Homework4_BACS

109090035 , helped by 109090023 , 109060082, 109090046

3/9/2023

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

Solution

```
# Create a normal distribution with mean=940 and sd=190
set.seed(123) # for reproducibility (number 123 can be changed)
rnorm_orig <- rnorm(n = 1000, mean = 940, sd = 190)

# Standardize the distribution
rnorm_std <- (rnorm_orig - mean(rnorm_orig)) / sd(rnorm_orig)

# Check the mean and sd of the standardized distribution
mean(rnorm_std)
```

```
## [1] 9.220402e-17
```

```
sd(rnorm_std)
```

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

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

Solution

We should expect the mean of rnorm_std to be 0 and the standard deviation to be 1, because that is what standardizing a normal distribution does.

When we standardize a normal distribution, we transform it into a new distribution that has a mean of 0 and a standard deviation of 1. This new distribution is called the standard normal distribution.

The formula for standardizing a normal distribution is:

$z = (x - \mu) / \sigma$

where z is the standardized value, x is the original value, μ is the mean of the original distribution, and σ is the standard deviation of the original distribution.

By applying this formula to each value in rnorm_orig, we are effectively transforming it into a standard normal distribution, which has a mean of 0 and a standard deviation of 1. Therefore, we should expect the mean of rnorm_std to be 0 and the standard deviation to be 1.

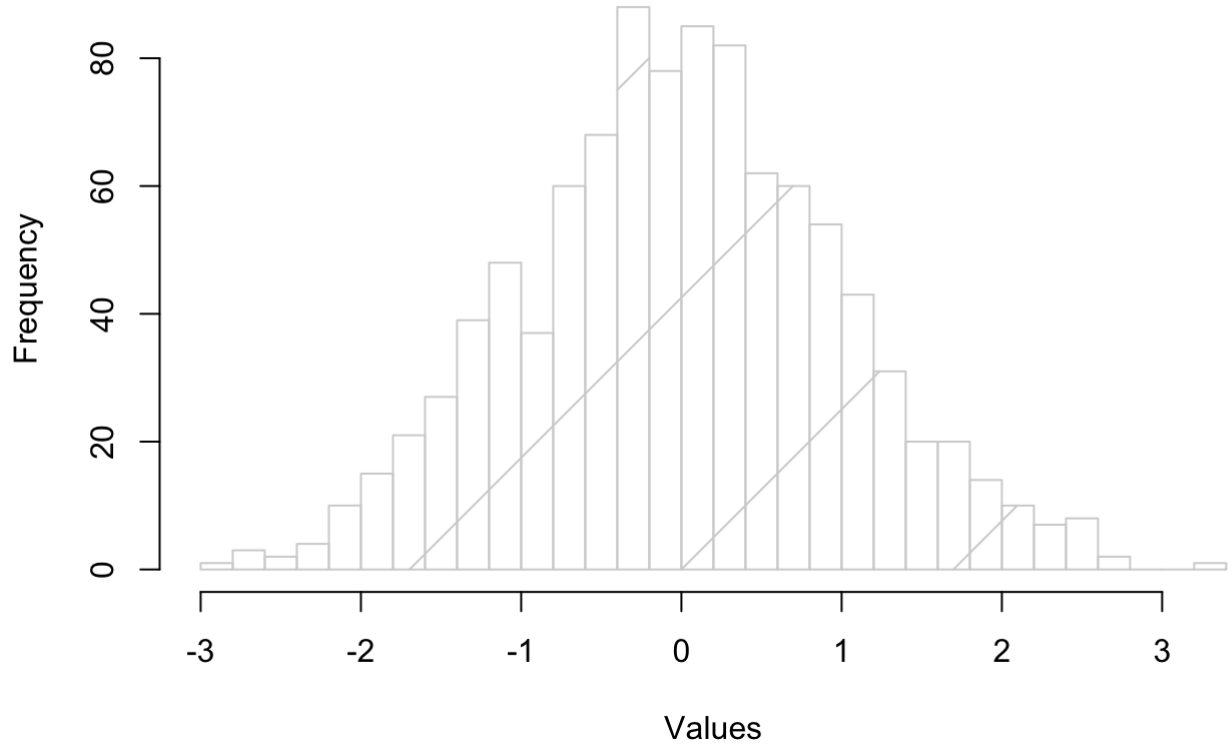
ii) What should the distribution (shape) of rnorm_std look like, and why?

Solution

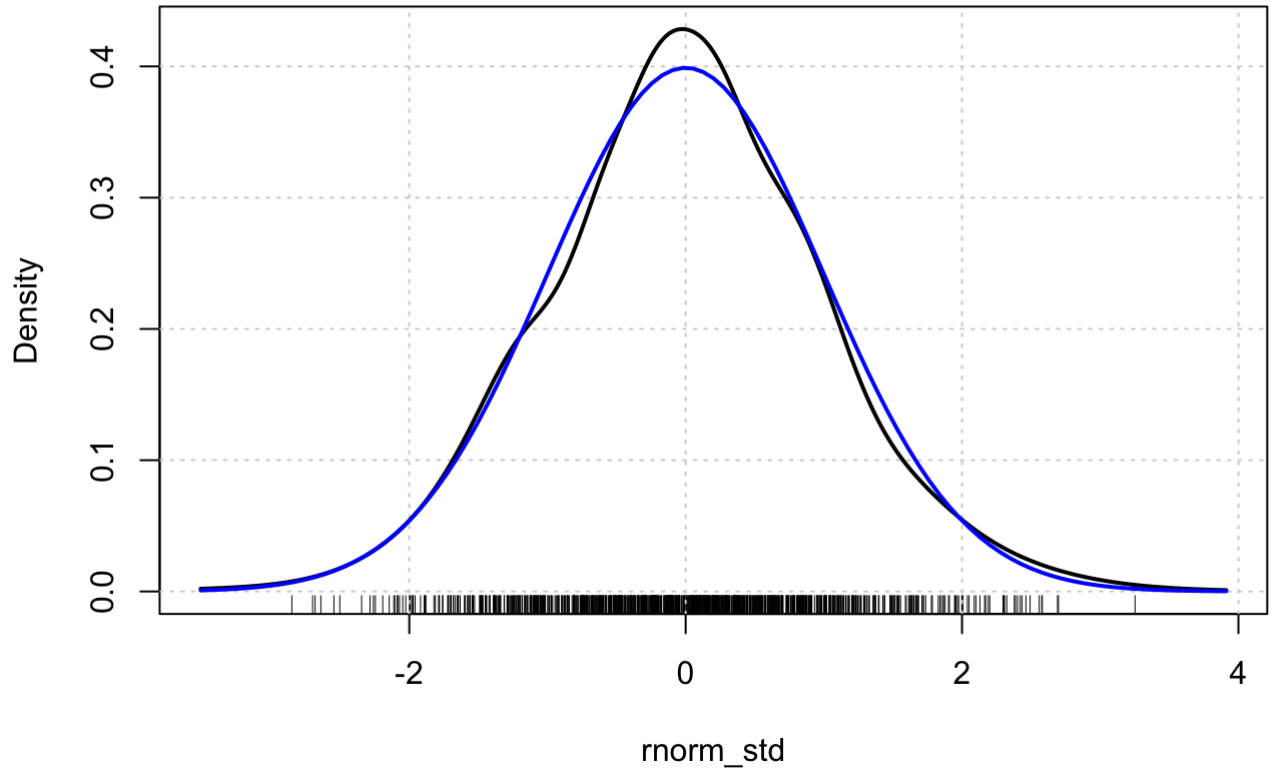
```
library(car)
```

```
## Loading required package: carData
```

```
# Plot the density histogram of rnorm_std
hist(rnorm_std, density = TRUE, breaks = 30, main = "Standard Normal Distribution", xlab = "Values")
```




```
densityPlot(rnorm_std)
# Add a blue line for the standard normal distribution density curve
curve(dnorm(x, mean = 0, sd = 1), add = TRUE, col = "blue", lwd = 2)
```



