HW3

108048110

3/4/2022

Why it is important to standardize your data?

Ans. Data standardization is a process of making sure that your data set can be compared to other data sets by putting different variables on the same scale. It's about making sure that the data is internally consistent. (ie. each datatype has the same content and format.)

```
# declaring a function that can plot the destribution of the data and output its mean and std.
plt_mean_sd <- function(data, title){
  plot(density(data), main=title)
  abline(v=mean(data), lwd=2)
  abline(v=median(data), lty=2)
  cat(paste("Mean: ", round(mean(data), 3), "Std: ", sd(data)))
}
```

Question 1) Standardize data

a)

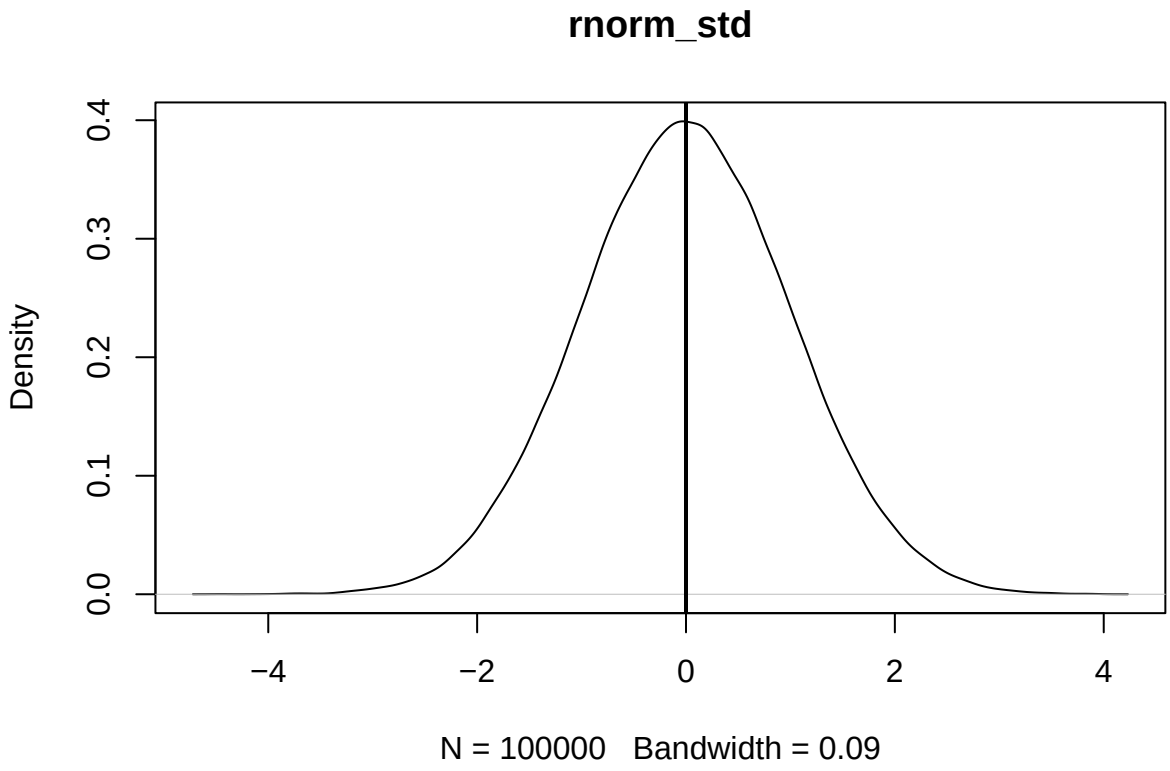
Create a normal distribution (mean=940, sd=190) and standardize it (let's call it rnorm_std)

```
rnorm_std <- rnorm(100000, mean=940, sd=190)

rnorm_std <- (rnorm_std-mean(rnorm_std))/sd(rnorm_std)
```

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

Ans. I would expect the mean of `rnorm_std` to be 0, and the standard deviation to be 1 because a standardized data set represents the number of standard deviations above or below the mean that a specific observation falls.

```
plt_mean_sd(rnorm_std, "rnorm_std")
```

