BACS - HW4

109006241

Question 1) Let’s reexamine how to standardize data: subtract the mean of a vector from all its values, and divide this difference by the standard deviation to get a vector of standardized values.

a. Create a normal distribution (mean=940, sd=190) and standardize it (let’s call it rnorm_std)

```
norm = rnorm(5000, mean=940, sd=190)
rnorm_std = (norm - mean(norm)) / sd(norm)
```

i) What should we expect the mean and standard deviation of rnorm_std to be, and why?

```
mean(rnorm_std); sd(rnorm_std)
```

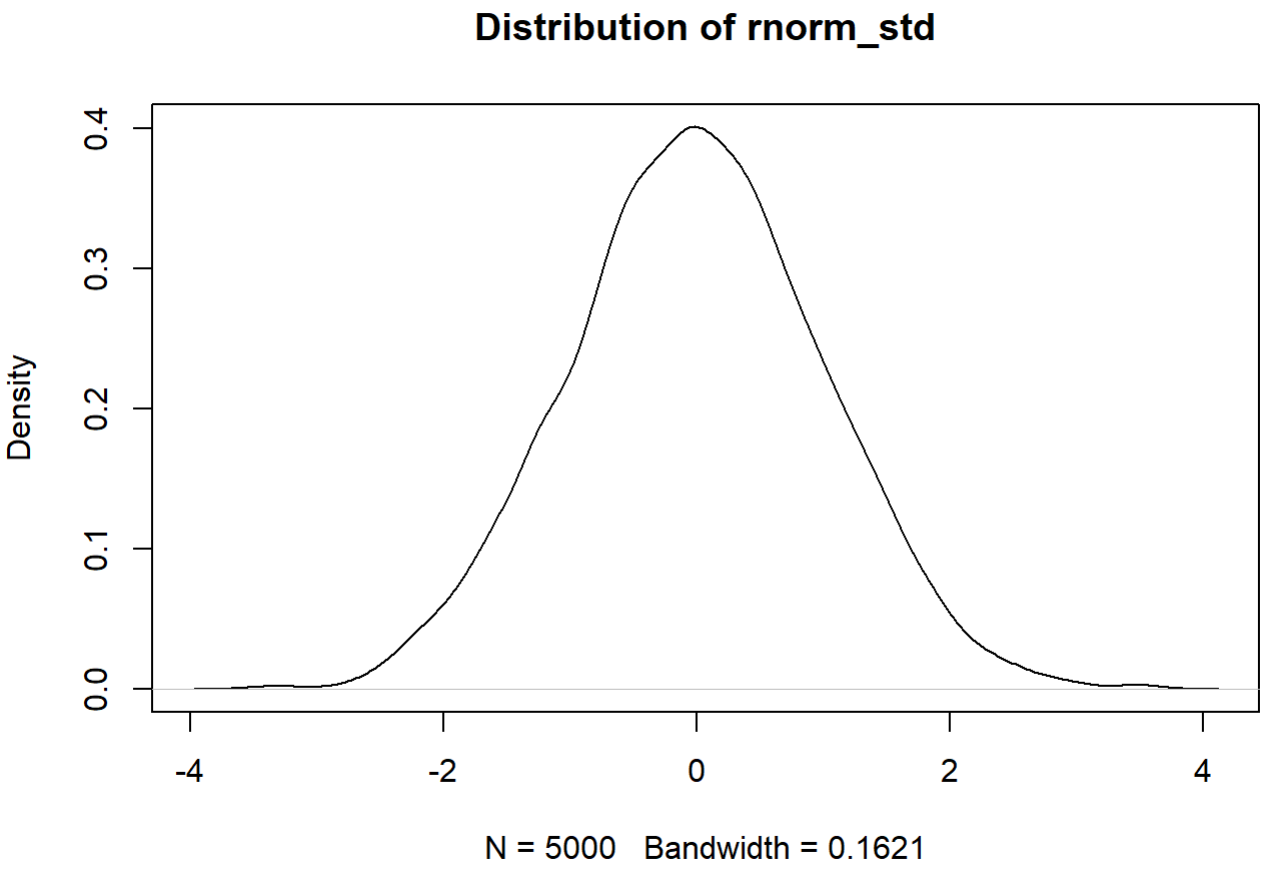
```
## [1] 1.127172e-16
```

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

> The mean will be 0 and the standard deviation will be 1. Because after standardizing, our original distribution will follow a normal distribution with mean 0 and sd 1 (standard normal distribution).

ii) What should the distribution (shape) of rnorm_std look like, and why?

```
plot(density(rnorm_std), main='Distribution of rnorm_std')
```



It will be a bell-shaped curve. Because rnorm_std follows a normal distribution.

iii) What do we generally call distributions that are normal and standardized?

