HW3

110078509

2022/3/4

```
rm(list=ls())
#remove the random variable to fresh the working environment
library(dplyr)

##
## 載入套件：'dplyr'

## 下列物件被遮斷自 'package:stats':
##
##     filter, lag

## 下列物件被遮斷自 'package:base':
##
##     intersect, setdiff, setequal, union

library(lubridate)

## Warning: 套件 'lubridate' 是用 R 版本 4.1.2 來建造的

##
## 載入套件：'lubridate'

## 下列物件被遮斷自 'package:base':
##
##     date, intersect, setdiff, union
```

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

```
a <- rnorm(10000, mean=940, sd=190)
rnorm_std <- (a - mean(a))/sd(a)
```

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

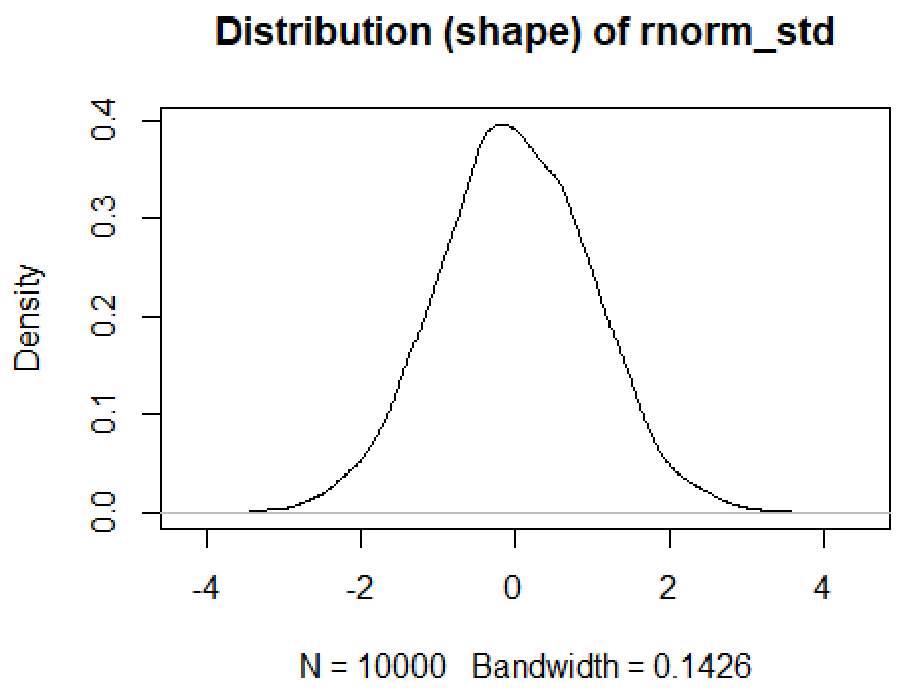
```
mean(rnorm_std)

## [1] -2.283153e-16

sd(rnorm_std)

## [1] 1
```

reason:

The standard deviation of rnorm_std must be 1 and its mean should be 0 (pretty close). Because it's what we called "Normalization"", which is being used to standardize the the distance of sample data away from the population mean, which is set as 940. In brief, it's doing a parallel swift and divide by the standard deviation. Therefore, the original mean would not affect the result of normalization.

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

```
ii <- density(rnorm_std)
plot(ii, main="Distribution (shape) of rnorm_std")
```



Reason:

It's a bell shape. Intuitively, due to the original data is standard bell shape , the closer the data are to the mean, the greater the amount of data and the lower the residual value.

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

Ans: Standard Normal Distribution.

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

```
bookings <- read.table("first_bookings_datetime_sample.txt", header=TRU
E)
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
```

```
sprintf("Orignial: The mean: %f, the standard deviation: %f",mean(minda
y) ,sd(minday))

## [1] "Orignial: The mean: 942.496350, the standard deviation: 189.663
078"

standardize <- function(i){
    i <- ((i - mean(i))/sd(i))
    return(i)
}


minday_std   =standardize(minday)
```

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

Ans:

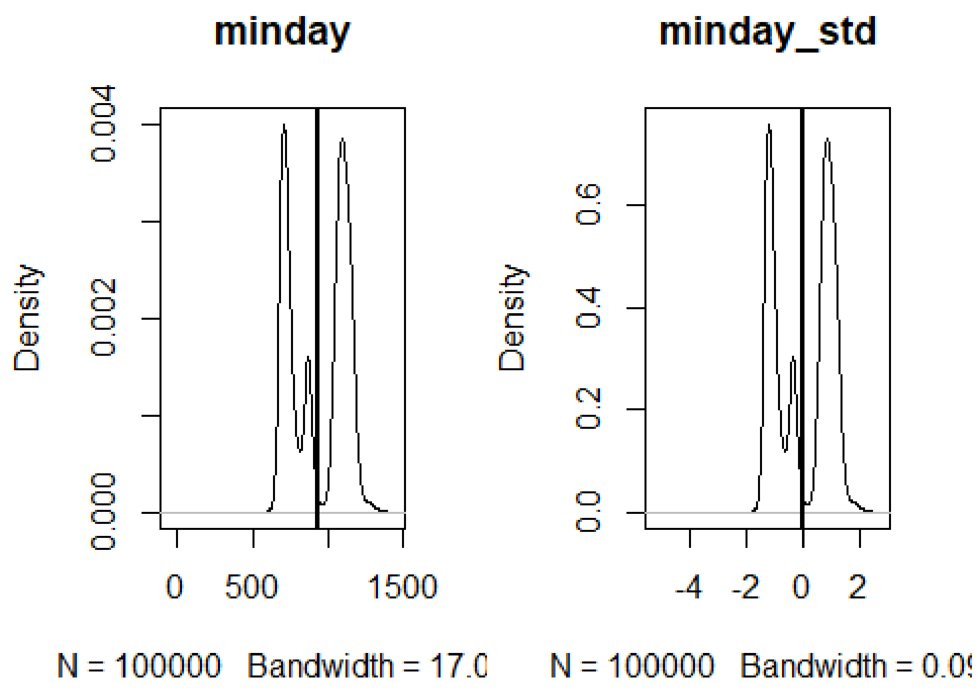
```
sprintf("The mean: %f, the standard deviation: %f",round(mean(minday_st
d),3) ,sd(minday_std) )

## [1] "The mean: -0.000000, the standard deviation: 1.000000"
```

We should expect that the mean of minday_std is around 942 (mean(minday)) and standard deviation of minday_std should be about 189.8 . (sd(minday)) Notice that sample mean is 937.527000, and the standard deviation is 189.663078.

According to the law of large number,the higher the chance that the sample mean closer to the expected value as the rising of sample size(n).

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

```
par(mfrow = c(1,2))
plot(density(minday), main="minday")+abline(v=mean(minday), lwd=2)

## integer(0)

plot(density(minday_std), main="minday_std")+abline(v=mean(minday_std),
 lwd=2)
```



```
## integer(0)
```

Explanation:

They look alike. The reason is that the deduction of the mean can be seen as a leftward parallel shift, which would not change the shape of the probability distribution. Then we divide it by the standard deviation, which only affect the scale.

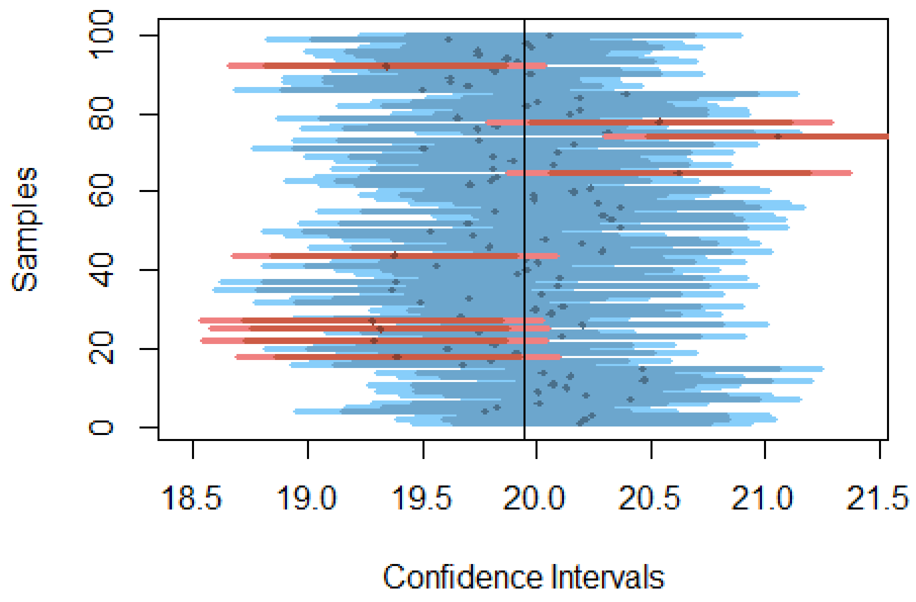
(Question 2)

