BACS HW4

Student ID: 107070014

Question 1) Let's reexamine what it means to *standardize* data. To *standardize* a vector, subtract the mean of the vector from all its values, and then divide them by the standard deviation.

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?

Answer:

We expect the mean is 0 and the standard deviation is 1.

Assume X is a normal distribution normal variable with mean E[X] and standard deviation sd(X). Assume Y is the standardized normal distribution for X, Y= $\frac{X-E[X]}{sd(X)}$.

1. For mean

E[Y] = E $[\frac{X-E[X]}{sd(X)}] = \frac{E[X]-E[X]}{sd(X)} = 0$

Since E[X] is not a random variable, E[E[X]] = E[X] and so does sd(X).

2. For standard deviation

The formula of standard deviation: sd[X] = $\sqrt{E[(X-E[X])^2]}$

sd[Y] = sd$[\frac{X-E[X]}{sd(X)}] = \sqrt{\mathrm{E}[(\frac{X-E[X]}{sd(X)} - \mathrm{E}[\frac{X-E[X]}{sd(X)}])^2]}$

$= \mathrm{E}[(\frac{X-E[X]}{sd(X)})^2 - 2(\frac{X-E[X]}{sd(X)})(\mathrm{E}\,[(\frac{X-E[X]}{sd(X)}]) + (\mathrm{E}[\frac{X-E[X]}{sd(X)}])^2]$

$= \mathrm{E}[(\frac{X-E[X]}{sd(X)})^2 - 2(\frac{X-E[X]}{sd(X)})0 + 0]$

$= \frac{\sqrt{E[(X-E[X])^2]}}{sd[X]} = \frac{sd[X]}{sd[X]} = 1$

Code:

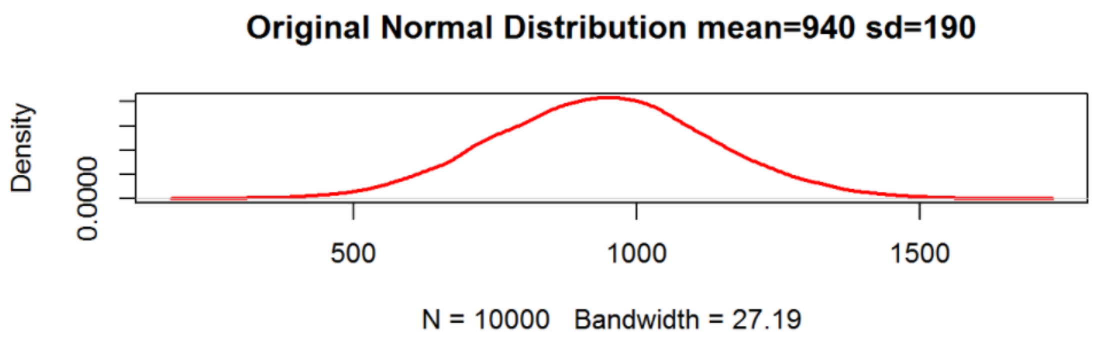
```
> norm <- rnorm(1000, mean = 940, sd = 190)
> mean_norm <- mean(norm)
> sd_norm <- sd(norm)
> rnorm_std <- (norm-mean_norm)/sd_norm
> mean(rnorm_std)
[1] -8.969832e-17
> sd(rnorm_std)
[1] 1
```

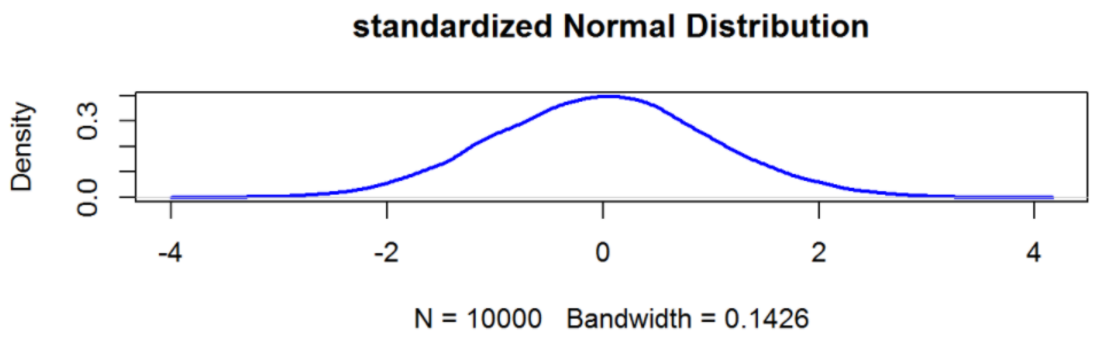
ii) What should the distribution (shape) of rnorm_std look like, and *why*?

