HW_Week4

109071503

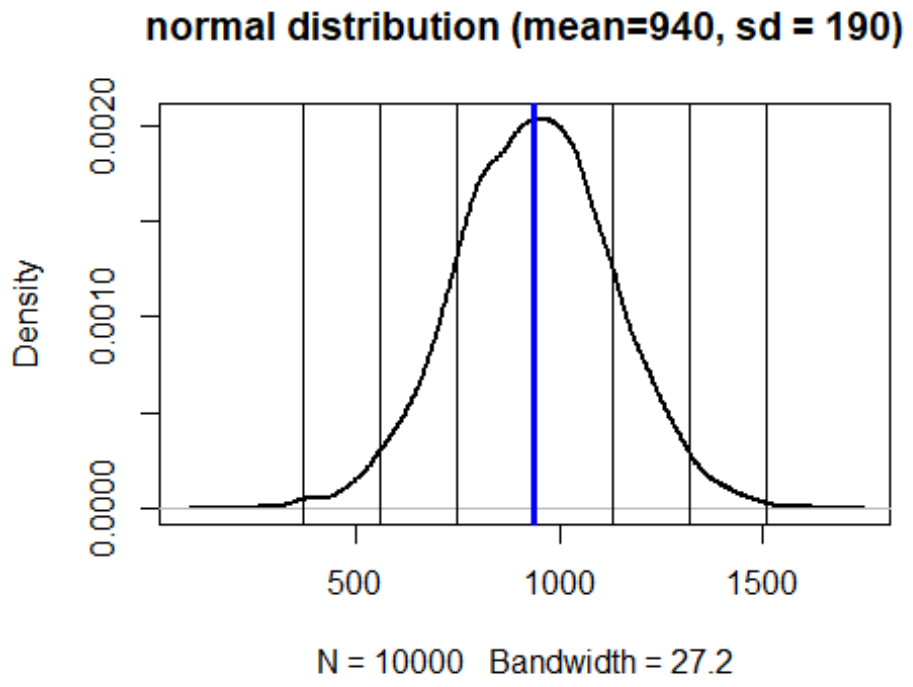
3/16/2021

Solution to HW(Week 4)

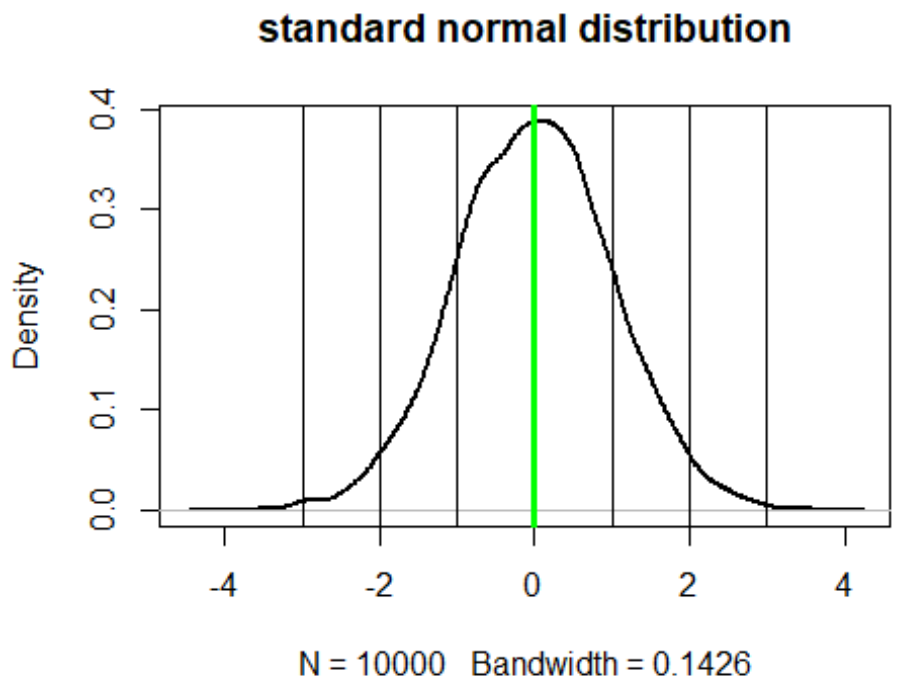
Question 1) Reexamine the meaning to standardize data