The distribution (shape) of rnorm_std should be a standard normal distribution, which is a bell-shaped curve centered at 0 and with a standard deviation of 1. This is because rnorm_std is the result of standardizing a normal distribution with mean 940 and standard deviation 190.

Standardizing a normal distribution involves subtracting the mean of the original distribution and dividing by its standard deviation, which effectively centers the distribution at 0 and scales it to have a standard deviation of 1. This transformation does not change the shape of the distribution, only its location and scale.

Therefore, since we have standardized a normal distribution to create rnorm_std, we should expect its shape to be a standard normal distribution, which is a symmetrical bell-shaped curve. The distribution should have a mean of 0 and a standard deviation of 1, which reflects the centering and scaling of the original distribution.

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

Solution

Distributions that are normal and standardized are generally called standard normal distributions or standard Gaussian distributions. These terms are often used interchangeably.

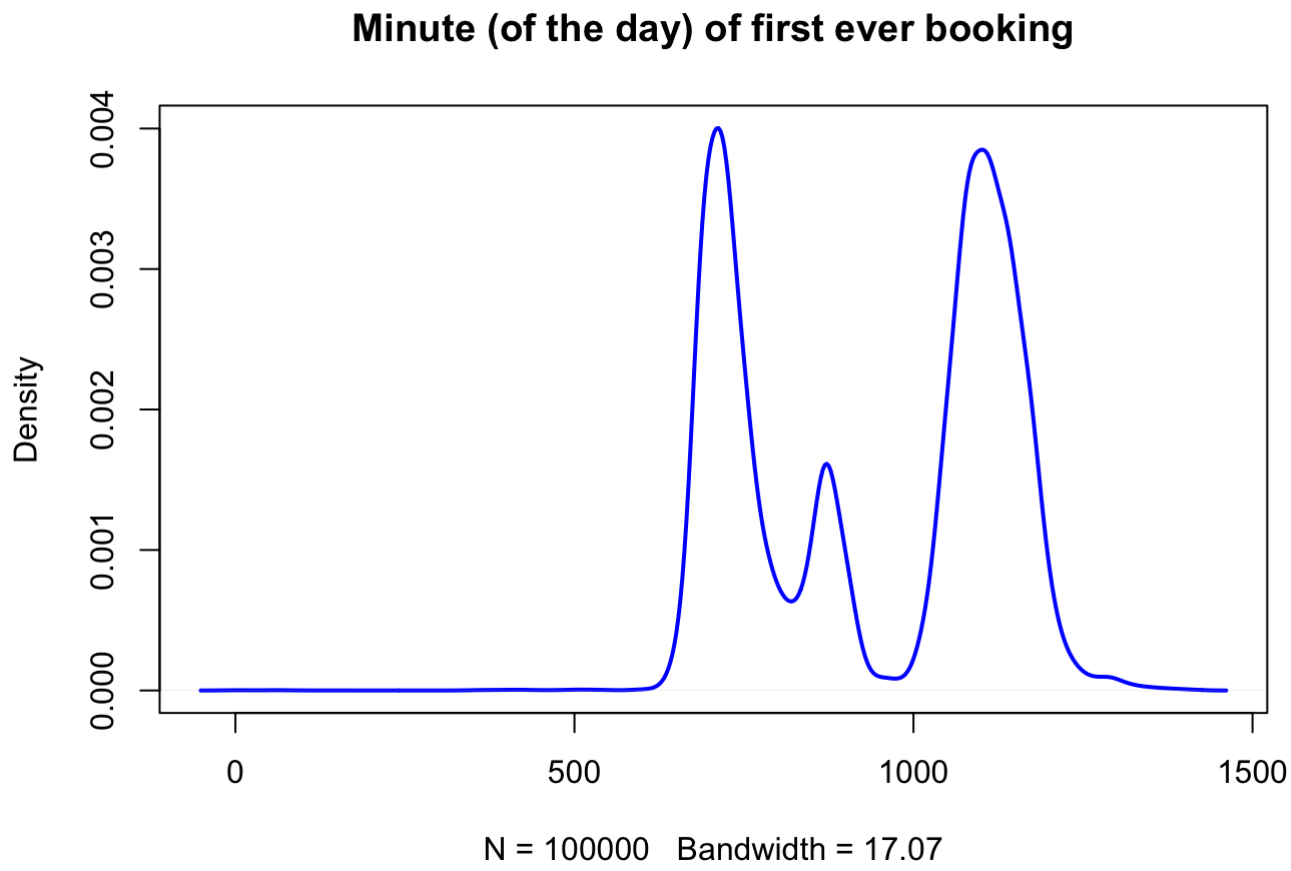
A standard normal distribution is a normal distribution with a mean of 0 and a standard deviation of 1. It is an important distribution in statistics because it is a reference distribution that allows us to compare other normal distributions to a common standard. The standard normal distribution is often used in hypothesis testing, confidence interval estimation, and other statistical analyses.

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=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)
```



Solution

```
# Create a standardized version of minday
minday_std <- (minday - mean(minday)) / sd(minday)

# Print the first six values of minday_std
head(minday_std)
```

```
## [1]  0.5668138  1.3576899 -1.1731137 -1.1731137  0.7249890 -0.2240623
```

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

Solution

The standardized minday variable minday_std should have a mean of 0 and a standard deviation of 1. This is because standardization transforms the original data into a new variable where the mean is centered at 0 and the standard deviation is scaled to 1.

By subtracting the mean from each observation and then dividing by the standard deviation. Since the mean of minday was approximately 956 and the standard deviation was approximately 130, standardizing this variable should result in a new variable with a mean of 0 and a standard deviation of 1.

We can check this by using the mean() and sd() functions in R to calculate the mean and standard deviation of minday_std, as shown in the following code:

```
# Calculate the mean and standard deviation of minday_std
mean(minday_std)
```