Standard normal distribution.

b. Create a standardized version of minday discussed in question 3 (let's call it minday_std)

```
PATH = "D:/Users/User/Documents/R/BACS/Homeworks/HW4/first_bookings_datetime_sample.txt"
bookings <- read.table(PATH, header=TRUE)
hours  <- as.POSIXlt(bookings$datetime, format="%m/%d/%Y %H:%M")$hour
mins   <- as.POSIXlt(bookings$datetime, format="%m/%d/%Y %H:%M")$min
minday <- hours*60 + mins

minday_std = (minday - mean(minday)) / sd(minday)
```

i) What should we expect the mean and standard deviation of minday_std to be, and why?

```
mean(minday_std); sd(minday_std)
```

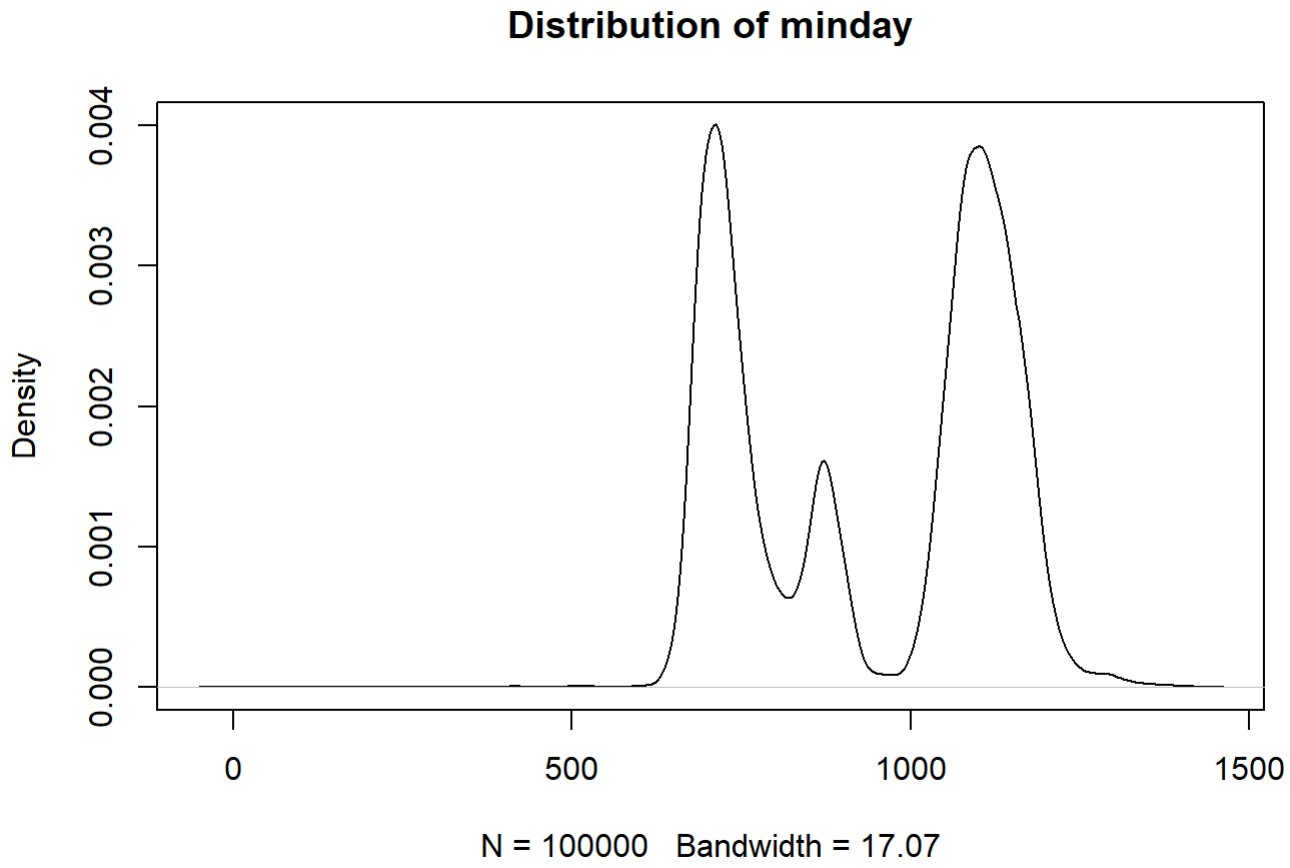
```
## [1] -4.25589e-17
```

