Bootstrapping and Simulation with R

Evans_G02504972

2023-11-14

```
library(tidyverse)
library(stringr)
library(broom)
library(ggthemes)
library(dplyr)
library(ggplot2)
```

Let's simulate the roll of a fair six-sided die

```
sample(x = 1:6, size = 1)
```

```
## [1] 5
```

We can also simulate multiple rolls with one line of code

```
sample(x = 1:6, size = 2, replace = TRUE)
```

```
## [1] 4 4
```

##Use the sample function to simulate 100000 dice rolls, storing the result in a vector called rolls. What proportion of rolls were each number?

```
set.seed(1994)
# Rolling a fair six-sided die 100,000 times
rolls <- sample(x = 1:6, size = 100000, replace = TRUE)

# Tidying up the result into a tibble
tidy_rolls <- tibble(value = rolls)

tidy_rolls %>%
  janitor::tabyl(value)
```

```
##  value     n percent
##      1 16680 0.16680
##      2 16825 0.16825
##      3 16669 0.16669
##      4 16616 0.16616
##      5 16654 0.16654
##      6 16556 0.16556
```

Using the rle() function, what are the most consecutive

rolls with the same die value?

```
# Calculating runs
runs <- rle(rolls)

# Tidying up into a tibble
tidy_runs <- tibble(Length = runs$lengths,
                    Value = runs$values)
tidy_runs
```

```
## # A tibble: 83,259 × 2
##    Length Value
##     <int> <int>
##  1      1     2
##  2      1     3
##  3      1     2
##  4      1     4
##  5      1     6
##  6      1     5
##  7      1     3
##  8      1     6
##  9      1     5
## 10      1     2
## # ℹ 83,249 more rows
```

What is the total number of runs of each length there were ‘r nrow(tidy_runs)’ unoque runs

What is the ratio of the number of runs of length 2 compared to the number of runs of length 1? What about the number of runs of length 3 compared to 2?

```
tidy_runs %>%
  janitor::tabyl(Length)
```

```
##  Length     n      percent
##       1 69293 8.322584e-01
##       2 11638 1.397807e-01
##       3  1958 2.351698e-02
##       4   308 3.699300e-03
##       5    50 6.005357e-04
##       6     9 1.080964e-04
##       7     3 3.603214e-05
```

2:1

```
11464/69607
```

```
## [1] 0.1646961
```

3:2

```
1914/11464
```

```
## [1] 0.1669574
```

What are these ratios close to?

The ratios are close to 1/6

#Sampling fruit vector ## Randomly select 10 values from the fruit vector 🍇 🍉 🍓 🥝 🍒 that is available from the stringr package, and run your code several times. Does sample() use sampling with or without replacement by default?

```
#Randomly select 10 fruits
sample(x = fruit, size = 10)
```

```
##  [1] "papaya"       "pomelo"       "loquat"       "blackberry"   "pamelo"
##  [6] "olive"        "tamarillo"    "blood orange" "cucumber"     "goji berry"
```

The 'sample' function samples without replacement by default.

##Randomly select 10 values from the fruit vector, sampling with replacement. Resample 10 fruits until you obtain at least one duplicate fruit within a sample.

```
#Randomly select 10 fruits
sample(x = fruit, size = 10, replace=TRUE)
```

```
##  [1] "tamarillo"    "currant"      "goji berry"   "durian"       "currant"
##  [6] "cantaloupe"   "pomegranate"  "boysenberry"  "blood orange" "grapefruit"
```

For more details regarding how R generates random numbers, submit ?Random to the Console. We can obtain a reproducible random sample by using the set.seed() function 🌱, which takes an integer value as its input and allows for reproducible random number generation. ## Set the seed value to 1989, and then sample 13 values from the fruit vector to reproduce the results below.

