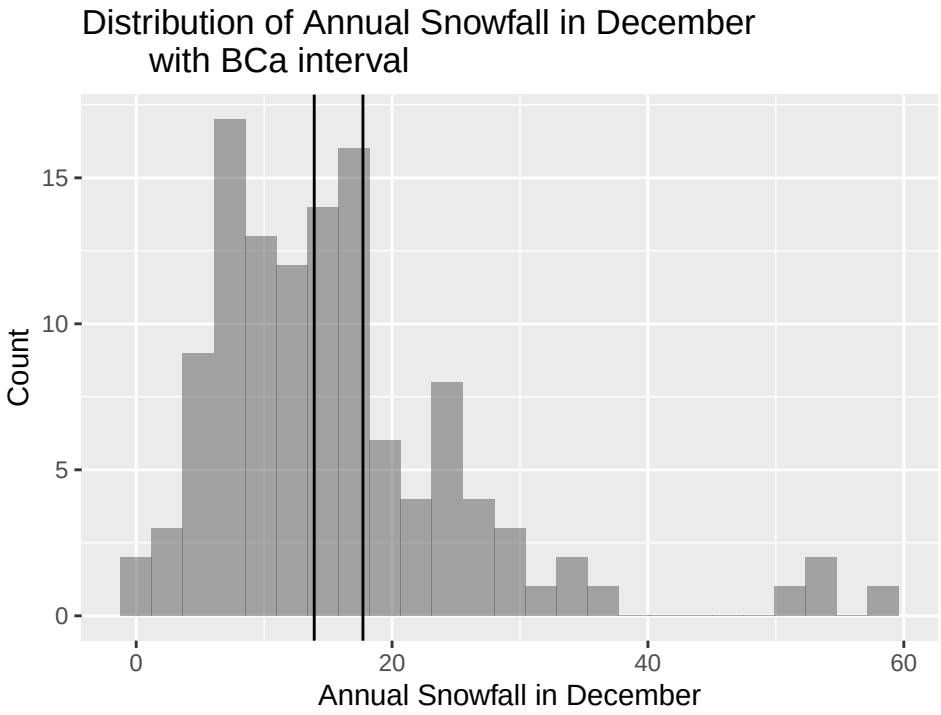

Fig. 6: Histogram with BCa Interval Shown

2.6 The Bootstrap-t (Studentized) Method

The final method we explain is the bootstrap-t interval (or the studentized method). Recall from earlier, our sample, X from which we can calculate an estimate ($\hat{\theta}(X)$) of the parameter of interest θ (Efron & Tibshirani 1986, Puth et al. (2015)). We can also estimate $\hat{\sigma}(X)$ for the standard error of θ. We can use these parameters to find the Student's t-statistic defined as:

$$T = \frac{\hat{\theta} - \theta}{\hat{\sigma}}.$$

As such, the αth percentile point of a confidence interval of θ would be $\hat{\theta} - \hat{\sigma}T_{(\alpha)}$ where $T_{(\alpha)}$ represents the αth percentile of the t-distribution, T. Unfortunately, the percentiles of this t-distribution are unknown in most cases, but we can use bootstrapping to estimate these percentiles. To do this, we perform a large number, B, of bootstrap samples, from which we can find the bootstrap replications of the parameter of interest, $\hat{\theta}^* = \hat{\theta}(X^*)$, and the standard error, $\hat{\sigma^*} = \hat{\sigma}(X^*)$. From these, we can calculate a t-statistic for each bootstrap sample:

$$T^* = \frac{\hat{\theta}^* - \hat{\theta}}{\hat{\sigma}^*}.$$

Using a large number of these bootstrap samples, we can estimate the percentiles of the t-distribution such that:

$$\hat{T}_{(\alpha)} = B * \alpha\text{th ordered value of all the bootstrap replications of } T^*.$$

This means that for $B = 2,000$ and $\alpha = 0.90$, then $\hat{T}_{(\alpha)}$ is the 1,800th ordered point of all of the bootstrap replications of T^*. It follows that the αth studentized confidence interval endpoint can be given with

$$\hat{\theta}_T[\alpha] = \hat{\theta} - \hat{\sigma} T_{(\alpha)},$$

where we can estimate the standard error using

$$\hat{\sigma} = \frac{1 - \hat{\theta}^2}{\sqrt{n}}.$$

One of the main factors why the studentized method is so popular is that we assume that our bootstrapped statistic is pivotal, which means that the confidence interval does not depend on any other parameters. Instead, we can calculate the appropriate confidence interval for the parameter of interest specifically from the bootstrapped statistics (Efron & Tibshirani 1986, Puth et al. (2015)).

