BACS HW(WEEK 4)

109090046 Helped by 109090035

2023-03-11

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)

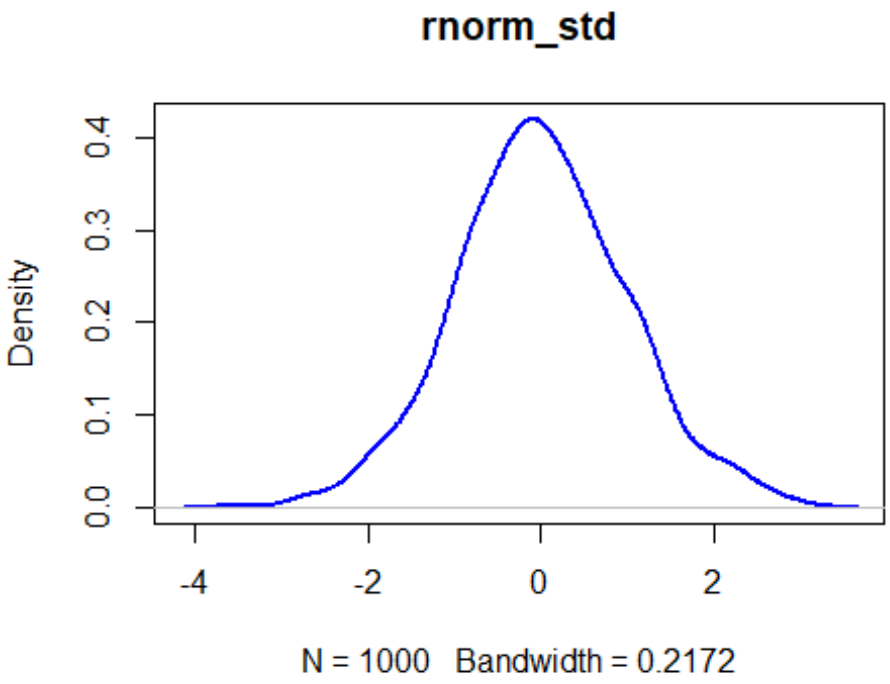
i) What should we expect the mean and standard deviation of rnorm_std to be, and why?

```
rnorm_std <- (rnorm(1000, mean=940, sd=190) - 940) / 190

mean(rnorm_std)

## [1] 0.007279167

sd(rnorm_std)

## [1] 0.9900744
```

- *The mean of standardized normal distribution should be close to zero.* Because the formula for standardization subtracts the mean of the original distribution from each value. Therefore, the mean is shifted to zero during the standardization process.
- *The standard deviation of standardized normal distribution should be close to one.* By definition, because the formula divides by the standard deviation. It is mean to change the standard deviation of the distribution to one.

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

```
plot(density(rnorm_std), lwd=2, col='blue', type="l", main='rnorm_std')
```



The distribution should be bell-shaped just like the plot above, which is similar to typical normal distribution.

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

A standard normal distribution. It is a type of normal distribution where the mean is 0 and the standard deviation is 1. It is also called the *Z-distribution* because it is used to calculate Z-scores, which are the number of standard deviations a given data point is from the mean of a distribution.

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

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

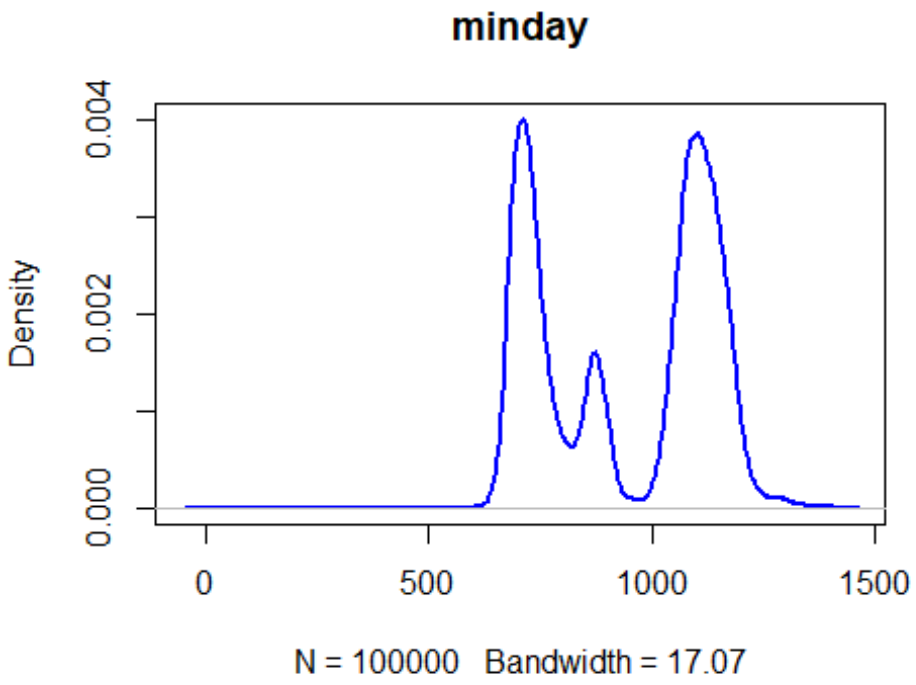
```
bookings <- read.table("D:/下載/first_bookings_datetime_sample.txt", he
ader=TRUE)
hours  <- as.POSIXlt(bookings$datetime, format="%m/%d/%Y %H:%M")$hour
mins   <- as.POSIXlt(bookings$datetime, format="%m/%d/%Y %H:%M")$min
minday <- hours*60 + mins
mean(minday)

## [1] 942.4964

sd(minday)
```

```
## [1] 189.6631

minday_std <- (minday - mean(minday)) / sd(minday)
mean(minday_std)

## [1] -4.25589e-17

sd(minday_std)

## [1] 1
```