(a) Create a normal distribution with mean = 940, sd = 190, and apply standardization to it.

```
c = c(-3, -2, -1, 1, 2, 3)
rdata = rnorm(10000, mean = 940, sd = 190)
plot(density(rdata), lwd = 2, main = "normal distribution (mean=940, sd = 190)")
abline(v=mean(rdata), col ="blue", lwd =3)
abline(v=mean(rdata)+c*sd(rdata))
```




```
rnorm_std = (rdata - mean(rdata))/sd(rdata)
plot(density(rnorm_std), lwd = 2, main = "standard normal distribution")
abline(v=mean(rnorm_std), col ="green", lwd =3)
abline(v=mean(rnorm_std)+c*sd(rnorm_std))
```



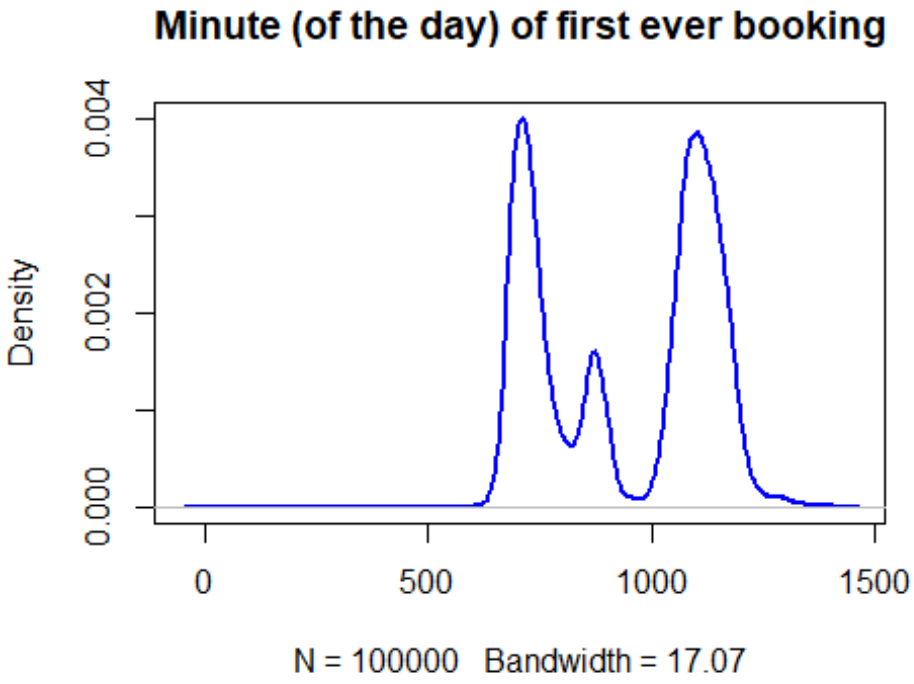
Comment :
(i) We should expect the mean and standard deviation of rnorm_std to be 0 and 1. Because when we apply standization, we shift our data by minus the amounts of mean, and then remove the scale of standard deviation with $Y = \frac{X - mean(X)}{sd(X)}$.

(ii) Bell shape. Since it is a simple linear transformation $Y = \frac{X - mean(X)}{sd(X)}$, in this case , $std = \frac{rdata - 940}{190}$, the new variables still follow normal distribution. So, it's probability distribution should look like a normal distribution, that is, "Bell shape".

