HW2

Joseph Goncalves Aeon Levy

2024-10-07

```
#####Problem 1:
#Copy paste and run the tribble given below.
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ dplyr     1.1.4     ✔ readr     2.1.5
## ✔ forcats   1.0.0     ✔ stringr   1.5.1
## ✔ ggplot2   3.5.1     ✔ tibble    3.2.1
## ✔ lubridate 1.9.3     ✔ tidyr     1.3.1
## ✔ purrr     1.0.2     
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
tribble( ~x, ~y, ~w, ~z,
         210,   300,   220,   180,
         102,   100,   119,   187,
         176,   175,   188,   173,
         87,    95,   91,     94,
         202,   210,  234,    218,
         110,   122,  131,    128,
) -> dt
dt
```

```
## # A tibble: 6 × 4
##       x     y     w     z
##   <dbl> <dbl> <dbl> <dbl>
## 1   210   300   220   180
## 2   102   100   119   187
## 3   176   175   188   173
## 4    87    95    91    94
## 5   202   210   234   218
## 6   110   122   131   128
```

```
## 1a) Use and show a map function to find the “mean” of each column of the dt data table
map_dbl(dt, mean)
```

```
##        x        y        w        z 
## 147.8333 167.0000 163.8333 163.3333
```

```
## 1b) Use and show a map function to find the “standard deviation” of each column of the dt data table.
map_dbl(dt, sd)
```

```
##        x        y        w        z 
## 54.45151 79.12016 58.40348 44.66617
```

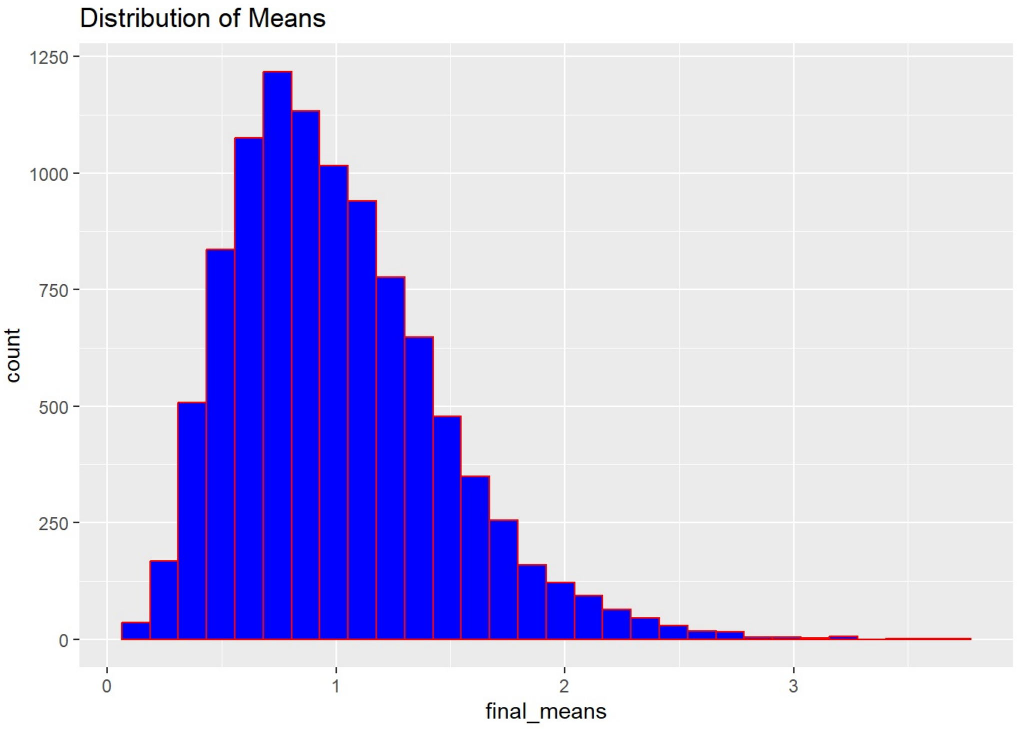
```
## 1c) Use and show a map function that will calculate the “square root” of each value of each column of the data
table dt.
map_df(dt, sqrt)
```

```
## # A tibble: 6 × 4
##       x     y     w     z
##   <dbl> <dbl> <dbl> <dbl>
## 1 14.5  17.3  14.8  13.4 
## 2 10.1  10    10.9  13.7 
## 3 13.3  13.2  13.7  13.2 
## 4  9.33  9.75  9.54  9.70
## 5 14.2  14.5  15.3  14.8 
## 6 10.5  11.0  11.4  11.3
## 1d) Use and show a map function that will calculate the “square root” of each value of each column of the data table dt.
summary(dt)
```

```
##        x               y               w               z        
##  Min.   : 87.0   Min.   : 95.0   Min.   : 91.0   Min.   : 94.0  
##  1st Qu.:104.0   1st Qu.:105.5   1st Qu.:122.0   1st Qu.:139.2  
##  Median :143.0   Median :148.5   Median :159.5   Median :176.5  
##  Mean   :147.8   Mean   :167.0   Mean   :163.8   Mean   :163.3  
##  3rd Qu.:195.5   3rd Qu.:201.2   3rd Qu.:212.0   3rd Qu.:185.2  
##  Max.   :210.0   Max.   :300.0   Max.   :234.0   Max.   :218.0
```