```
## [1] 2.302986e-15
```

```
sd(minday_std)
```

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

If minday_std was created correctly, the output of these commands should be approximately 0 and 1, respectively.

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

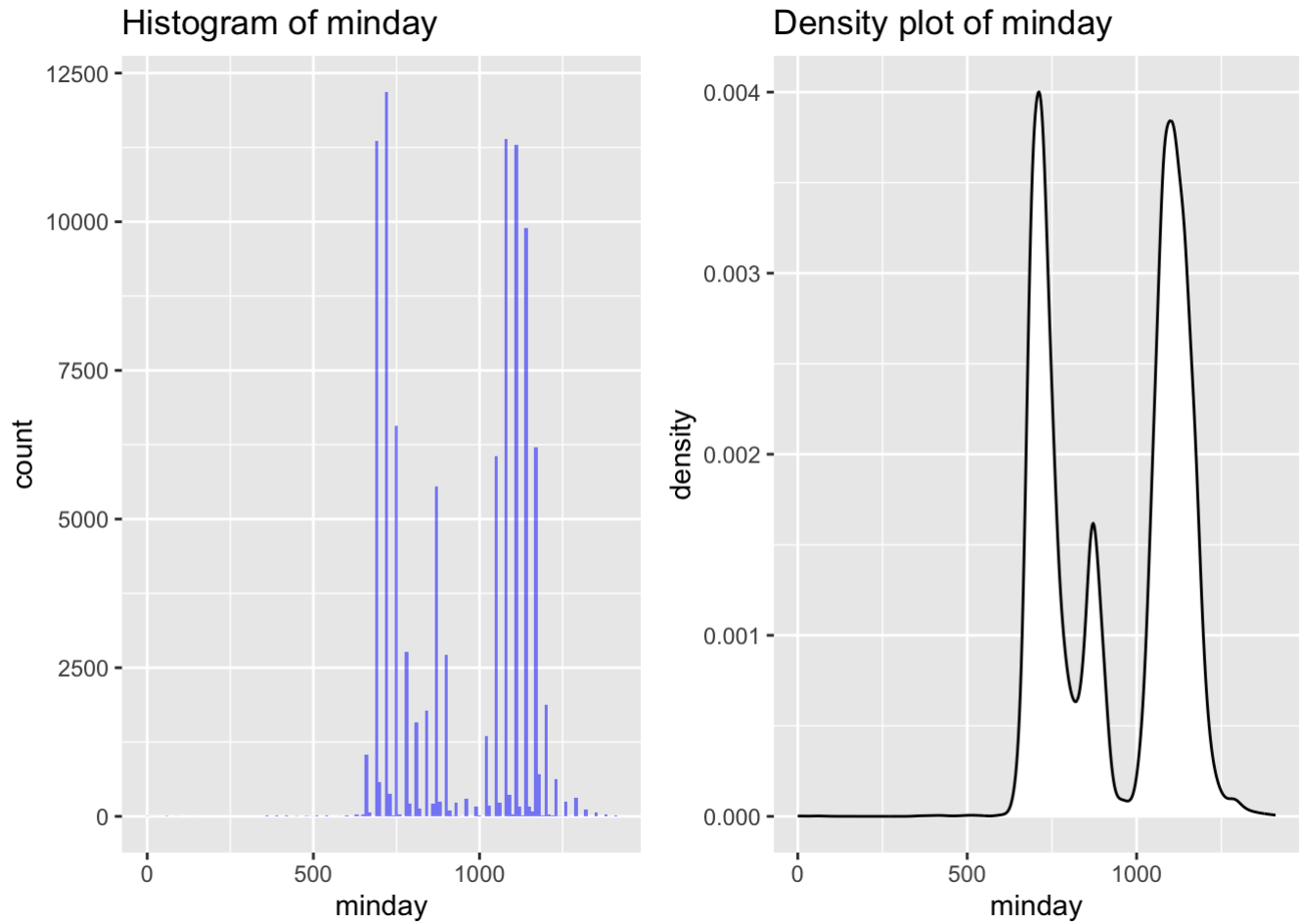
Solution

The distribution of minday_std should look like a normal distribution with a mean of 0 and a standard deviation of 1. This is because standardization transforms the original data into a new variable with a known mean and standard deviation, which results in a specific shape of the distribution.

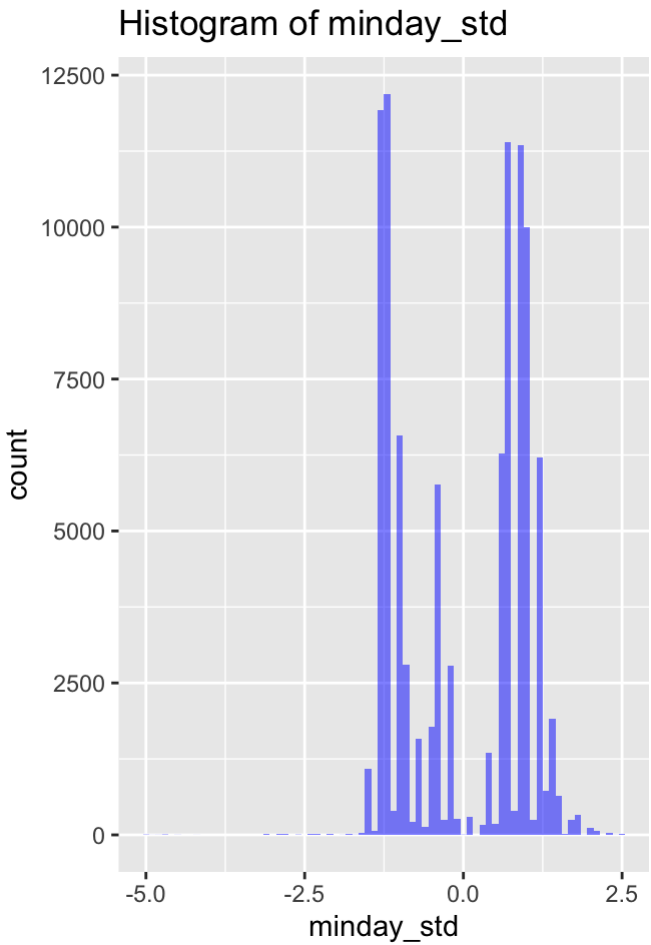
In contrast, the distribution of minday may or may not be normal. It depends on the data and the underlying process that generated it. If the data is normally distributed, then minday may also have a normal distribution. However, if the data is skewed or has outliers, the distribution of minday may be non-normal.

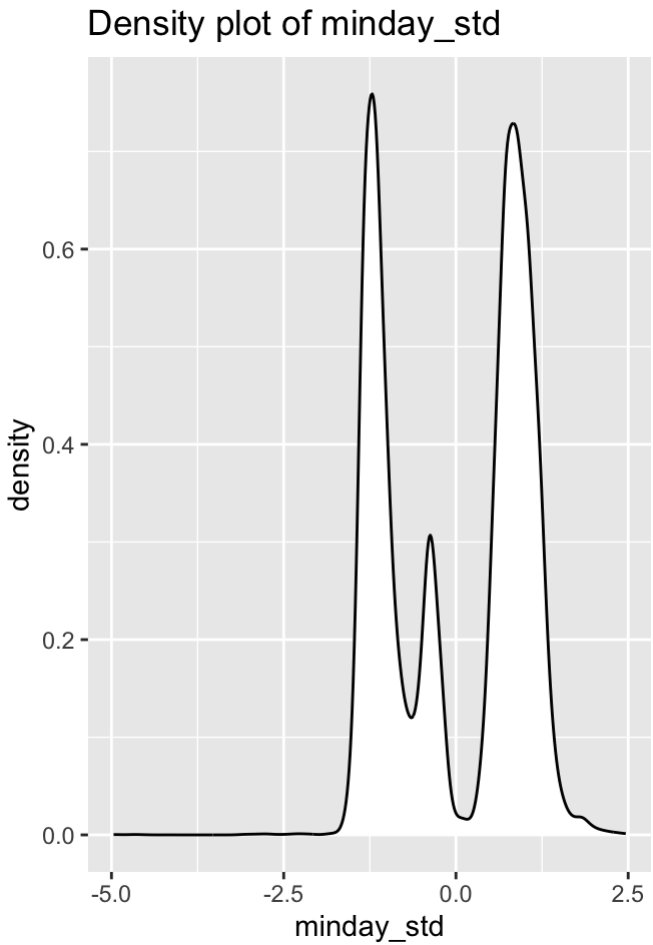
To see the difference between the distribution of minday and minday_std, we can create histograms of both variables using the ggplot2 package in R, as shown in the following code:

```
library(ggplot2)
library(gridExtra)
# Histogram of minday
p1 <- ggplot(data = bookings, aes(x = minday)) +
  geom_histogram(binwidth = 10, fill = "blue", alpha = 0.5) +
  ggtitle("Histogram of minday")

# create density plot
p2 <- ggplot(data = bookings, aes(x = minday)) +
  geom_density(color = "black", fill = "white") +
  labs(title = "Density plot of minday")

# Histogram of minday_std
p3 <- ggplot(data = bookings, aes(x = minday_std)) +
  geom_histogram(binwidth = 0.1, fill = "blue", alpha = 0.5) +
  ggtitle("Histogram of minday_std")

# create density plot
p4 <- ggplot(data = bookings, aes(x = minday_std)) +
  geom_density(color = "black", fill = "white") +
  labs(title = "Density plot of minday_std")

grid.arrange(p1, p2,  ncol = 2)
```




```
grid.arrange(p3, p4,  ncol = 2)
```

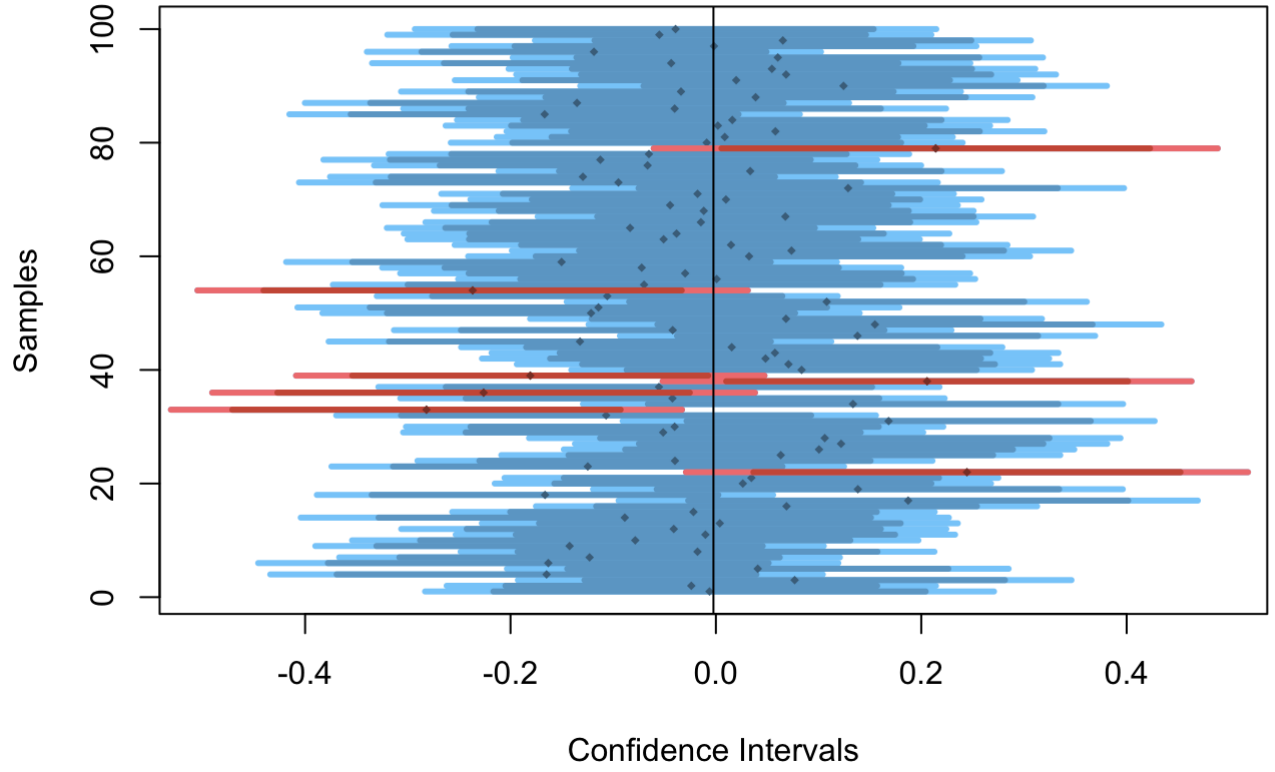

Histogram of minday_std
count
12500
10000
7500
5000
2500
0
-5.0
-2.5
0.0
2.5
minday_std


Density plot of minday_std
density
0.6
0.4
0.2
0.0
-5.0
-2.5
0.0
2.5
minday_std

Question 2) Install the compstatslib package from Github (see class notes) and run the plot_sample_ci() function that simulates samples drawn randomly from a population.

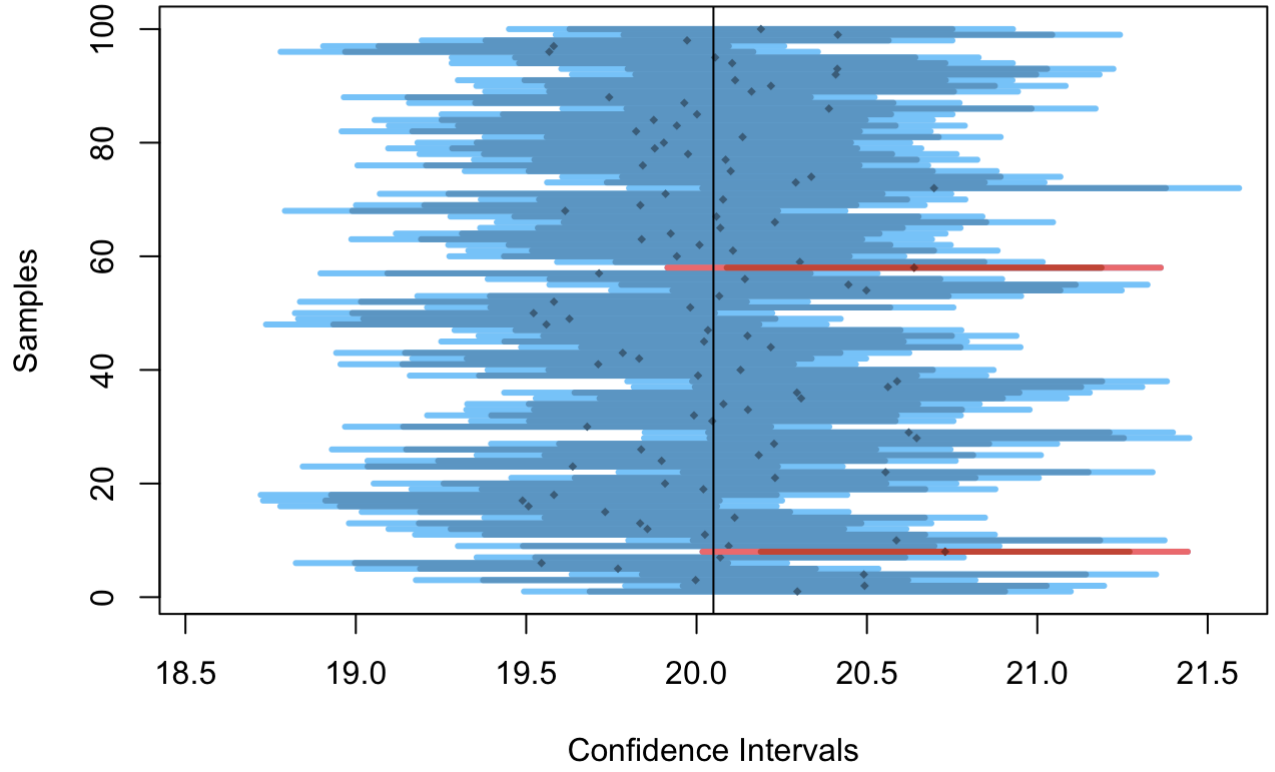
Solution