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

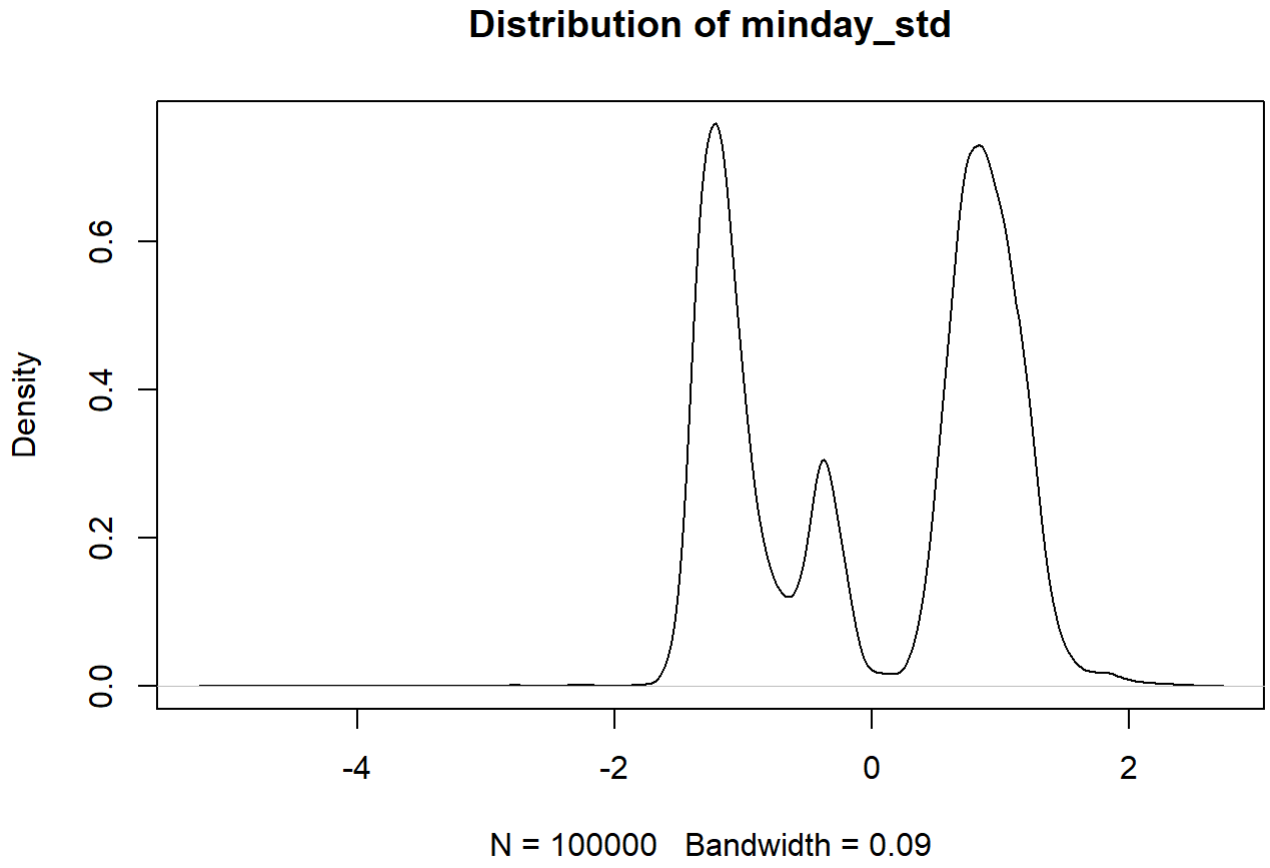
> The mean will be 0 and the standard deviation will be 1. Because after standardizing, the mean and sd will be 0 and 1 respectively.

ii) What should the distribution of minday_std look like compared to minday, and why?

```
plot(density(minday), main='Distribution of minday')
```




```
plot(density(minday_std), main='Distribution of minday_std')
```



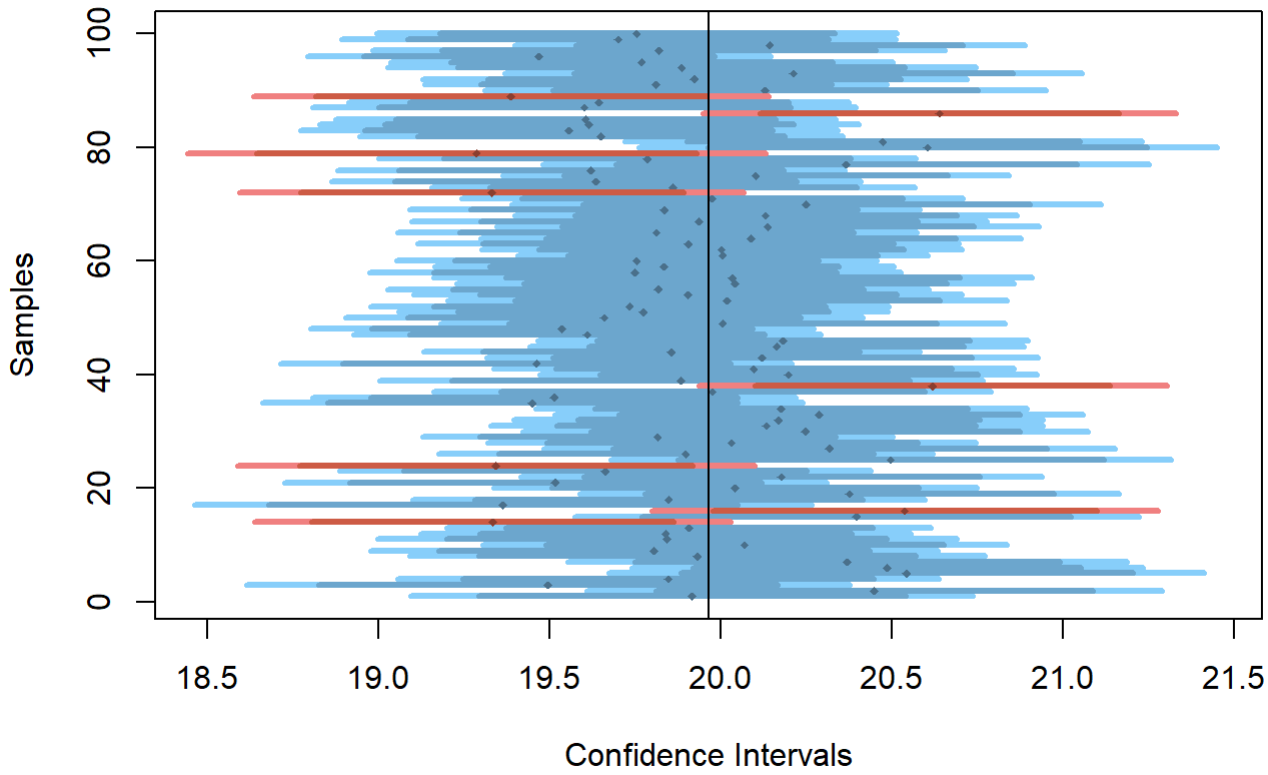
Those two distributions have the same shape. Because when standardizing, we are only scaling the values in our dataset such that its mean will be 0 and sd will be 1, without changing the shape of the distribution of our data.