2.6.1 Studentized Confidence Interval in Use

Following the process above, we can use the following to get our studentized confidence interval (Lau 2020):

```
set.seed(495)
# sample statistics
orig_mean <- mean(~ Dec, data = SnowGR) # theta hat
orig_se <- sd(~ Dec, data = SnowGR)/sqrt(nrow(SnowGR))
# initialize storage vectors
se <- rep(0,nboot)
t_stat <- rep(0,nboot)
# use bootstrap replication statistics to find se and t-stat
for (i in 1:nboot) {
```

```
  se[i] <- sd[i]/sqrt(nrow(SnowGR))
  t_stat[i] <- (mean[i] - orig_mean)/se[i]
}
#find interval
dec_t_stat <- as.data.frame(t_stat)
q <- unname(quantile(dec_t_stat$t_stat, c(0.025, 0.975)))
lower <- q[1]
upper <- q[2]
c(orig_mean - upper*orig_se, orig_mean - lower*orig_se)
```

```
## [1] 14.03467 18.01709
```

So we are 95% confident that the true mean number of inches of snow that fall in December in Grand Rapids, Michigan is between 14.03 inches and 18.02 inches. This would give us the following confidence interval (Figure 7):

```
gf_histogram(~ Dec, data = SnowGR) +
  labs(x = "Annual Snowfall in December",
       title = "Distribution of Annual Snowfall in December
       with Studentized Interval", y = "Count",
       caption = "Fig. 7: Histogram with Studentized Interval Shown") +
  geom_vline(aes(xintercept= 14.03467)) +
  geom_vline(aes(xintercept= 18.01709))
```

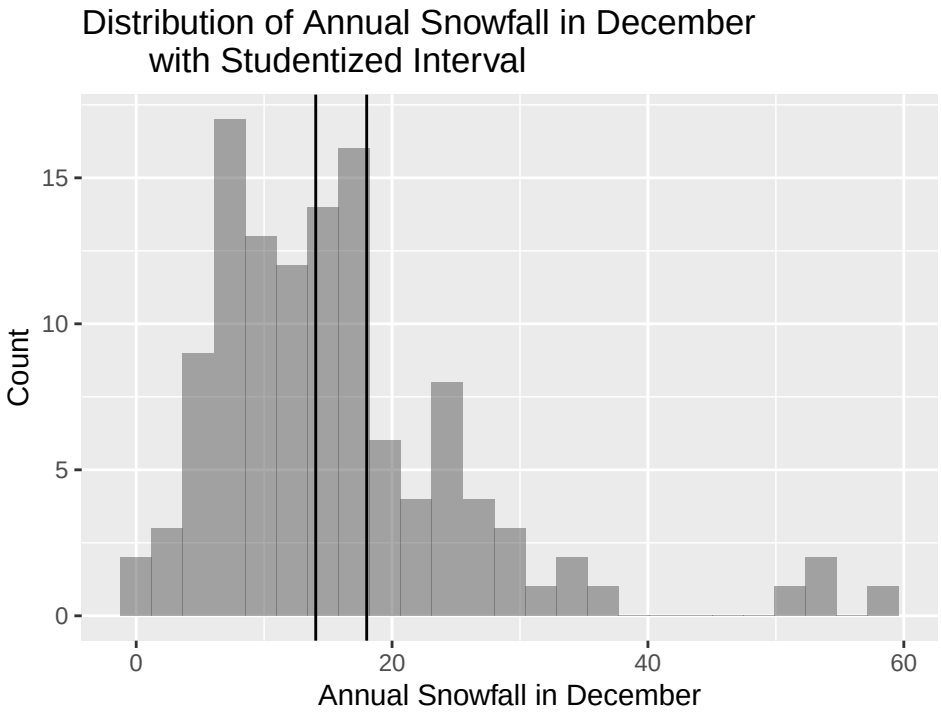

Fig. 7: Histogram with Studentized Interval Shown

3 Simulation

3.1 Code

For our simulation, we will randomly sample from distributions in which we know the true population mean. For example, we will be sampling from a Gamma distribution where we can find the true population mean using the shape and rate parameters of the sample distribution. After we sample from the distribution, we will create a bootstrapped confidence interval for a few of the methods we described in the exposition. Since we know the true population mean of the distribution, we will determine if the true population mean is contained in the confidence interval for each of the methods we are testing (Rousselet et al. 2021). We will perform this process a large number of times (10,000 simulations) and then average by the number of iterations to determine, on average, the proportion that each bootstrapped confidence interval method contains the true population mean for a given distribution. We will do this for a couple of different distributions and ranges of sample sizes to show how the intervals perform with skewed distributions and larger sample