Run visualize_sample_ci(), which simulates samples drawn randomly from a population. 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.

```
#Gitub code Start Here
# Visualize the confidence intervals of samples drawn from a population
#   e.g.,
#     visualize_sample_ci(sample_size=300, distr_func=rnorm, mean=50, s
d=10)
#     visualize_sample_ci(sample_size=300, distr_func=runif, min=17, ma
x=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=F
ALSE))

    # 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 Interv
als")
    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_h
igh[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])
}

# 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)
#Gitub code end Here
```

(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)*

```
visualize_sample_ci(num_samples = 100, sample_size = 100, pop_size=1000
0,
                    distr_func=rnorm, mean=20, sd=3)
```



(a-i)

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

Ans: we expect (1-0.95)*100 = 5 not include in this 95% C.I.

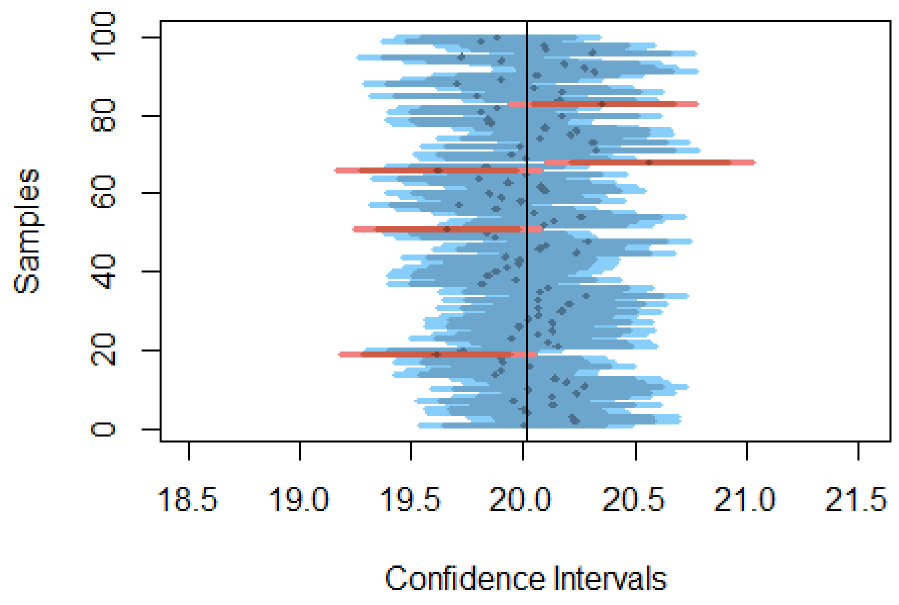
(a-ii) How many samples do we expect to NOT include the population mean in their 99% CI?

(a-ii)-Ans We expected (1-0.99)*100 = 1 not include in this 99% C.I.

(b)-(i)