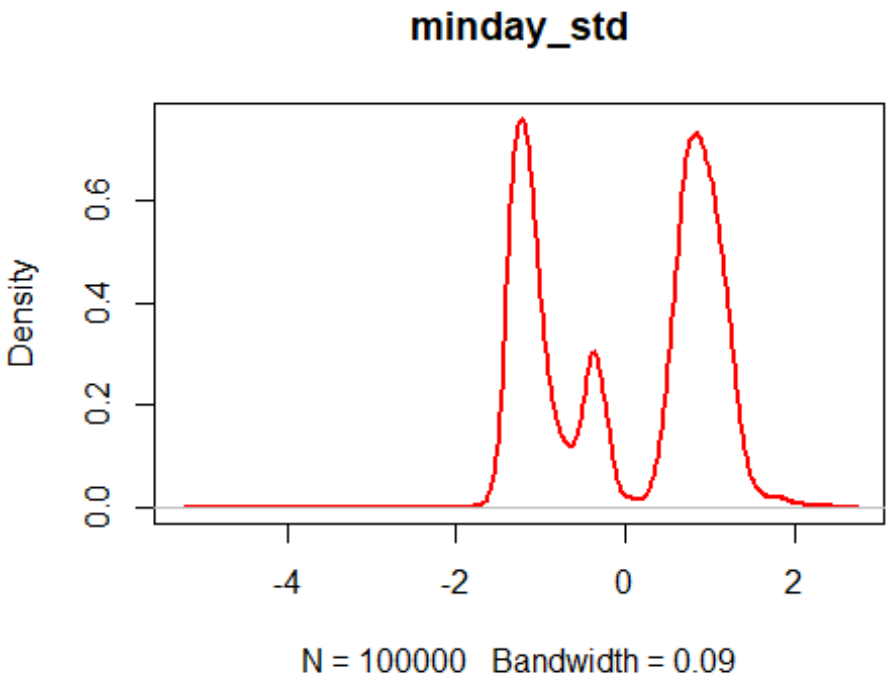
The mean should be close to zero and standard deviation should be close to one. Because the formula for standardization subtracts the mean of the original distribution from each value and then divides by the standard deviation. So, the mean is shifted to zero and standard deviation is changed to one.

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

```
plot(density(minday), type="l", lwd=2, main='minday', col='blue')
```




```
plot(density(minday_std), type="l", lwd=2, main='minday_std', col='red')
```