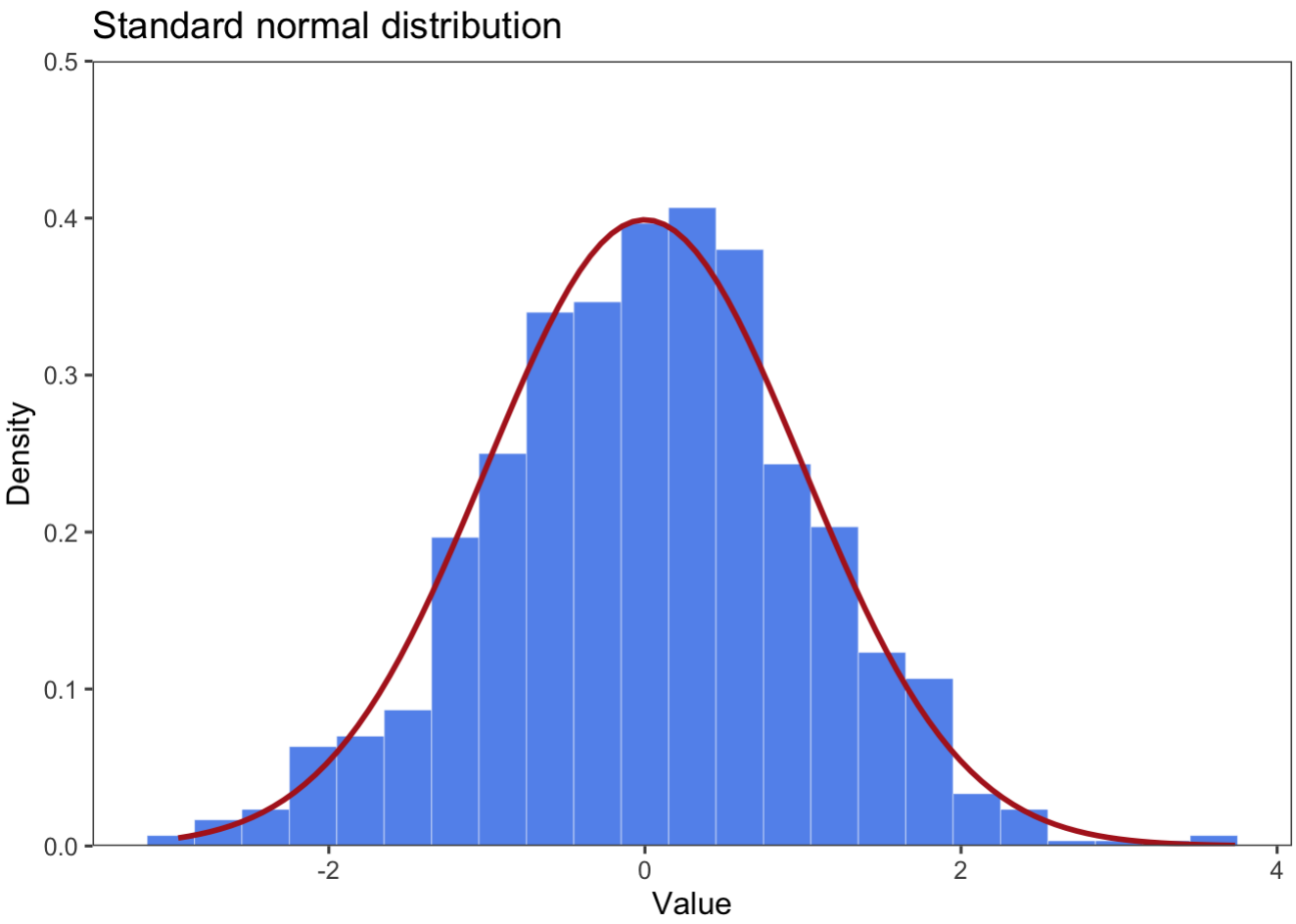
```
set.seed(1989)
sample(x = fruit, size = 10)
```

```
##  [1] "salal berry"       "purple mangosteen" "durian"
##  [4] "rambutan"          "breadfruit"        "currant"
##  [7] "tamarillo"         "blackberry"        "raisin"
## [10] "dragonfruit"
```