```
library(compstatslib)

set.seed(123)
plot_sample_ci()
```



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

```
plot_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?

Solution

we would expect about 5% of the samples to not include the population mean in its 95% CI.

With 100 samples, we can expect approximately 5% of them, or 5 samples, to not include the population mean in its 95% CI. However, this is only an expected value, and the actual number of samples that do not include the population mean may vary in any particular simulation.

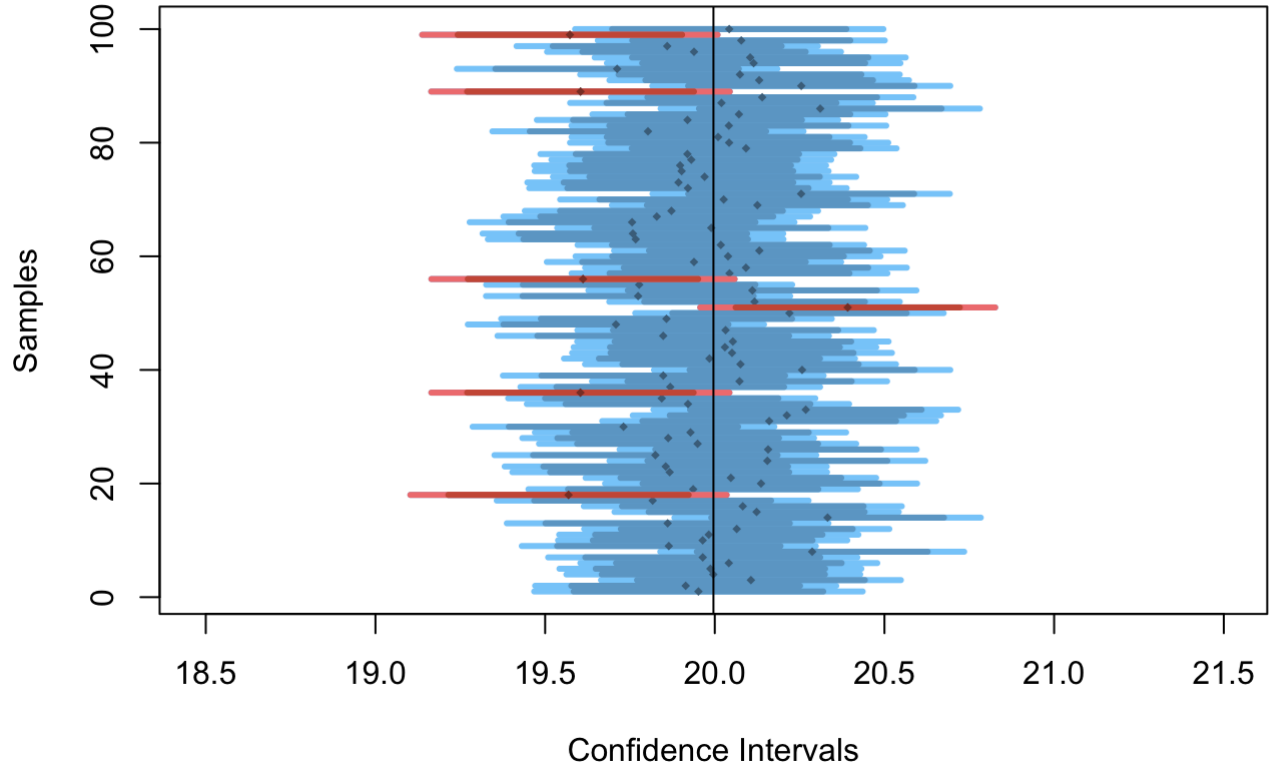
ii) How many samples do we expect to NOT include the population mean in their 99% CI?

Same as below, we would expect about 1% of the samples to not include the population mean in its 99% CI.

With 100 samples, we can expect approximately 1% of them, or 1 samples, to not include the population mean in its 99% CI. However, this is only an expected value, and the actual number of samples that do not include the population mean may vary in any particular simulation.