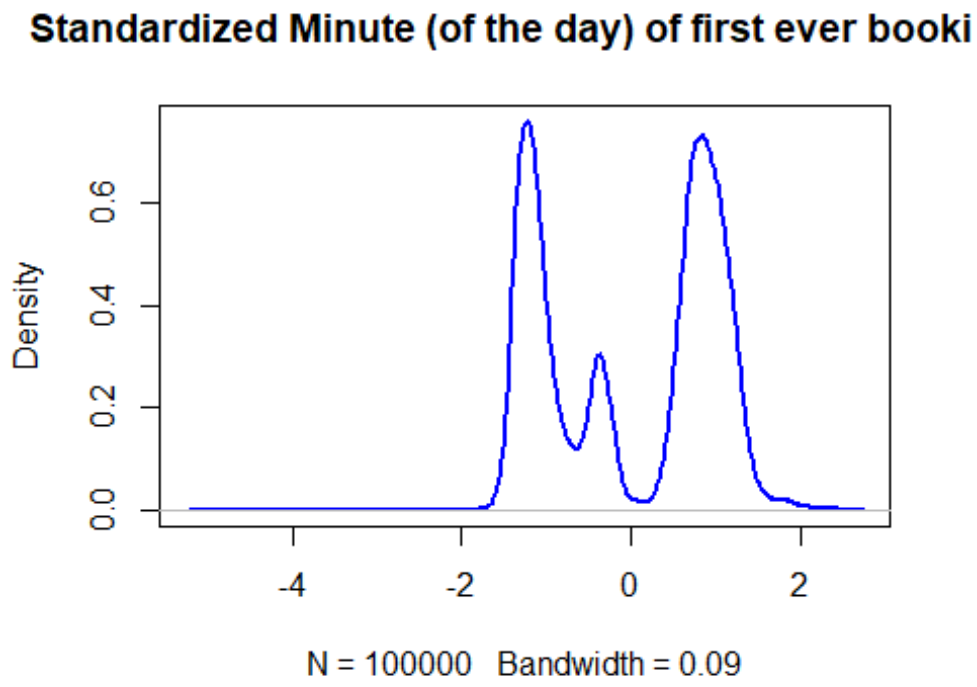
(iii) Yes. we do generally call this kind of distributions are normal and standardized.

(b) Create a standardized version of minday

```
# First, I need to load the data and generate our "minday" variable
bookings <- read.table("first_bookings_datetime_sample.txt", header=TRUE)
# bookings$datetime[1:9]
hours  <- as.POSIXlt(bookings$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)
```




```
# Then, we create a standardized version of minday, and call it "minday_std"
minday_std = (minday-mean(minday))/sd(minday)
cat("Mean of minday_std is: ",mean(minday_std))

## Mean of minday_std is:  -4.25589e-17

cat("Standard deviation of minday_std is:", sd(minday_std))

## Standard deviation of minday_std is: 1

plot(density(minday_std), main="Standardized Minute (of the day) of first ever book
ing", col="blue", lwd=2)
```



Comment:

(i) We should expect the mean and standard deviation of minday_std to be 0 and 1. Because of the same reason describe in Question 1 (a), we just shift our data by minus the amounts of mean, and then remove the scale of standard deviation with $Y = \frac{X - mean(X)}{sd(X)}$.

(ii) The distribution of minday_std should look like the distribution of minday. However, the range of standardized data is smaller because we remove the scale of standard deviation. Therefore, the density is relative high. In conclusion, the shape of both variables are similar, the range of variable "minday_std" is smaller, the density of "minday_std" shown in the plot is higher.

