HW3_Report

Student ID: 107023058

Questioin 1:

(a) reexamine what it means to standardize data.

```
norm <- rnorm(500,mean=940,sd = 190)
#create a normal distribution
rnorm_std <- (norm - mean(norm))/sd(norm)
#standardization
```

(i)

- What should we expect the mean and standard deviation of rnorm_std ?
- We expect the value of the mean and standard deviation of the rnorm_std will be 0 and 1 respectively. Because we subtract the original rnorm by its mean and divide by its standard deviation, according to the property of normal distribution, the two operations will make its mean and standard deviation become 0 and 1.

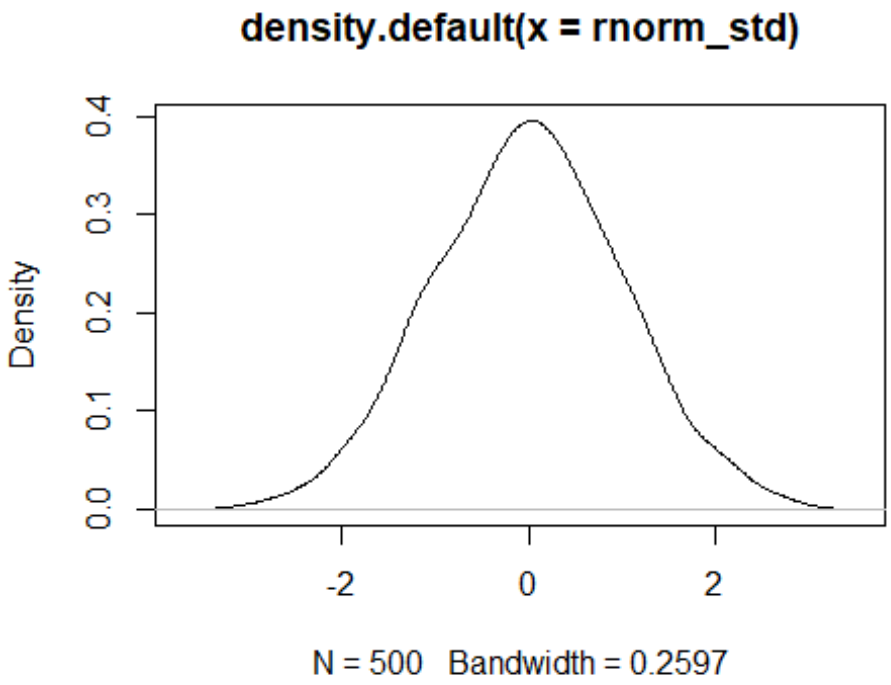
```
mean(rnorm_std) # mean of rnorm_std

## [1] -2.889052e-16

sd(rnorm_std) # standard deviation of rnorm_std

## [1] 1
```

(ii)

- Q: What should the shape of the distribution looks like?

- It looks like a bell-shape graph , which central line is on the 0 point. As the graph below shows.

```
plot(density(rnorm_std))

# show the graph of rnorm_std
```



(iii)

- Q: What do we generally call distribution that are normalization?
- We would call it standard normal distribution.

(b) Create a standardized version of minday discussed in question 3

```
bookings <- read.table("D:/Retro/NTHU/課程講義/大三/計算統計於商業分析之應用/HW3/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"

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
```

(i)