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

```
visualize_sample_ci(num_samples = 100, sample_size = 300, pop_size=1000
0,distr_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?

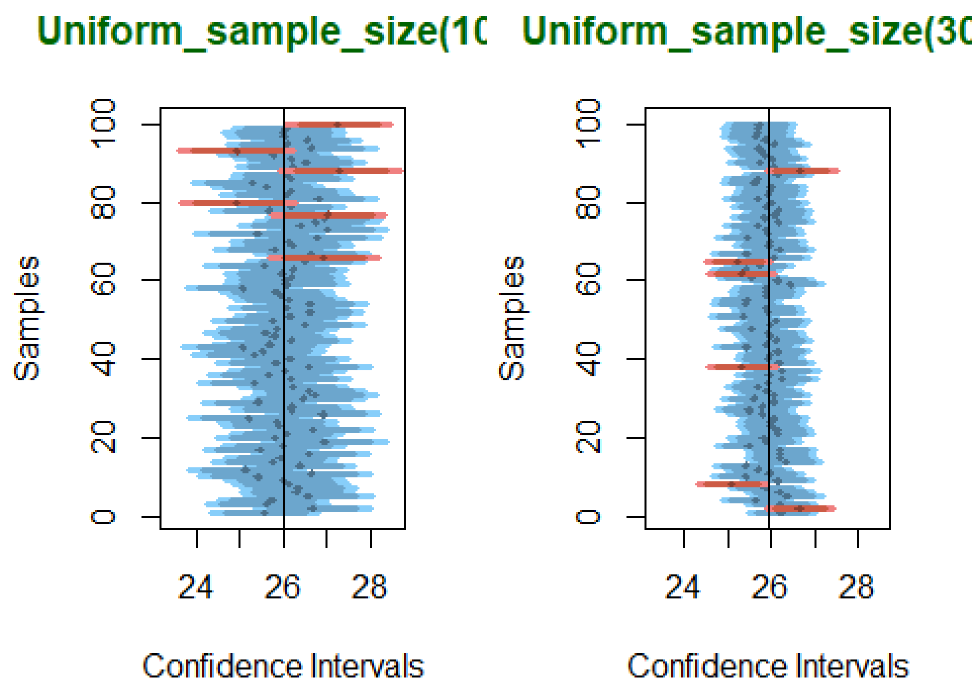
(i)-Ans Narrower than before.

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

(ii) -Ans: This times, we expected (1-0.95)*300 = 15 not include in this 95% C.I.

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

```
par(mfrow = c(1,2))
visualize_sample_ci(sample_size =100, distr_func=runif,min = 17, max =
35)+
    title(main = "Uniform_sample_size(100)",col.main = "darkgreen")

## integer(0)

visualize_sample_ci(sample_size = 300,distr_func=runif,min = 17, max =
35)+
title(main = "Uniform_sample_size(300)",col.main = "darkgreen")
```

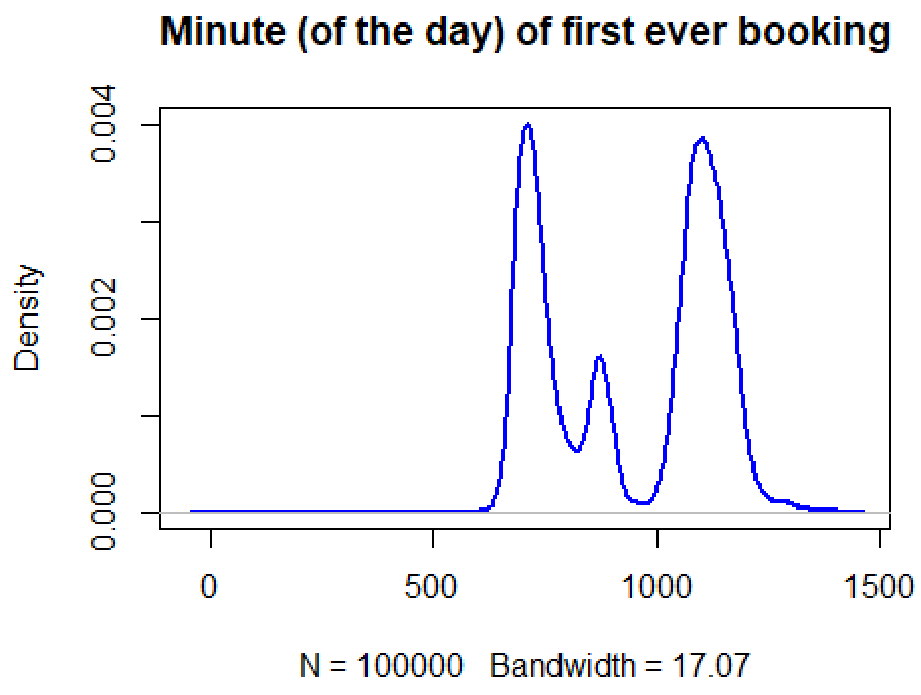


```
## integer(0)
```

Ans: (c) As the increasing of sample size, no matter whether it's a normal distribution or a uniform distribution,the 99% and 95% CI still get narrower. To make it rolling, I set min = 17, and max = 35 as provided for the uniformly distributed population. The mean of the distribution is (17+35)/2 = 26, standard deviation = (35-17)**2/12=27. The calculation of 95% CI and 99% CI is still. Therefore, (a), (b) answer in this scenario does not change.

##(Question 3) ### 3-(a) What is the "average" booking time for new members making their first restaurant booking?

