Module 5 video code

Prof. Bolton

Fall 2021

```
library(tidyverse)
```

Video code

Setting up the flights data

```
install.packages("nycflights13", repos = "https://cloud.r-project.org")
```

```
##
## The downloaded binary packages are in
##  /var/folders/9z/mqg8cp0j0xl6t3hk0n9c02_c0000gn/T//RtmpHXr3hq/downloaded_packages
```

```
library(tidyverse)
library(nycflights13)
# Save data in a data frame called SF
SF <- flights %>% filter(dest=="SFO" & !is.na(arr_delay))
dim(SF)
```

```
## [1] 13173    19
```

Summarise the flights data

```
SF %>% summarise(
  mean_delay = mean(arr_delay),
  median_delay = median(arr_delay),
  max_delay = max(arr_delay))
```

```
## # A tibble: 1 x 3
##   mean_delay median_delay max_delay
##        <dbl>        <dbl>     <dbl>
## 1       2.67           -8      1007
```

```
# We'll save the population mean,
# so we can use it later on
population_mean <- SF %>%
  summarize(population_mean_delay =
              mean(arr_delay))

population_mean <-
  as.numeric(population_mean)
```

Take a sample

```
# sample of 25 flights from our population
# by default, replace = FALSE (i.e. sampling without replacement)
sample25 <- SF %>% sample_n(size=25, replace = FALSE)
```

What is the difference between `sample()` and `sample_n()`?

```
sample(c("H", "T"), probs=c(0.5, 0.5),
       size=10, replace=TRUE)
sample(1:6, replace=FALSE`)
```

The `sample()` function samples elements from a **vector**, with or without replacement

```
# Create our sample
SF %>% sample_n(size=25, replace=FALSE)
```

The `sample_n()` samples rows (observations) from a data frame, with or without replacement

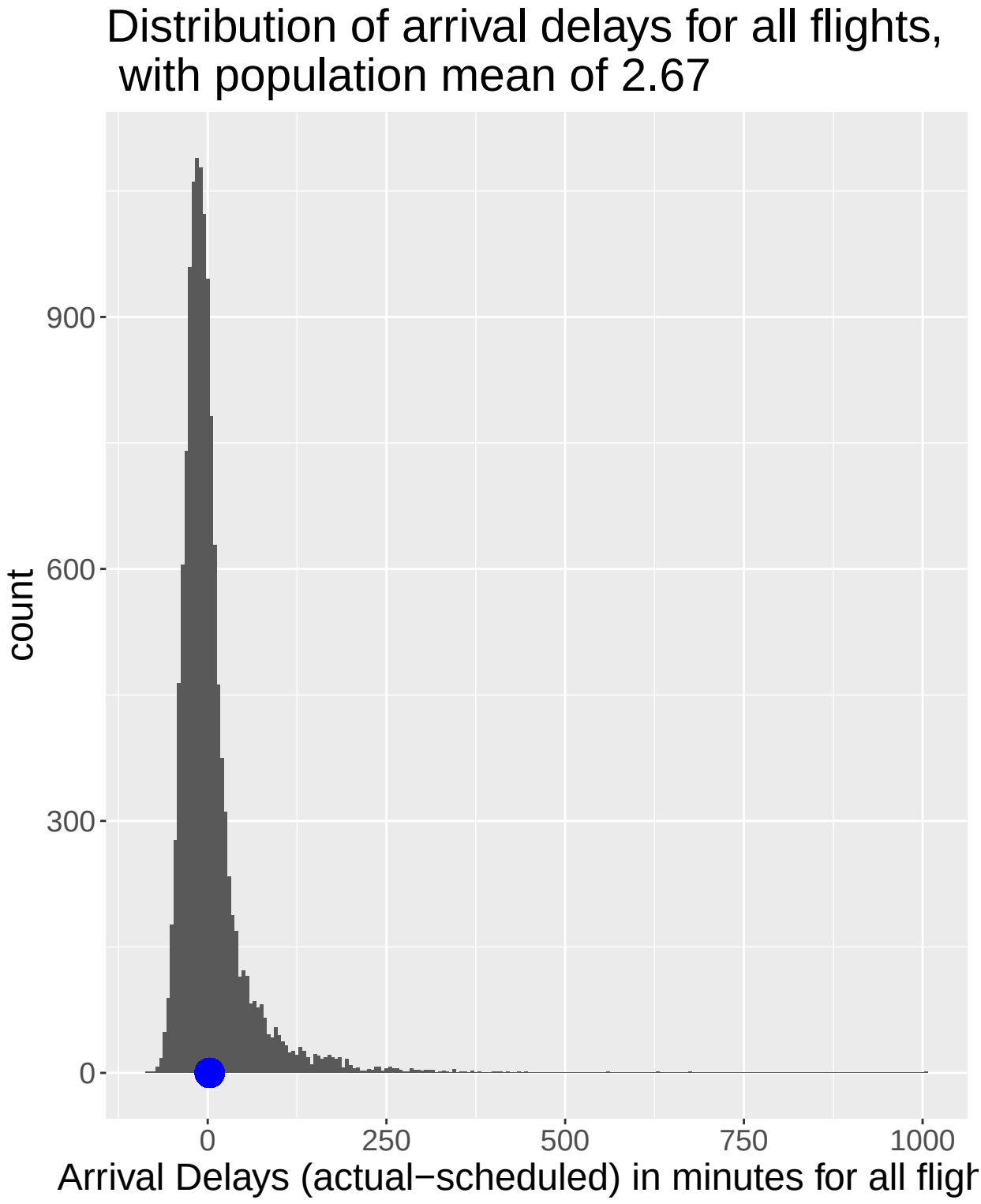
Calculate summary values for this sample

```
sample25 %>% summarise(mean_delay = mean(arr_delay),
                 median_delay = median(arr_delay),
                 max_delay = max(arr_delay))
```

```
## # A tibble: 1 x 3
##   mean_delay median_delay max_delay
##        <dbl>        <dbl>     <dbl>
## 1        1.8          -10       208
```