sizes. We will define the accuracy of each method as the proportion that each bootstrapped confidence interval method contains the true population mean for a distribution

To perform this simulation, we will use the `boot` package (Canty & Ripley 2022). This package can compute a bootstrap object which contains the output from a bootstrap calculation of a given sample and then produces confidence intervals from the bootstrap object. In particular, we will use the `boot.ci` function to create four confidence intervals: the standard (or normal) confidence interval, the studentized confidence interval, the percentile confidence interval, and the bias-corrected with acceleration confidence interval.

To start, we will declare some constants, including the sample sizes and the number of bootstrap replications we use. We will be using a range of sample sizes from 5 to 150. For this simulation, we will use a confidence level of 95%.

```
sample_sizes <- c(5, 10, 15, 20, 30, 40, 60, 100, 150)
num_sample_sizes <- length(sample_sizes)
boot_reps <- 2000
```

Next, we create a simulation that randomly selects a sample from a Normal distribution and performs a bootstrap with 2,000 bootstrap replications. Then, we create the confidence intervals from this bootstrap and determine if the true mean, our population parameter, is contained within our intervals. Since we are using a $Normal(0, 1)$ distribution, our true population mean equals 0. We will repeat this process 10,000 times as specified for each sample size we are testing. Since we are initially sampling from a Normal distribution, the assumptions we hold for the standard interval are true. Thus, we would expect all of the methods to perform comparably to each other at large sample sizes.

```
run_sim_normal <- function(sim_reps) {
  # create function to return mean and sd of bootstrap replication
  boot.stat <- function(d, i) {
    d <- X[i]
    return (c(mean(d), sd(d)))
  }
  # initialize matrix to hold different bootstrap prop.
```

```r
# for each sample size
master <- matrix(nrow = num_sample_sizes, ncol = 5)
# iterate through each sample size
for(i in 1:num_sample_sizes) {
  # initialize vectors that hold if the CI contains true mean
  # by iteration
  norm_contains <- rep(0, sim_reps)
  student_contains <- rep(0, sim_reps)
  percentile_contains <- rep(0, sim_reps)
  bca_contains <- rep(0, sim_reps)

  n <- sample_sizes[i]
  true_mean <- 0
  # perform 10,000 iterations of bootstraps
  for (j in 1:sim_reps) {
    X <- rnorm(n, 0, 1)
    values <- data.frame(X)
    # perform bootstrap and CIs
    results <- boot(data = values, statistic = boot.stat, R = boot_reps)
    ci <- boot.ci(results, conf = 0.95,
                  type = c("norm", "perc", "bca", "stud"))
    # update vectors depending on if the CI contains true mean
    norm_contains[j] <- true_mean >= ci$normal[2] &
      true_mean <= ci$normal[3]
    student_contains[j] <- true_mean >= ci$student[4] &
      true_mean <= ci$student[5]
    percentile_contains[j] <- true_mean >= ci$percent[4] &
      true_mean <= ci$percent[5]
    bca_contains[j] <- true_mean >= ci$bca[4] &
      true_mean <= ci$bca[5]
```

```
    }
    # add the sample size and average proportion to matrix
    master[i, 1] <- sample_sizes[i]
    master[i, 2] <- sum(norm_contains)/sim_reps
    master[i, 3] <- sum(student_contains)/sim_reps
    master[i, 4] <- sum(percentile_contains)/sim_reps
    master[i, 5] <- sum(bca_contains)/sim_reps
  }
  # some wrangling
  master_df <- as.data.frame(master) %>%
      rename("Sample_Size" = V1, "Normal" = V2, "Studentized" = V3,
             "Percentile" = V4, "BCa" = V5)
  master_df
}
```