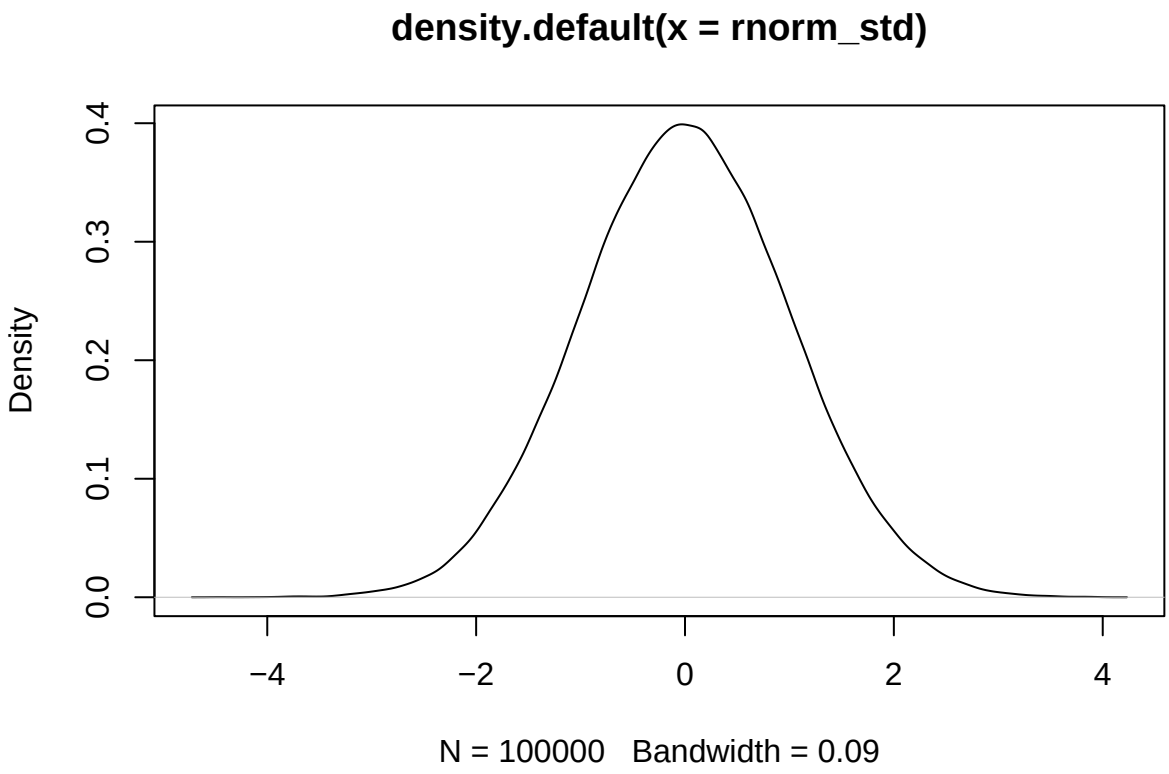



```
## Mean:  0 Std:  1
```

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

Ans. The distribution of `rand_data` should be normal. Since standardization will not change the distribution of your data, and the origianl data is normally distributed, thus I would expect the shape of `rnorm_std` to be bell-shaped.

```
plot(density(rnorm_std))
```



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

Ans. Standard normal distribution.

Prerequisite

```
bookings <- read.table("first_bookings_datetime_sample.txt", 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
```

(b)

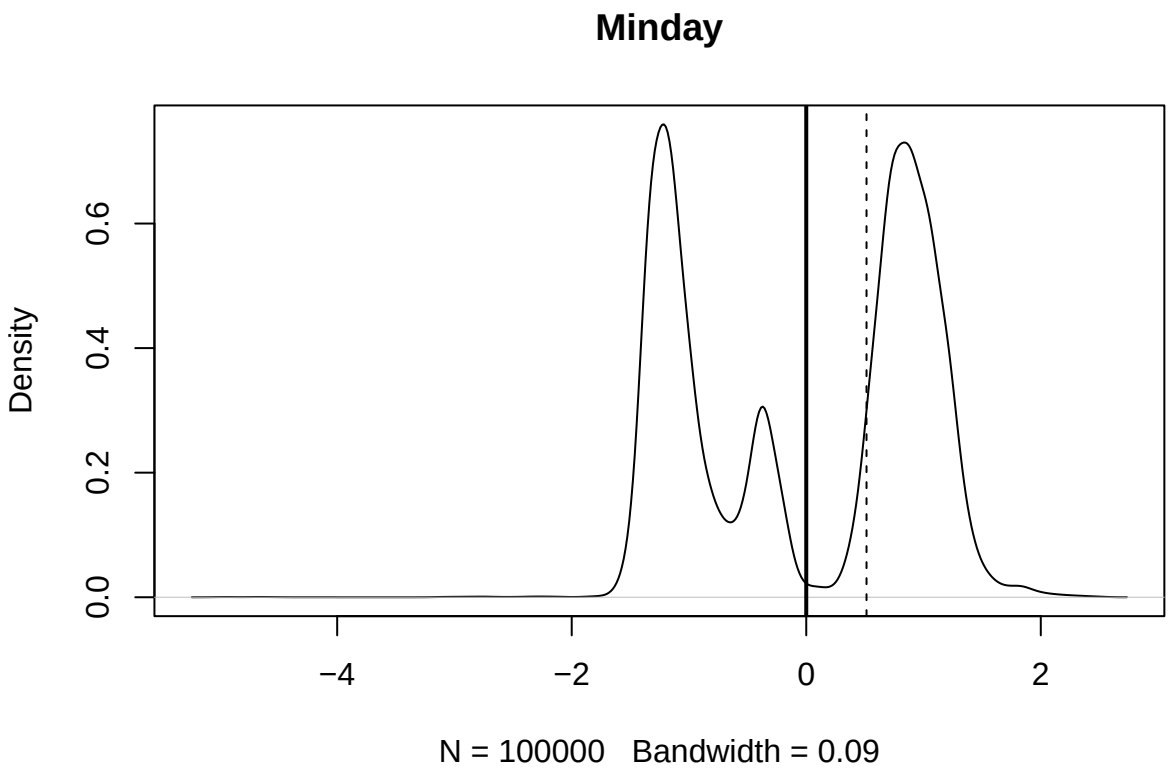
Create a standardized version of minday.