```
bookings <- read.table("first_bookings_datetime_sample.txt", header=TRU
E)
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

par(mfrow = c(1,1))
plot(density(minday), main="Minute (of the day) of first ever booking",
 col="blue", lwd=2)
```



```
mean(minday)

## [1] 942.4964

#Ans  942.4964
```

(3-a-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

```
#Estimate μ of minday via sample mean
x <- mean(minday);x  #sample mean

## [1] 942.4964

sprintf("The sample mean is  %f ,which can be used to estimate the popu
lation mean", x)

## [1] "The sample mean is  942.496350 ,which can be used to estimate t
he population mean"
```

sample standard error & the 95% confidence interval (CI) of the sampling means(s)*

```
n <-length(minday)
SE <- sd(minday)/sqrt(n)
sprintf("The sample Standard error: is  %f ", SE)

## [1] "The sample Standard error: is  0.599767 "
```

Formula: 95% CI

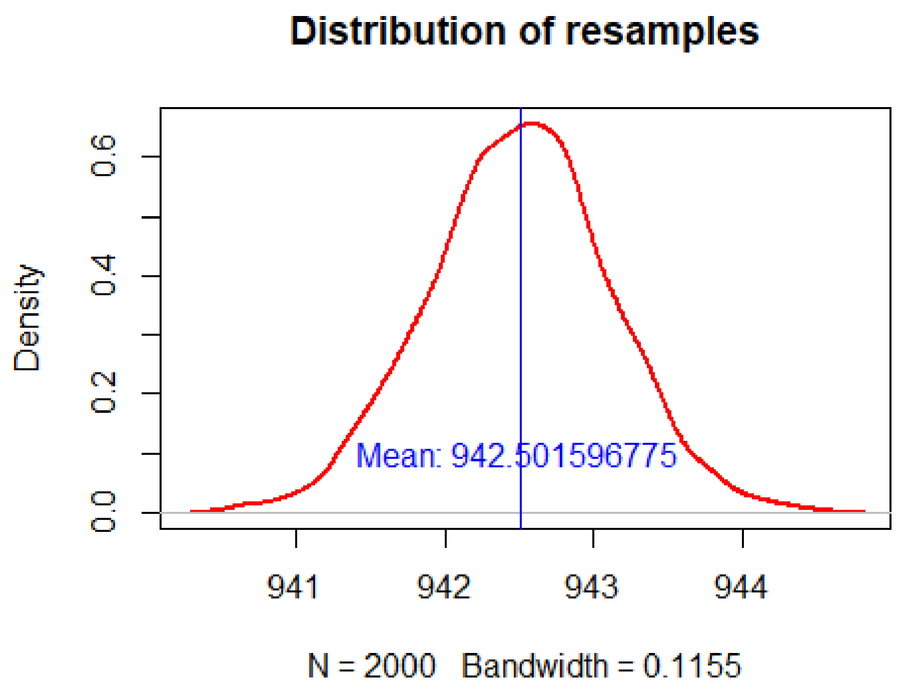
```r
z <- c(-1.96,1.96)
CI <- x+z*(SE)
sprintf("CI of sample mean: is  [%f,%f] ", CI[1], CI[2])

## [1] "CI of sample mean: is  [941.320806,943.671894] "
```

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

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

resamples <- replicate(2000,compute_sample_mean(minday))
```

(3-iii) Visualize the means of the 2000 bootstrapped samples

```r
head(resamples)

## [1] 942.5064 941.2403 942.3968 942.2870 941.8269 942.8246

plot(density(resamples), lwd=2,
     main="Distribution of resamples",  col = 'red')
abline(v = mean(resamples), lwd = 1, col = 'blue')

plot_resample_density <- function(sample_i) {
    lines(density(sample_i), col=rgb(0.0, 0.35, 0.05, 0.1))
    return(mean(sample_i))
}
m_text <- paste('Mean:', mean(resamples))
text(x = 942.5, y =0.1 , m_text, col = "blue", cex = 1)
```