Question 2) Install the compstatslib package from Github (see class notes) and run the plot_sample_ci() function that simulates samples drawn randomly from a population. Each sample is a horizontal line with a dark band for its 95% CI, and a lighter band for its 99% CI, and a dot for its mean. The population mean is a vertical black line. Samples whose

95% CI includes the population mean are blue, and others are red.

a. Simulate 100 samples (each of size 100), from a normally distributed population of 10,000:

```
plot_sample_ci(num_samples = 100, sample_size = 100, pop_size=10000, distr_func=rnorm, mean=20, sd=3)
```



i) How many samples do we expect to NOT include the population mean in its 95% CI?

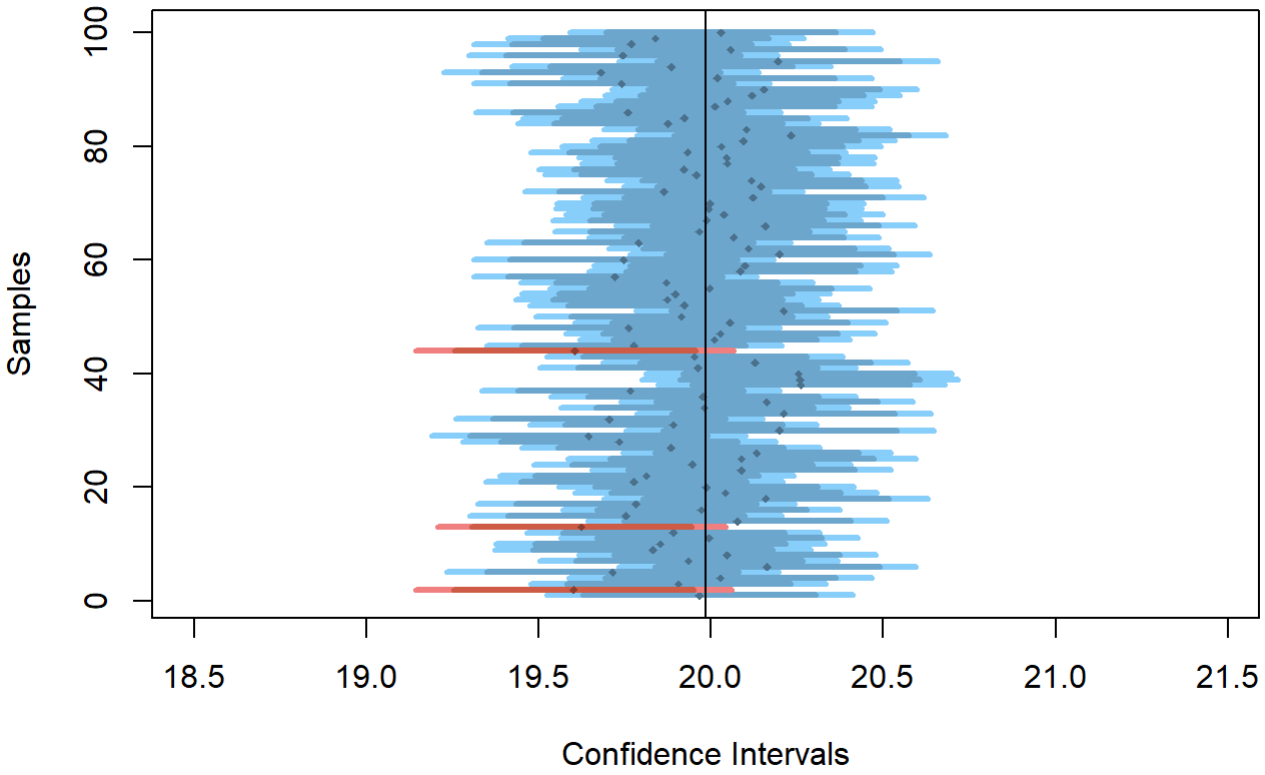
> According to theory, we expect 0.05*100 = 5 samples that doesn't include the population mean in its 95% CI. But, according to practice, it ranges from around 3-7 samples (Sampled from 10 simulations).

ii) How many samples do we expect to NOT include the population mean in their 99% CI?