##Simulating from statistical distribution

```
# Set ggplot theme for visualizations
theme_set(ggthemes::theme_few())

# Set parameters that will be passed to `stat_function()`
n <- 1000
mean <- 0
sd <- 1
binwidth <- 0.3 # passed to geom_histogram and stat_function
myData <- data.frame(x = rnorm(n, mean, sd))

ggplot(myData, aes(x = x, mean = mean, sd = sd, binwidth = binwidth, n = n)) +
  geom_histogram(aes(y = after_stat(density)), binwidth = binwidth,
       colour = "white", fill = "cornflowerblue", size = 0.1) +
stat_function(fun = function(x) dnorm(x, mean = mean, sd = sd),
    color = "firebrick", size = 1) +
  labs(title = "Standard normal distribution",
       x = "Value",
       y = "Density") +
  scale_y_continuous(limits = c(0, 0.5), expand = c(0, 0))
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## ℹ Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



##Generate

100,000 values from a standard normal distribution, and calculate an estimate of the median (50th percentile) of the standard normal distribution using the quantile() function.

```
#Generating 100000 values from a standard normals
stan_normal <- tibble(value = rnorm(n=100000,
                                    mean=0,
                                    sd=1))
#monte carlo estimate of the median
quantile(stan_normal$value, probs=0.5)
```

```
##          50%
## -0.003227824
```

Also calculate an estimate of the 97.5th percentile of the standard normal distribution using the quantile() function. Note that the exact value of the 97.5th percentile can be found using qnorm(p = 0.975).

```
quantile(stan_normal$value, probs=0.975)
```

```
##    97.5%
## 1.975728
```

```
#exact value form the qnorm( function)

qnorm(p = 0.975, mean=0,sd =1)
```

```
## [1] 1.959964
```

```
set.seed(1994)
#number of monte carlo simulations
N <- 10000

waiting_times <- tibble(spongebob = runif(n=N,
                                          min=0,
                                          max=20))
mean(waiting_times$spongebob)
```

```
## [1] 9.922599
```

```
quantile(waiting_times$spongebob, probs =0.50)
```

```
##      50%
## 9.889624
```

```
#Simulating patrict's waiting time

waiting_times <- waiting_times %>%
  mutate(
    patrick = runif(n = N, min = 0, max = 30),
    total = patrick + 10 + spongebob
  )
#calculate monte carlo estimate for spongebob's total waiting time
mean(waiting_times$total)
```

