BACS HW3

109062710

March 17, 2021

Question 1

Here is the helper functions for Q1

```
standardize <- function(data) {
  standardized <- (data - mean(data)) / sd(data)
  return(standardized)
}

create_density <- function(data, title) {
  mean <- mean(data)

  sd_values = c(
    mean(data) - 2 * sd(data),
    mean(data) - sd(data),
    mean(data) + sd(data),
    mean(data) + 2 * sd(data)
  )

  ggplot(mapping = aes(data)) +
    geom_density(
      fill="#69b3a2",
      color="#e9ecef",
    ) +
    geom_vline(xintercept = mean, col="black") +
    geom_vline(xintercept = sd_values, col="red") +
    ggtitle(title)
}

create_histogram <- function(data, title) {
  n = length(data)

  # Freidman-Darconis' BiUnwidth Rule
  binwidth <- (2 * IQR(data)) / n^(1/3)
  bins <- ceiling(max(data) - min(data)) + binwidth

  ggplot(mapping = aes(data)) +
    geom_histogram(
      fill="#69b3a2",
      color="#e9ecef",
      bins = bins,
      binwidth = binwidth
    ) +
    ggtitle(title)
}
```

A. create a normal distribution (`mean = 940, sd = 190`) and standardize it

```
rnorm <- rnorm(1000, mean = 940, sd = 190)
rnorm_std <- standardize(rnorm)
```

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

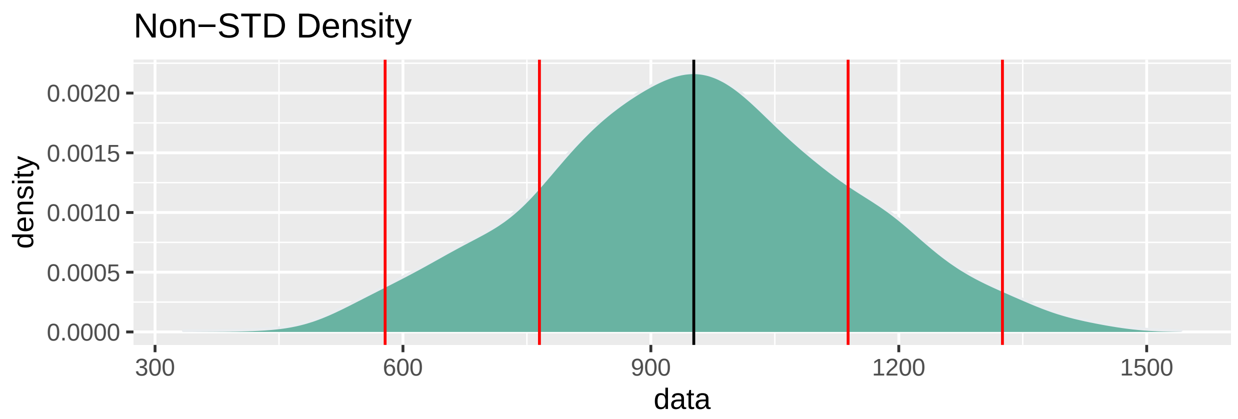
```
glue(
  "The mean of rnorm is {nonstd_rnorm_mean},
  and its standard deviation is {nonstd_rnorm_sd}."
)
```