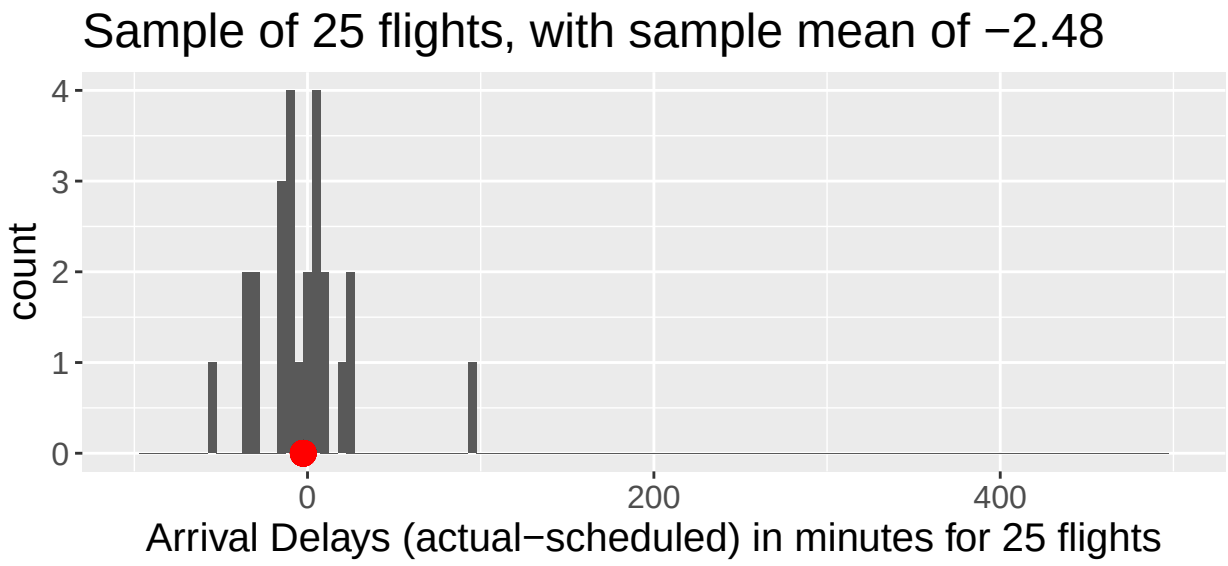
Looking at multiple samples of size n=25

```
## Warning: Use of `SF$arr_delay` is discouraged. Use `arr_delay` instead.
```



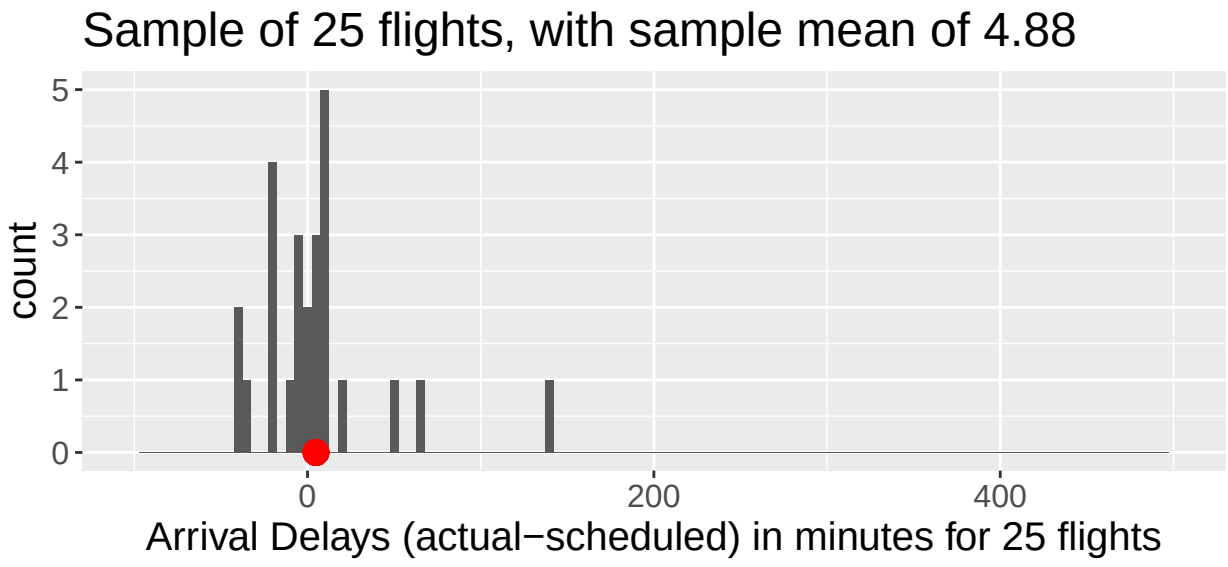

```
## Warning: Use of `d25$arr_delay` is discouraged. Use `arr_delay` instead.
```

```
## Warning: Removed 2 rows containing missing values (geom_bar).
```



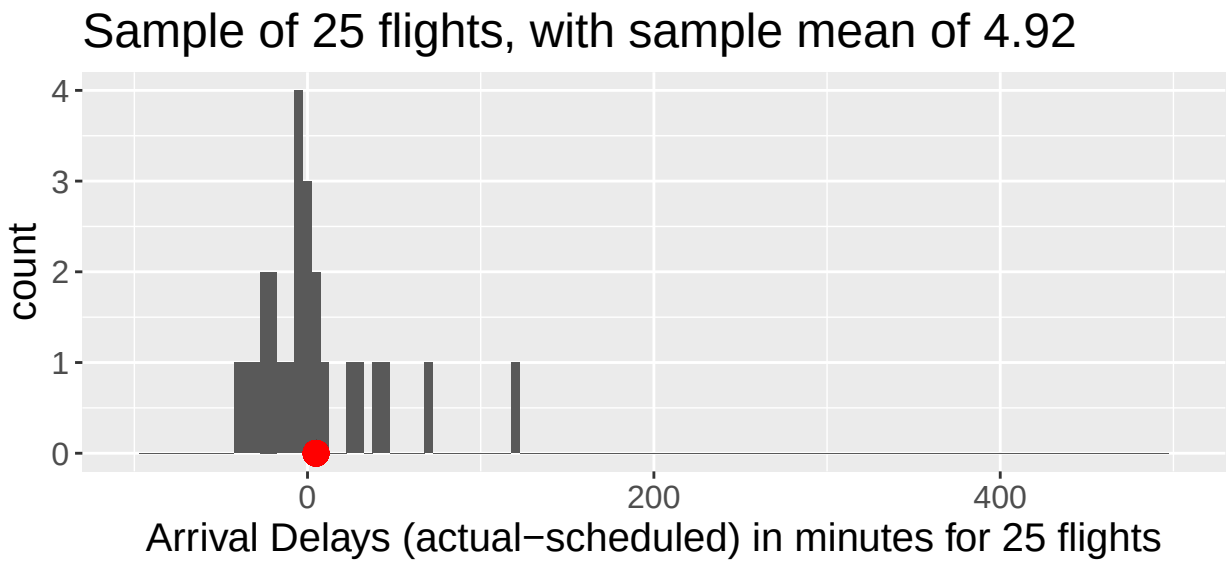

```
## Warning: Use of `d25$arr_delay` is discouraged. Use `arr_delay` instead.
```

```
## Warning: Removed 2 rows containing missing values (geom_bar).
```




```
## Warning: Use of `d25$arr_delay` is discouraged. Use `arr_delay` instead.
```

```
## Warning: Removed 2 rows containing missing values (geom_bar).
```



Review: Sampling distributions

Recall, the **sampling distribution** of the mean of `arr_delay` is the distribution of all the values that `mean_delay` could be for random samples of size $n = 25$