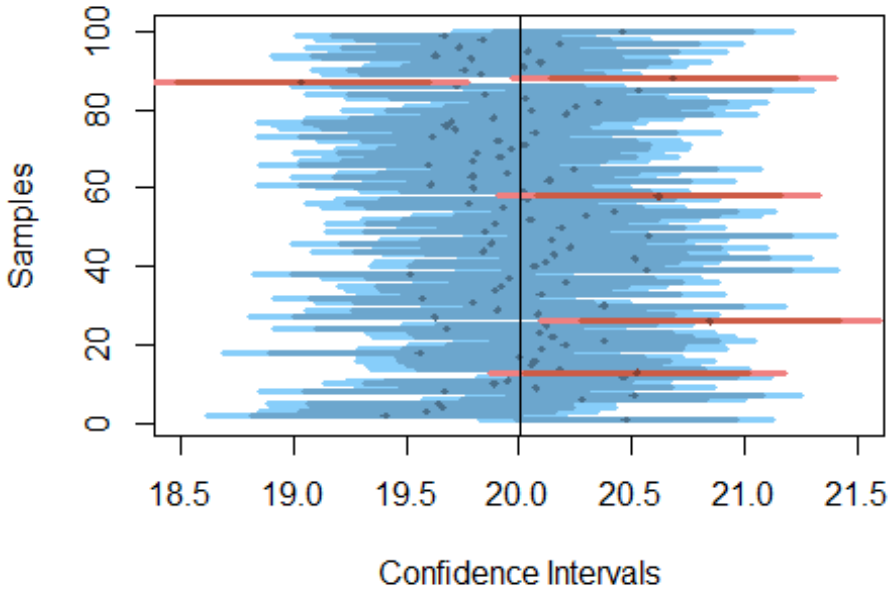
The shape of both distribution look exactly the same because standardization won't change the distribution. But we can see the scale of two distribution are different because of the differences between two datasets' mean and standard deviation. For `minday`, it is more widely spread and the x value's range can be very large, while `minday_std` only range from about -5 to 3.

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)`

```
library(compstatslib)
plot_sample_ci(num_samples = 100, sample_size = 100, pop_size=10000, di
str_func=rnorm, mean=20, sd=3)
```



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

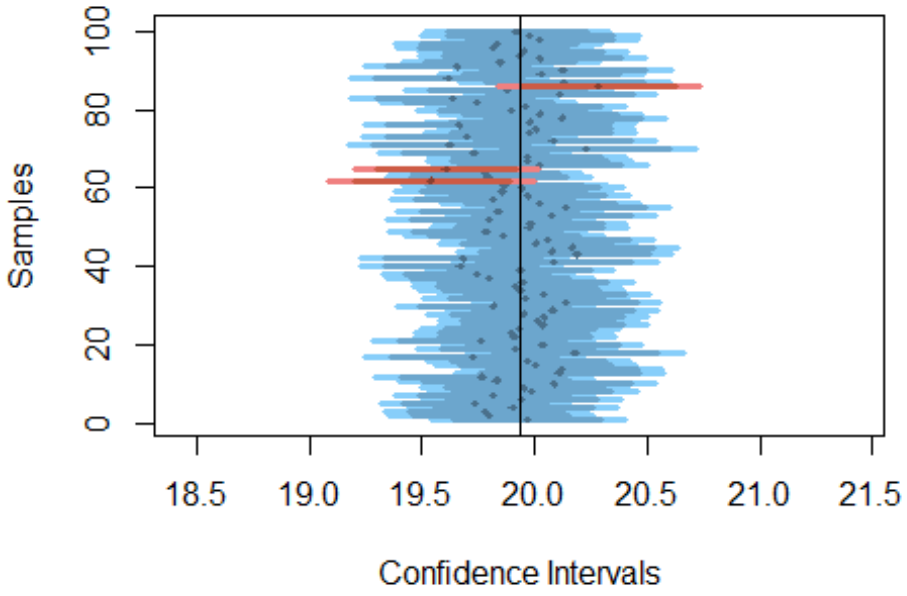
5% of number of samples which is 5.

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

1% of number of samples which is 1.