> According to theory, we expect 0.01*100 = 1 samples that doesn't include the population mean in its 95% CI. But, according to practice, it ranges from around 0-3 samples (Sampled from 10 simulations).

b. Rerun the previous simulation with the same number of samples, but larger sample size (sample_size=300):

```
plot_sample_ci(num_samples = 100, sample_size = 300, pop_size=10000, distr_func=rnorm, mean=20, sd=3)
```



i) Now that the size of each sample has increased, do we expect their 95% and 99% CI to become wider or narrower than before?

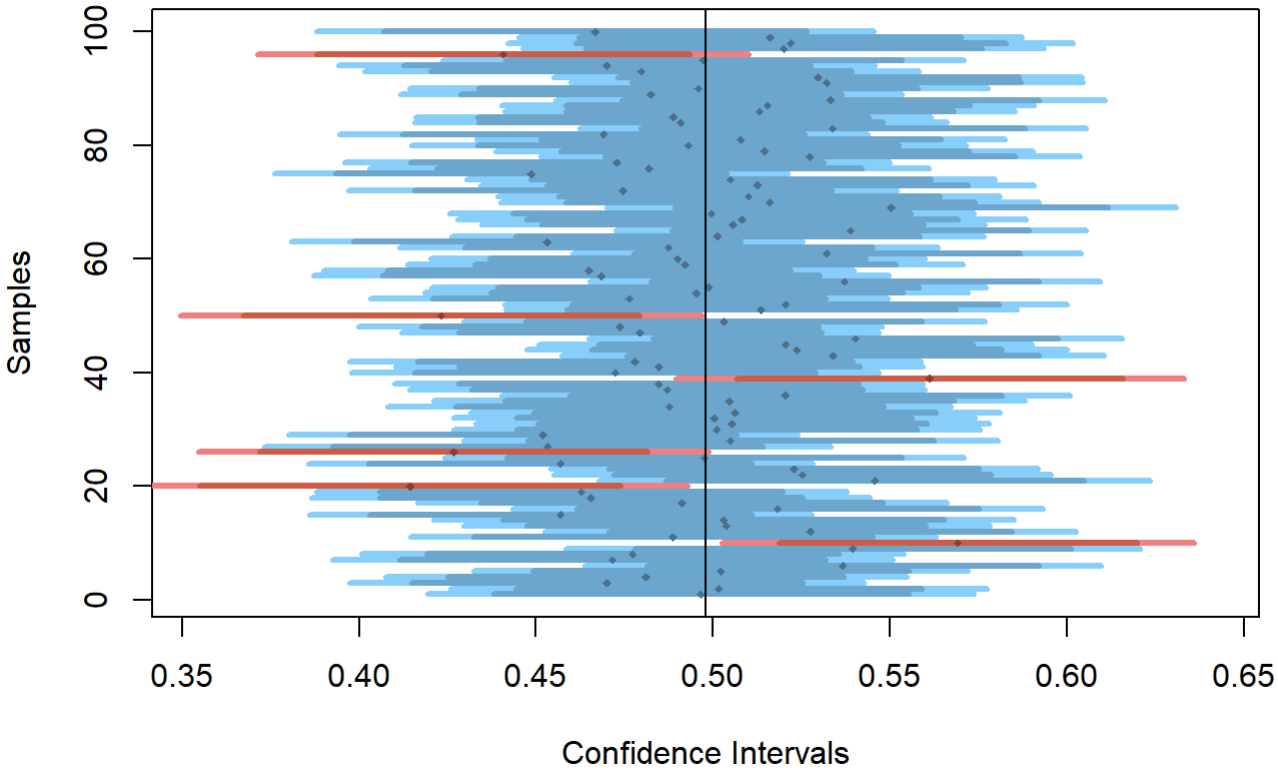
It becomes narrower.

ii) This time, how many samples (out of the 100) would we expect to NOT include the population mean in its 95% CI?