To estimate the sampling distribution, let's look at 1000 values of `mean_delay`, calculated from 1000 random samples of size $n = 25$ from our population

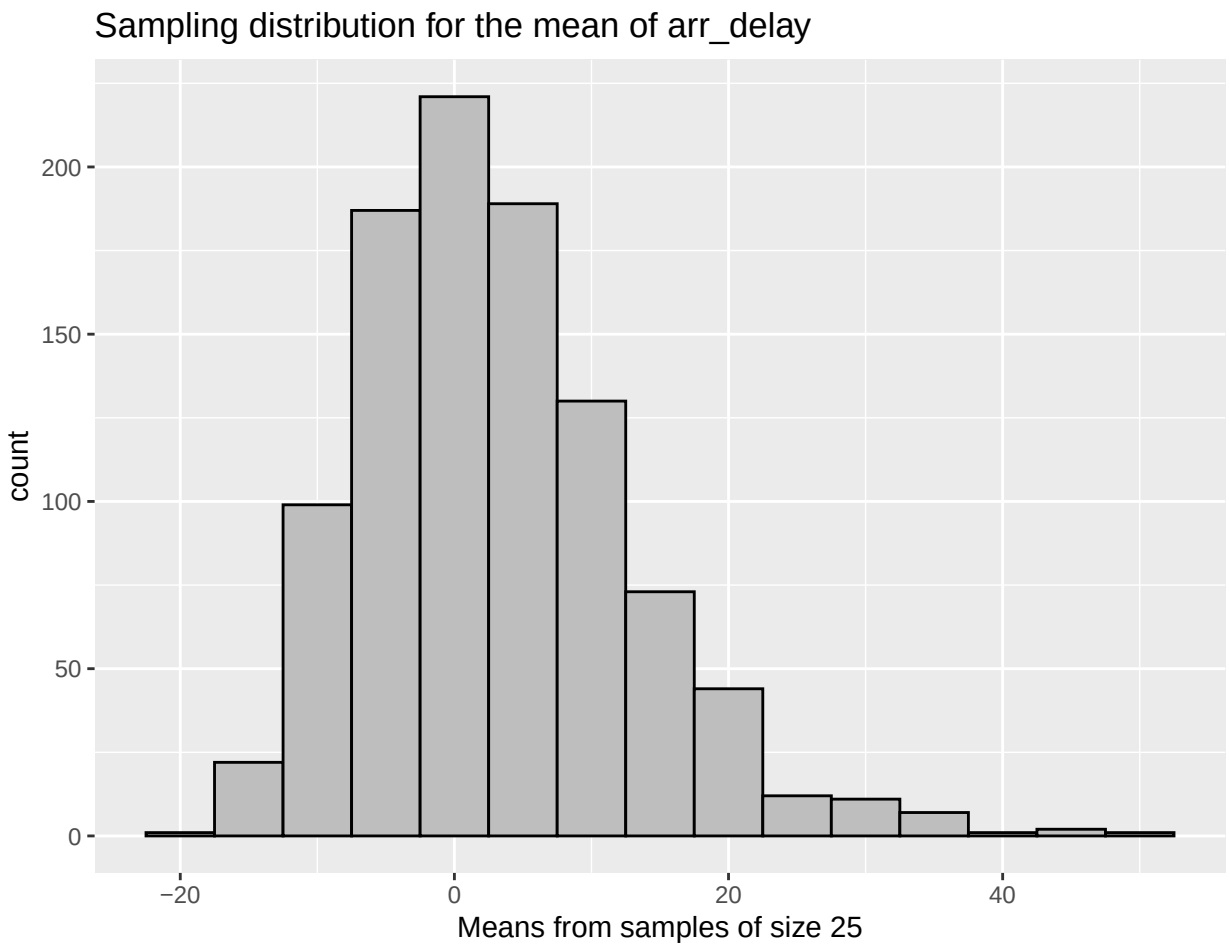
```
sample_means <- rep(NA, 1000) # where we'll store the means

for(i in 1:1000){
  sample25 <- SF %>% sample_n(size=25)
  sample_means[i] <- as.numeric(sample25 %>%
                          summarize(mean(arr_delay)))
}

sample_means <- tibble(mean_delay = sample_means)
```

Sampling distribution of the mean

```
ggplot(sample_means, aes(x=mean_delay)) +
  geom_histogram(binwidth=5, color="black", fill="gray") +
  labs(x="Means from samples of size 25",
  title="Sampling distribution for the mean of arr_delay")
```