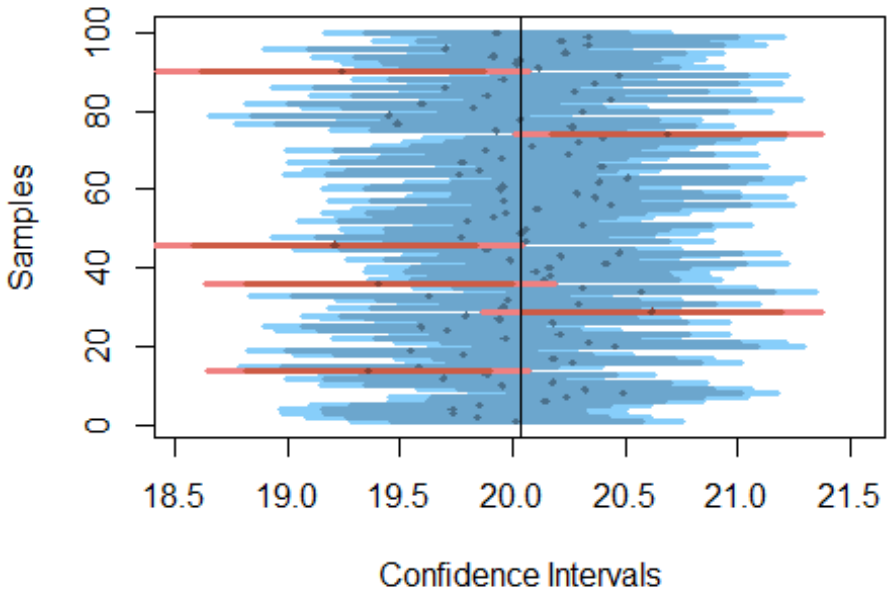
Question 2)

```
# 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))
}

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])
}
```

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

```
set.seed(2)
visualize_sample_ci(num_samples = 100, sample_size = 100, pop_size=10000,
                    distr_func=rnorm, mean=20, sd=3)
```



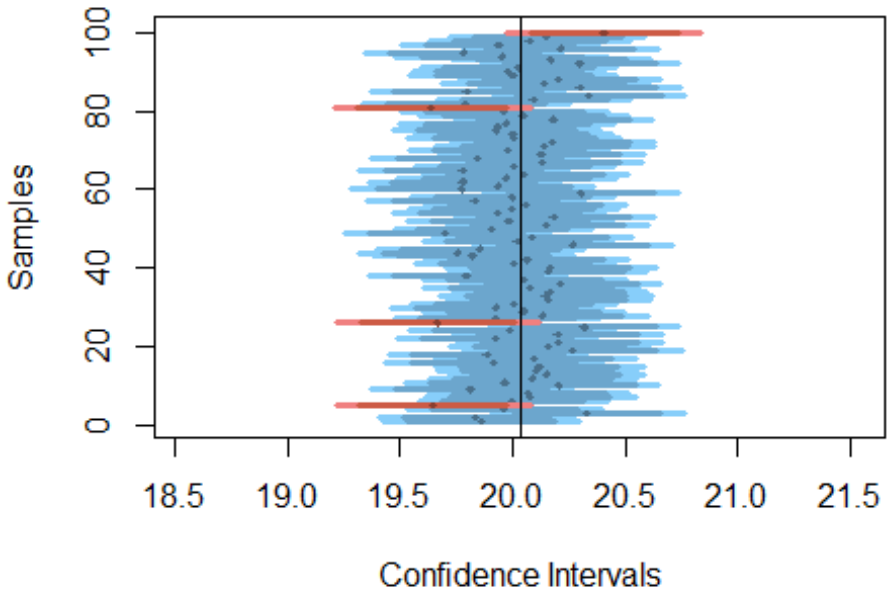
Comment:

(i) 5% of num_samples, that is 5 (5%*100=5) samples are expected to NOT include the population mean in its 95% CI. And in our experiment by set random seed (2), we have 6 samples that are not include the population mean in its 95% CI.

(ii) 1% of num_samples, that is 1 samples. In this case, we have 0 samples that are not include the population mean in its 99% CI.