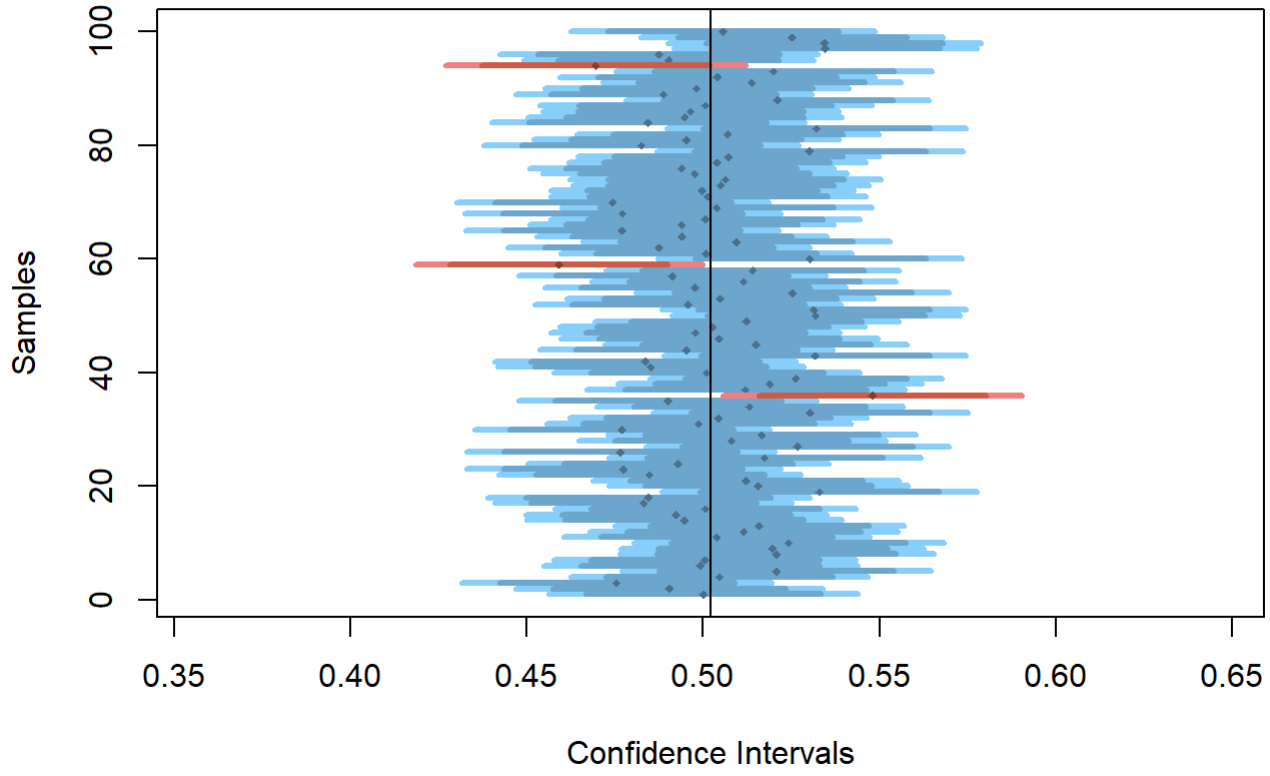
According to theory, we still expect 0.05*100 = 5 samples that doesn't include the population mean in its 95% CI. But, according to practice, it ranges from around 3-7 samples (Sampled from 10 simulations).

c. If we ran the above two examples (a and b) using a uniformly distributed population (specify parameter distr_func=runif for plot_sample_ci), how do you expect your answers to (a) and (b) to change, and why?

```
# a
plot_sample_ci(num_samples = 100, sample_size = 100, pop_size=10000, distr_func=runif)
```




```
# b
plot_sample_ci(num_samples = 100, sample_size = 300, pop_size=10000, distr_func=runif)
```



By changing from normal distribution to uniform distribution, we still expect the same answers as (a) and (b). It is still expected that 5 samples won't include the population mean in its' 95% CI, and 1 sample won't include the population mean in its' 99% CI. Changing into uniform distribution will only change our dataset values, thus changing the mean value from 20 into 0.5 instead.

Question 3) The company EZTABLE has an online restaurant reservation platform that is accessible by mobile and web. Imagine that EZTABLE would like to start a promotion for new members to make their bookings earlier in the day.

We have a sample of data about their new members, in particular the date and time for which they make their first ever booking (i.e., the booked time for the restaurant) using the EZTABLE platform.

a. What is the “average” booking time for new members making their first restaurant booking? (use minday, which is the absolute minute of the day from 0-1440)

i) Use traditional statistical methods to estimate the population mean of minday, its standard error, and the 95% confidence interval (CI) of the sampling means

```
n = length(minday)
minday_mean = sum(minday) / n
minday_sd = sqrt(sum((minday - mean(minday))^2) / (n - 1))
minday_stderror = minday_sd / sqrt(n)
minday_95ci_low = minday_mean - 1.96 * minday_stderror
minday_95ci_high = minday_mean + 1.96 * minday_stderror
cat(" Mean =", minday_mean, "\n",
     "Std. Error =", minday_stderror, "\n",
     "95% CI =", minday_95ci_low, "-", minday_95ci_high)
```

```
##  Mean = 942.4964
##  Std. Error = 0.5997673
##  95% CI = 941.3208 - 943.6719
```

ii) Bootstrap to produce 2000 new samples from the original sample

```
minday_sample = sample(minday, 10000)

minday_bootstrap = replicate(2000, sample(minday_sample, length(minday_sample), replace=TRUE))
```