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

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

Ans. I would expect the mean of `minday` to be 0, and the standard deviation to be 1 because a standardized data set represents the number of standard deviations above or below the mean that a specific observation falls.

```
plt_mean_sd(minday_std, "Minday")
```




```
## Mean:  0 Std:  1
```

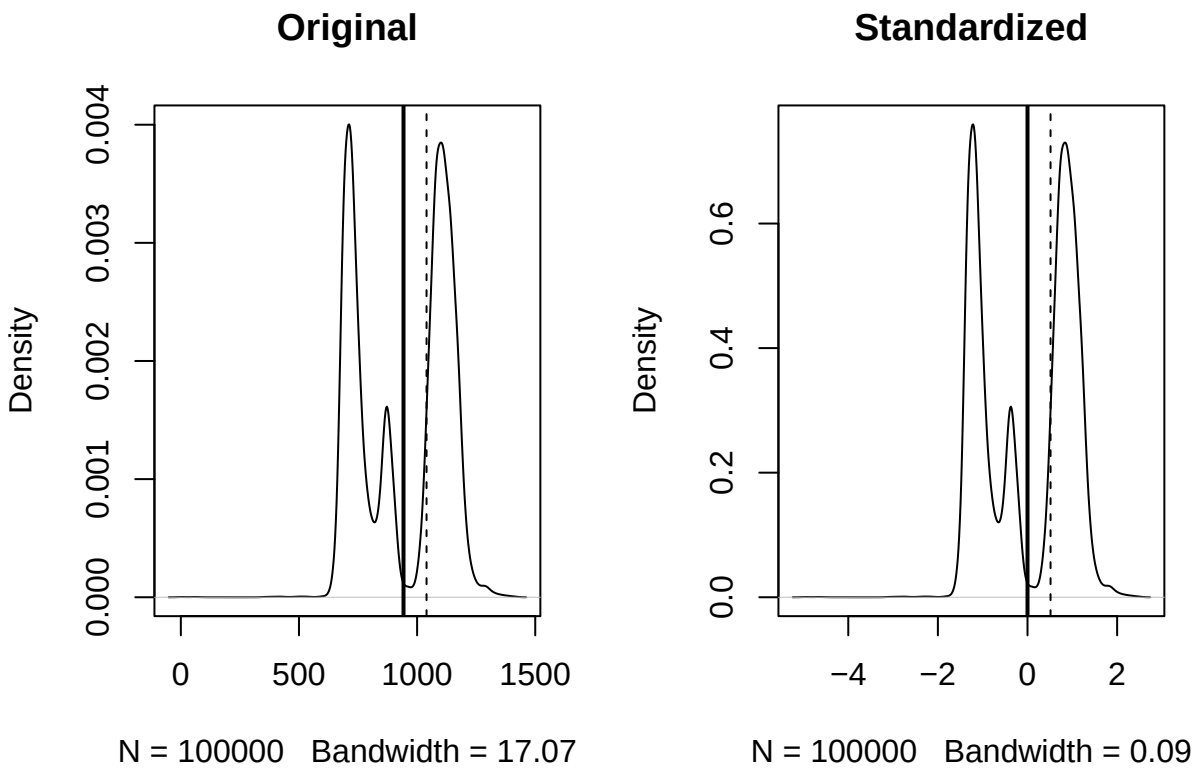
ii) What should the distribution of minday_std look like compared to minday, and why?

Ans. As I mentioned above, the distribution of `minday` should be the same as it's original distribution. Standardization will not change the distribution of data. However, `minday` has a much wider x range compared to `minday_std`.

```
# Here's the comparison
par(mfrow=c(1, 2))
plt_mean_sd(minday, title="Original")
```

```
## Mean:  942.496 Std:  189.663078134648
```

```
plt_mean_sd(minday_std, title="Standardized")
```




```
## Mean:  0 Std:  1
```

Note. two data have almost the same shape of distribution!