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

```
set.seed(2)
visualize_sample_ci(num_samples = 100, sample_size = 300, pop_size=10000,
                        distr_func=rnorm, mean=20, sd=3)
```



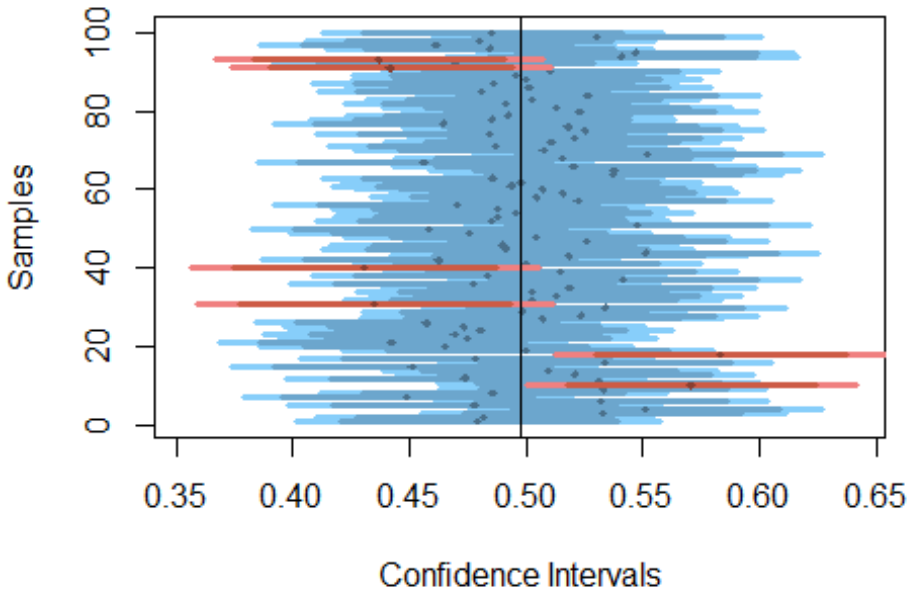
Comment:

(i) Narrower than before.

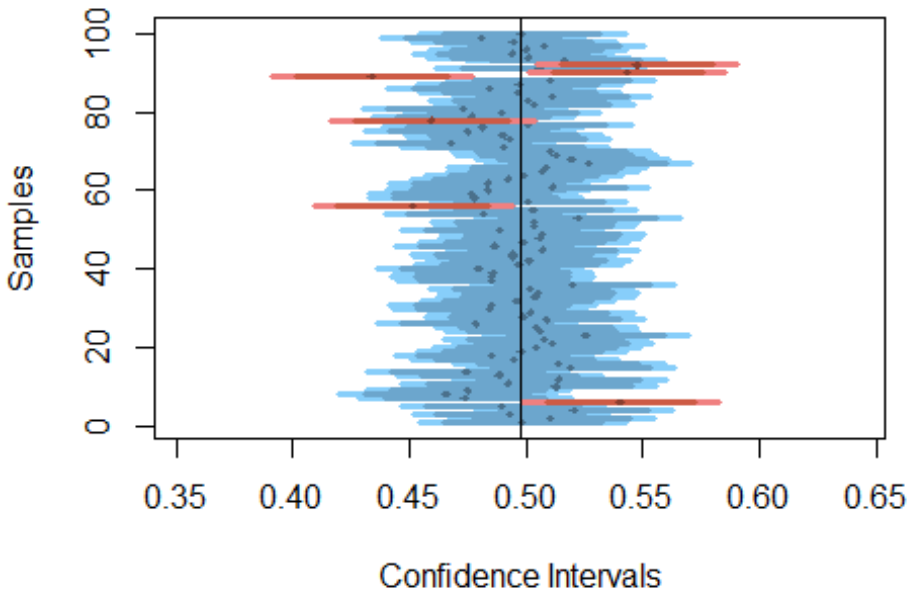
(ii) 5% of num_samples, that is 5 samples. In this case, we have 4 samples that are not include the population mean in its 99% CI.