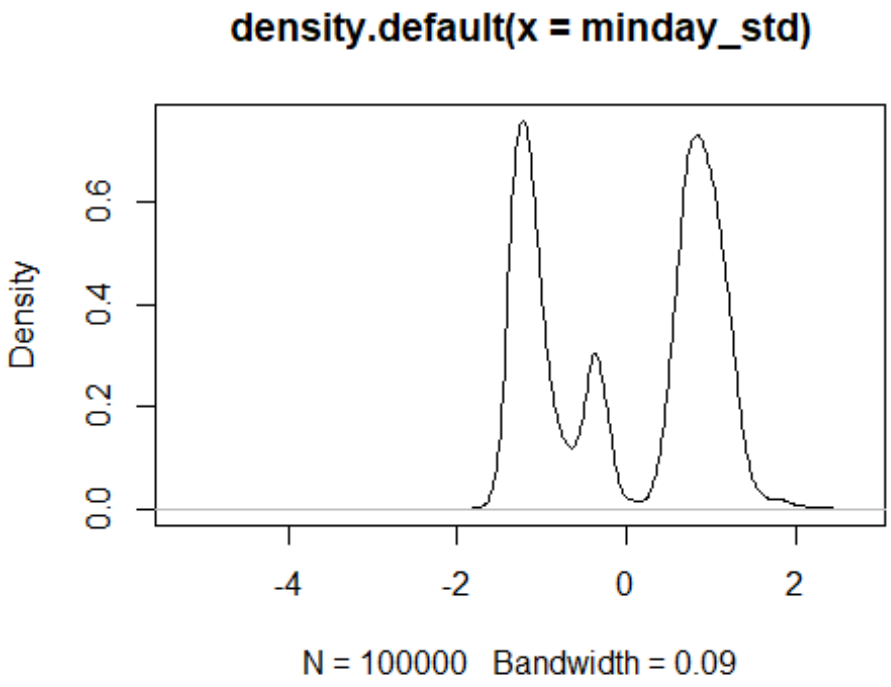
- Q: What should we expect the mean and standard deviation of minday_std?

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

## [1] -4.25589e-17

sd(minday_std) # standard deviation

## [1] 1
```

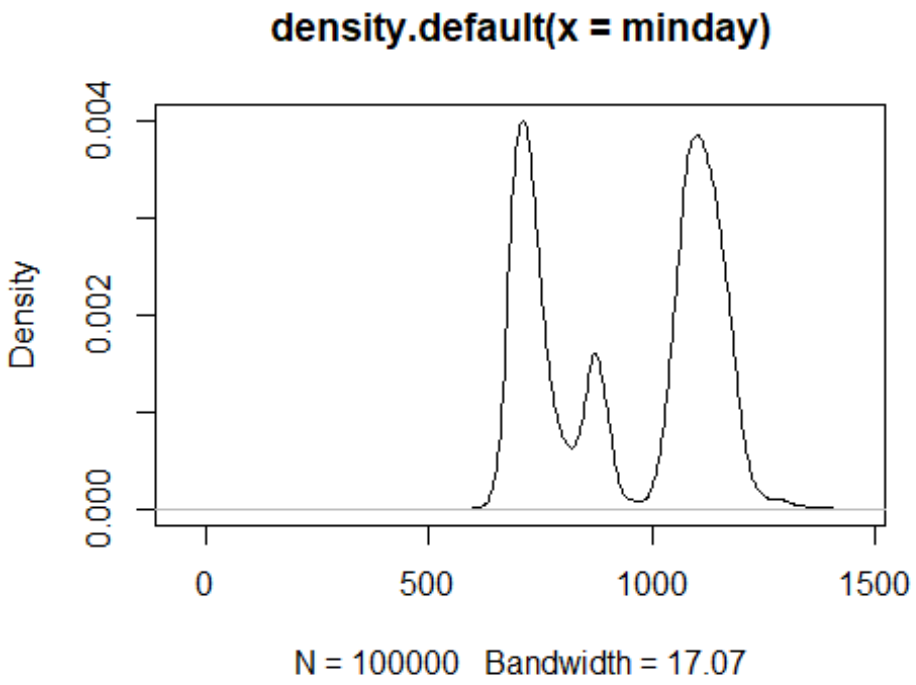
(ii)

- Q: What should the minday_std look like compare to minday?

```
plot(density(minday_std))

# the distribution of minday_std
```



```
plot(density(minday))
# the distribution of minday
```



- the two distrbution look the same except for their scales.

Question 2

- Copy and run the visualize_sample_ci()