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

```
plot_sample_ci(num_samples = 100, sample_size = 300, pop_size=10000,
               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?

Solution

If we rerun the previous simulation with a larger sample size of 300, we expect the 95% and 99% confidence intervals to become narrower than before. This is because as the sample size increases, the standard error of the mean decreases, which in turn leads to narrower confidence intervals.

We can compare the widths of the 95% and 99% confidence intervals with the previous simulation with sample size 100 to confirm that the intervals have become narrower with the larger sample size.

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

Solution

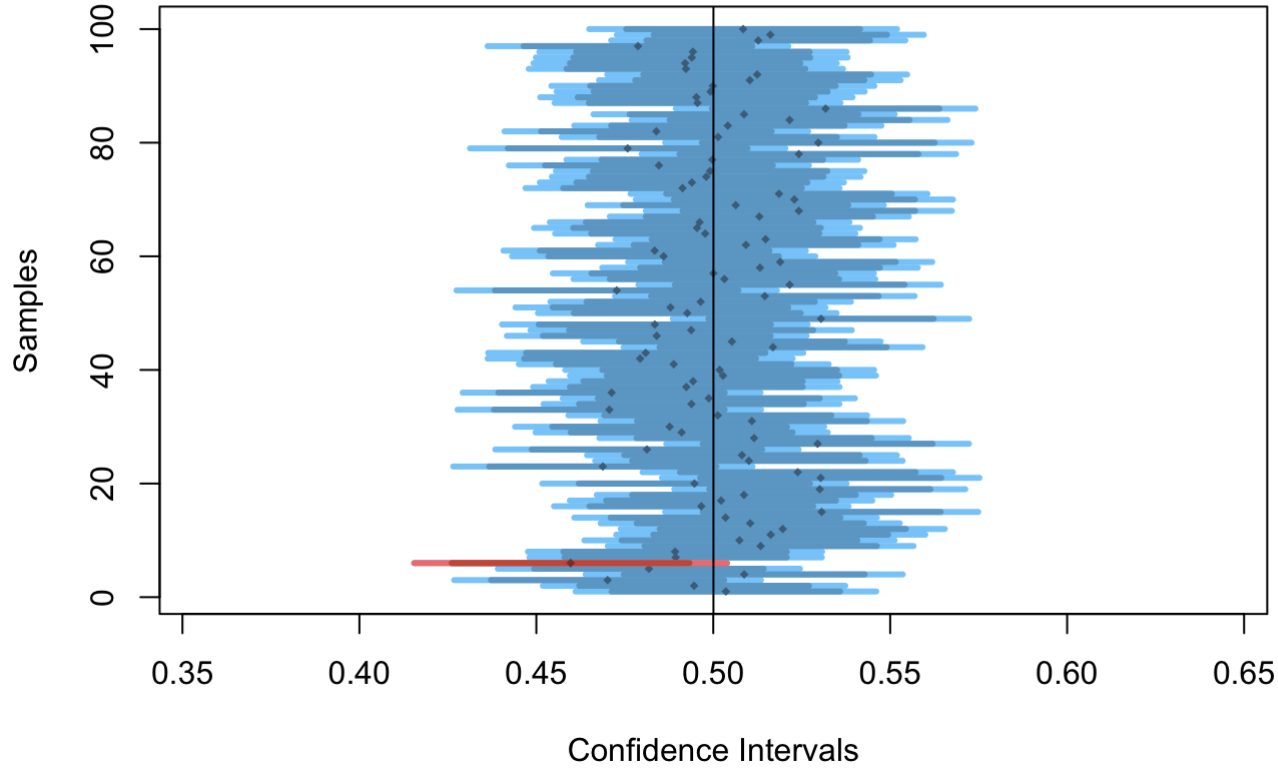
If we increase the sample size to 300, we would expect a smaller proportion of samples to not include the population mean in its 95% CI, compared to the previous simulation with sample size of 100. However, the exact number of samples that would not include the population mean in its 95% CI would depend on the specific random samples drawn from the population.

we would expect about 5% of the samples to not include the population mean in its 95% CI.

With 300 samples, we can expect approximately 5% of them, or 15 samples, to not include the population mean in its 95% CI. However, this is only an expected value, and the actual number of samples that do not include the population mean may vary in any particular simulation.

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

```
plot_sample_ci(num_samples = 100, sample_size = 300, pop_size=10000,distr_func=runif)
```



Solution

If we use a uniformly distributed population instead of a normally distributed population, we would expect the following changes:

a. The standard deviation of the population would be smaller than in the normal case, since the range of the uniform distribution is limited by its minimum and maximum values. As a result, the distribution of sample means would have a smaller standard error, leading to narrower confidence intervals for the same sample size and confidence level. Therefore, we would expect a smaller margin of error for the mean in the 95% and 99% confidence intervals compared to the normal case.

b. Since the population distribution is no longer normal, the distribution of sample means would not be normal either. Specifically, the distribution of sample means would follow a central limit theorem approximation to a normal distribution only if the sample size is large enough. For smaller sample sizes, the distribution of sample means would be skewed and have heavier tails than a normal distribution.

Therefore, we would expect a higher proportion of samples to not include the population mean in its 95% confidence interval, especially for small sample sizes.

Question 3a) 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

Solution

To estimate the population mean of minday using traditional statistical methods, we can calculate the sample mean and use it as an estimate for the population mean. We can also calculate the standard error of the mean (SEM), which is the standard deviation of the sampling distribution of the mean and is equal to the standard deviation of the population divided by the square root of the sample size. Finally, we can use the t-distribution to calculate the 95% confidence interval (CI) of the sampling means, which gives us a range of values within which we are 95% confident the true population mean lies

```
# Calculate the sample mean and SEM
n <- length(minday)
mean_minday <- mean(minday)
sd_minday <- sd(minday)
sem_minday <- sd(minday)/sqrt(n)

t_val <- qt(0.975, df = n -1) # t-value for 95% CI with n length degrees of freedom
ci_lower <- mean_minday  - t_val * sem_minday
ci_upper <- mean_minday  + t_val * sem_minday

# Print the estimated 95% CI
cat("The 95% CI of the minday is [", ci_lower, ",", ci_upper, "]\n")
```

```
## The 95% CI of the minday is [ 941.3208 , 943.6719 ]
```

```
# Print the results
cat("Sample mean: ", mean_minday, "\n")
```

```
## Sample mean:  942.4963
```

```
#SD is the standard deviation
cat("SD: ", sd_minday, "\n")
```

```
## SD:  189.6631
```

```
#SEM is the standard error of the mean
cat("SEM: ", sem_minday, "\n")
```

```
## SEM:  0.5997673
```

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

Solution