(c) Compare to above questions, we ran the above two examples using a uniformly distributed population

```
set.seed(3)
visualize_sample_ci(num_samples = 100, sample_size = 100, pop_size=10000,
                    distr_func=runif)
```




```
set.seed(3)
visualize_sample_ci(num_samples = 100, sample_size = 300, pop_size=10000,
                    distr_func=runif)
```

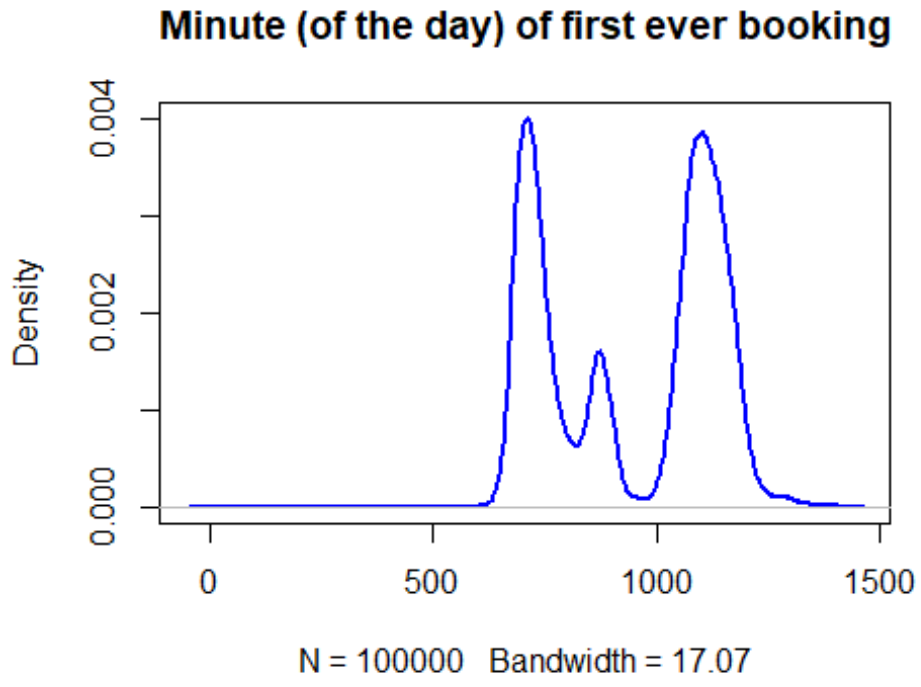


Comment: I expect that my answer will not change. Because samples from different distribution will not change the properties of 95% CI and 99% CI. Confidence Interval are calculated bases on the assumption of distributions.

Question 3) The following question are related to the startup company EZTABLE

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  8/8/2013 12:00
## [5] 2/16/2013 18:00  5/25/2014 15:00  12/18/2013 19:00 12/23/2012 12:00
## [9] 10/18/2013 20:00
## 18416 Levels: 1/1/2012 17:15 1/1/2012 19:00 1/1/2013 11:00 ... 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) Calculate the "average" booking time for minday by following instructions

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

```
mean(minday)
```

```
## [1] 942.4964
```

```
sd(minday)
```

```
## [1] 189.6631
```

```
CI = mean(minday) + c(1.96, -1.96)*sd(minday)
cat("95% confidence interval of minday is: "," [", CI[1], "~", CI[2], "]")
```

```
## 95% confidence interval of minday is:   [ 1314.236 ~ 570.7567 ]
```