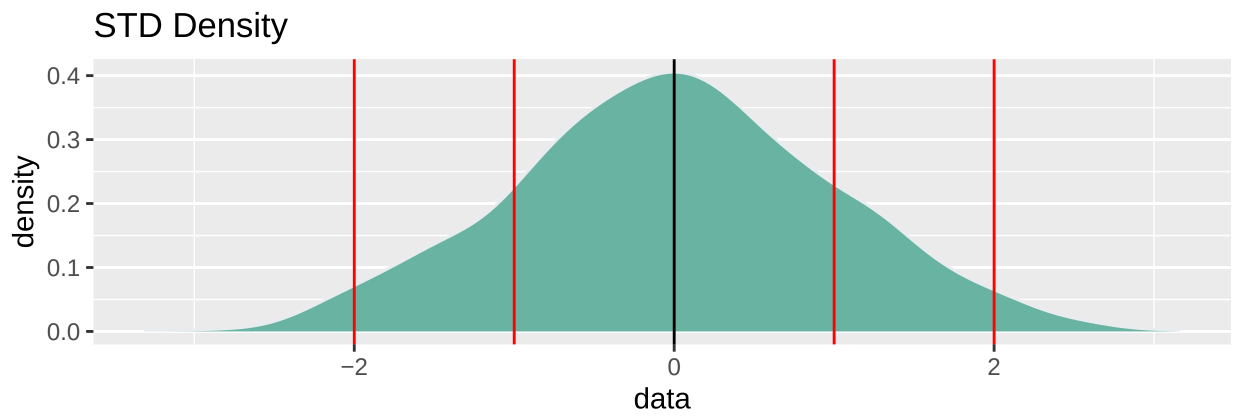
```
## The mean of rnorm is 951.925411442816,
## and its standard deviation is 186.754548334887.
```

```
glue(
  "The mean of rnorm_std is {std_rnorm_mean},
  and its standard deviation is {std_rnorm_sd}."
)
```

```
## The mean of rnorm_std is 1.20935569501365e-16,
## and its standard deviation is 1.
```

```
grid.arrange(
  rnorm_density,
  rnorm_std_density,
  ncol=1,
  nrow=2
)
```