Question 2) Run `visualize_sample_ci`

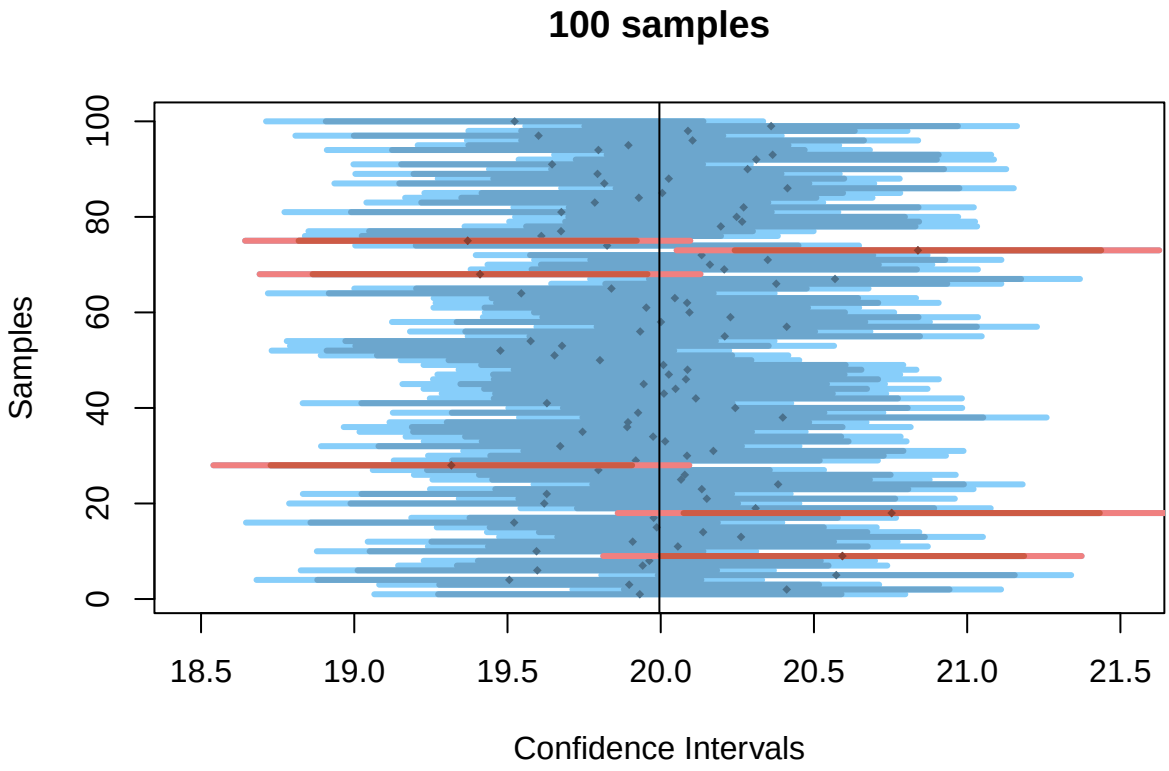
```r
add_ci_segment <- function(ci95_low, ci95_high, ci99_low, ci99_high,
                           sample_means, indices, good=TRUE) {
  segment_colors <- list(c("lightcoral", "coral3", "coral4"),
                         c("lightskyblue", "skyblue3", "skyblue4"))
  color <- segment_colors[[as.integer(good)+1]]

  segments(ci99_low, indices, ci99_high, indices, lwd=3, col=color[1])
  segments(ci95_low, indices, ci95_high, indices, lwd=3, col=color[2])
  points(sample_means, indices, pch=18, cex=0.6, col=color[3])
}

# Visualize the confidence intervals of samples drawn from a population
#   e.g.,
#     visualize_sample_ci(sample_size=300, distr_func=rnorm, mean=50, sd=10)
#     visualize_sample_ci(sample_size=300, distr_func=runif, min=17, max=35)
visualize_sample_ci <- function(num_samples = 100, sample_size = 100, title="density plot",
                                pop_size=10000, distr_func=rnorm, ...) {
```

```
  # Simulate a large population
  population_data <- distr_func(pop_size, ...)
  pop_mean <- mean(population_data)
  pop_sd <- sd(population_data)

  # Simulate samples
  samples <- replicate(num_samples,
                       sample(population_data, sample_size, replace=FALSE))

  # Calculate descriptives of samples
  sample_means = apply(samples, 2, FUN=mean)
  sample_stdevs = apply(samples, 2, FUN=sd)
  sample_stderrs <- sample_stdevs/sqrt(sample_size)
  ci95_low  <- sample_means - sample_stderrs*1.96
  ci95_high <- sample_means + sample_stderrs*1.96
  ci99_low  <- sample_means - sample_stderrs*2.58
  ci99_high <- sample_means + sample_stderrs*2.58

  # Visualize confidence intervals of all samples
  # Blank Corpus
  plot(NULL, xlim=c(pop_mean-(pop_sd/2), pop_mean+(pop_sd/2)), main=title,
       ylim=c(1,num_samples), ylab="Samples", xlab="Confidence Intervals")

  add_ci_segment(ci95_low, ci95_high, ci99_low, ci99_high,
                 sample_means, 1:num_samples, good=TRUE)

  # Visualize samples with CIs that don't include population mean
  bad = which(((ci95_low > pop_mean) | (ci95_high < pop_mean)) |
                ((ci99_low > pop_mean) | (ci99_high < pop_mean)))
  add_ci_segment(ci95_low[bad], ci95_high[bad], ci99_low[bad], ci99_high[bad],
                 sample_means[bad], bad, good=FALSE)

  # Draw true population mean
  abline(v=mean(population_data))
}
```

a)

Simulate 100 samples, from a normally distributed population of 10,000.

```
visualize_sample_ci(distr_func=rnorm, mean=20, sd=3, title="100 samples")
```



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

```
(1-0.95)*100
```

```
## [1] 5
```