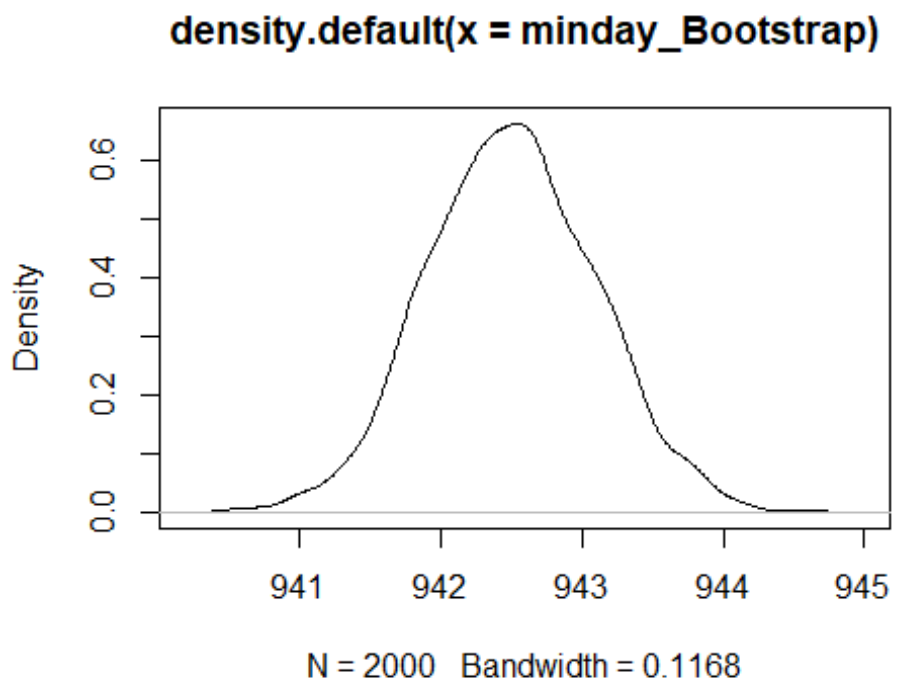
Note: 1. We use sample mean and sample standard deviation to calculate the estimated parameter of population 2. We calculate the 95% CI of population mean by 95% CI = $\overline{Y} \pm 1.96 * \overline{\sigma_Y}$

(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)
}
minday_Bootstrap = replicate(2000, compute_sample_mean(minday))
```

(iii) Visualize the means of the 2000 bootstrapped samples

```
plot(density(minday_Bootstrap))
```




```
mean(minday_Bootstrap)
```

```
## [1] 942.5039
```

```r
sd(minday_Bootstrap)
```

```
## [1] 0.5936019
```

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

```r
CI = mean(minday_Bootstrap) + c(1.96, -1.96)*sd(minday_Bootstrap)
cat("95% confidence interval of minday_Bootstrap is: [", CI[1], "~", CI[2], "]")
```

```
## 95% confidence interval of minday_Bootstrap is: [ 943.6673 ~ 941.3404 ]
```

(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

```r
compute_sample_median <- function(sample0) {
  resample <- sample(sample0, length(sample0), replace=TRUE)
  median(resample)
}
minday_median_BS = replicate(2000, compute_sample_median(minday))

mean(minday_median_BS)
```

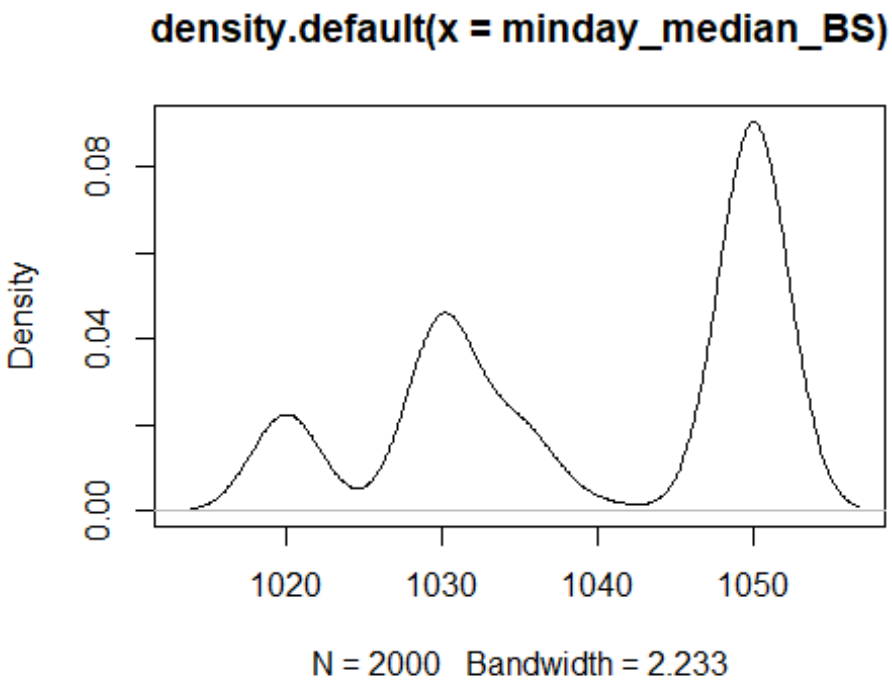
```
## [1] 1039.539
```