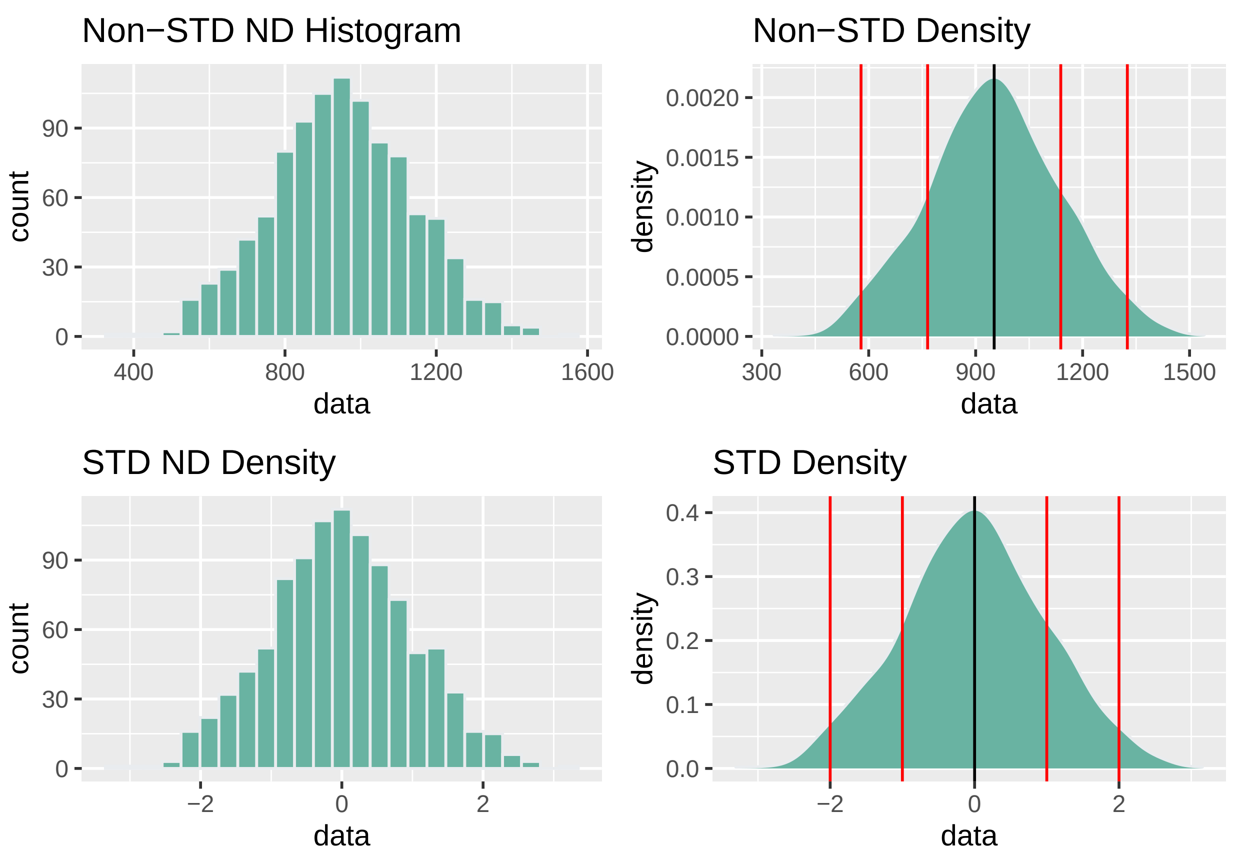




As we can see from the result above and the graph above, mean value and standard deviation value are concentrated around `-3` to `3`, instead of `0` to `1600` which before standardization. After standardization, each `x_value` in the graph represents how far each instance from the mean in STD unit. This happens because standardization scales down everything to STD unit scale.

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

```
grid.arrange(
  rnorm_hist,
  rnorm_density,
  rnorm_std_hist,
  rnorm_std_density,
  ncol=2,
  nrow=2
)
```



Basically, `rnorm_std` and `rnorm` plots should look entirely the same, but they are not. Let's take the graph above as a reference.

However, there is a worth mentioning here:

1. Non-standardized and standardized histograms look almost the same, but there is a slight difference if you take a close look.
2. The `x_values` range becomes smaller in standardized density plot because standardization scales down everything to STD unit scale.

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

It's called **bell-shaped curved** distribution.

B. Create a standardized version of `minday` from the earlier question (let's call it `minday_std`)

```r
minday_std <- standardize(minday)
```

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

```
glue("The mean of minday_std {minday_std_mean}, while its SD is {minday_std_sd}.")
```

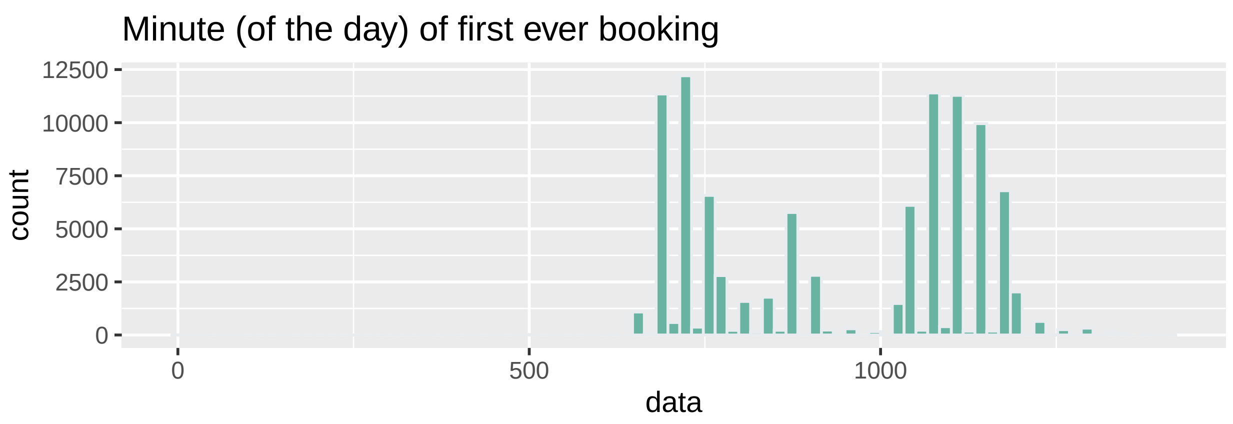
```
## The mean of minday_std -4.25589034500073e-17, while its SD is 1.
```