iii) Visualize the means of the 2000 bootstrapped samples

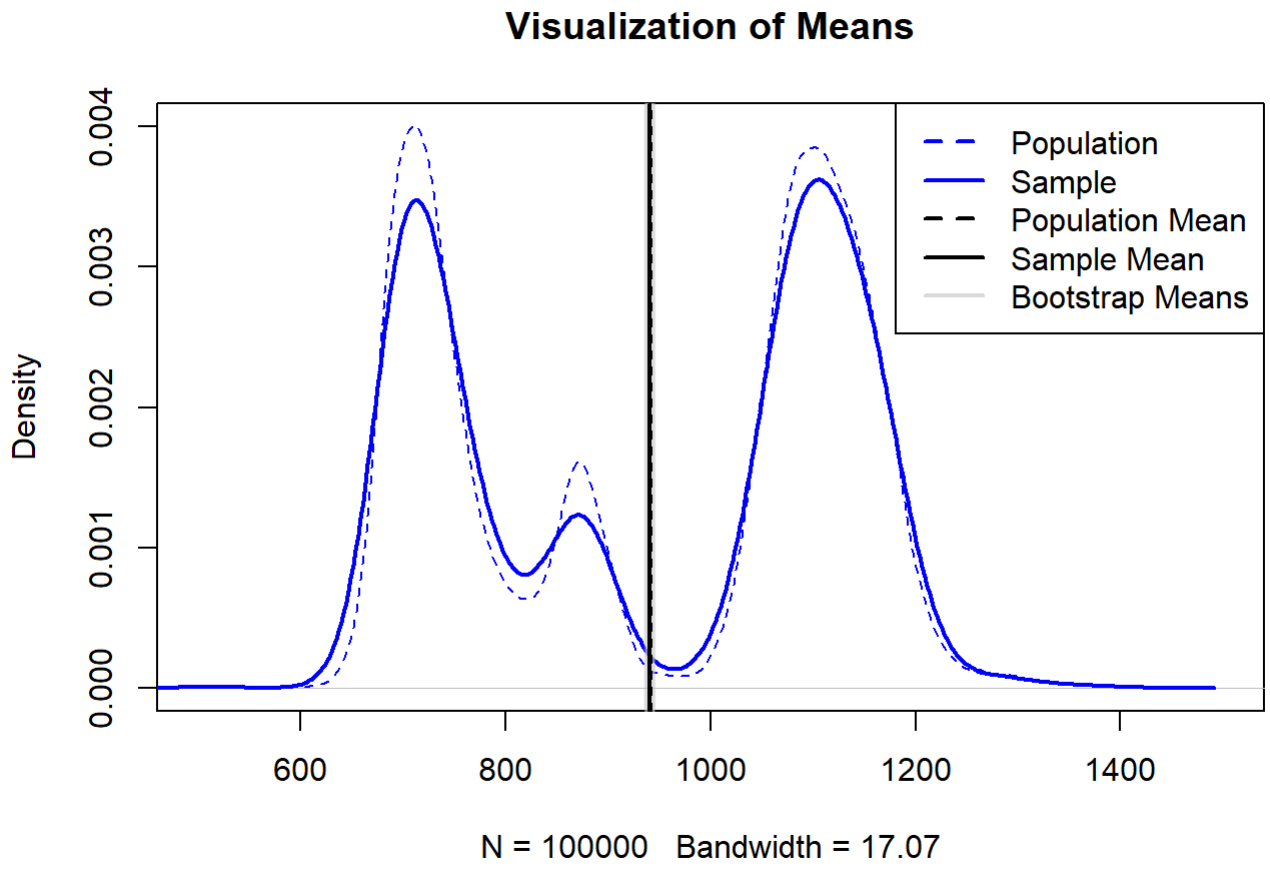
```
plot_bootstrap_density <- function(sample_i) {
  return(mean(sample_i))
}

sample_means = apply(minday_bootstrap, 2, FUN=plot_bootstrap_density)
```

```
plot(density(minday), col="blue", lty="dashed",
     main='Visualization of Means', xlim=c(500, 1500))
lines(density(minday_sample), col="blue", lwd=2)

abline(v=sample_means, col=rgb(0.7, 0.7, 0.7, 0.01))
abline(v=mean(sample_means), lwd=2)
abline(v=minday_mean, lty="dashed")

legend(x="topright",
       legend=c("Population", "Sample", "Population Mean", "Sample Mean", "Bootstrap Means"),
       lty=c(2, 1, 2, 1, 1),
       col=c("blue", "blue", "black", "black", rgb(0.7, 0.7, 0.7, 0.5)),
       lwd=2)
```



iv) Estimate the 95% CI of the bootstrapped means using the quantile function

```
boostrap_mean_95ci = quantile(sample_means, probs=c(0.025, 0.975))
cat("95% CI of the bootstrapped means =", boostrap_mean_95ci[1], "-", boostrap_mean_95ci[2])
```

```
## 95% CI of the bootstrapped means = 937.1221 - 944.4533
```

b. By what time of day, have half the new members of the day already arrived at their restaurant?