Secondly, we select from a Gamma Distribution with a shape parameter of one and a scale parameter of four ($Gamma(1, 4)$). Therefore, the true population mean is 4 because the product of the shape and scale parameters gives the true mean. This simulation follows the same process as above, where we sampled from the Normal distribution, except we are sampling from a Gamma distribution this time. Since our Gamma distribution is heavily left-skewed, we expect the studentized confidence interval and the BCa confidence interval to perform the best.

```
run_sim_gamma <- function(sim_reps) {
  boot.stat <- function(d, i) {
    d <- X[i]
    return (c(mean(d), sd(d)))
  }
  master <- matrix(nrow = num_sample_sizes, ncol = 5)
  for(i in 1:num_sample_sizes) {
    norm_contains <- rep(0, sim_reps)
```

```
  student_contains <- rep(0, sim_reps)
  percentile_contains <- rep(0, sim_reps)
  bca_contains <- rep(0, sim_reps)
  n <- sample_sizes[i]
  true_mean <- 4
  for (j in 1:sim_reps) {
    X <- rgamma(sample_sizes, shape = 1, scale = 4)
    values <- data.frame(X)
    results <- boot(data = values, statistic = boot.stat, R = boot_reps)
    ci <- boot.ci(results, conf = 0.95,
                  type = c("norm", "perc", "bca", "stud"))
    norm_contains[j] <- true_mean >= ci$normal[2] &
      true_mean <= ci$normal[3]
    student_contains[j] <- true_mean >= ci$student[4] &
      true_mean <= ci$student[5]
    percentile_contains[j] <- true_mean >= ci$percent[4] &
      true_mean <= ci$percent[5]
    bca_contains[j] <- true_mean >= ci$bca[4] &
      true_mean <= ci$bca[5]
  }
  master[i, 1] <- sample_sizes[i]
  master[i, 2] <- sum(norm_contains)/sim_reps
  master[i, 3] <- sum(student_contains)/sim_reps
  master[i, 4] <- sum(percentile_contains)/sim_reps
  master[i, 5] <- sum(bca_contains)/sim_reps
}
master_df <- as.data.frame(master) %>%
    rename("Sample_Size" = V1, "Normal" = V2, "Studentized" = V3,
           "Percentile" = V4, "BCa" = V5)
master_df
```

```
}
```

Lastly, we will be sampling from a Log-Normal distribution. This means that if we have a random variable, $X \sim Lognormal$, then the random variable, $Y = ln(X) \sim Normal$. Therefore, the distribution is left-skewed, although not as heavily as the previously used Gamma distribution. The true mean of this distribution can be found using $E(X) = e^{u+\frac{\sigma^2}{2}}$ which means that for a $Lognormal(0, 1)$ distribution, we expect the population mean to be $e^{\frac{1}{2}}$. Again, we expect the studentized confidence interval and the BCa confidence interval to perform the best.

```
run_sim_log_normal <- function(sim_reps) {
  boot.stat <- function(d, i) {
    d <- X[i]
    return (c(mean(d), sd(d)))
  }
  master <- matrix(nrow = num_sample_sizes, ncol = 5)
  for(i in 1:num_sample_sizes) {
    norm_contains <- rep(0, sim_reps)
    student_contains <- rep(0, sim_reps)
    percentile_contains <- rep(0, sim_reps)
    bca_contains <- rep(0, sim_reps)
    n <- sample_sizes[i]
    true_mean <- exp(1/2)
    for (j in 1:sim_reps) {
      X <- rlnorm(n, 0, 1)
      values <- data.frame(X)
      results <- boot(data = values, statistic = boot.stat, R = boot_reps)
      ci <- boot.ci(results, conf = 0.95,
                    type = c("norm", "perc", "bca", "stud"))
      norm_contains[j] <- true_mean >= ci$normal[2] &
        true_mean <= ci$normal[3]
```

```
      student_contains[j] <- true_mean >= ci$student[4] &
        true_mean <= ci$student[5]
      percentile_contains[j] <- true_mean >= ci$percent[4] &
        true_mean <= ci$percent[5]
      bca_contains[j] <- true_mean >= ci$bca[4] &
        true_mean <= ci$bca[5]
    }
    master[i, 1] <- sample_sizes[i]
    master[i, 2] <- sum(norm_contains)/sim_reps
    master[i, 3] <- sum(student_contains)/sim_reps
    master[i, 4] <- sum(percentile_contains)/sim_reps
    master[i, 5] <- sum(bca_contains)/sim_reps
  }
  master_df <- as.data.frame(master) %>%
      rename("Sample_Size" = V1, "Normal" = V2, "Studentized" = V3,
             "Percentile" = V4, "BCa" = V5)
  master_df
}
```