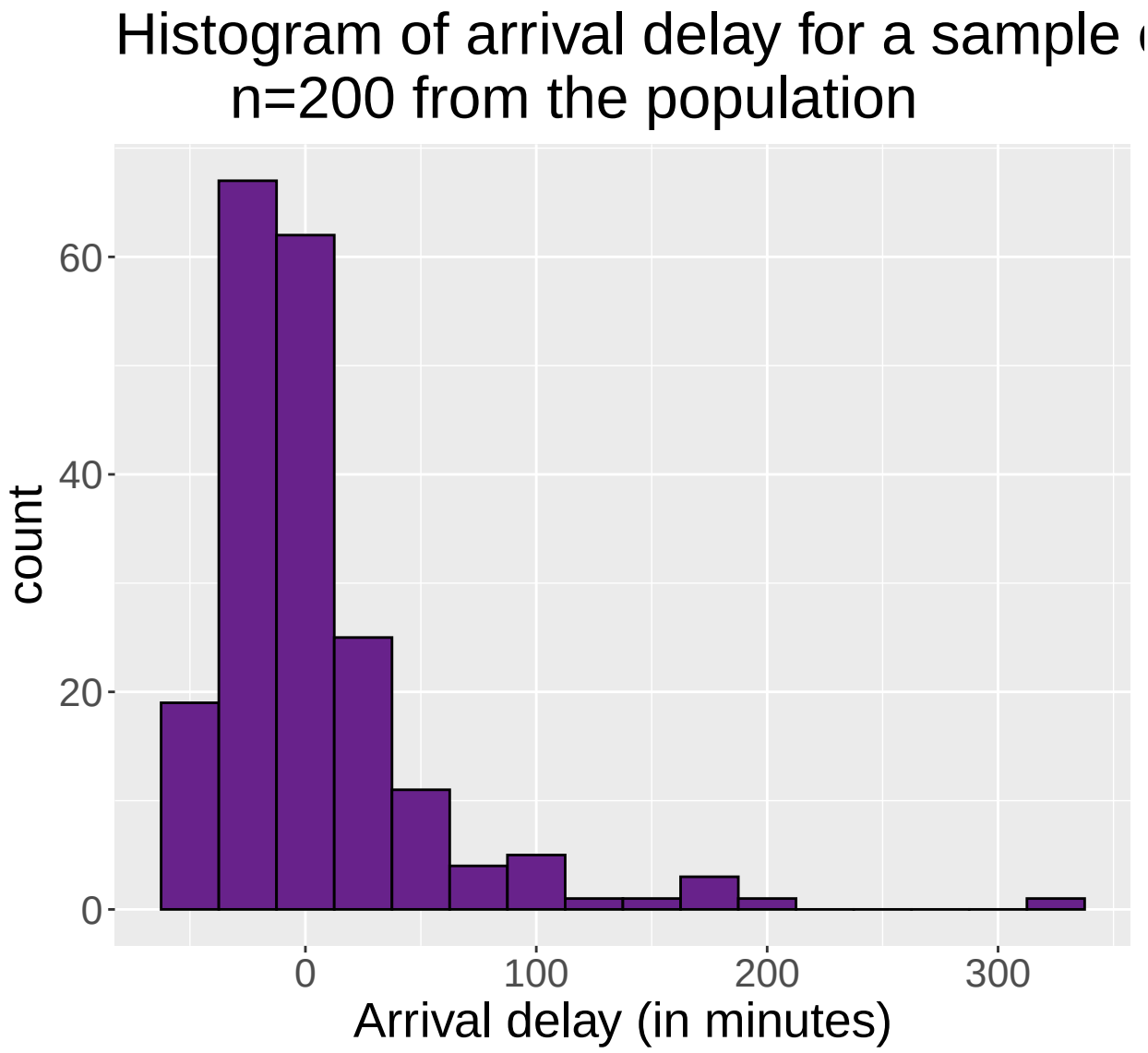


3 histograms for question prompt

Bootstrapping with R

Suppose we do not observe the full population, and have only observed **one sample of size 200**

```
observed_data <- SF %>%
  sample_n(size=200)
```



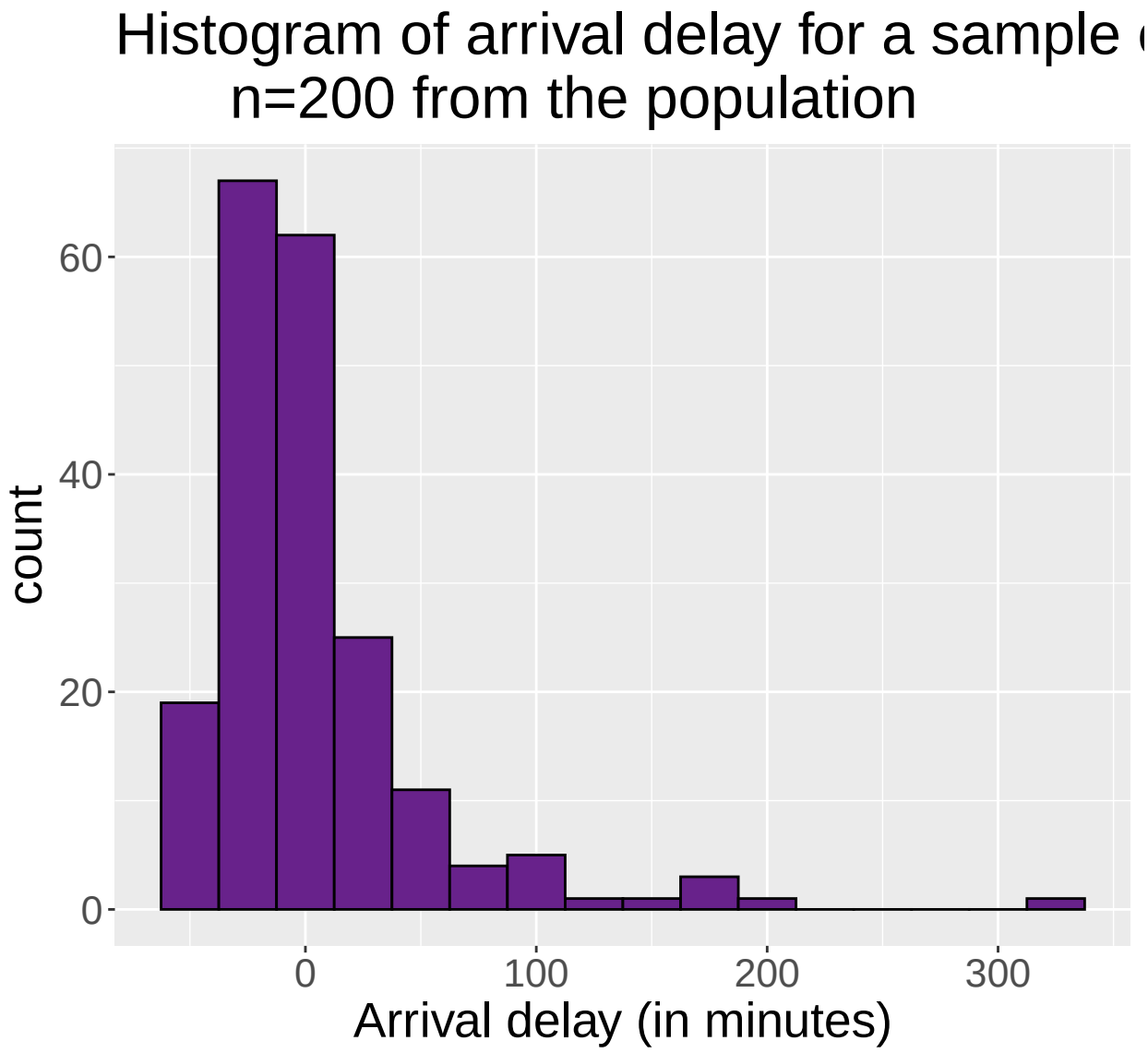
Let's calculate the mean arrival delay for this sample

```
obs_mean <- observed_data %>%
  summarize(mean(arr_delay))
as.numeric(obs_mean)
```

```
## [1] 4.485
```