```
## [1] 34.79465
```

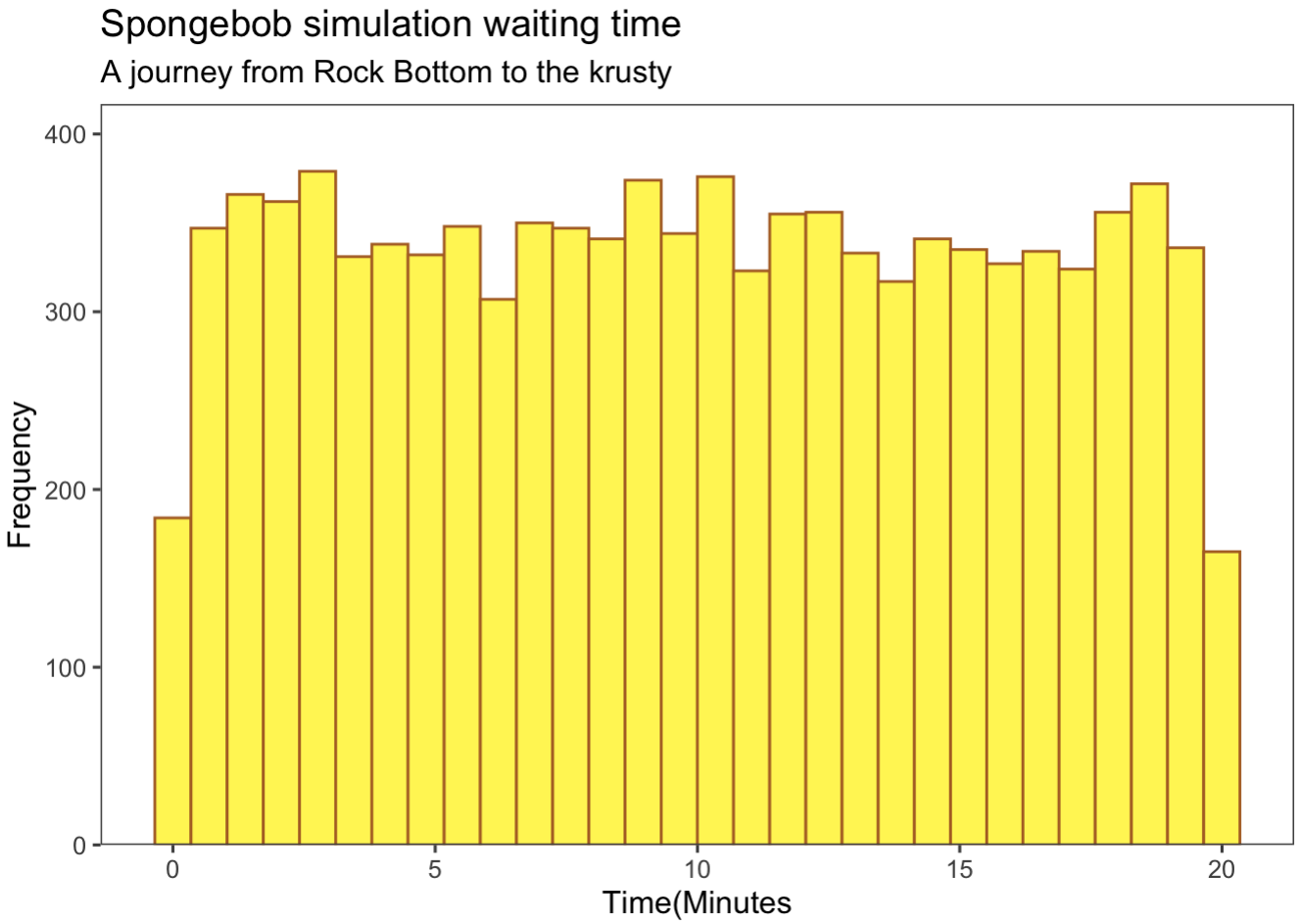
```
#calculate monte carlo estimate for spongebob's median waiting time
quantile(waiting_times$total, probs =0.50)
```