```
# 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)
```

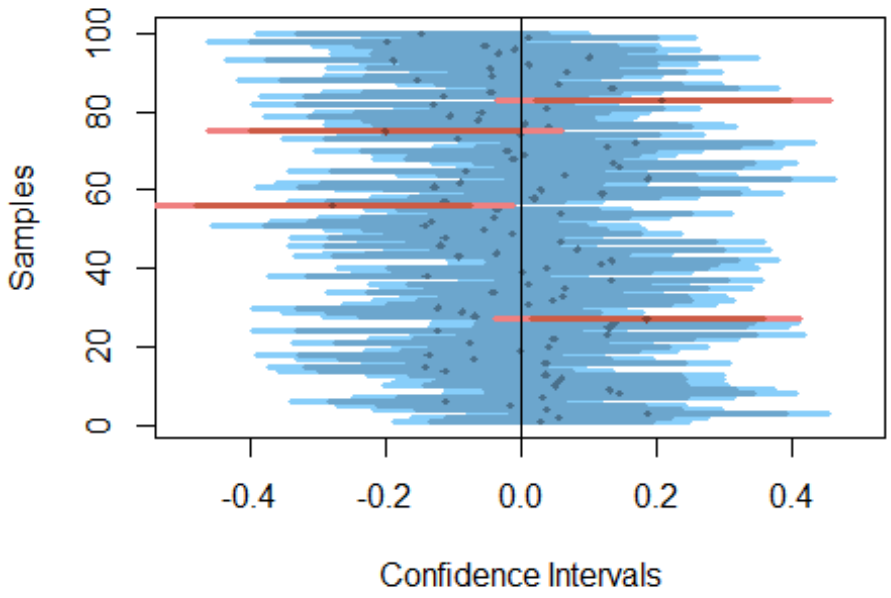
```
  # Simulate samples
  samples <- replicate(num_samples,
                       sample(population_data, sample_size, replace=FAL
SE))

  # 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 Interval
s")
  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_hig
h[bad],
                 sample_means[bad], bad, good=FALSE)

  notInclude_95 <- which((ci95_low > pop_mean) | (ci95_high < pop_mea
n))
  # find which is not include in 95% CI
  notInclude_99 <- which((ci99_low > pop_mean) | (ci99_high < pop_mea
n))
  # find which is not include in 99% CI
  #cat("# of not including in 95% inteval:",length(notInclude_95)," # o
f not including in 99% inteval:",length(notInclude_99))
  width_of_95 <- mean(abs(ci95_high-ci95_low))
  width_of_99 <- mean(abs(ci99_high-ci99_low))
  # Draw true population mean
  abline(v=mean(population_data))
  return(c(length(notInclude_95),length(notInclude_99),width_of_95,widt
h_of_99))
}
```

```
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

```
num <- c()
num <- visualize_sample_ci(num_samples = 100, sample_size = 100, pop_si
ze=10000,distr_func = rnorm)
```



```
num[1] # the number of not including mean in 95% CI

## [1] 4

num[2] # the number of not including mean in 99% CI

## [1] 1
```

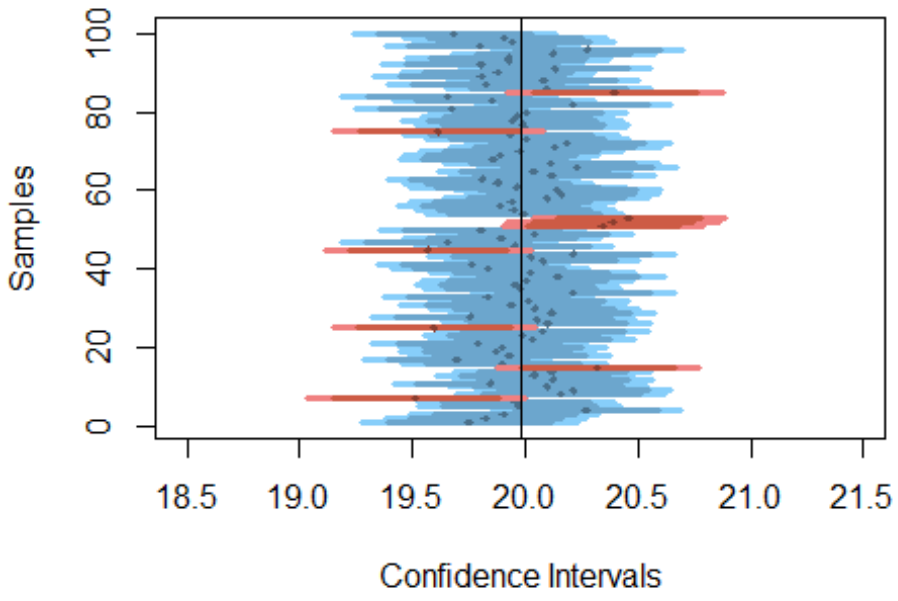
- I simply add some line of code to return the number that don't include in 95% and 99% CI and its 95% & 99% CI width.
- Compare to theoretical answer, the theoretical value should be:

```
num_samples <- 100
theoretical_95<- num_samples * 0.05
theoretical_95

## [1] 5

theoretical_99<- num_samples * 0.01
theoretical_99

## [1] 1
```