```
set.seed(123) # for reproducibility
# Number of bootstrap samples to draw
n_boot <- 2000
# Initialize vector to store bootstrap sample means
boot_means <- numeric(n_boot)
boot_medians <- numeric(n_boot)
# Perform the bootstrap
for (i in 1:n_boot) {
  # Draw a bootstrap sample with replacement
  boot_sample <- sample(minday, size = length(minday), replace = TRUE)
  # Calculate the mean of the bootstrap sample
  boot_mean <- mean(boot_sample)
  boot_median <- median(boot_sample)
  # Store the mean in the boot_means vector
  boot_means[i] <- boot_mean
  boot_medians[i] <- boot_median
}
```

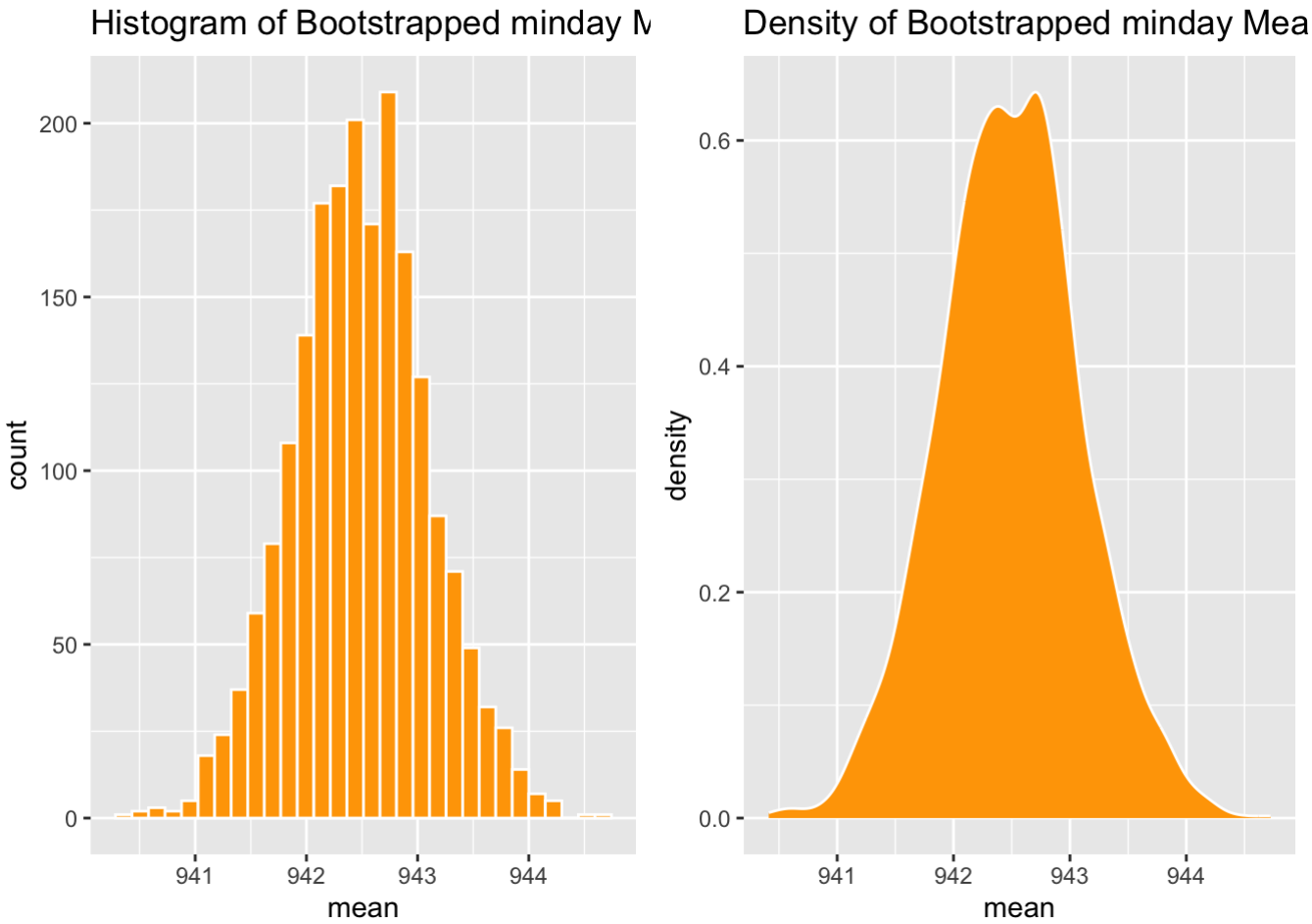
iii) Visualize the means of the 2000 bootstrapped samples

Solution

```
library(ggplot2)

# Create a data frame of the means
means_df <- data.frame(mean = boot_means)

# Create a histogram of the means
p1 <- ggplot(means_df, aes(x = mean)) +
  geom_histogram(bins = 30, fill = "orange", color = "white") +
  ggtitle("Histogram of Bootstrapped minday Means")

p2 <-ggplot(means_df, aes(x = mean)) +
  geom_density(bins = 30, fill = "orange", color = "white") +
  ggtitle("Density of Bootstrapped minday Means")
```

```
## Warning in geom_density(bins = 30, fill = "orange", color = "white"): Ignoring
## unknown parameters: `bins`
```

```
grid.arrange(p1,p2,ncol = 2 )
```



iv) Estimate the 95% CI of the bootstrapped means using the quantile function

Solution

To estimate the 95% confidence interval (CI) of the bootstrapped means using the quantile function, we can simply calculate the 2.5th and 97.5th percentiles of the means of the 2000 bootstrapped samples. This will give us a range of values within which we can be 95% confident that the true population mean falls

```
# calculate 95% CI of bootstrapped means using quantile function
ci <- quantile(boot_means, c(0.025, 0.975))

# print CI
cat("95% CI of bootstrapped means: [", round(ci[1], 2), ", ", round(ci[2], 2), "]", sep="")
```

```
## 95% CI of bootstrapped means: [941.3, 943.73]
```

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

Solution