Code:

```
par(mfrow = c(2,1))
plot(density(norm), lwd = 2, col = "red", main =
"Original Normal Distribution")
plot(density(rnorm_std), lwd = 2, col = "blue",
main = "Standardized Normal Distribution")
```

Results:





Answer:

The shape of **rnorm_std** should look like a bell, since it is a normal distribution.

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

Answer:

We call them “Standard Normal 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*?

Answer:

We expect the mean of **minday_std** to be 0 and standard deviation to be 1. The reason is similar to Question 1 (a)(i).

Code:

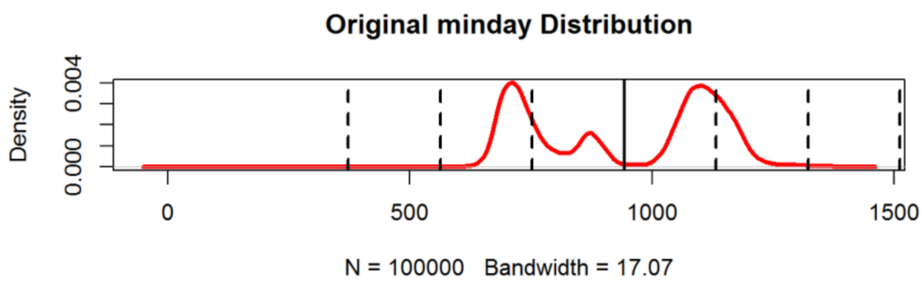
```
> bookings <- read.table("first_bookings_datetime_sample.txt", header = TRUE)
>
> hours <- as.POSIXlt(bookings$datetime, format="%m/%d/%Y %H:%M")$hour
> mins <- as.POSIXlt(bookings$datetime, format="%m/%d/%Y %H:%M")$min
> minday <- hours*60 + mins
> mean_minday <- mean(minday)
> sd_minday <- sd(minday)
> minday_std <- (minday-mean_minday)/sd_minday
> mean_minday_std <- mean(minday_std)
> mean_minday_std
[1] -4.25589e-17
> sd_minday_std <- sd(minday_std)
> sd_minday_std
[1] 1
```

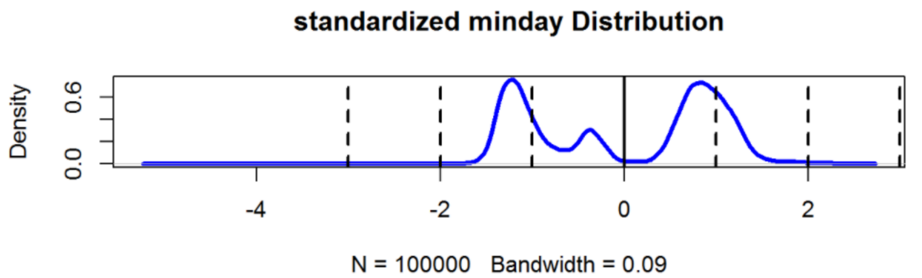
ii) What should the distribution of `minday_std` look like compared to `minday`, and *why*?