(b)

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

```
num_2 <- c()
num_2 <- visualize_sample_ci(num_samples = 100, sample_size = 300, pop_
size=10000,distr_func=rnorm, mean=20, sd=3)
```



change the sample_size, do we expect their 95% and 99% CI to become wider or narrower than before ?

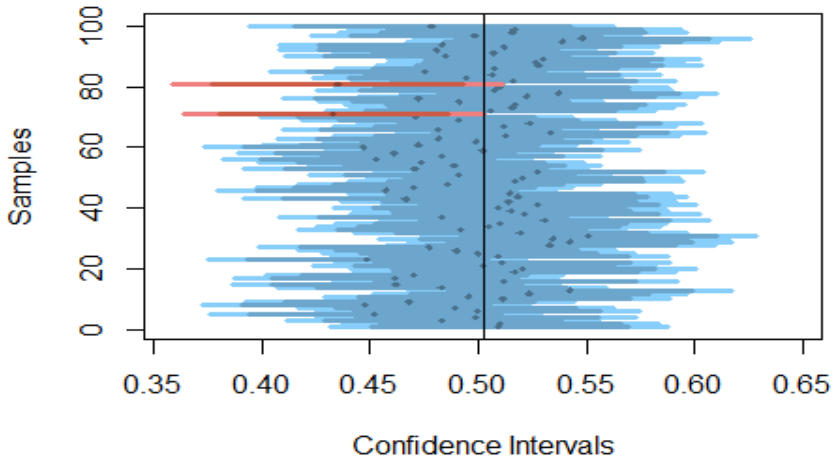
```
num_2[3] # the width of 95% CI while sample_size = 300

## [1] 0.6782519

num_2[4] # the width of 99% CI while sample_size = 300

## [1] 0.892801

num[3] # the width of 95% CI while sample_size = 100

## [1] 0.3856807

num[4] # the width of 99% CI while sample_size = 100

## [1] 0.5076818
```

- From the result, we can find that changing sample size will affect its C.I width. Although it look like more narrow in sample_size = 300, indeed it is wider than sample_size = 100. We can check it from the value above and also in the values of the horizontal axis in the graph.

(c)

- Using runif to run the function, how do you expect your answer to (a) and (b)?

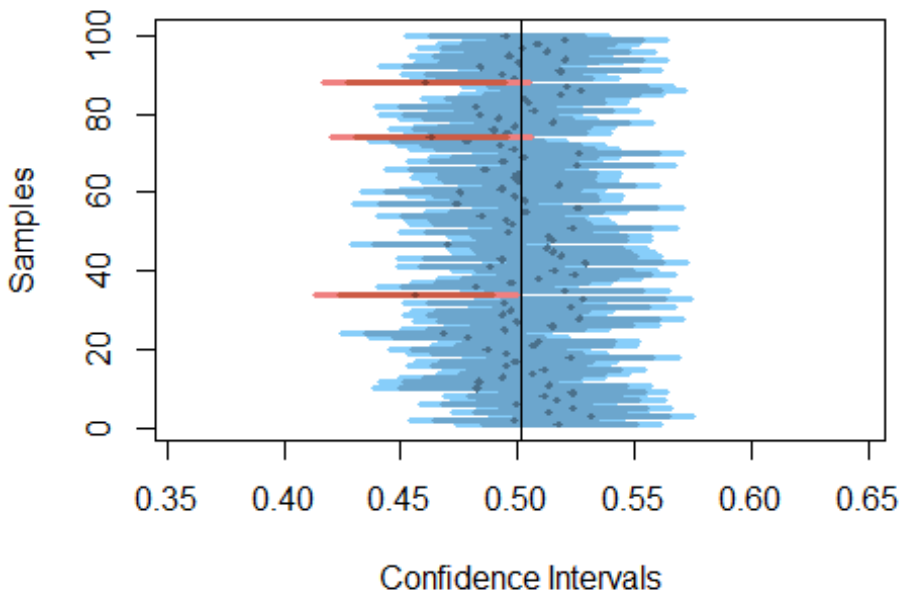
```
num_3 <- c()
num_3 <- visualize_sample_ci(num_samples = 100, sample_size = 100, pop_
size=10000,distr_func = runif)
```



```
num_3[1] # the number of not including mean in 95% CI

## [1] 2

num_3[2] # the number of not including mean in 99% CI

## [1] 0

num_3[3] # the width of 95% CI with runif

## [1] 0.1136426

num_3[4] # the width of 99% CI with runif

## [1] 0.1495907
```