```
print(median(minday))
```

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

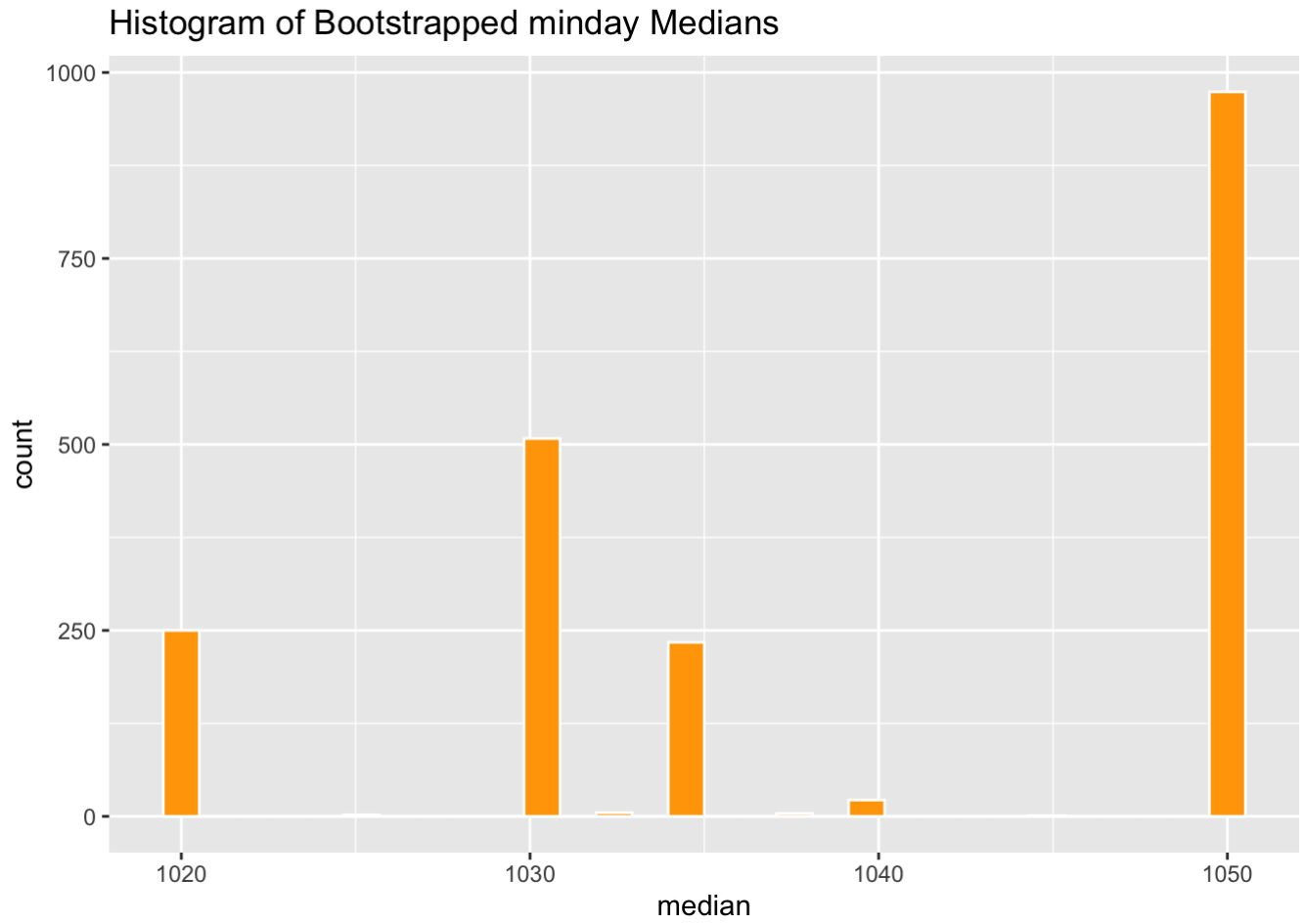
ii) Visualize the medians of the 2000 bootstrapped samples

Solution

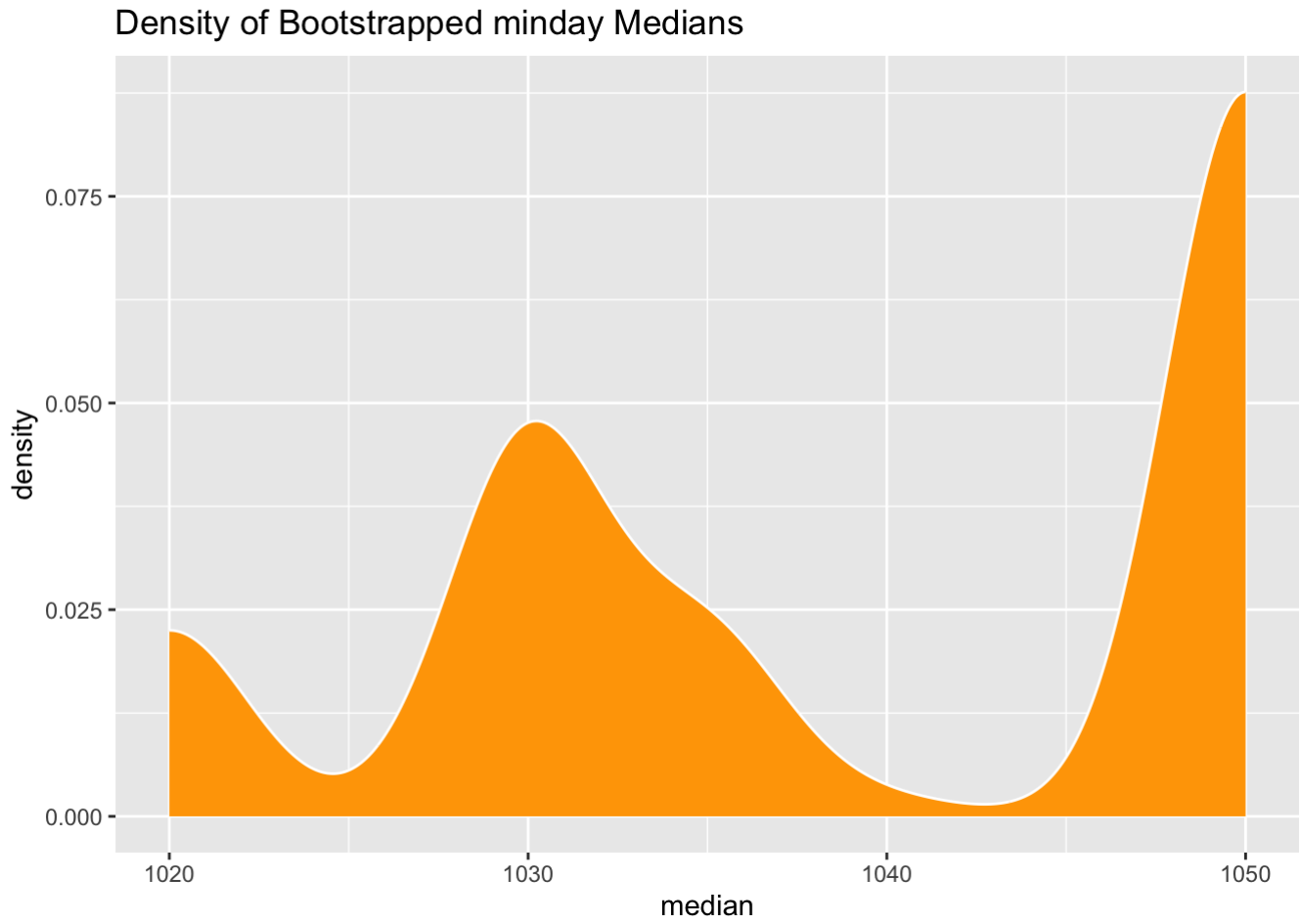
```
# Create a data frame of the means
median_df <- data.frame(median = boot_medians)

# Create a histogram of the means
p1 <- ggplot(median_df, aes(x = median)) +
  geom_histogram(bins = 30, fill = "orange", color = "white") +
  ggtitle("Histogram of Bootstrapped minday Medians")

p2 <- ggplot(median_df, aes(x = median)) +
  geom_density(bins = 30, fill = "orange", color = "white") +
  ggtitle("Density of Bootstrapped minday Medians")
```

```
## Warning in geom_density(bins = 30, fill = "orange", color = "white"): Ignoring
## unknown parameters: `bins`
```

```
p1
```


Histogram of Bootstrapped minday Medians
1000
750
500
250
0
count
1020
1030
1040
1050
median

```
p2
```



iii) Estimate the 95% CI of the bootstrapped medians using the quantile function

Solution

```
# Compute the lower and upper bounds of the 95% CI
lower_bound <- quantile(boot_medians, 0.025)
upper_bound <- quantile(boot_medians, 0.975)

# Print the estimated 95% CI
cat("The 95% CI of the bootstrapped medians is [", lower_bound, ",", upper_bound, "]\n")
```

```
## The 95% CI of the bootstrapped medians is [ 1020 , 1050 ]
```