Once we have all the simulations, we can run them below.

```
set.seed(495)
normal_df <- run_sim_normal(10000)
write.csv(normal_df,"data/normal_data_df.csv", row.names = FALSE)

log_normal_df <- run_sim_log_normal(10000)
write.csv(log_normal_df,"data/log_normal_df.csv", row.names = FALSE)

gamma_df <- run_sim_gamma(10000)
write.csv(gamma_df,"data/gamma_data_df.csv", row.names = FALSE)
```

After, we can create the graphs and output for each of the distributions and sample

sizes.

```
normal_df <- read.csv(file = 'data/normal_data_df.csv')
gamma_df <- read.csv(file = 'data/gamma_data_df.csv')
log_normal_df <- read.csv(file = 'data/log_normal_df.csv')

# create the figure that shows the normal output
normal_df_long <- normal_df %>%
  pivot_longer(cols = 2:5, names_to = "method", values_to = "prop")

normal_fig <- ggplot(data = normal_df_long,
                     aes(x = Sample_Size, y = prop, color = method)) +
  geom_point() + geom_line() +
  labs(
    title = "Accuracy of Bootstrap CIs using Normal Distibution",
    x = "Sample Size",
    y = "Proportion (of 10,000 Simulations)", color = "Method",
    caption = "Fig. 8: Graph showing the Accuracy of Bootstrap CI
    Methods for different sample sizes on Normal Distribution") +
  theme_light() +
  theme(legend.position="bottom")

# create the table for the normal output
normal_table <- normal_df %>%
  kable(booktabs = TRUE, align = "c",
        caption = "From a Normal distribution",
        col.names = gsub("[_]", " ", names(normal_df)), digits = 3)

# create the figure that shows the gamma output
gamma_df_long <- gamma_df %>%
  pivot_longer(cols = 2:5, names_to = "method", values_to = "prop")
```

```r
gamma_fig <- ggplot(data = gamma_df_long,
                    aes(x = Sample_Size, y = prop, color = method)) +
  geom_point() + geom_line() +
  labs(title = "Accuracy of Bootstrap CIs using Gamma Distibution",
       x = "Sample Size",
       y = "Proportion (of 10,000 Simulations)", color = "Method",
       caption = "Fig. 9: Graph showing the Accuracy of Bootstrap CI
       Methods for different sample sizes on Gamma Distribution") +
  theme_light() +
  theme(legend.position="bottom")

# create the table for the gamma output
gamma_table <- gamma_df %>%
  kable(booktabs = TRUE, align = "c",
        caption = "From a Gamma distribution",
        col.names = gsub("[_]", " ", names(gamma_df)), digits = 3)

# create the figure that shows the log normal output
log_normal_df_long <- log_normal_df %>%
  pivot_longer(cols = 2:5, names_to = "method", values_to = "prop")

log_normal_fig <- ggplot(data = log_normal_df_long,
                    aes(x = Sample_Size, y = prop, color = method)) +
  geom_point() + geom_line() +
  labs(title ="Accuracy of Bootstrap CIs using Log-Normal
       Distibution",
       x = "Sample Size",
       y = "Proportion (of 10,000 Simulations)", color = "Method",
       caption = "Fig. 10: Graph showing the Accuracy of Bootstrap CI
```

```
       Methods for different sample sizes on Log-Normal Distribution") +
  theme_light() +
  theme(legend.position="bottom")

# create the table for the log normal output
log_normal_table <- log_normal_df %>%
  kable(booktabs = TRUE, align = "c",
        caption = "From a Log-Normal distribution",
        col.names = gsub("[_]", " ", names(log_normal_df)), digits = 3)
```

3.2 Results

3.2.1 Normal Distribution Simulation

First, we have the results of the simulation where we randomly sampled from a Normal distribution.

```
normal_fig
```