- From the result, we can find that the number that we *expect* to not include the population mean in its 95% CI is 2, and in its 99% CI is 0 .

```
# change the sample_size to 300
num_4 <- c()
num_4 <- visualize_sample_ci(num_samples = 100, sample_size =
300, pop_size=10000,distr_func = runif)
```

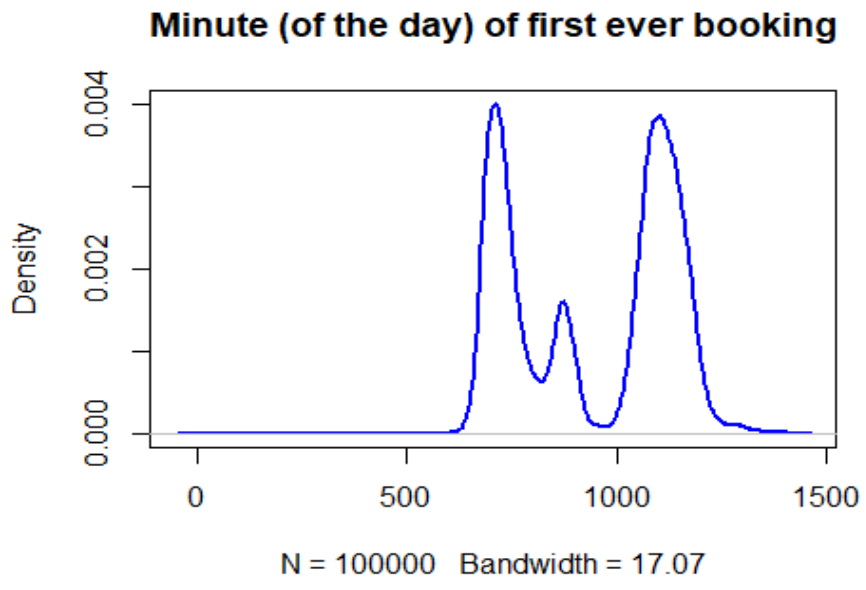


```
num_4[3] # the width of 95% CI with runif

## [1] 0.06575437

num_4[3] # the width of 99% CI with runif

## [1] 0.06575437
```

From the result above, we can find that the width of runif in sample_size = 300 is more narrow than in sample_size = 100. Different from the result in rnorm.

Question 3

```
bookings <- read.table("D:/Retro/NTHU/課程講義/大三/計算統計於商業分析之應用/HW3/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"

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)

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

(i)

- Using traditional statistical methods to estimate the mean、standard error and 95% CI.

```
minday_mean <- mean(minday) # mean
minday_se <- sd(minday)/sqrt(length(minday)) # standard error
minday_CI <- c(minday_mean - 1.96*minday_se, minday_mean + 1.96 * minda
y_se)
minday_CI # the confidence interval of 95%