Code:

```
par(mfrow = c(2,1))
plot(density(minday),lwd=3,col="red", main="Original minday 
Distribution")
abline(v = mean_minday,lwd=2)
abline(v = mean_minday-sd_minday,lty=2,lwd=2)
abline(v = mean_minday-2*sd_minday,lty=2,lwd=2)
abline(v = mean_minday-3*sd_minday,lty=2,lwd=2)
abline(v = mean_minday+sd_minday,lty=2,lwd=2)
abline(v = mean_minday+2*sd_minday,lty=2,lwd=2)
abline(v = mean_minday+3*sd_minday,lty=2,lwd=2)

plot(density(minday_std),lwd=3, col="blue", main="standardized minday 
Distribution")
abline(v = mean_minday_std,lwd=2)
abline(v = mean_minday_std-sd_minday_std,lty=2,lwd=2)
abline(v = mean_minday_std-2*sd_minday_std,lty=2,lwd=2)
abline(v = mean_minday_std-3*sd_minday_std,lty=2,lwd=2)
abline(v = mean_minday_std+sd_minday_std,lty=2,lwd=2)
abline(v = mean_minday_std+2*sd_minday_std,lty=2,lwd=2)
abline(v = mean_minday_std+3*sd_minday_std,lty=2,lwd=2)
```

Results:





Question 2) Copy and run the code we used in class to create simulations of confidence intervals. Run *visualize_sample_ci()*, which 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:

```
visualize_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?

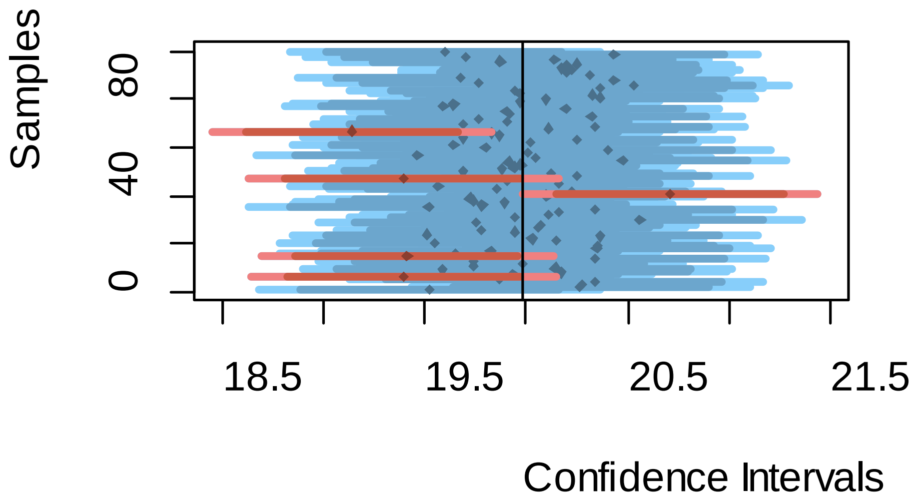
Code:

```
# 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,
                                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
  plot(NULL, xlim=c(pop_mean-(pop_sd/2), pop_mean+(pop_sd/2)),
       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))

  #number exclude ci95 & ci99
  exclude_ci95<-which((ci95_low > pop_mean) | (ci95_high < pop_mean))
  exclude_ci99<-which((ci99_low > pop_mean) | (ci99_high < pop_mean))
  #The return Value
  list(number_exclude_ci95 = length(exclude_ci95), number_exclude_ci99 =
length(exclude_ci99),width_of_ci95=mean(abs(ci95_high -
ci95_low)),width_of_ci99=mean(abs(ci99_high -ci99_low)))
  }

add_ci_segment <- function(ci95_low, ci95_high, ci99_low, ci99_high,
                           sample_means, indices, good=TRUE) {
  segment_colors <- list(c("lightcoral", "coral3", "coral4"),
```

```
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])
}
list[number_exclude_ci95,number_exclude_ci99,width_of_ci95,width_of_ci99]<-
visualize_sample_ci(num_samples = Num_sample, sample_size = Sample_size,
pop_size=10000, distr_func=rnorm, mean=20, sd=3)
```

Results:




```
> expected_number_exclude_ci95 <- Num_sample*0.05
> expected_number_exclude_ci95
[1] 5
> number_exclude_ci95
[1] 4
```

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

Results:

```
> expected_number_exclude_ci99 <- Num_sample*0.01
> expected_number_exclude_ci99
[1] 1
> number_exclude_ci99
[1] 1
```

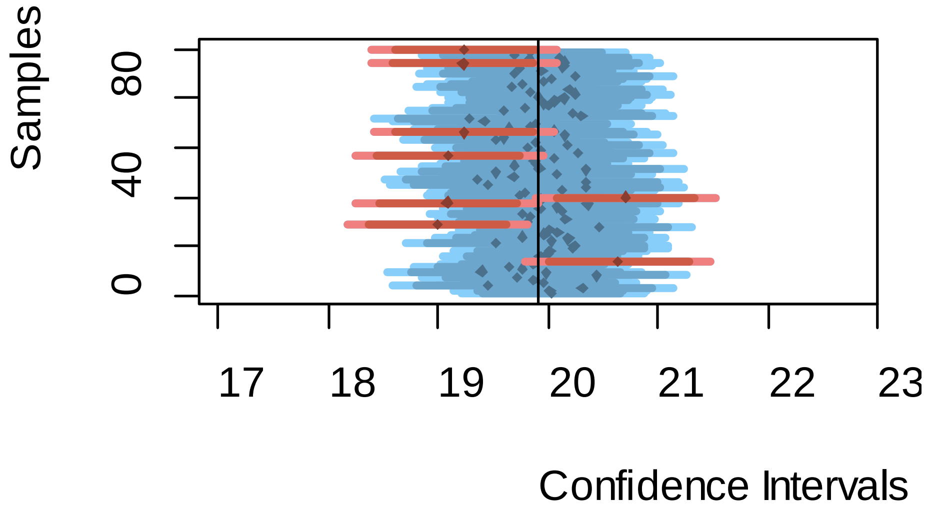
b) Rerun the previous simulation with larger samples (sample_size=300):

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?

Code:

```
Num_sample <- 100
Sample_size <- 300
list[number_exclude_ci95,
number_exclude_ci99,width_of_ci95,width_of_ci99]
<-visualize_sample_ci(num_samples = Num_sample,
sample_size = Sample_size, pop_size=10000,
distr_func= runif, min=10, max=30)
```

Results:



```
> Num_sample <- 100
> Sample_size <- 300
>
list[number_exclude_ci95,number_exclude_ci99,width_of_ci95,
width_of_ci99]<-visualize_sample_ci(num_samples =
Num_sample, sample_size = Sample_size, pop_size=10000,
distr_func=runif,min=10, max=30)
>
> Ave_ci95_Width_SampleSize300<-width_of_ci95
> Ave_ci99_Width_SampleSize300<-width_of_ci99
>
> Ave_ci95_Width_SampleSize100
[1] 2.254943
> Ave_ci95_Width_SampleSize300
[1] 1.298121
>
> Ave_ci99_Width_SampleSize100
[1] 2.968241
> Ave_ci99_Width_SampleSize300
[1] 1.708751
```

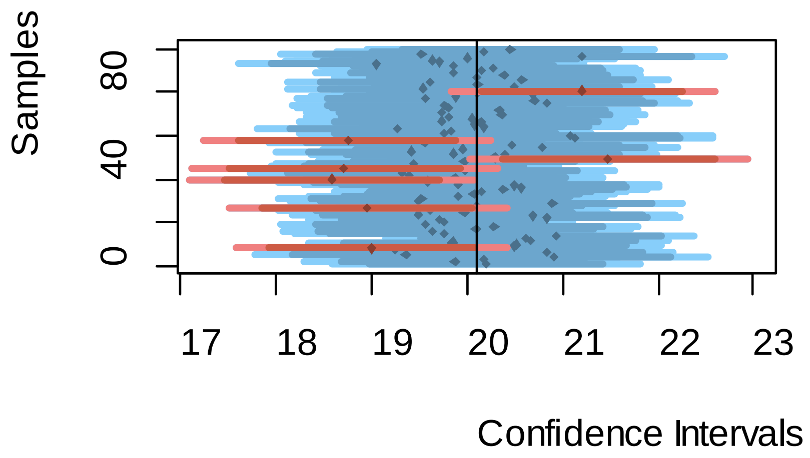
ii) This time, how many samples (out of the 100) would we *expect* to NOT include the population mean in its 95% CI?

Results:

```
> expected_number_exclude_ci95 <- Num_sample*0.05
> expected_number_exclude_ci95
[1] 5
>
> number_exclude_ci95
[1] 6
```

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

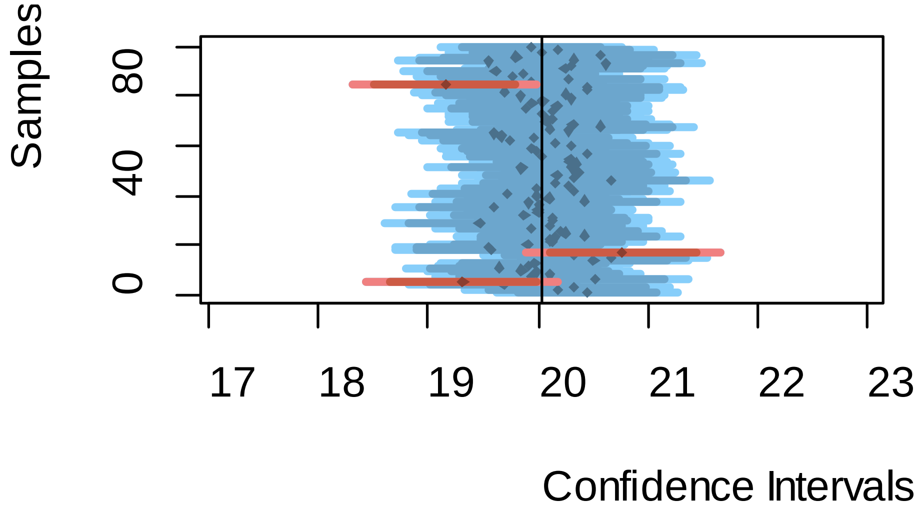
Results for (a):




```
> Num_sample <- 100
> Sample_size <- 100
>
list[number_exclude_ci95,number_exclude_ci99,width_of_ci95,
width_of_ci99]<-visualize_sample_ci(num_samples =
Num_sample, sample_size = Sample_size, pop_size=10000,
distr_func=runif,min=10, max=30)
>
> expected_number_exclude_ci95 <- Num_sample*0.05
> expected_number_exclude_ci95
[1] 5
> number_exclude_ci95
[1] 7
```

```
> expected_number_exclude_ci99 <- Num_sample*0.01
> expected_number_exclude_ci99
[1] 1
> number_exclude_ci99
[1] 0
```

Results of (b):




```
> Num_sample <- 100
> Sample_size <- 300
>
list[number_exclude_ci95,number_exclude_ci99,width_of_ci95,
width_of_ci99]<-visualize_sample_ci(num_samples =
Num_sample, sample_size = Sample_size, pop_size=10000,
distr_func=runif,min=10, max=30)
>
> Ave_ci95_Width_SampleSize300<-width_of_ci95
> Ave_ci99_Width_SampleSize300<-width_of_ci99
>
> Ave_ci95_Width_SampleSize100
[1] 2.254943
> Ave_ci95_Width_SampleSize300
[1] 1.310848
>
> Ave_ci99_Width_SampleSize100
[1] 2.968241
> Ave_ci99_Width_SampleSize300
[1] 1.725504
```

```
> expected_number_exclude_ci95 <- Num_sample*0.05
> expected_number_exclude_ci95
[1] 5
> number_exclude_ci95
[1] 3
```

Question 3) The startup company EZTABLE has an online restaurant reservation system 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. Here is some sample code to explore the data:

```
bookings <- read.table("first_bookings_datetime_sample.txt", header=TRUE)
bookings$datetime[1:9]
  [1] 4/16/2014 17:30  1/11/2014 20:00  3/24/2013 12:00 ...
  18416 Levels: 1/1/2012 17:15 1/1/2012 19:00 ... 9/9/2014 19:30
hours  <- as.POSIXlt(bookings$datetime, format="%m/%d/%Y %H:%M")$hour
mins   <- as.POSIXlt(bookings$datetime, format="%m/%d/%Y %H:%M")$min
minday <- hours*60 + mins
plot(density(minday), main="Minute (of the day) of first ever booking", col="blue", lwd=2)
```

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

Results:

```
> bookings <- read.table("first_bookings_datetime_sample.txt", header=TRUE)
> bookings$datetime[1:9]
[1] "4/16/2014 17:30"  "1/11/2014 20:00"  "3/24/2013 12:00"
[4] "8/8/2013 12:00"   "2/16/2013 18:00"  "5/25/2014 15:00"
[7] "12/18/2013 19:00" "12/23/2012 12:00" "10/18/2013 20:00"
```

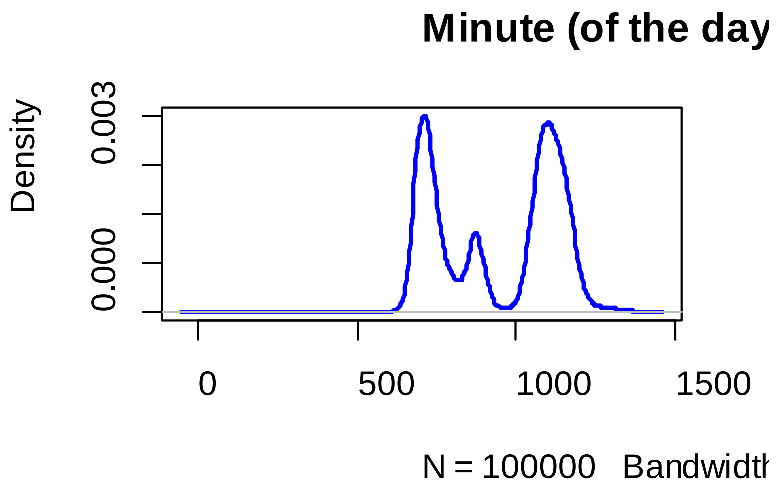
```
> hours  <- as.POSIXlt(bookings$datetime, format="%m/%d/%Y %H:%M")$hour
> mins   <- as.POSIXlt(bookings$datetime, format="%m/%d/%Y %H:%M")$min
> minday <- hours*60 + mins
> plot(density(minday), main="Minute (of the day) of first ever booking",
col="blue", lwd=2)
>
> mean(minday)
[1] 942.4964
> std_error <- sd(minday)/sqrt(length(minday))
> std_error
[1] 0.5997673
> ci95_mean1 <- mean(minday) + 1.96*(std_error)
> ci95_mean2 <- mean(minday) - 1.96*(std_error)
> ci95_mean1
[1] 943.6719
> ci95_mean2
[1] 941.3208
```

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

Code:

```
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
plot(density(minday), main="Minute (of the day) of first ever booking",
col="blue", lwd=2)

sample0 = sample(minday, sample_size)
sample_size = 300

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

Results:

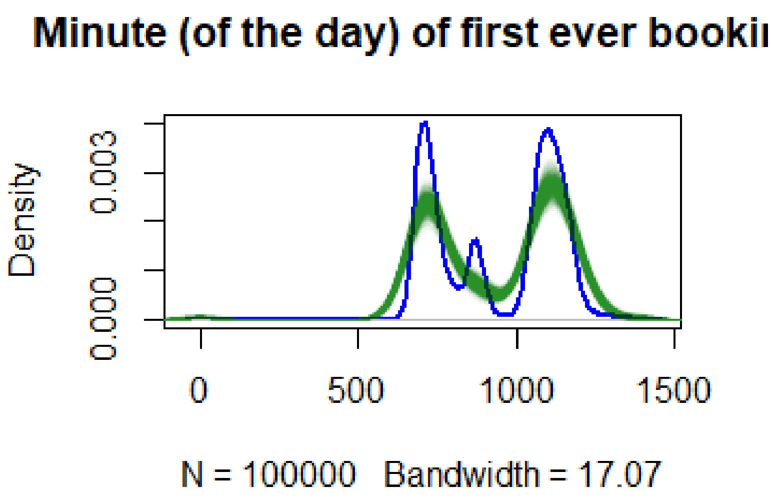


iii) Visualize the means of the 2000 bootstrapped samples

Code:

```
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
plot(density(minday), main="Minute (of the day) of first ever booking",
col="blue", lwd=2)

sample0 = sample(minday, sample_size)
sample_size = 300

compute_sample_mean <- function(sample0) {
  resample <- sample(sample0, length(sample0), replace=TRUE)
  mean(resample)
}

plot_resample_density <- function(sample_i) {
  lines(density(sample_i), col=rgb(0.0, 0.4, 0.0, 0.01))
  return(mean(sample_i))
}
sample_means <- apply(resamples, 2, FUN=plot_resample_density)
resamples <- replicate(2000,
                       sample(sample0, length(sample0), replace=TRUE))
```