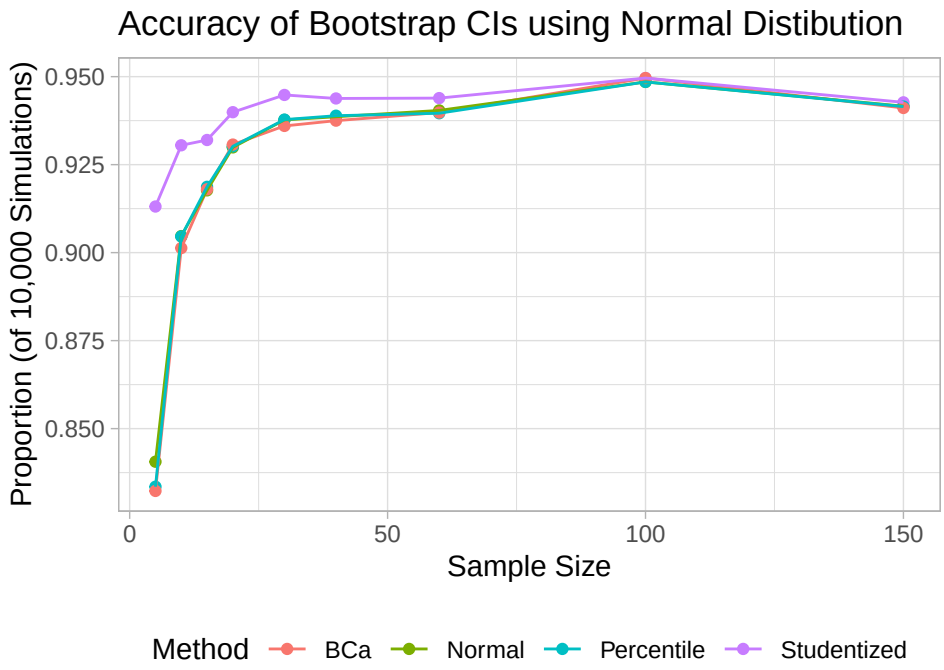

Fig. 8: Graph showing the Accuracy of Bootstrap CI Methods for different sample sizes on Normal Distribution

Table 1: From a Normal distribution

Sample Size	Normal	Studentized	Percentile	BCa
5	0.841	0.913	0.834	0.832
10	0.905	0.930	0.905	0.901
15	0.918	0.932	0.919	0.918
20	0.930	0.940	0.930	0.931
30	0.938	0.945	0.938	0.936
40	0.939	0.944	0.939	0.938
60	0.940	0.944	0.940	0.940
100	0.949	0.950	0.949	0.950
150	0.942	0.943	0.942	0.941

At small sample sizes, the studentized confidence interval performed much better than the other intervals, which all performed similarly. For a sample size of 5, the studentized interval had an accuracy of around 91.3%. In comparison, the other three intervals had an accuracy of around 83% to 84% of creating an interval that contained the true population mean. The standard, percentile, and BCa confidence interval methods all have similar processes, especially when the distribution we are sampling from is Normal because the assumptions we made for the standard interval are correct. As the sample size increases, the performance of the intervals increases and rises to stay consistent at around 94% to 95% accuracy. The data (Table 1) is shown for reference.

```
normal_table
```

3.2.2 Gamma Distribution Simulation

Second, we have the results from the simulation where we sampled from a Gamma distribution.

```
gamma_fig
```