### (3-a-iv)

Estimate the 95% CI of the bootstrapped means.

```
x_ <- mean(resamples);
n_boot <- length(resamples);
s_boot <- sd(resamples)/sqrt(n_boot)
z_boot <- c(-1.96, 1.96)
CI_boot <- x_+(z_boot*s_boot)
sprintf("The 95 percent CI of the bootstrapped means is [ %f, %f ] ",CI
_boot[1], CI_boot[2])

## [1] "The 95 percent CI of the bootstrapped means is [ 942.475090, 94
2.528104 ] "
```

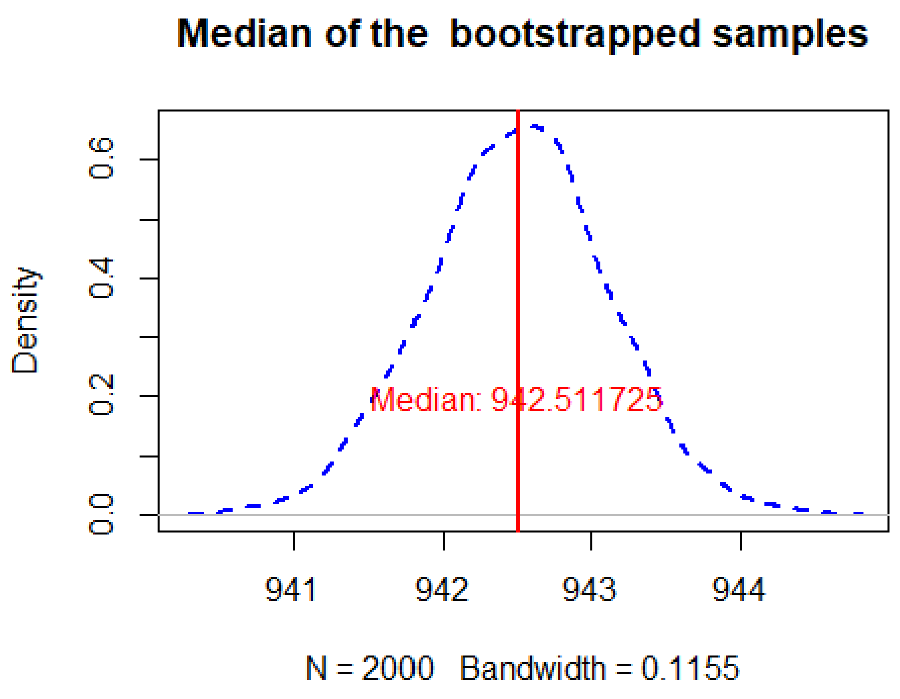
3-b) By what time of day, have half the new members of the day already arrived at their restaurant?

3-b-i) Estimate the median of minday

```
# median(minday)#[1] 1040 this value is incorrect.
wilcox.test(minday, alternative = "two.sided", correct = TRUE, conf.int
 = TRUE,conf.level = .95)

##
##  Wilcoxon signed rank test with continuity correction
##
## data:  minday
## V = 4999450015, p-value < 2.2e-16
## alternative hypothesis: true location is not equal to 0
## 95 percent confidence interval:
##   930 930
## sample estimates:
## (pseudo)median
##            930

#correct = true means doing the continually recorrection
```

(3-b-ii) Visualize the medians of the 2000 bootstrapped samples

```
# Plot population and original sample densities
plot(density(resamples),main = "Median of the  bootstrapped samples",
     col="blue", lty="dashed",lwd=2.25)
abline(v=median(resamples), lwd=2.5, col = 'red')

md_text <- paste('Median:', median(resamples))
text(x = mean(resamples), y =0.2 , md_text, col = "red", cex =1)
```



### (3-b-iii)

Estimate the 95% CI of the bootstrapped medians.

```
wilcox.test(resamples, alternative = "two.sided", correct = TRUE, conf.
int = TRUE,conf.level = .95)

##
##  Wilcoxon signed rank test with continuity correction
##
## data:  resamples
## V = 2001000, p-value < 2.2e-16
## alternative hypothesis: true location is not equal to 0
## 95 percent confidence interval:
##  942.4777 942.5313
## sample estimates:
## (pseudo)median
##       942.5046
```