```
##      50%
## 34.81804
```

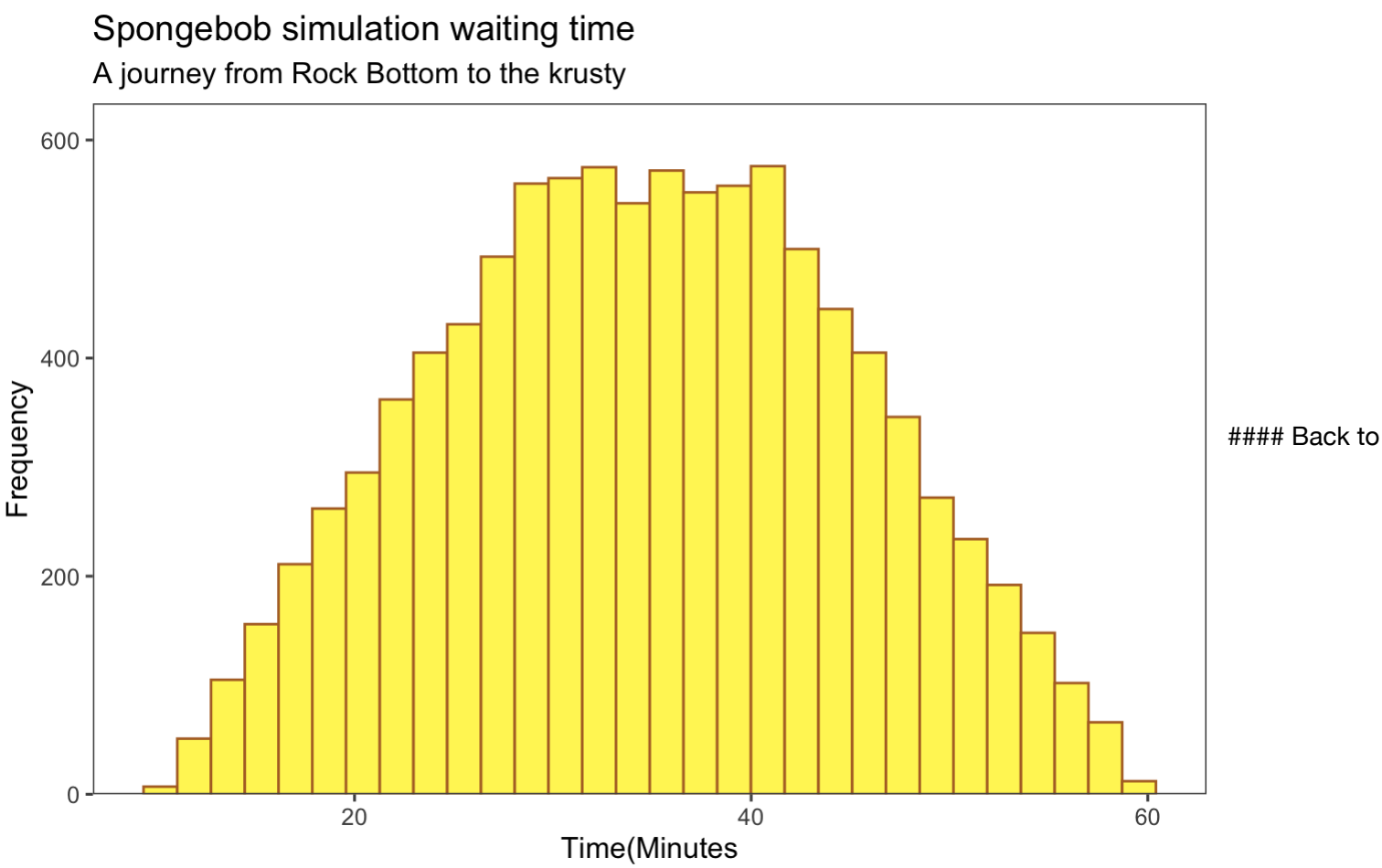
Construct and interpret a 95% confidence interval for the total waiting time in minutes for Spongebob until a bus comes for him and Patrick to take to the Krusty Krab using the quantile() function.

First, let's visualize spongebob's simulated waiting times for sponge and patrick

```
waiting_times %>%
  ggplot(aes(x = spongebob)) +
  geom_histogram(color = "#b26e2d", fill = "#fff463") +  # You can adjust the number of bins
  scale_y_continuous(expand = expansion(mult = c(0, 0.1))) +
  labs(title = "Spongebob simulation waiting time",
       subtitle = "A journey from Rock Bottom to the krusty",
       x = "Time(Minutes",
       y = "Frequency")
```