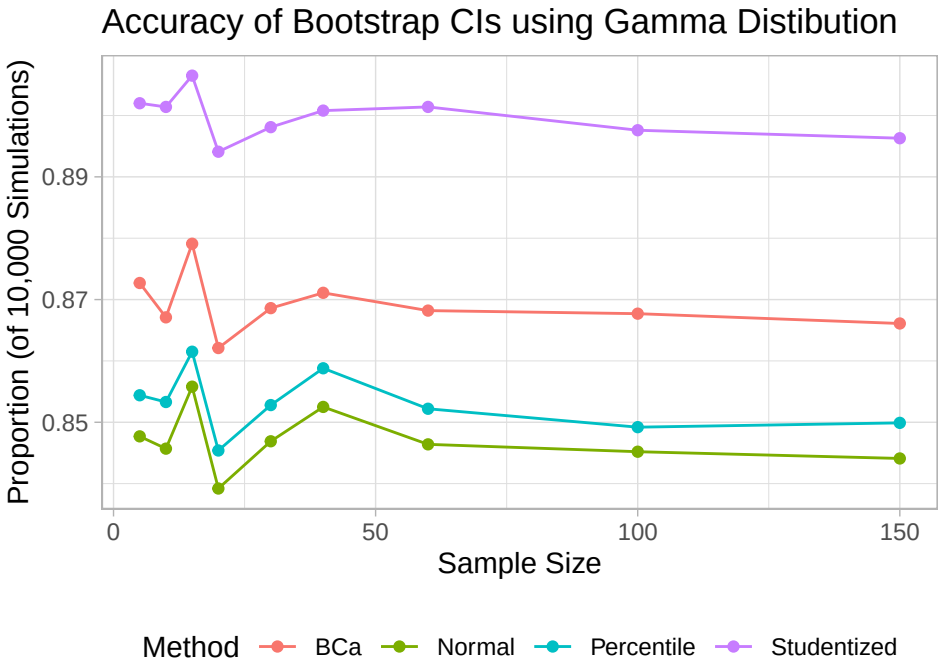

Fig. 9: Graph showing the Accuracy of Bootstrap CI Methods for different sample sizes on Gamma Distribution

The $Gamma(1, 4)$ distribution we randomly sampled from is heavily-skewed, so we did not expect all of the bootstrap confidence interval methods to work comparably. As shown in Figure 9, the studentized method performed the best at around 90% accuracy for creating a confidence interval that contained the true mean. The BCa method performed the second best at around 87% accuracy. The percentile method performed the following best with around 85% accuracy, while the standard method performed the worst with an accuracy of around 84.5%. As the sample size increased, the accuracy of each of the methods became less variable and evened out to a constant accuracy. The table containing the proportions (Table 2) is shown for reference.

```
gamma_table
```

3.2.3 Log-Normal Distribution Simulation

Lastly, we have the results from the simulation where we sampled from a Log-Normal distribution.

Table 2: From a Gamma distribution

Sample Size	Normal	Studentized	Percentile	BCa
5	0.848	0.902	0.854	0.873
10	0.846	0.901	0.853	0.867
15	0.856	0.906	0.862	0.879
20	0.839	0.894	0.845	0.862
30	0.847	0.898	0.853	0.869
40	0.853	0.901	0.859	0.871
60	0.846	0.901	0.852	0.868
100	0.845	0.898	0.849	0.868
150	0.844	0.896	0.850	0.866

```
log_normal_fig
```

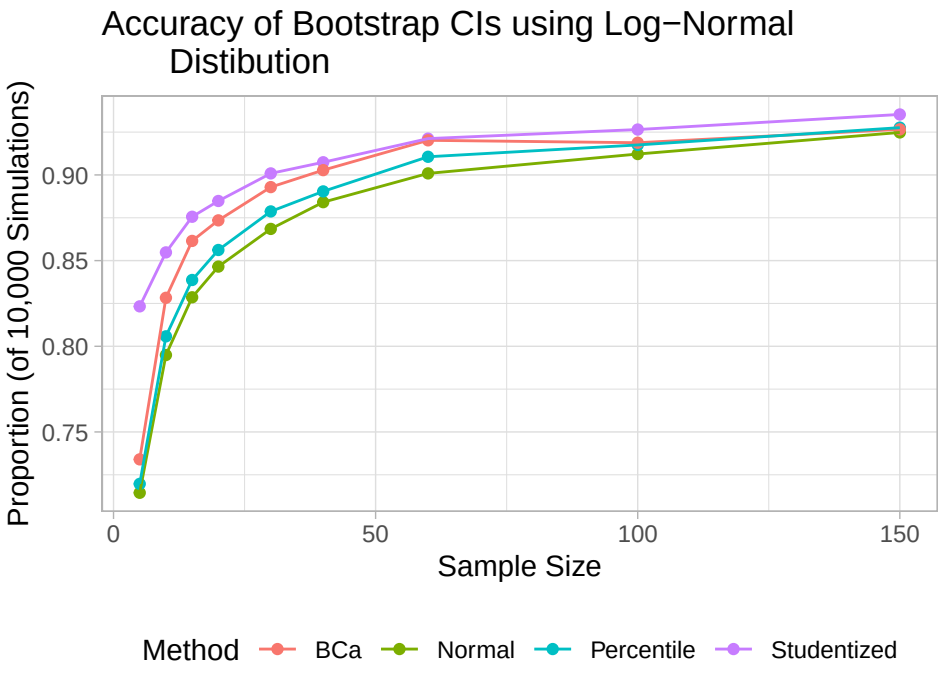

Fig. 10: Graph showing the Accuracy of Bootstrap CI Methods for different sample sizes on Log−Normal Distribution