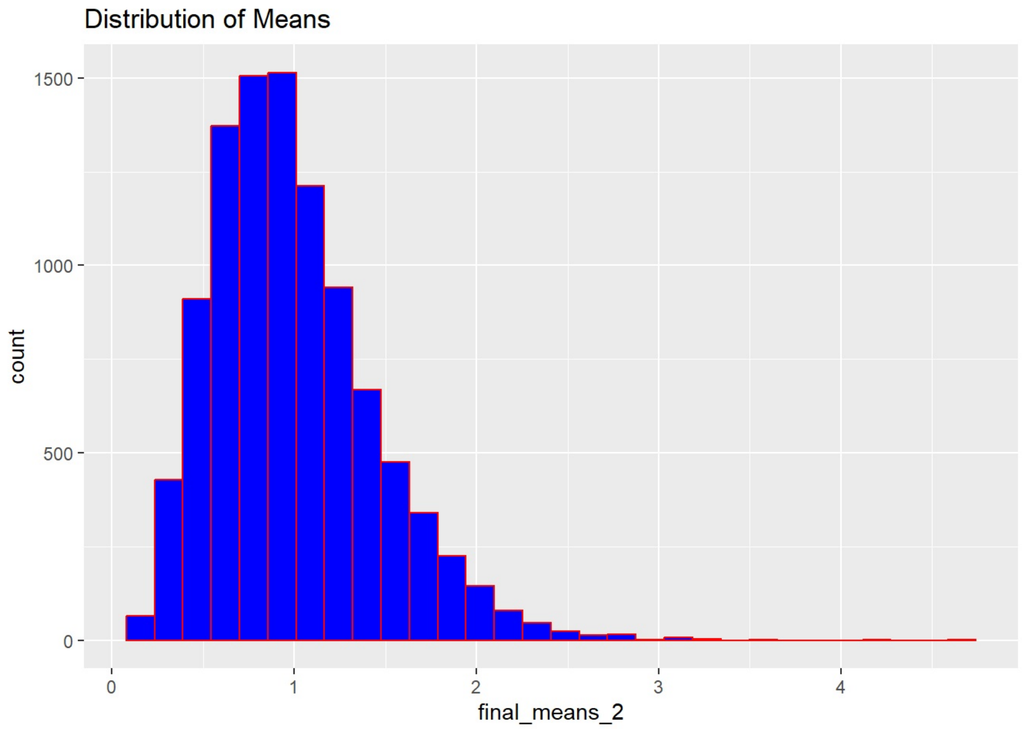
```
##### Problem 2:
myfunction <- function(){
  s_means <- numeric(10000)
  for(i in seq_len(10000)) {
    s <- rexp(5, rate = 1)
    s_means[i] <- mean(s)
  }
  return(s_means)
}
final_means <- myfunction()
ggplot(data=data.frame(final_means), aes(x=final_means))+
  geom_histogram(color = "red", fill = "blue")+
  labs(title = "Distribution of Means")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



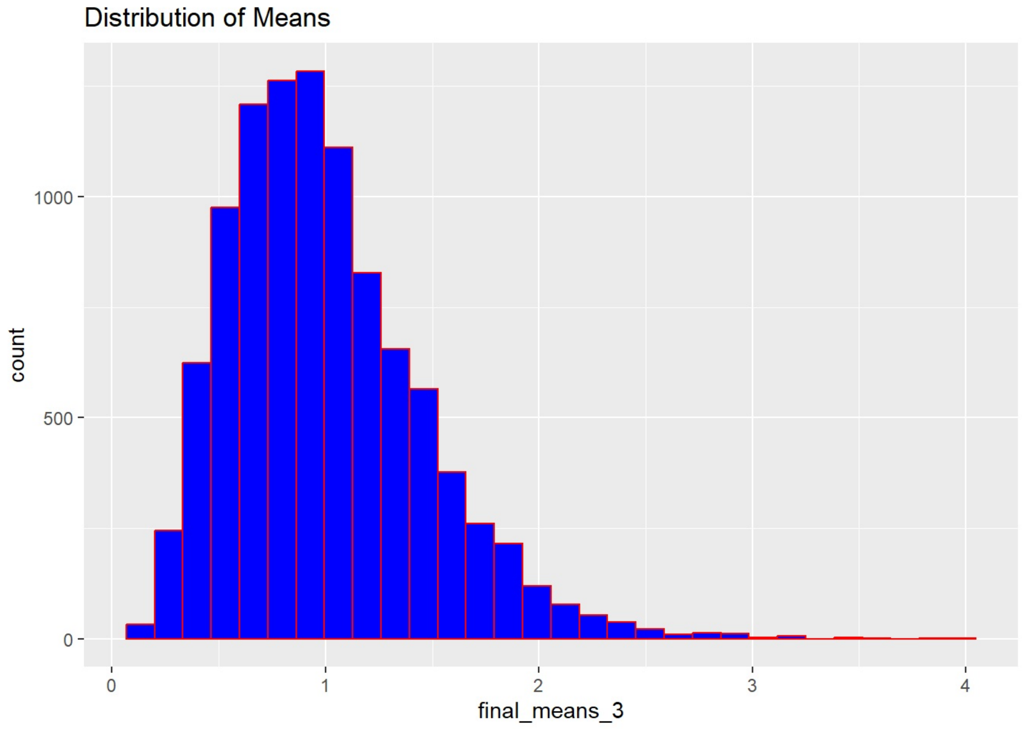
```
## 2a) Repeat part 1 by using a map_*() function
map_function <- function(){
  s_means <- map_dbl(1:10000, ~mean(rexp(5, rate = 1)))
  return(s_means)
}
final_means_2 <- map_function()