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

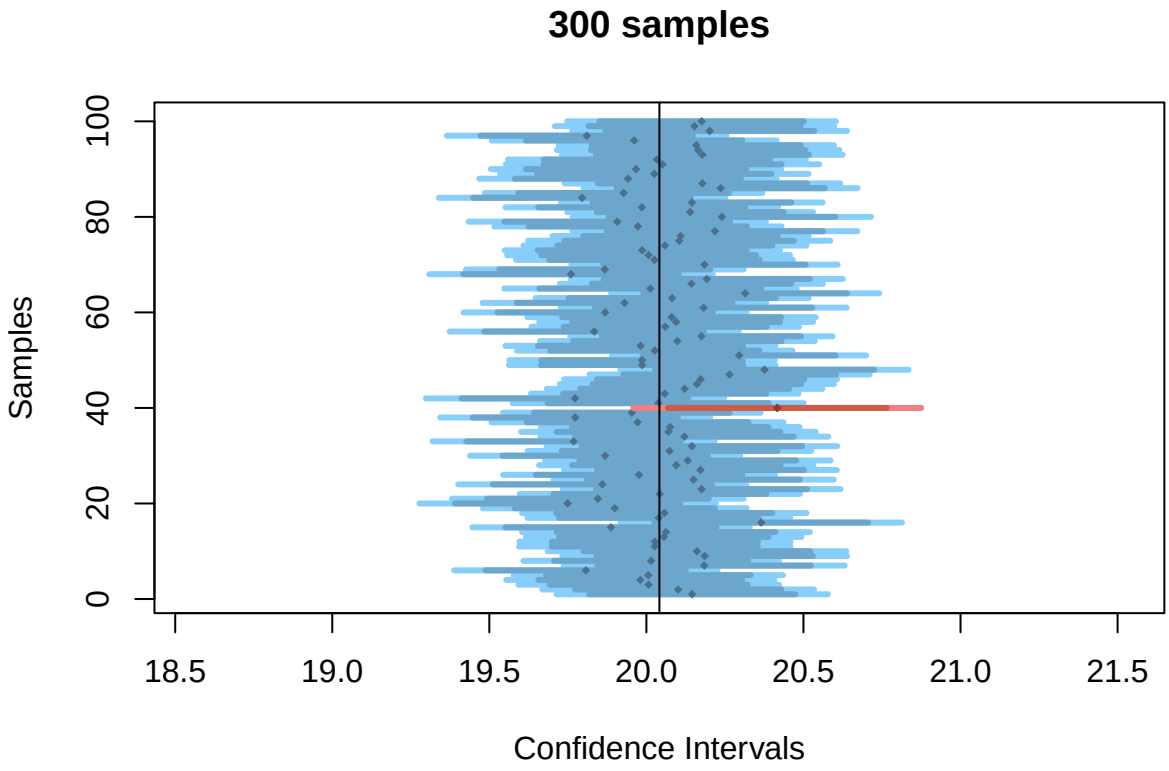
```
(1-0.99)*100
```

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

b)

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

```
visualize_sample_ci(sample_size = 300, distr_func=rnorm, mean=20, sd=3, title="300 samples")
```



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?

Ans. The larger the size of the sample the more information I can gain on the data; therefore, I can guess the population mean and sd more accurately, narrower the range of 95% and 99% of confidence interval.

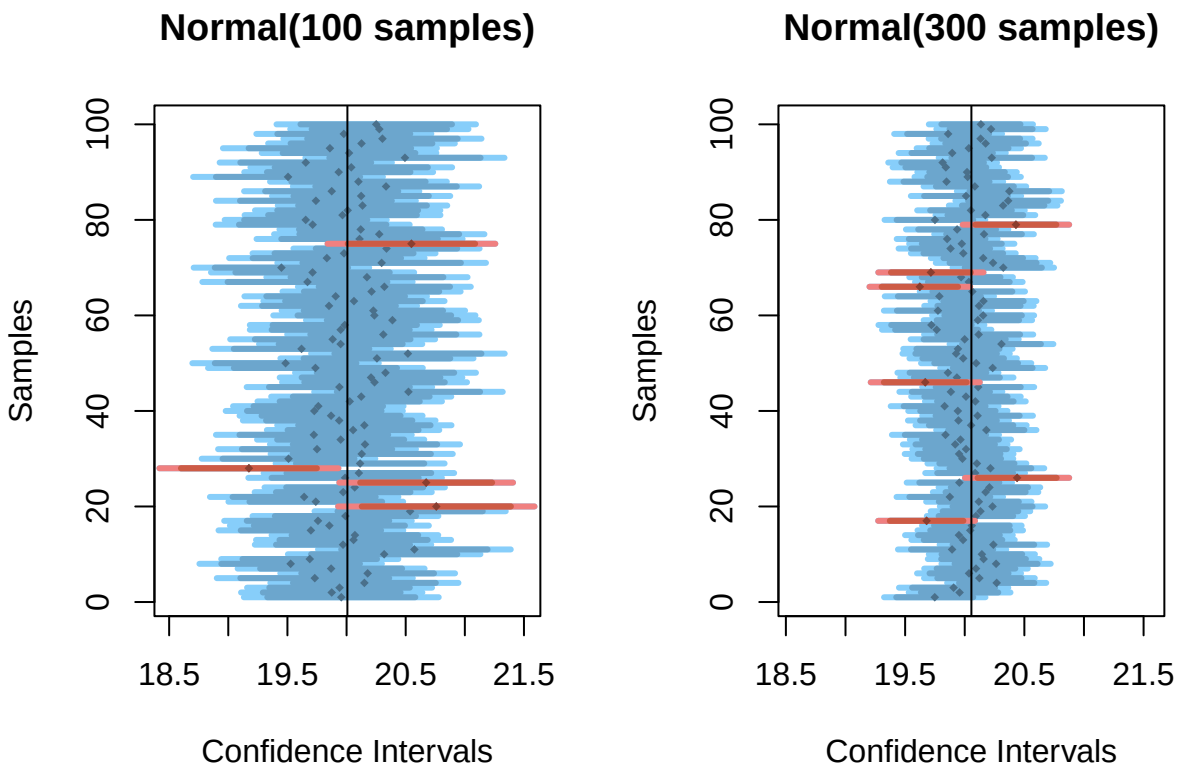
ii) This time, how many samples would we expect to NOT include the population mean in its 95% CI?

```
300*(1-0.95)
```

```
## [1] 15
```

Plots Obviously we can observe from the graph, as the sample size gets larger, we can estimate the population more clearly. The confidence interval does get narrower as the sample size increases.

```
# m*n array
par(mfrow=c(1,2))
visualize_sample_ci(distr_func=rnorm, mean=20, sd=3, title="Normal(100 samples)")
visualize_sample_ci(sample_size = 300, distr_func=rnorm, mean=20, sd=3, title="Normal(300 samples)")
```