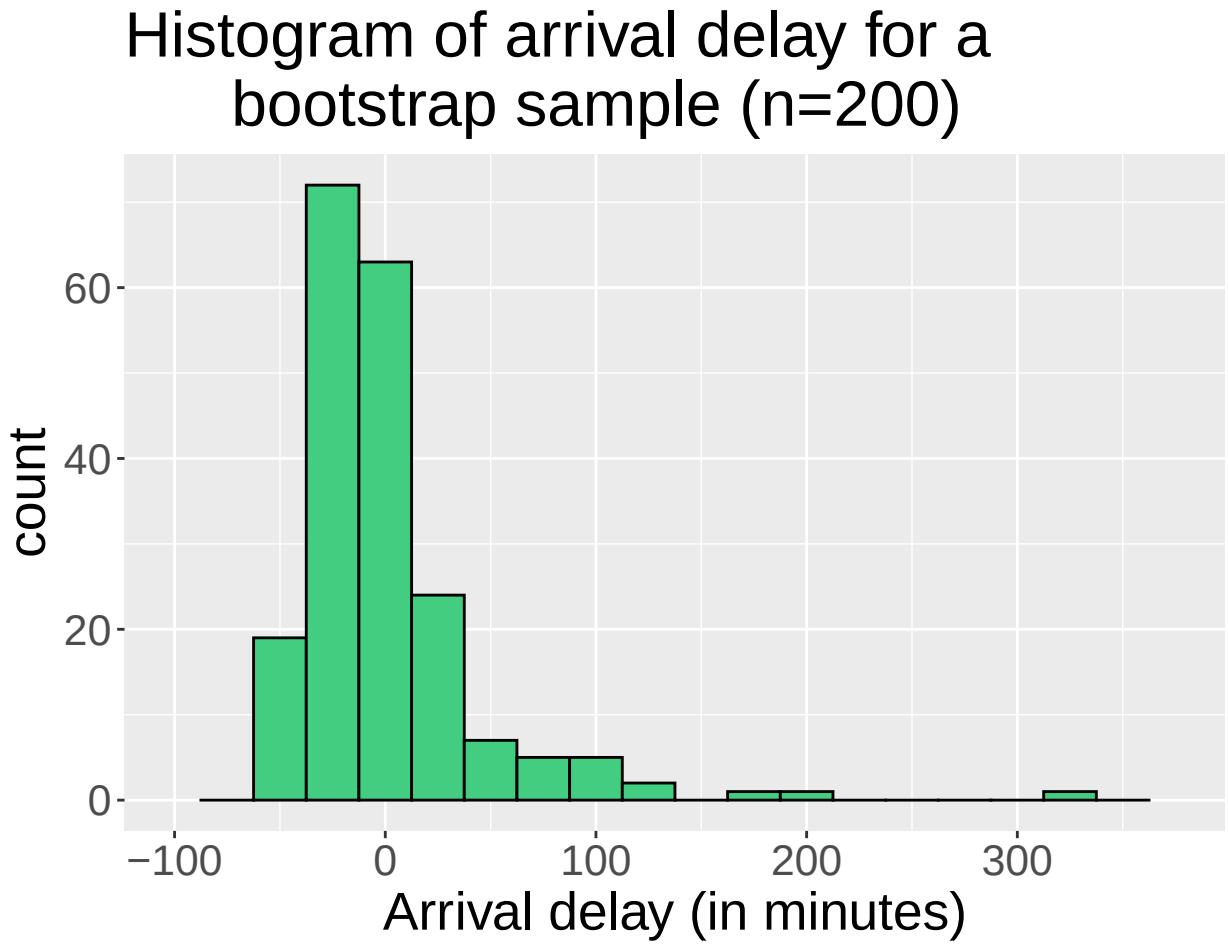
A bootstrap sample from our observed data

.pull-left[




```
boot_samp <- observed_data %>%
  sample_n(size=200, replace=TRUE)
```

```
## Warning: Removed 2 rows containing missing values (geom_bar).
```

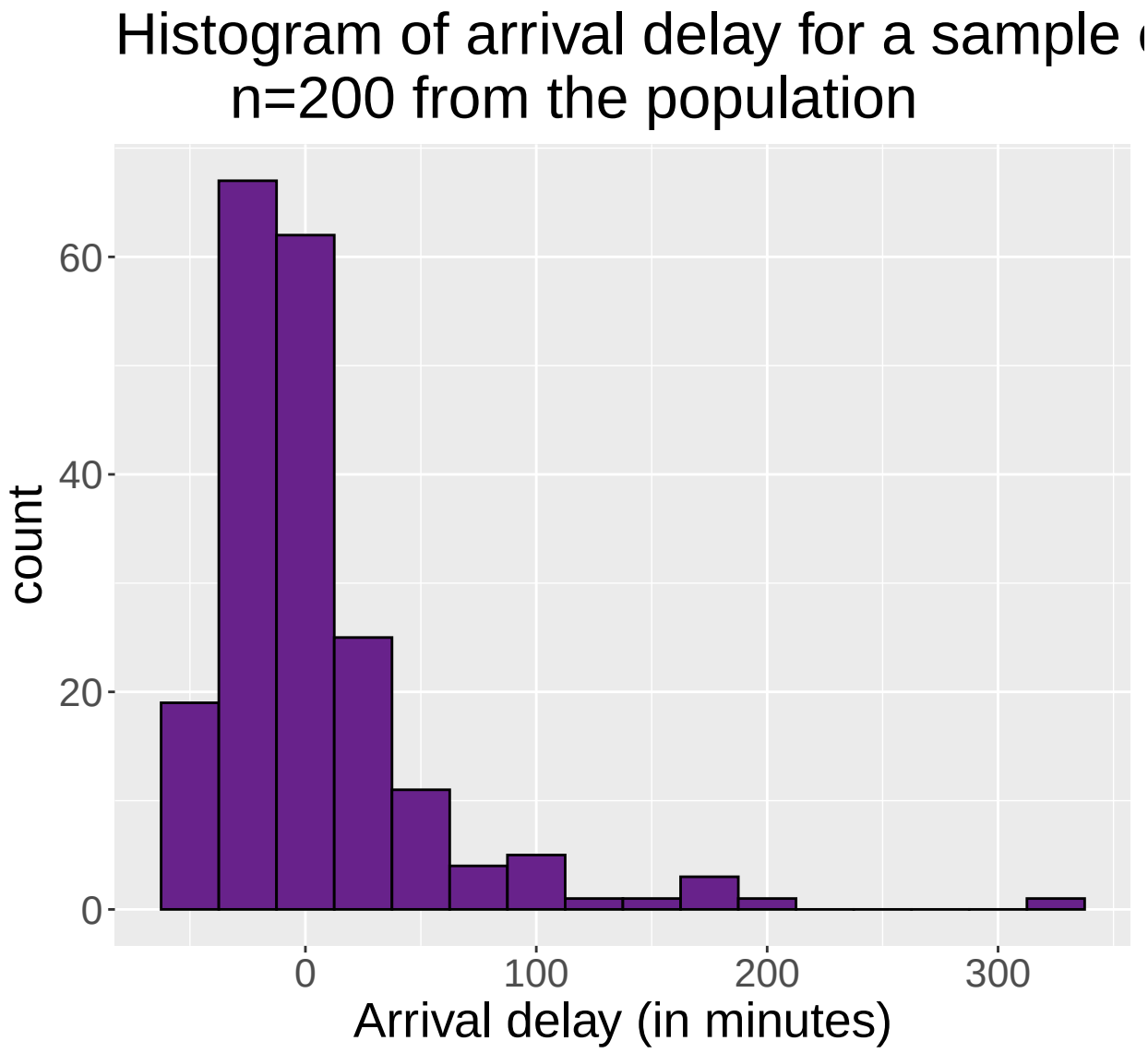



```
boot_mean <- boot_samp %>%
  summarize(mean_delay =
              mean(arr_delay))
as.numeric(boot_mean)
```

```
## [1] 1.18
```