## [1] 941.3208 943.6719
```

(ii)

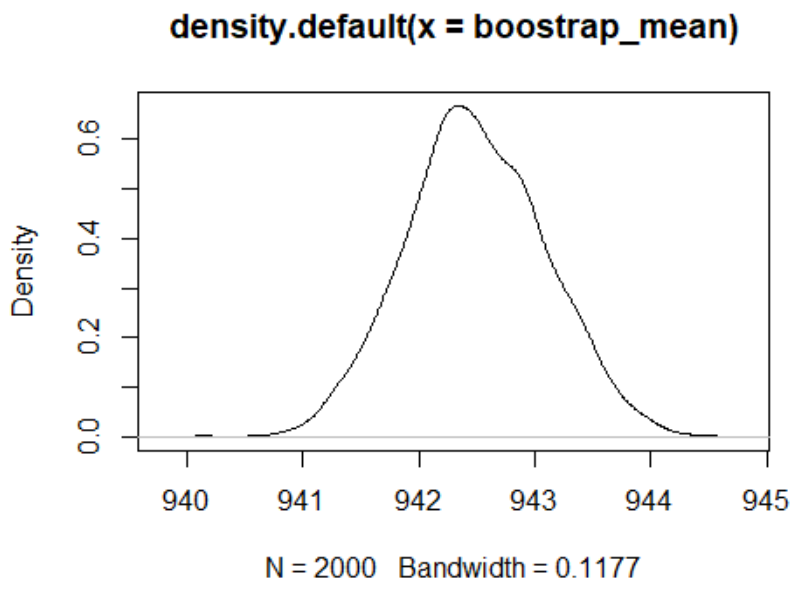
- Bootstrap to produce 2000 new samples from the original sample

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

boostrap_mean<- replicate(2000,compute_sample_mean(minday))
# calculate bootstrapped mean
```

(iii)

- Visualize the means of the 2000 bootstrapped samples.

```
plot(density(boostrap_mean)) # visualization
```



(iv)

- Estimate 95% CI of the bootstrapped mean.

```
mean_CI_95 <- c(mean(boostrap_mean)-1.96*(sd(boostrap_mean)/sqrt(length
(boostrap_mean))),
                mean(boostrap_mean)+1.96*(sd(boostrap_mean)/sqrt(length
(boostrap_mean))))

mean_CI_95 # 95% CI of the bootstrapped mean

## [1] 942.4670 942.5195
```

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

(i)

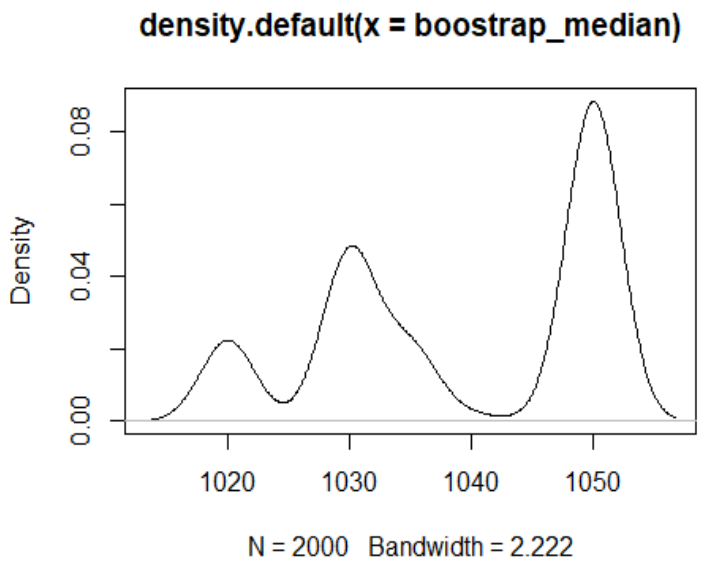
- Estimate the median of minday.

```
median(minday)

## [1] 1040
```

(ii)

- Visualize the medians of the 2000 bootstrapped median.

```
compute_sample_median <- function(sample0) {
  resample <- sample(sample0, length(sample0), replace=TRUE)
  median(resample)
}
# change the function to apply median function.
boostrap_median<- replicate(2000,compute_sample_median(minday))
# calculate bootstrapped median
plot(density(boostrap_median)) # visualization
```



(iii)

- Estimate 95% CI of the bootstrapped median.

```
median_CI_95 <- c(mean(boostrap_median)-1.96*(sd(boostrap_median)/sqrt
(length(boostrap_median))),
                  mean(boostrap_median)+1.96*(sd(boostrap_median)/sqrt(le
ngth(boostrap_median))))

median_CI_95 # the 95% CI of the bootstrapped median

## [1] 1038.79 1039.78
```