ggplot(data=data.frame(final_means_2), aes(x=final_means_2))+
  geom_histogram(color="red", fill = "blue")+
  labs(title = "Distribution of Means")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



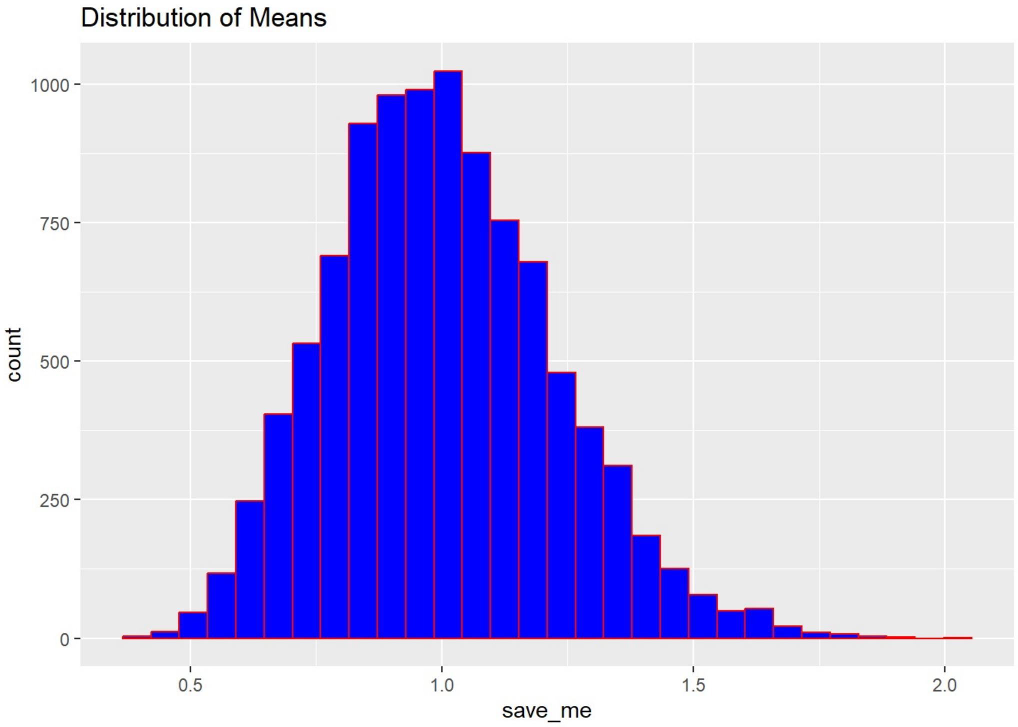
```
## 2b) Repeat part 1 by using the replicate() function
replicate_function <- function(){
  s_means <- replicate(10000, mean(rexp(5, rate = 1)))
  return(s_means)
}
final_means_3 <- replicate_function()

ggplot(data=data.frame(final_means_3), aes(x=final_means_3))+
  geom_histogram(color="red", fill = "blue")+
  labs(title = "Distribution of Means")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
## 2c) Use a another for loop that will print out plots for sample sizes of 5, 10, and 20 observations (instead o
f just 5).
new_numbers <- c(5,10,20)
more_means <- function(new_numbers,n_iterations=10000) {
  s_means <- numeric(n_iterations)
  for(i in 1:n_iterations){
    s <- rexp(new_numbers, rate = 1)
    s_means[i] <- mean(s)
  }
  return(s_means)
}
for (size in new_numbers)
  save_me <-more_means(size)
ggplot(data=data.frame(save_me), aes(x=save_me))+
  geom_histogram(color = "red", fill = "blue")+
  labs(title = "Distribution of Means")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
#Problem 3:

data(mtcars)

# Initialize an output vector to store standard deviation values
output <- vector("double", ncol(mtcars))

# Loop each column mtcars
for (i in seq_along(mtcars)) {
  # Calculate sd and store
  output[i] <- sd(mtcars[[i]])
}

# Print sd for column
output
```

```
##  [1]   6.0269481   1.7859216 123.9386938  68.5628685   0.5346787   0.9784574
##  [7]   1.7869432   0.5040161   0.4989909   0.7378041   1.6152000
```