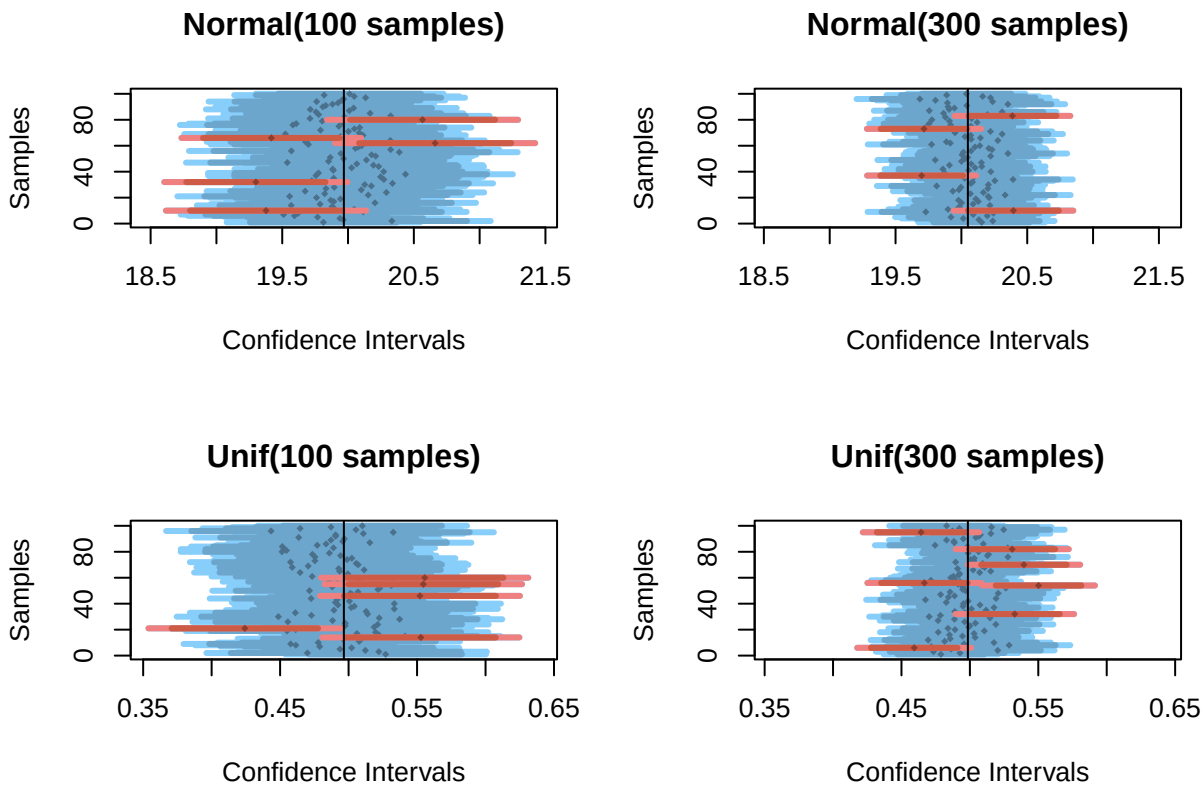


c)

If we ran the above two examples using a uniformly distributed population, how do you expect your answers to (a) and (b) to change, and why?

Ans. Based on central limit theorem, the distribution of sample means approximates a normal distribution as the sample size gets larger, regardless of the population's distribution. As the sample size increases, you can get more information about the population's parameters; in other words, you can estimate the unknown parameters (population means, sd, etc...) more accurately. Once you can evaluate the parameters more precisely, the range estimates for unknown parameters (CI) get narrower.

```
par(mfrow=c(2,2))

visualize_sample_ci(distr_func=rnorm, mean=20, sd=3, title="Normal(100 samples)")
visualize_sample_ci(sample_size = 300, distr_func=rnorm, mean=20, sd=3, title="Normal(300 samples)")
visualize_sample_ci(distr_func=runif, title="Unif(100 samples)")
visualize_sample_ci(sample_size = 300, distr_func=runif, title="Unif(300 samples)")
```



Question 3) Analysis a startup company EZTABLE's data.

a)

What is the "average" booking time for new members making their first restaurant booking?

```
sample_sd <- function(x){
  variance <- sum((x-mean(x))^2)/(length(x)-1)
  sqrt(variance)
}

property <- function(data, type="s", CI=0.95){
  if(CI==0.90) t <- c(-1.65, 1.65)
  if(CI==0.95) t <- c(-1.96, 1.96)
  if(CI==0.99) t <- c(-2.58, 2.58)

  Mean <- mean(data)
  if(type=="p"){
    Std <- sd(data)

    cat("Population")
  }else{
    Std = sqrt(sum((data-mean(data))^2)/(length(data)-1))
```

```
    Stderr <- Std/sqrt(length(data))
    ci <- (Mean+(t*Std)/sqrt(length(data)))

    cat("Sample", "\nStderr: ", Stderr, "\nCI range: ", ci)
  }
  cat(paste("\nMean: ",Mean , "\nStd: ", Std))
}

# population mean and sd
property(minday, "p")
```

```
## Population
## Mean:  942.49635
## Std:  189.663078134648
```

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.

Note. If we want to estimate a population mean from a sample, confidence interval will be a useful variable. The 95% confidence interval is: xbar +- 2*sd/sqrt(n)

```
resample <- sample(minday, 500, replace=TRUE)
property(resample)
```

```
## Sample
## Stderr:  8.476627
## CI range:  907.6458 940.8742
## Mean:  924.26
## Std:  189.543149734361
```