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

```
library(compstatslib)
plot_sample_ci(num_samples = 100, sample_size = 300, pop_size=10000, di
str_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?

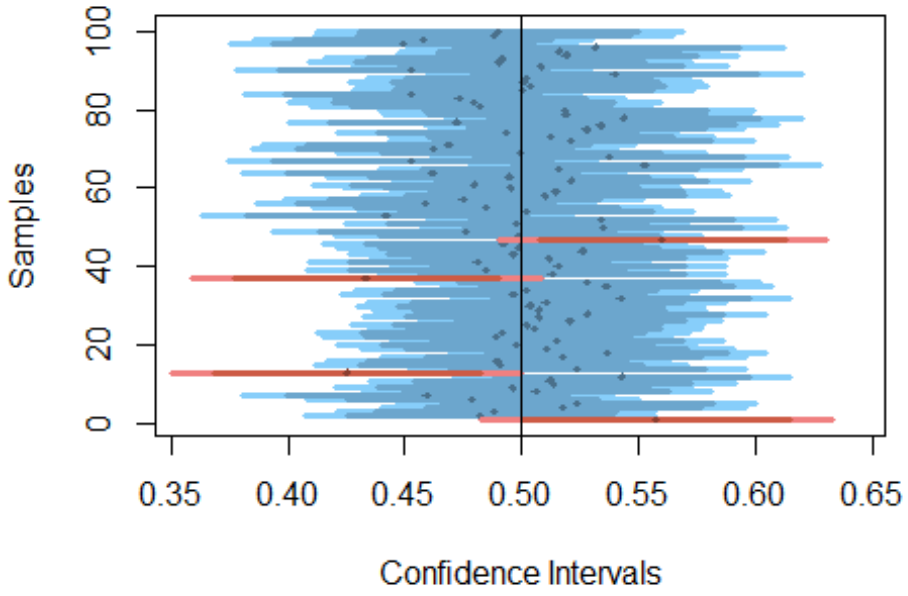
They becomes narrower than before.

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

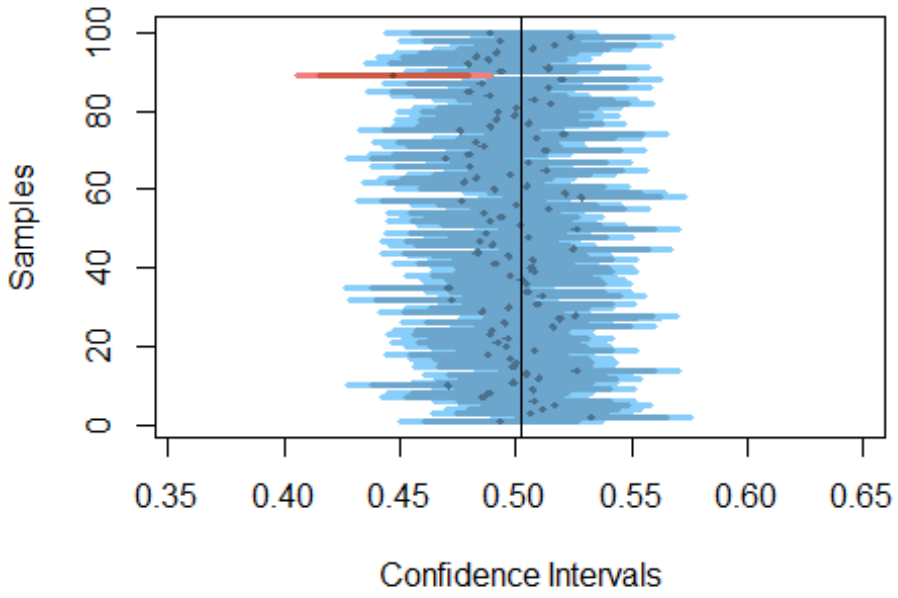
It should be less than the answer of (a) because of larger sample size.

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?

```
library(compstatslib)
plot_sample_ci(num_samples = 100, sample_size = 100, pop_size=10000, di
str_func=runif)
```




```
plot_sample_ci(num_samples = 100, sample_size = 300, pop_size=10000, di
str_func=runif)
```



The answers in (a) and (b) will be smaller with same sample size and confidence level because of following: The standard deviation of the population would be smaller than in the normal distribution cases, since the range of the uniform distribution is limited by its minimum and maximum values. As a result, the distribution of sample means would have a smaller standard error, leading to narrower confidence intervals for the same sample size and confidence level. Therefore, we would expect a smaller number of samples which NOT include the population mean in its 95% and 99% CI compared to the normal case.

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

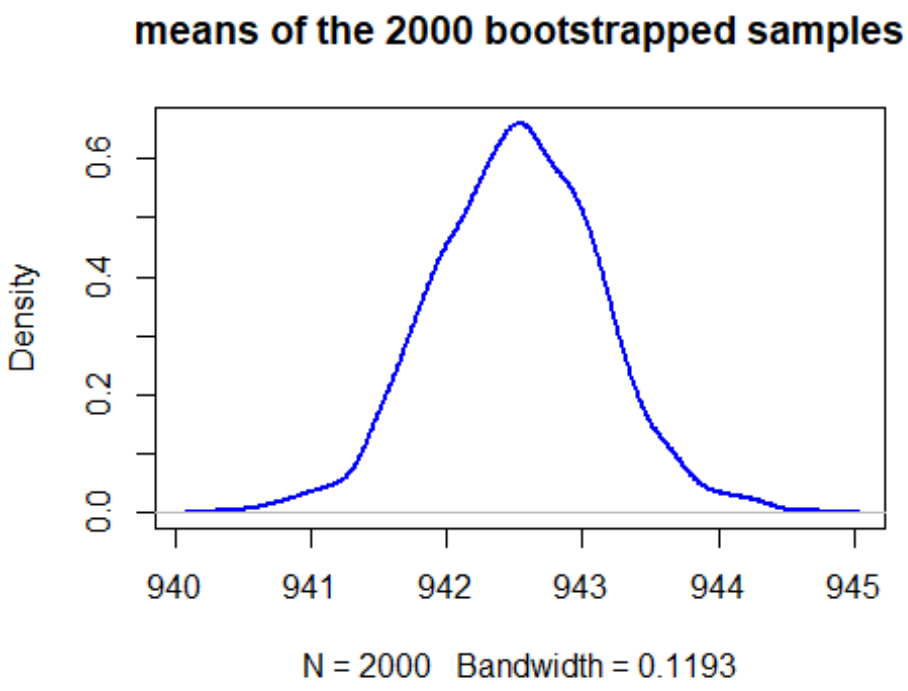
```
n <- length(minday)
pop_mean <- mean(minday)
se <- sd(minday)/ sqrt(n)