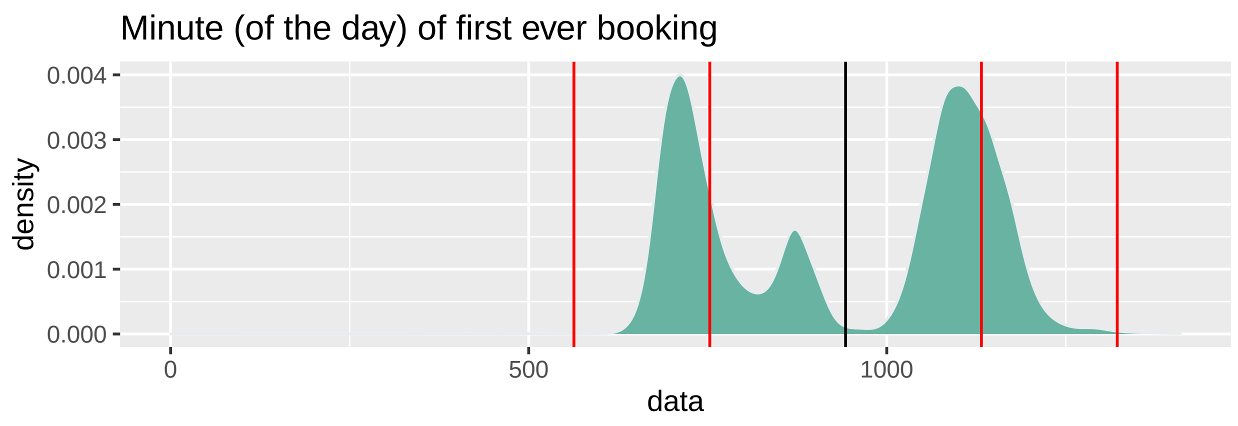
We expect the mean and the STD values to be really small which are within `-2.5` to `2.5` range after standardization because standardization scales down everything to STD unit scale.

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

Before standardization,

```
grid.arrange(
  minday_hist,
  minday_density,
  ncol=1,
  nrow=2
)
```