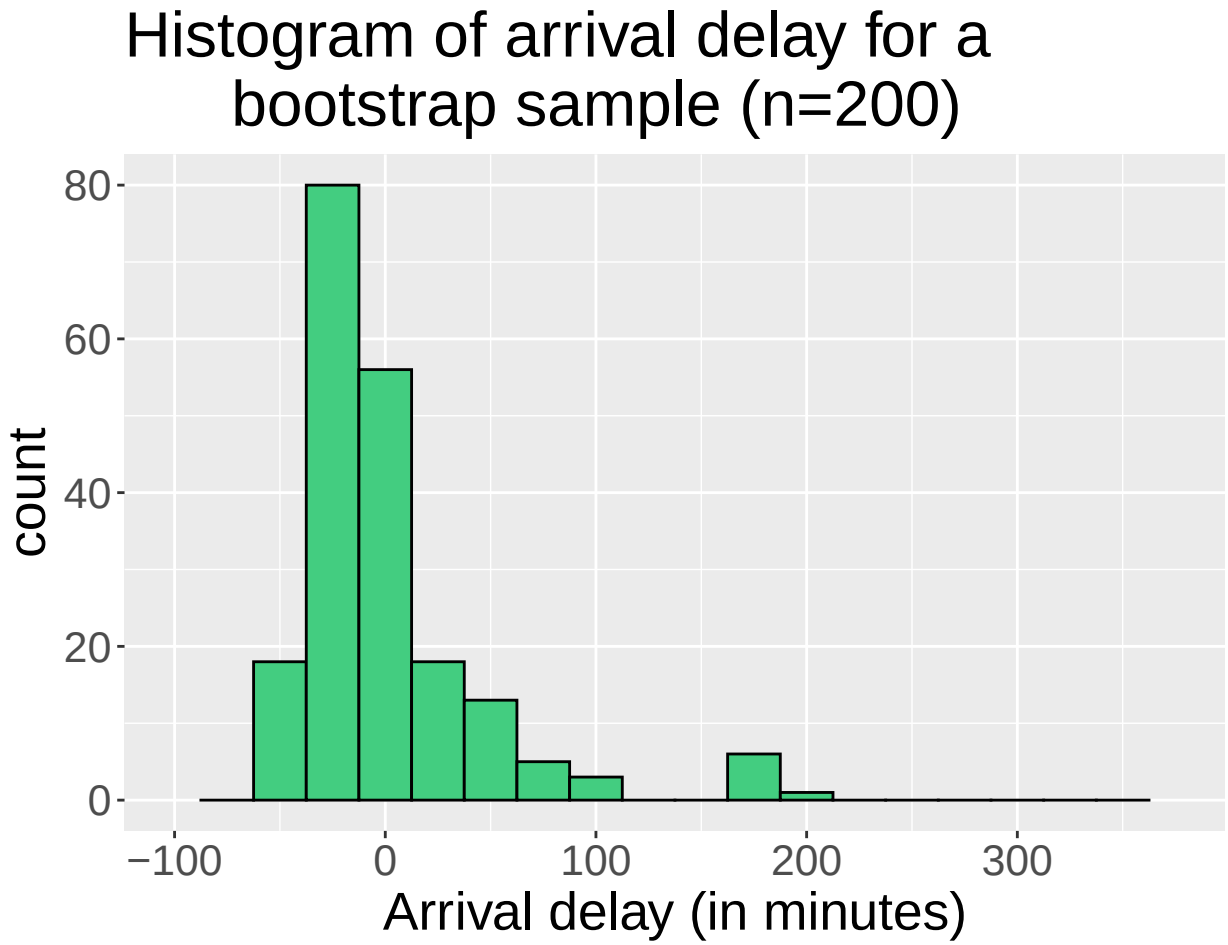
Another bootstrap sample from our observed data

.pull-left[




```
boot_samp <- observed_data %>%
  sample_n(size=200, replace=TRUE)
```

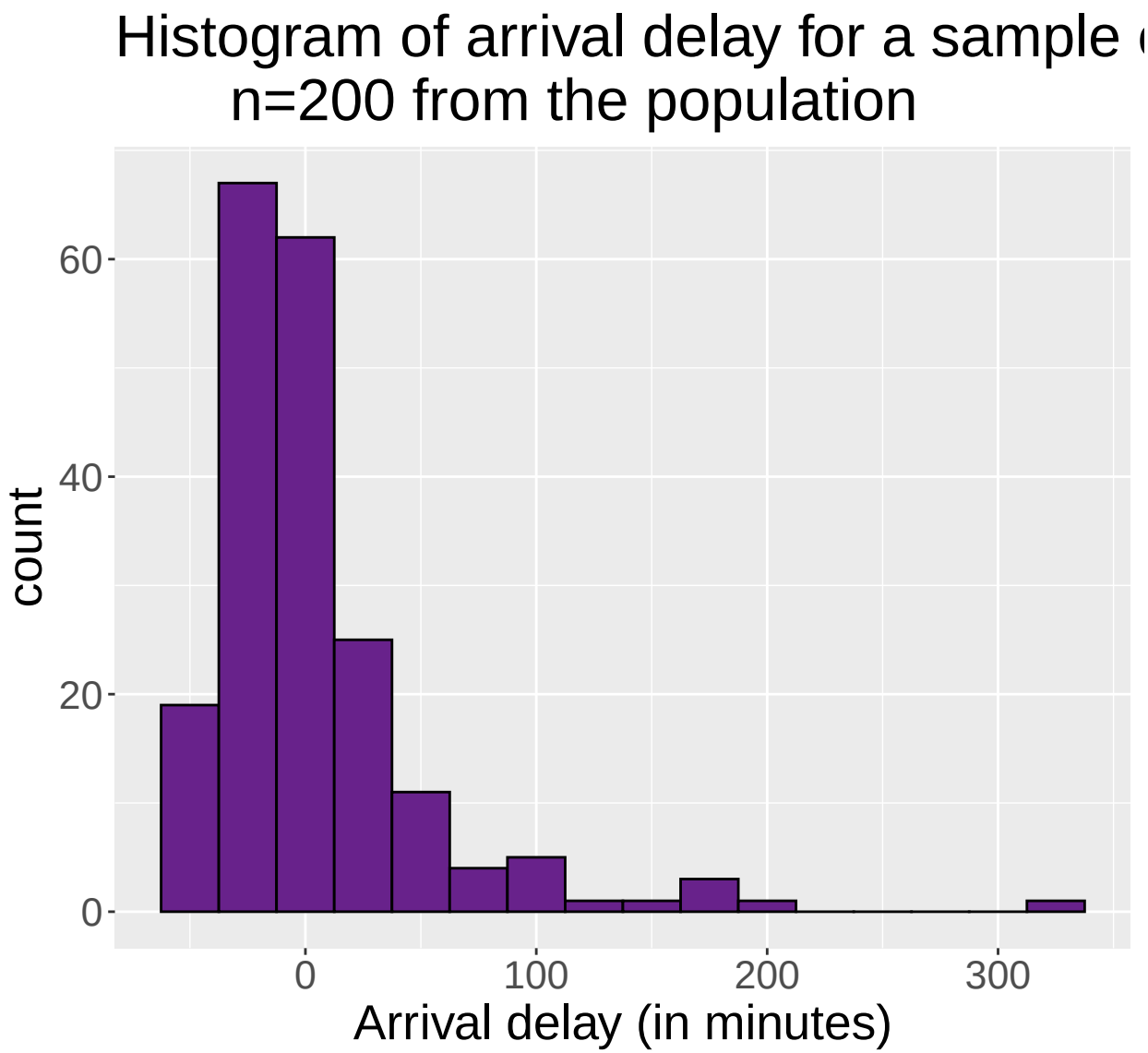
```
## Warning: Removed 2 rows containing missing values (geom_bar).
```




```
boot_mean <- boot_samp %>%
  summarize(mean_delay =
              mean(arr_delay))
as.numeric(boot_mean)
```