```
waiting_times %>%
  ggplot(aes(x = total)) +
  geom_histogram(color = "#b26e2d", fill = "#fff463") +  # You can adjust the number of bins
  scale_y_continuous(expand = expansion(mult = c(0, 0.1))) +
  labs(title = "Spongebob simulation waiting time",
       subtitle = "A journey from Rock Bottom to the krusty",
       x = "Time(Minutes",
       y = "Frequency")
```



Back to

"*Construct and interpret a 95% confidence interval for the total waiting time in minutes for Spongebob until a bus comes for him and Patrick to take to the Krusty Krab using the quantile() function.*"

```
quantile(waiting_times$total, probs = c(0.025, 0.95))
```

```
##     2.5%      95%
## 15.60744 52.05422
```

We expect sponge's total waiting time untill he and patrick are on bus heading towards the krusty Krab to be between 15.45 and 54.52 minutes, 95% of the time

PERMUTATION & RANDOMIZATION TESTS

```
# Simulating clinical trial data with participant tumor changes
set.seed(1994)
n1 <- 8
n2 <- 7
trialData <- tibble(Participant = 1:(n1 + n2),
                    Treatment = rep(c("T-cell", "Standard of care"), times = c(n1, n2)),
                    Prop_Shrinkage = c(rbeta(n = n1, shape1 = 4, shape2 = 12),
                                       runif(n = n2, min = -0.2, max = 0.25)))
```

Next, implement Welch's two-sample t-test to test if the reduction in the tumor volume is larger for the T-cell treatment group compared to the standard of care.

```
library("tidymodels")
#impliment the two-sample t-test for our clinical trial data

tResult <- t_test(x=trialData,
                  response = Prop_Shrinkage,
                  explanatory =Treatment,
                  order= c("T-cell","Standard of care"),
                  alternative="greater")
```

Our hypothesis for this test are

$$\mathrm{H-O}: \mu_{\text{T-cell}} > \mu_{\text{SOC}} \text{vs.} \mathrm{H_a}: \mu_{\text{textT-cell}} > \mu_{\text{SOC}},$$

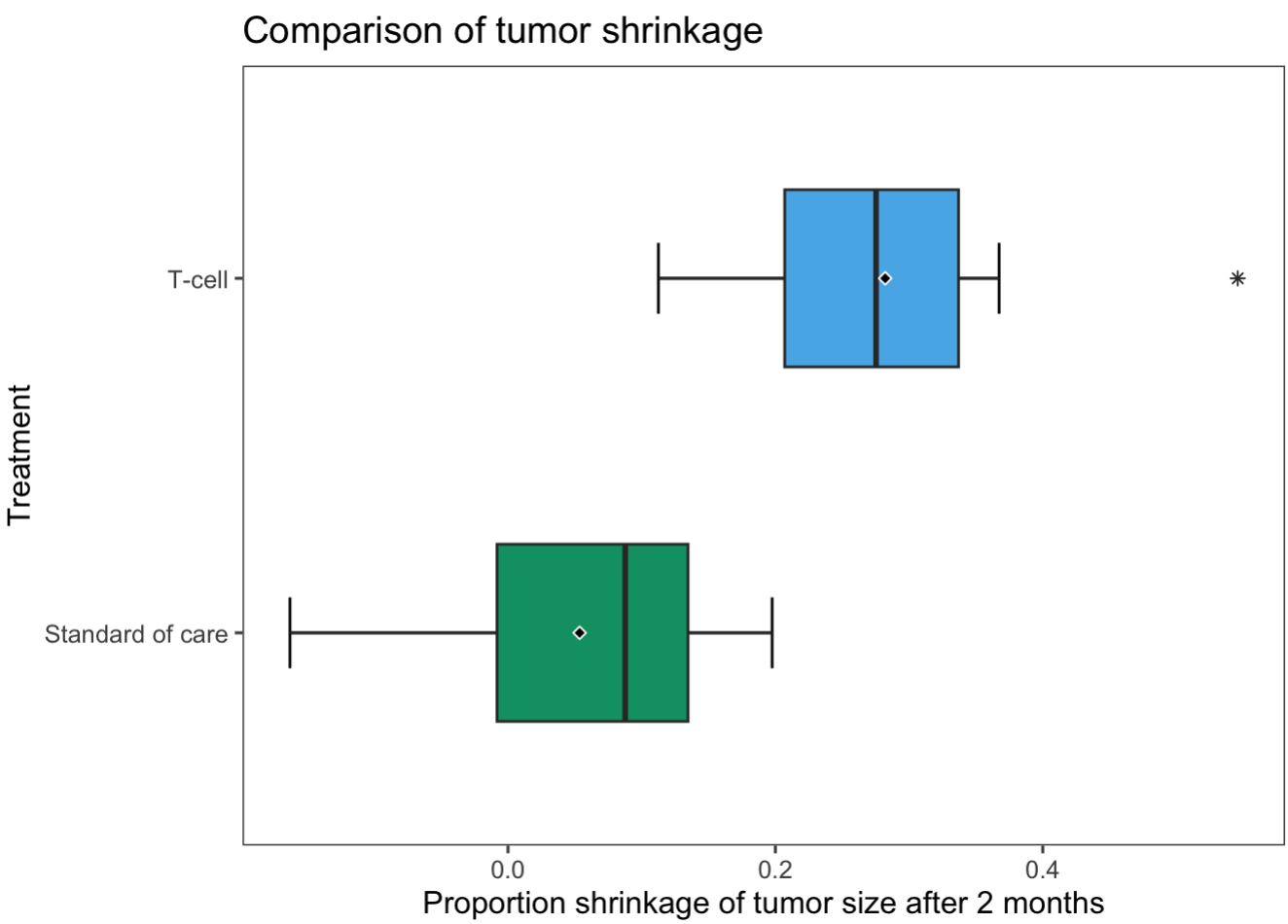
where μ_k is the average propotion reduction in the volume of a participant's tumor afyter two months of treatment for treatment k