i) Estimate the median of minday

```
median(minday)
```

```
## [1] 1040
```

ii) Visualize the medians of the 2000 bootstrapped samples

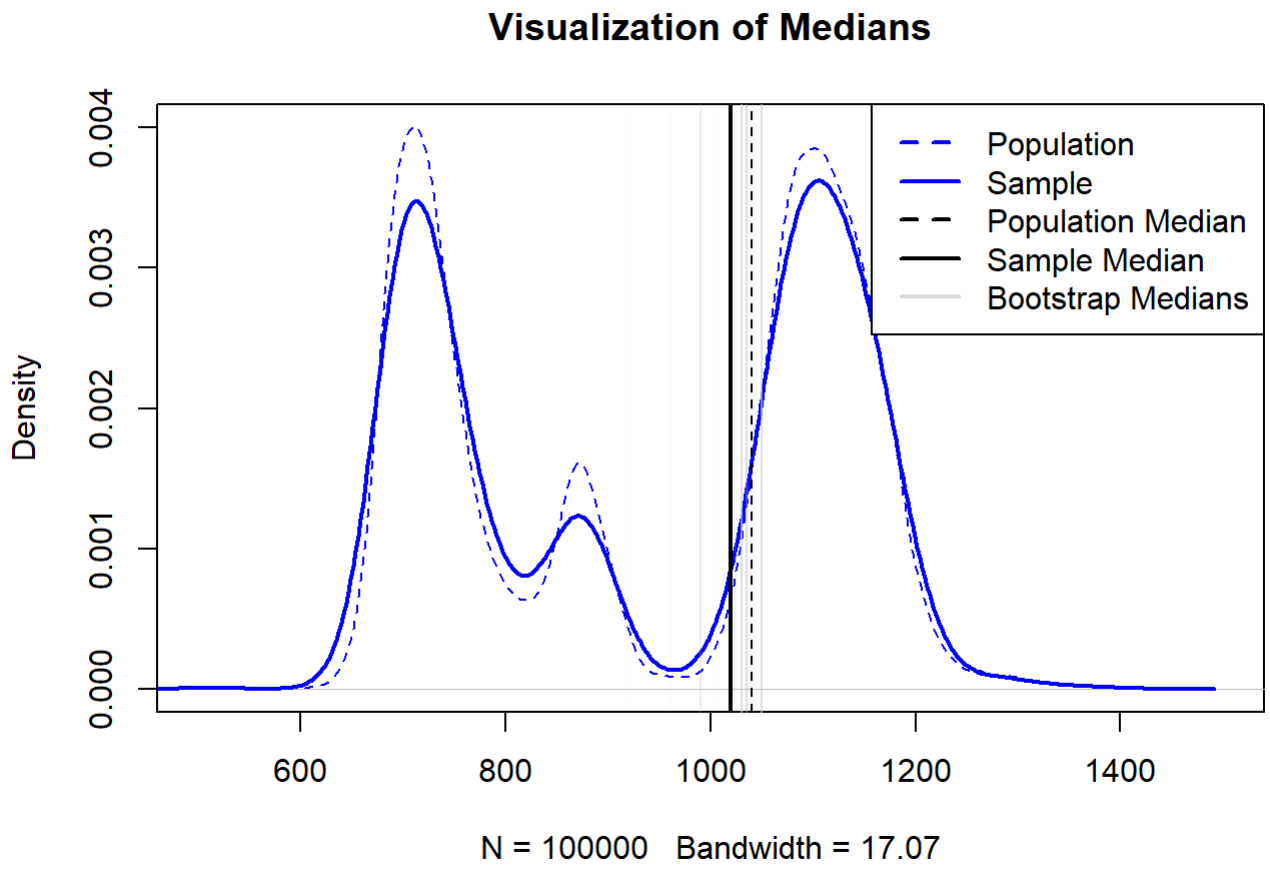
```
get_median <- function(sample_i) {
  return(median(sample_i))
}

sample_medians = apply(minday_bootstrap, 2, FUN=get_median)
```

```
plot(density(minday), col="blue", lty="dashed",
     main='Visualization of Medians', xlim=c(500, 1500))
lines(density(minday_sample), col="blue", lwd=2)

abline(v=sample_medians, col=rgb(0.7, 0.7, 0.7, 0.01))
abline(v=median(sample_medians), lwd=2)
abline(v=median(minday), lty="dashed")

legend(x="topright",
       legend=c("Population", "Sample", "Population Median", "Sample Median", "Bootstrap Median
s"),
       lty=c(2, 1, 2, 1, 1),
       col=c("blue", "blue", "black", "black", rgb(0.7, 0.7, 0.7, 0.5)),
       lwd=2)
```



iii) Estimate the 95% CI of the bootstrapped medians using the quantile function

```
boostrap_median_95ci = quantile(sample_medians, probs=c(0.025, 0.975))
cat("95% CI of the bootstrapped medians =", boostrap_median_95ci[1], "-", boostrap_median_95ci
[2])
```

```
## 95% CI of the bootstrapped medians = 1020 - 1050
```