# critical value of 95% CI and n-1 degree of freedom
critical_val <- qt(0.975, df=n-1)
upper_bdd <- pop_mean + critical_val * se
lower_bdd <- pop_mean - critical_val * se

## Population mean: 942.4964

## Standard error: 0.5997673

## The 95% confidence interval (CI) of the sampling means: [ 941.3208 ,
 943.6719 ]
```

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

```
compute_sample_mean <- function(sample0) {
  resample <- sample(sample0, length(sample0), replace=TRUE)
  mean(resample)
}
means_2000 <- replicate(2000, compute_sample_mean(minday))
```

iii) Visualize the means of the 2000 bootstrapped samples

```
plot(density(means_2000), type="l", lwd=2, main='means of the 2000 boot
strapped samples', col='blue')
```



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

```
ci <- quantile(means_2000, c(0.025, 0.975))
ci

##    2.5%   97.5%
## 941.358 943.695

## the 95% CI of the bootstrapped means: [ 941.358 , 943.695 ]
```

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

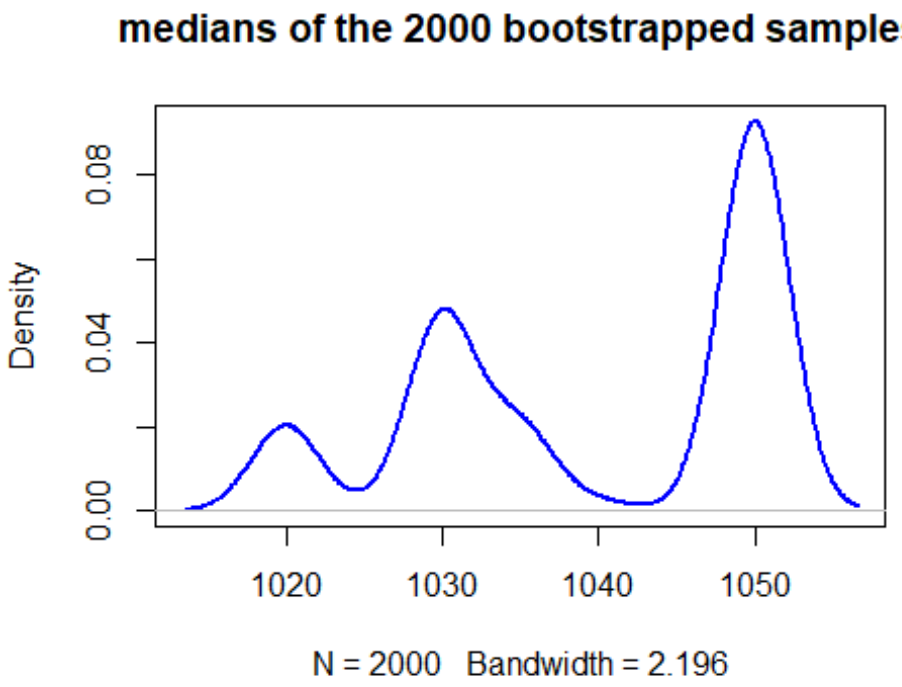
i) Estimate the median of `minday`

```
median(minday)

## [1] 1040
```

ii) Visualize the medians of the 2000 bootstrapped samples

```
compute_sample_median <- function(sample0) {
  resample <- sample(sample0, length(sample0), replace=TRUE)
  median(resample)
```

```
}
medians_2000 <- replicate(2000, compute_sample_median(minday))
plot(density(medians_2000), type="l", lwd=2, main='medians of the 2000
bootstrapped samples', col='blue')
```



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

```
ci_med <- quantile(medians_2000, c(0.025, 0.975))
ci_med

##  2.5% 97.5%
##  1020  1050

## the 95% CI of the bootstrapped medians: [ 1020 , 1050 ]
```