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

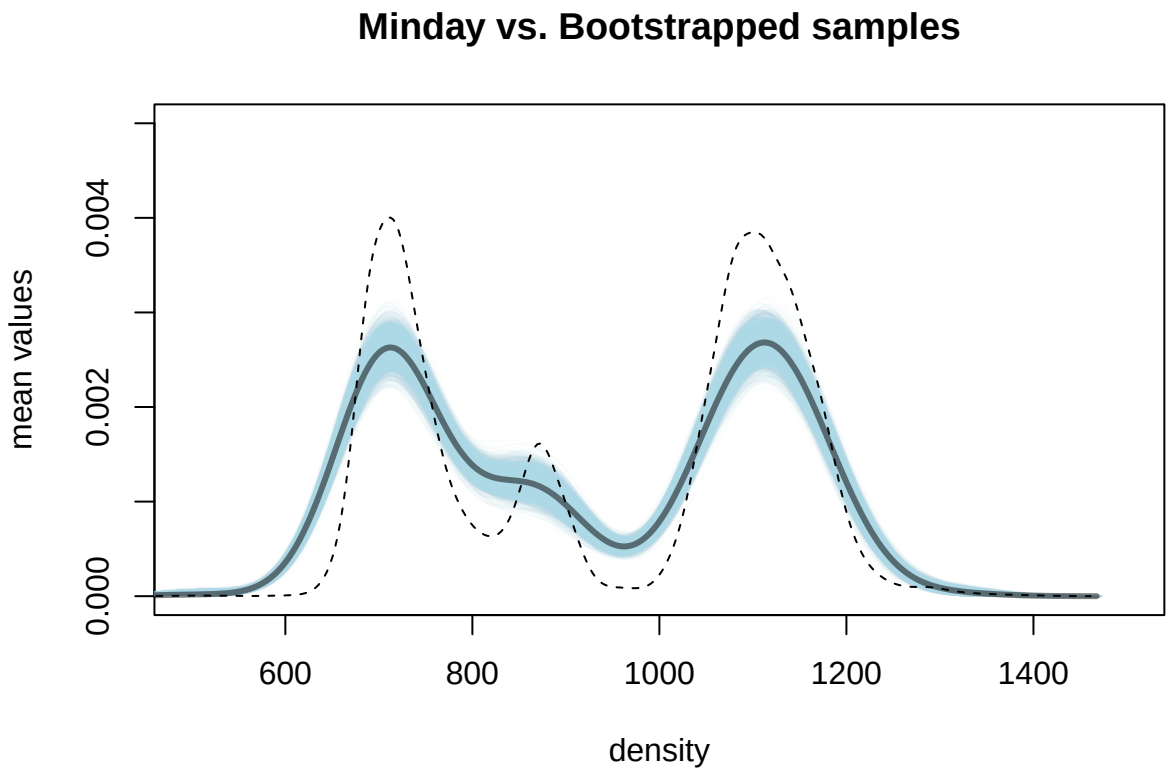
```
resamples <- replicate(2000, sample(resample, length(resample), replace=TRUE))

property(resamples)
```

```
## Sample
## Stderr:  0.1892202
## CI range:  924.1085 924.8502
## Mean:  924.479345
## Std:  189.220223014518
```

iii) Visualize the means of the 2000 bootstrapped samples

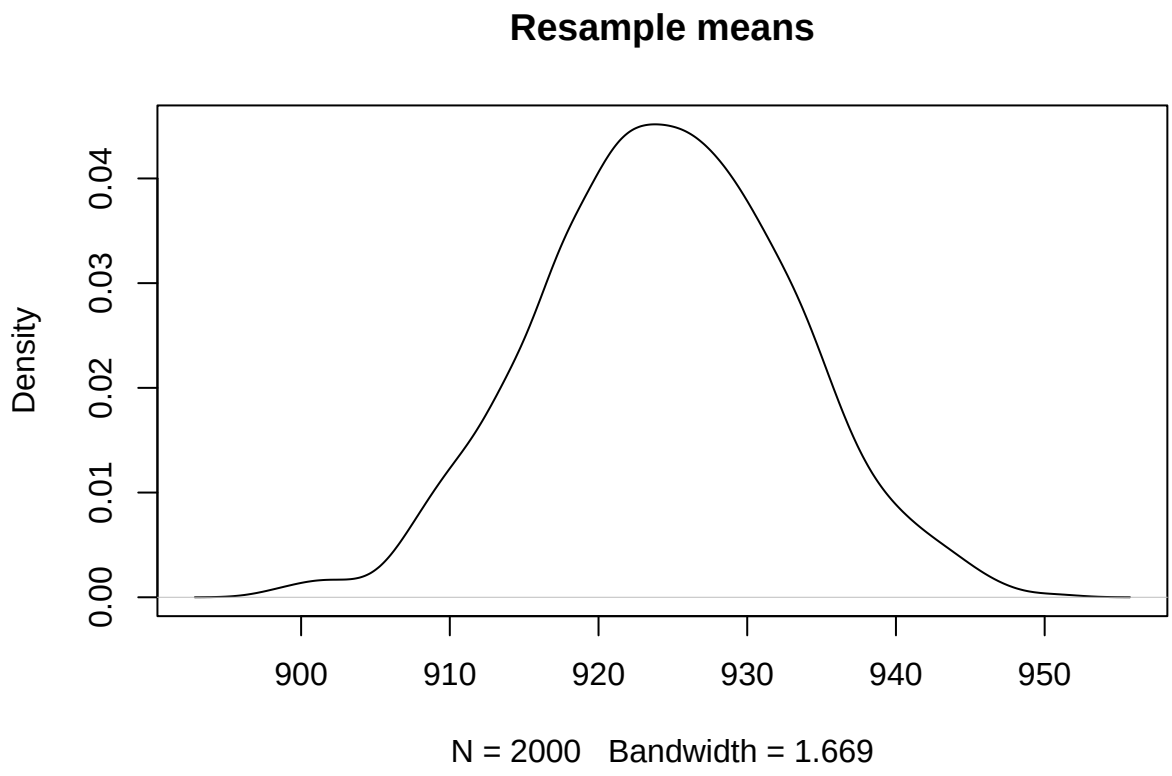
```
plot(NULL, lwd=0,xlim=c(500, 1500), ylim=c(0, 0.005), main="Minday vs. Bootstrapped samples", xlab="dens

plt_resample_mean <- function(data){
  lines(density(data), col=adjustcolor("lightblue", alpha.f = 0.05))
  return(mean(data))
}
plt_resample_median <- function(data){
```

```
  lines(density(data), col=adjustcolor("lightblue", alpha.f = 0.05))
  return(median(data))
}

resamples_mean <- apply(resamples, 2, FUN=plt_resample_mean)

lines(density(resample), lwd=3, col=adjustcolor("black", alpha.f = 0.5))
lines(density(minday), lty="dashed")
```




```
# underneath is a plot of hidden population and all sample distributions.
```

```
plot(density(resamples_mean), main="Resample means")
```