```r
median(minday)
```

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

(iii) Visualize the medians of the 2000 bootstrapped samples

```r
plot(density(minday_median_BS))
```

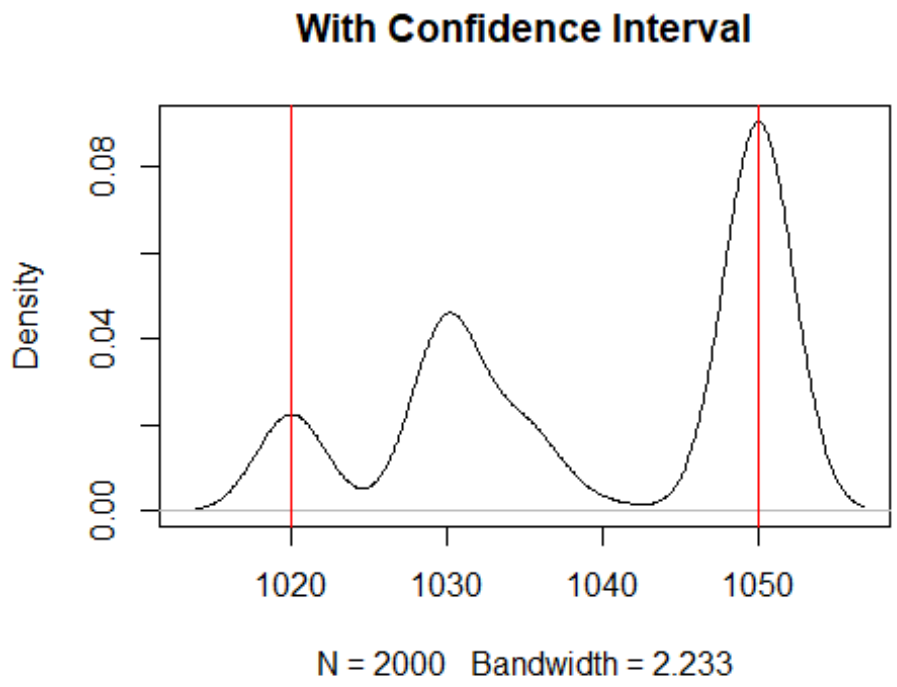


(iv) Estimate the 95% CI of the bootstrapped medians.

```
sort_median_BS=  minday_median_BS[order(minday_median_BS)]
sort_median_BS[0:9]
```

```
## [1] 1020 1020 1020 1020 1020 1020 1020 1020 1020 1020
```

```
lower_bound = sort_median_BS[round(0.025*length(minday_median_BS))]
upper_bound = sort_median_BS[round(0.975*length(minday_median_BS))]

plot(density(minday_median_BS), main = "With Confidence Interval")
abline(v=c(sort_median_BS[50],sort_median_BS[1950]) ,col="red")
```




```
cat("95% confidence interval of the bootstrapped medians is: "," [", lower_bound, "
~", upper_bound, "]")
```

```
## 95% confidence interval of the bootstrapped medians is:   [ 1020 ~ 1050 ]
```