As shown in Figure 10, the studentized method had the greatest accuracy in creating

Table 3: From a Log-Normal distribution

Sample Size	Normal	Studentized	Percentile	BCa
5	0.715	0.823	0.720	0.734
10	0.795	0.855	0.806	0.828
15	0.829	0.876	0.839	0.862
20	0.847	0.885	0.856	0.874
30	0.869	0.901	0.879	0.893
40	0.884	0.907	0.890	0.903
60	0.901	0.921	0.911	0.920
100	0.912	0.926	0.917	0.919
150	0.925	0.935	0.928	0.927

the confidence intervals that captured the true mean. At small sample sizes such as 5, the studentized method has an accuracy of 82.3%, while the following best is the BCa method with a 73.4% accuracy. As the sample size increases, all methods' accuracy increases quickly and performs comparably, although the studentized and BCa methods perform slightly better. All the intervals stay constant at around 92.5% to 93.5% accuracy for large sample sizes. Again, the proportions are shown (Table 3) for each sample size for reference.

```
log_normal_table
```

4 Conclusion

In this paper, we demonstrated the power of using bootstrapped confidence intervals to estimate a true population parameter without using any information about the underlying distribution of the sample. In the exposition, we introduced the standard method, which relies on three conditions: 1) that the distribution is Normal, 2) that there is no bias, and 3) that there is a constant standard error. If we take away the first condition and use the bootstrap percentiles to create the interval, we are left with the percentile method. If we add a bias-correcting value not equal to zero, then we have the bias-correcting method. Furthermore, if we allow for a non-constant standard error (through the acceleration), we have the bias-corrected with acceleration method. Using the studentized method, we can also use bootstrap resampling to estimate a t-distribution from which we can create a confidence interval. From our simulation, the methods all performed reasonably the same for the Normal distribution at large sample sizes. However, when we introduced skew into the distributions, the studentized and BCa methods performed better than the others at large sample sizes. It is important to note that even at large sample sizes, none of our methods reached a coverage of 95%, which was our specified confidence level. The closest we were was when we sampled from a Normal distribution. This may be because our sample sizes need to be bigger; however, we can reach a close approximation using the largest sample size we tested. Thanks to the high computational power available, we can use bootstrapping to create confidence intervals for a population parameter without knowing much about the actual sample.

References

Canty, A. & Ripley, B. D. (2022), *boot: Bootstrap R (S-Plus) Functions*. R package version 1.3-28.1.

Efron, B. & Hastie, T. (2021), *Computer Age Statistical Inference, Student Edition: Algorithms, Evidence, and Data Science*, Vol. 6, Cambridge University Press.

Efron, B. & Tibshirani, R. (1986), 'Bootstrap methods for standard errors, confidence

intervals, and other measures of statistical accuracy', *Statistical Science* **1**(1), 54–75.
URL: *http://www.jstor.org/stable/2245500*

Flowers-Cano, R. S., Ortiz-Gómez, R., León-Jiménez, J. E., López Rivera, R. & Perera Cruz, L. A. (2018), 'Comparison of bootstrap confidence intervals using monte carlo simulations', *Water* **10**(2).
URL: *https://www.mdpi.com/2073-4441/10/2/166*

Lau, Gonzalez, N. (2020), *The Studentized Bootstrap*, chapter 18.4.

Pruim, R., Kaplan, D. T. & Horton, N. J. (2017), 'The mosaic package: Helping students to 'think with data' using r', *The R Journal* **9**(1), 77–102.
URL: *https://journal.r-project.org/archive/2017/RJ-2017-024/index.html*

Puth, M.-T., Neuhäuser, M. & Ruxton, G. D. (2015), 'On the variety of methods for calculating confidence intervals by bootstrapping', *Journal of Animal Ecology* **84**(4), 892–897.
URL: *https://besjournals.onlinelibrary.wiley.com/doi/abs/10.1111/1365-2656.12382*

Rousselet, G. A., Pernet, C. R. & Wilcox, R. R. (2021), 'The percentile bootstrap: A primer with step-by-step instructions in r', *Advances in Methods and Practices in Psychological Science* **4**(1), 2515245920911881.
URL: *https://doi.org/10.1177/2515245920911881*