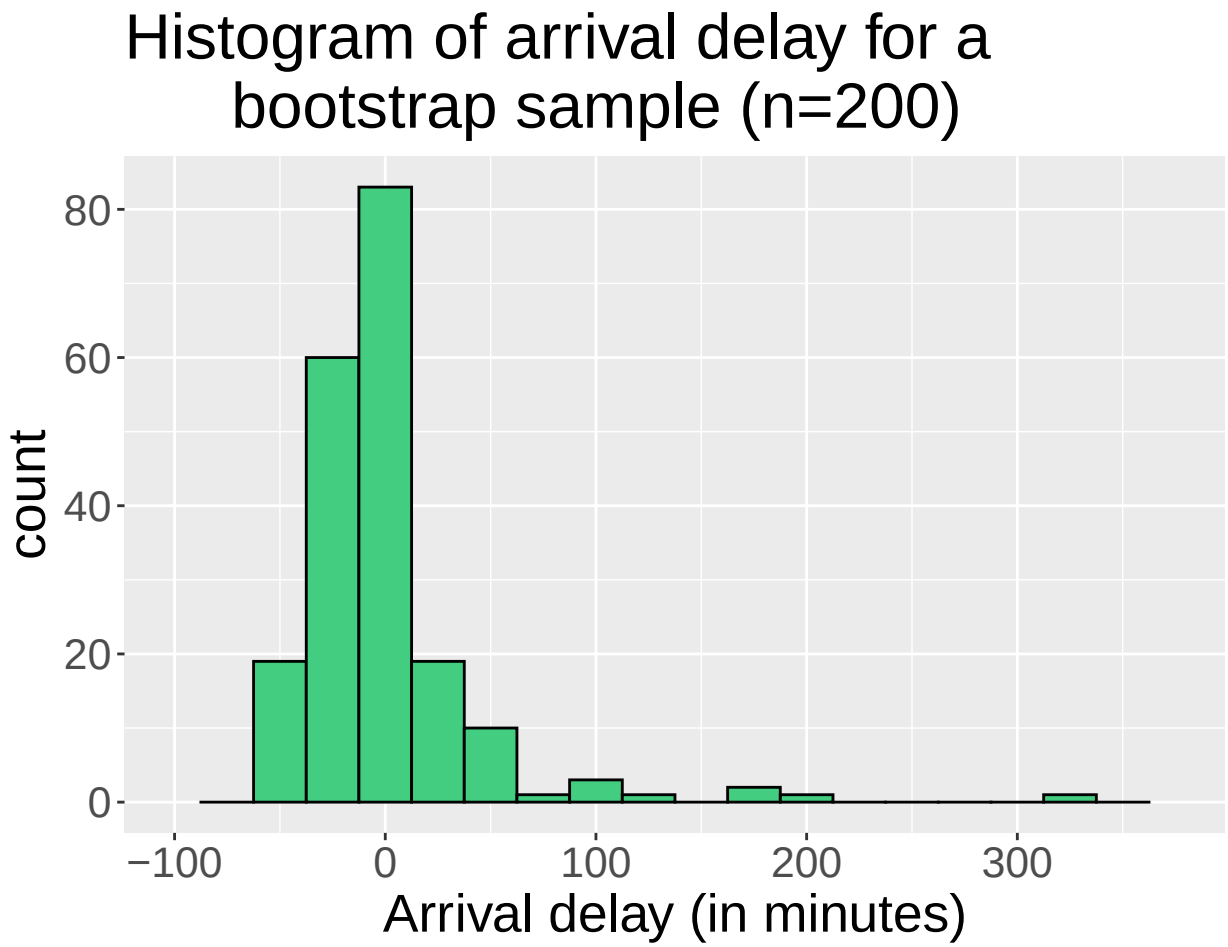
```
## [1] 2.24
```

And another bootstrap sample...




```
boot_samp <- observed_data %>%
  sample_n(size=200, replace=TRUE)
```

```
## Warning: Removed 2 rows containing missing values (geom_bar).
```

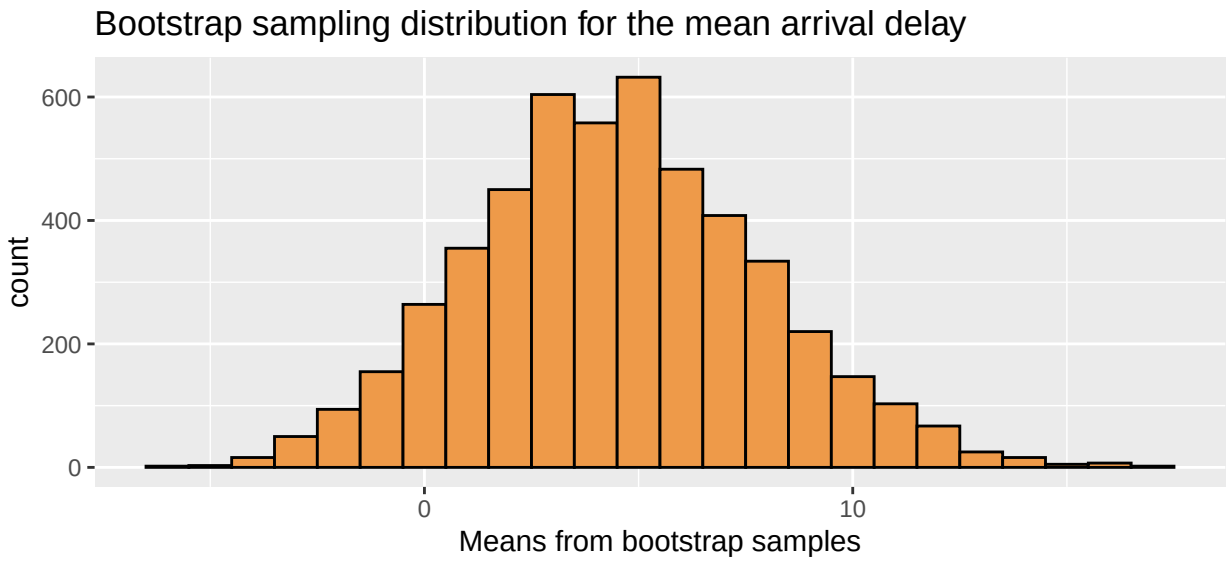



```
boot_mean <- boot_samp %>%
  summarize(mean_delay =
              mean(arr_delay))
as.numeric(boot_mean)
```

```
## [1] -0.15
```

```
boot_means <- rep(NA, 5000) # where we'll store the means
for(i in 1:5000){
  boot_samp <- observed_data %>% sample_n(size=200, replace=TRUE)
  boot_means[i] <-
    as.numeric(boot_samp %>%
                 summarize(mean_delay = mean(arr_delay)))
}
boot_means <- tibble(mean_delay = boot_means)
```

```
ggplot(boot_means, aes(x=mean_delay)) +
  geom_histogram(binwidth=1, fill="tan2", color="black") +
  labs(x="Means from bootstrap samples",
       title="Bootstrap sampling distribution for the mean arrival delay")
```