iv) Estimate the 95% CI of the bootstrapped means.

```
property(resamples_mean)
```

```
## Sample
## Stderr:  0.1895933
## CI range:  924.1077 924.8509
## Mean:  924.479345
## Std:  8.47887213946252
```

b)

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

```
unname(quantile(minday, 1/2))
```

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

i) Estimate the median of minday

```
median(minday)
```

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

ii) Visualize the medians of the 2000 bootstrapped samples.

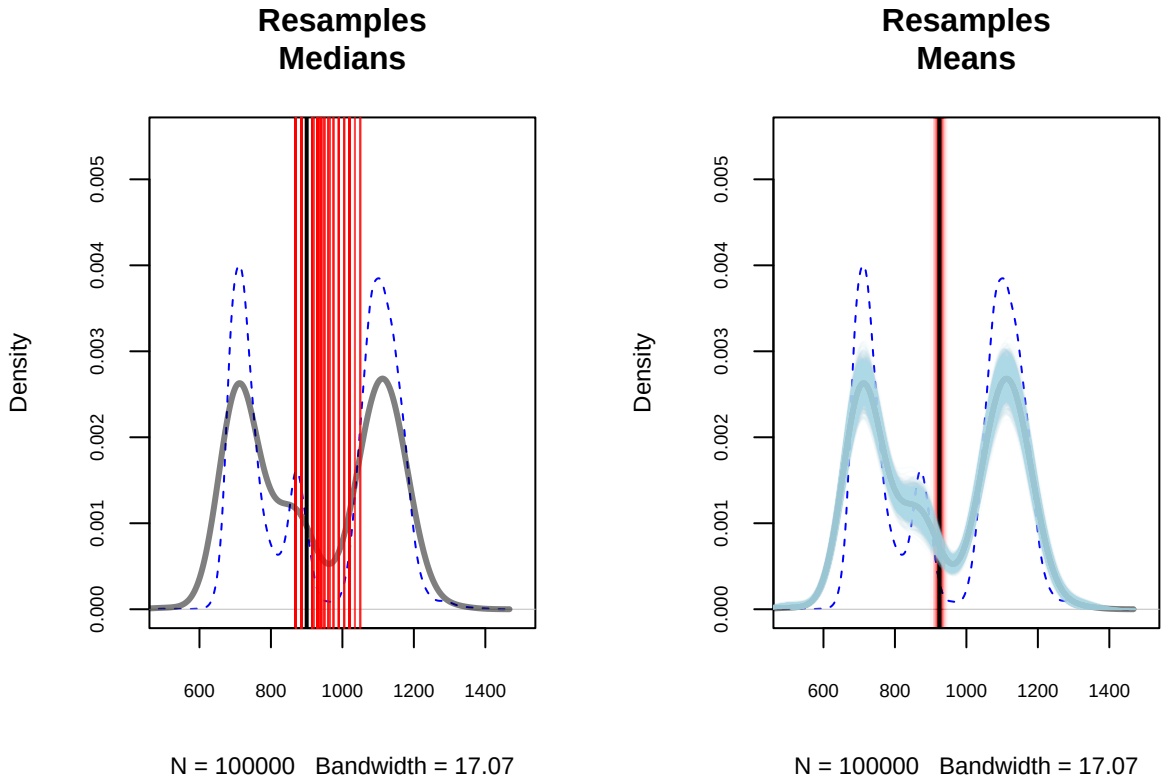
```
par(mfrow=c(1, 2))
plt_resample_statistics <- function(data, resamples, method){

  cat(method(data))
  if(mean(data)==method(data)) {
    title<-"Means"
    transparent=0.005}
  if(median(data)==method(data)){
    title<-"Medians"
    transparent=0.2}

  plot(density(data), xlim=c(500, 1500), ylim=c(0, 0.0055),
       main=c("Resamples", title),cex.lab=0.75, cex.axis=0.6, cex.main=1,
       col="blue", lty="dashed")

  lines(density(resample), lwd=3, col=adjustcolor("black", alpha.f = 0.5))

  plt_resampling <- function(data){
    abline(v = method(data), col=adjustcolor("red", alpha.f = transparent))
  }
  apply(resamples, 2, FUN=plt_resampling)
  abline(v = method(resamples), lwd=2)
}
plt_resample_statistics(minday, resamples, median)
```

```
## 1040
```

```
plt_resample_statistics(minday, resamples, mean)
```

```
## 942.4964
```

```
resamples_median <- apply(resamples, 2, FUN=plt_resample_median)
```



iii) Estimate the 95% CI of the bootstrapped medians.

```
property(resamples_median)
```

```
## Sample
## Stderr:  0.79071
## CI range:  895.8752 898.9748
## Mean:  897.425
## Std:  35.3616260556362
```

Note. You can zoom in a little to see some red lines representing the confidence interval of the sampling means.