Results:



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

Results:

```
> 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
> sample0 = sample(minday, sample_size)
> sample_size = 300
> compute_sample_mean <- function(sample0) {
+   resample <- sample(sample0, length(sample0), replace=TRUE)
+   mean(resample)
+ }
> minday_mean <- mean(minday)
> minday_mean
[1] 942.4964
> sample_means <- apply(resamples, 2, FUN=plot_resample_density)
>
> resamples <- replicate(2000,
+                        sample(sample0, length(sample0), replace=TRUE))
> quantile(sample_means, probs=c(0.05, 0.95))
      5%      95%
913.2958 948.5358
```

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

Results:

```
> 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
>
> sample0 = sample(minday, sample_size)
> sample_size = 300
>
> compute_sample_mean <- function(sample0) {
+   resample <- sample(sample0, length(sample0), replace=TRUE)
+   mean(resample)
+ }
>
> minday_median <- median(minday)
> minday_median
[1] 1040
```

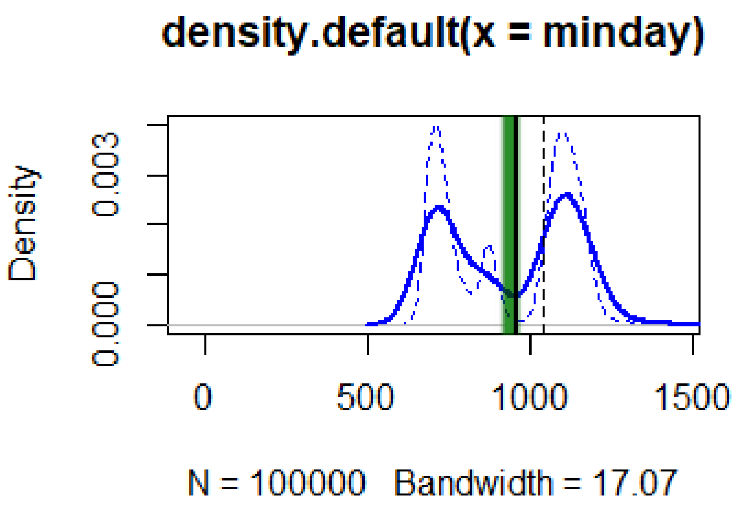
ii) Visualize the *medians* of the 2000 bootstrapped samples

Code:

```
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
plot(density(minday), main="Minute (of the day) of first ever booking",
col="blue", lwd=2)

sample0 = sample(minday, sample_size)
sample_size = 300
```

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

minday_median <- median(minday)

plot_resample_density <- function(sample_i) {
  lines(density(sample_i), col=rgb(0.0, 0.4, 0.0, 0.01))
  return(median(sample_i))
}
sample_medians <- apply(resamples, 2, FUN=plot_resample_density)

resamples <- replicate(2000,
                       sample(sample0, length(sample0), replace=TRUE))

plot(density(minday), col="blue", lty="dashed")
lines(density(sample0), col="blue", lwd=2)

plot_resample_median <- function(sample_i) {
  abline(v=median(sample_i), col=rgb(0.0, 0.4, 0.0, 0.01))
}
apply(resamples, 2, FUN=plot_resample_mean)

abline(v = median(sample_means), lwd=2)
abline(v = minday_median, lty="dashed")
```

Results:



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

Results:

```
> 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
>
> sample0 = sample(minday, sample_size)
> sample_size = 300
>
> compute_sample_median <- function(sample0) {
+   resample <- sample(sample0, length(sample0), replace=TRUE)
+   median(resample)
+ }
>
> minday_median <- median(minday)
>
> sample_medians <- apply(resamples, 2, FUN=plot_resample_density)
>
> resamples <- replicate(2000,
+                        sample(sample0, length(sample0), replace=TRUE))
>
> quantile(sample_medians, probs=c(0.05, 0.95))
  5%  95%
 900 1080
```