Percentiles (quantiles): an extension of quartiles

For a number p between 0 and 100, the pth percentile is the smallest value that is larger or equal to p% of all the values

- Median (Q_2): 50th percentile
- First quartile (Q_1): 25th percentile
- Third quartile (Q_3): 75th percentile

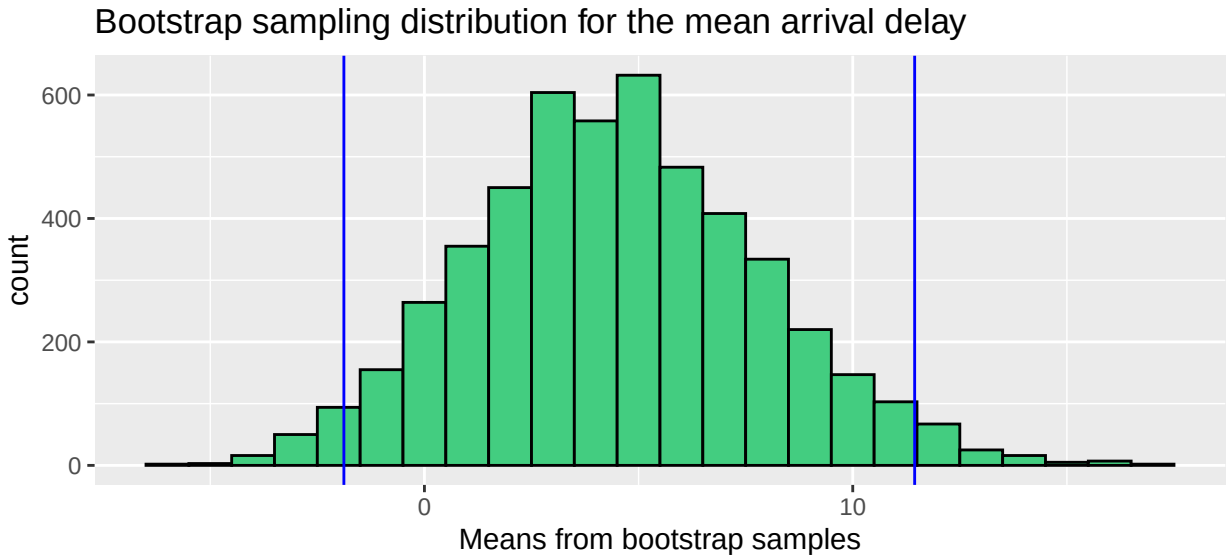
Use the `quantile()` funciton in R to calculate these:

```
# Calculate Q1, median, and Q3
quantile(boot_means$mean_delay, c(0.25, 0.5, 0.75))
```

```
##   25%   50%   75%
## 2.205 4.395 6.695
```

```
# Can also calculate any other percentiles
quantile(boot_means$mean_delay, c(0.025, 0.4, 0.57))
```

```
##      2.5%       40%       57%
## -1.880125  3.520000  4.970000
```



2.5th and 97.5th percentiles:

```
quantile(boot_means$mean_delay,
         c(0.025, 0.975))
```

```
##      2.5%     97.5%
## -1.880125 11.445625
```

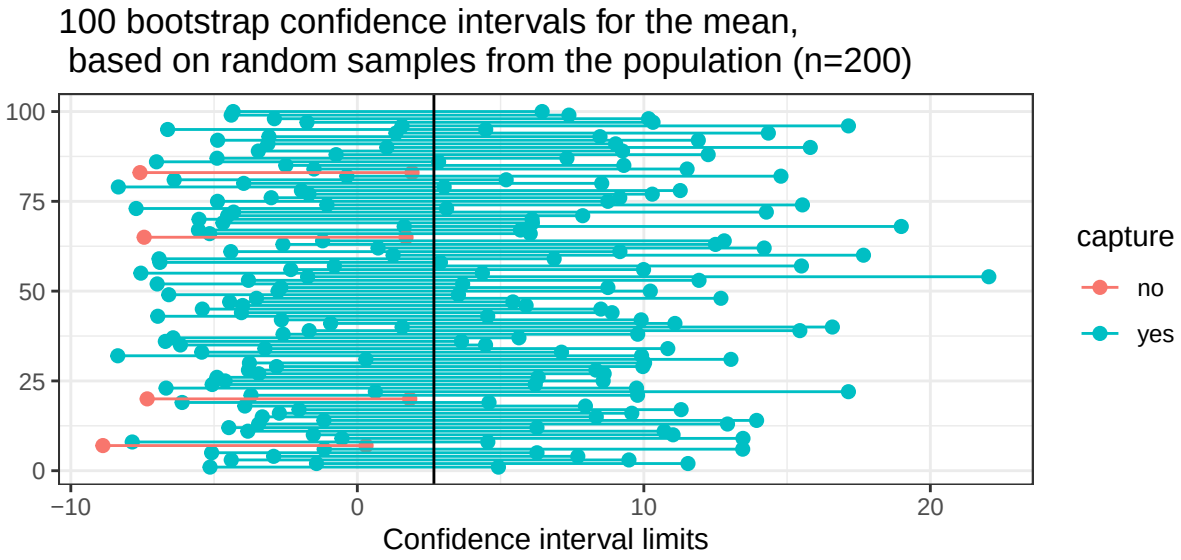
Recall true population mean:

```
as.numeric(population_mean)
```

```
## [1] 2.672892
```

How often does this procedure give an interval that captures the population mean?

This code is for the curious but NOT something we'll ask you to be able to make yourself. It also take aaaaaages to run, so that is why we have saved the output as a csv for you.


100 bootstrap confidence intervals for the mean,
based on random samples from the population (n=200)
100
75
50
25
0
−10
0
10
20
Confidence interval limits
capture
no
yes