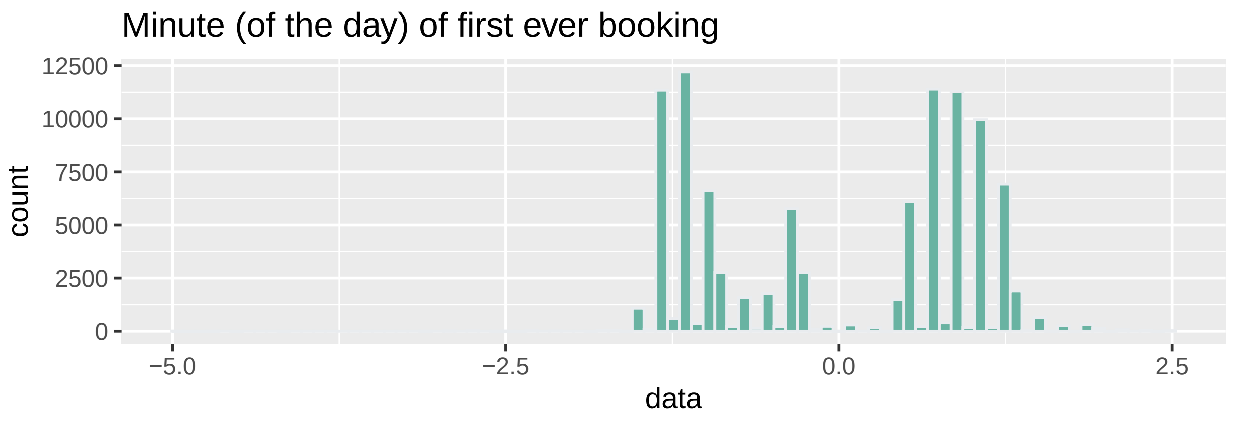


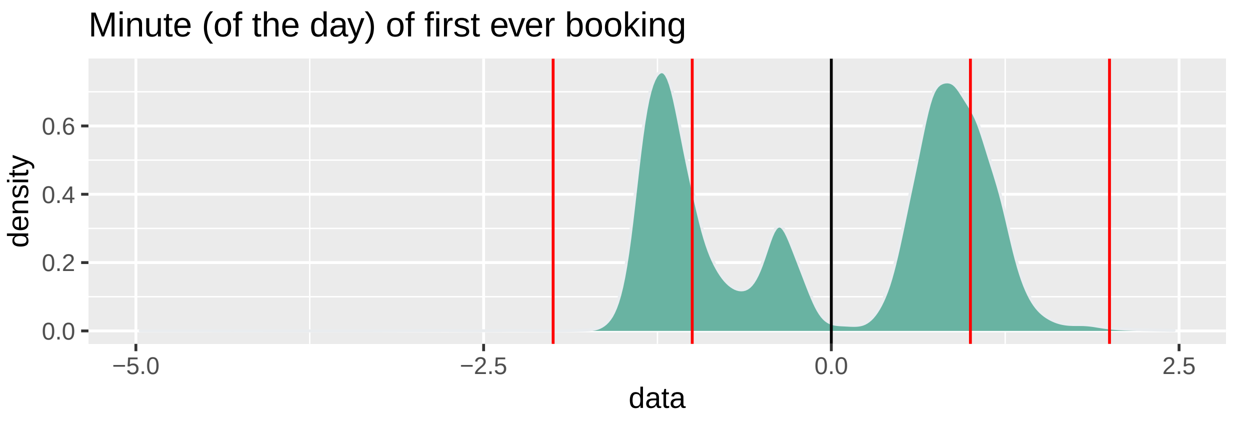


After standardization,

```
grid.arrange(
  minday_std_hist,
  minday_std_density,
```

```r
  ncol=1,
  nrow=2
)
```



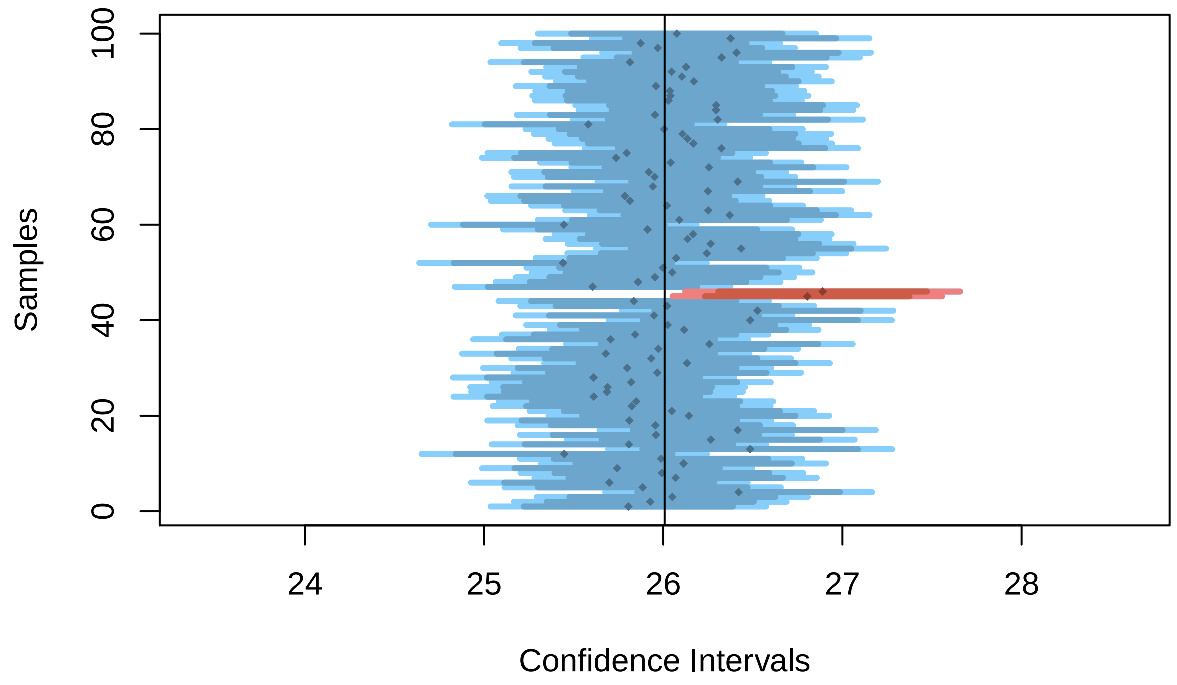


The situation is the similar to the section a, part ii. In the non-standardized data set, the STD lines are far away when we expect them to be. Besides, we have a huge range of `x_value` which is from `0` to `1500`.

However, in the standardized data set, the mean line is exactly in between the STD lines. In addition, we have a smaller range of `x_value` which is from `-4` to `4`.

Question 2

```r
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])
}

visualize_sample_ci(sample_size=300, distr_func=runif, min=17, max=35)
```



Question 3