```
tResult %>%
  flextable::flextable()
```

statistic	t_df	p_value	alternative	estimate	lower_ci	upper_ci
3.235097	12.88542	0.003289609	greater	0.2285642	0.1033601	Inf

But are these results reliable?

```
trialData %>%
  ggplot(aes(x = Treatment, y = Prop_Shrinkage,
                         fill = Treatment)) +
  stat_boxplot(geom = "errorbar", width = 0.2, coef = 1.5) +
  stat_boxplot(geom = "boxplot", width = 0.5, coef = 1.5,
               outlier.shape = 8) +
  stat_summary(fun = "mean", geom = "point", shape = 23, fill = "black",
               color = "white") +
  scale_fill_manual(values = c("#009E73", "#56B4E9")) +
    coord_flip() +
    labs(y = "Proportion shrinkage of tumor size after 2 months",
         title = "Comparison of tumor shrinkage") +
  theme(legend.position = "none")
```




```
# Calculating standard deviations and variances for each group
trialData %>%
  group_by(Treatment) %>%
  summarize(Mean = mean(Prop_Shrinkage),
            n = n(),
            SD = sd(Prop_Shrinkage),
            Variance = var(Prop_Shrinkage)) %>%
  flextable::flextable()
```

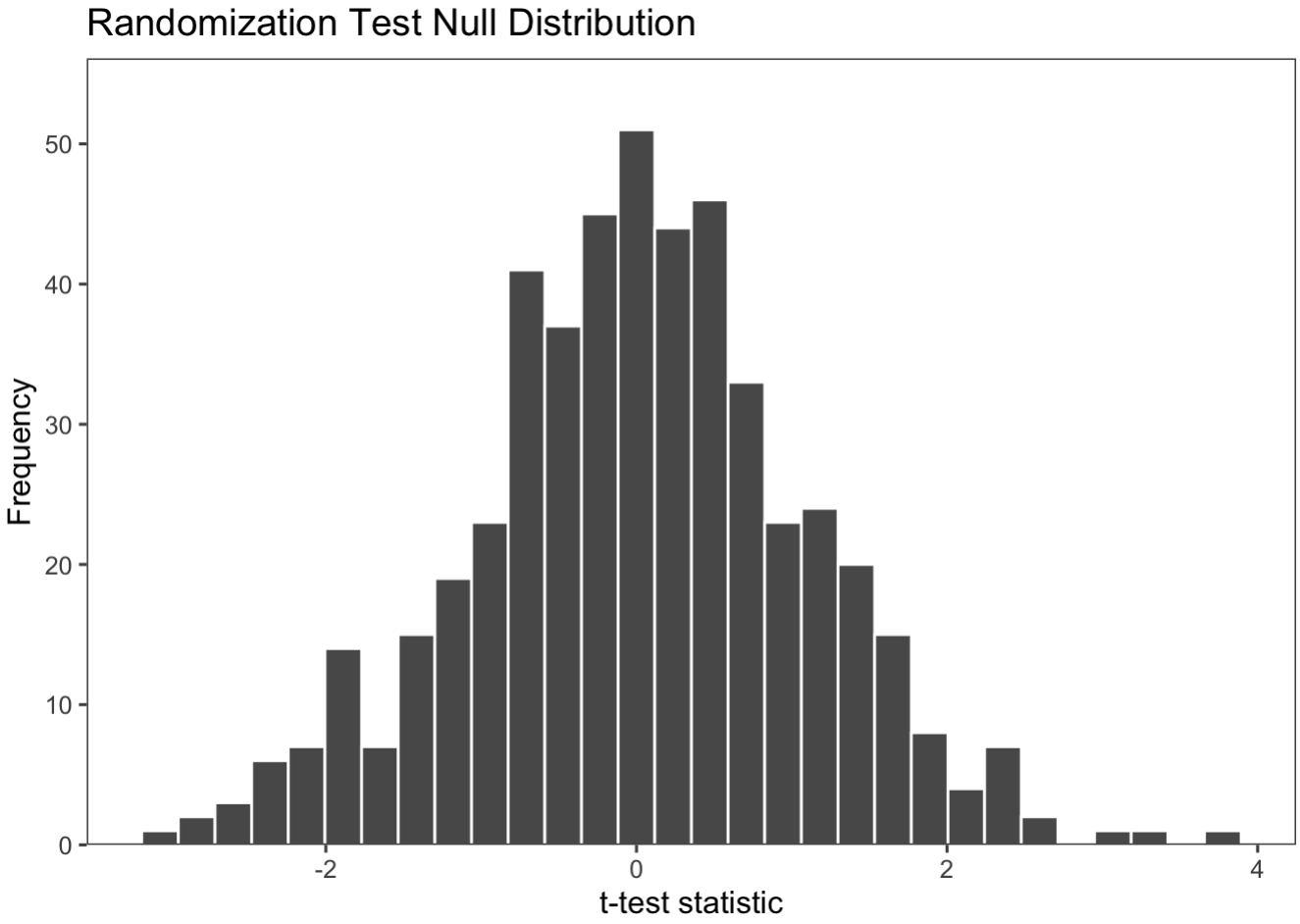
Treatment	Mean	n	SD	Variance
Standard of care	0.05357786	7	0.1333019	0.01776940
T-cell	0.28214204	8	0.1400896	0.01962511

Let's implement the randomization test

```
# Number of permutations to do
nperms <- 500

# Instantiating vector for test statistics
permTs <- vector(length = nperms)

# Calculating t-test statistic for each permutation
for(p in 1:nperms) {
  permTs[p] <- trialData %>%
    mutate(Treatment = sample(Treatment, replace = FALSE)) %>%
  t_test(response = Prop_Shrinkage,
         explanatory = Treatment,
         order = c("T-cell", "Standard of care"),
         alternative = "greater") %>%
    pull(statistic)
}
```

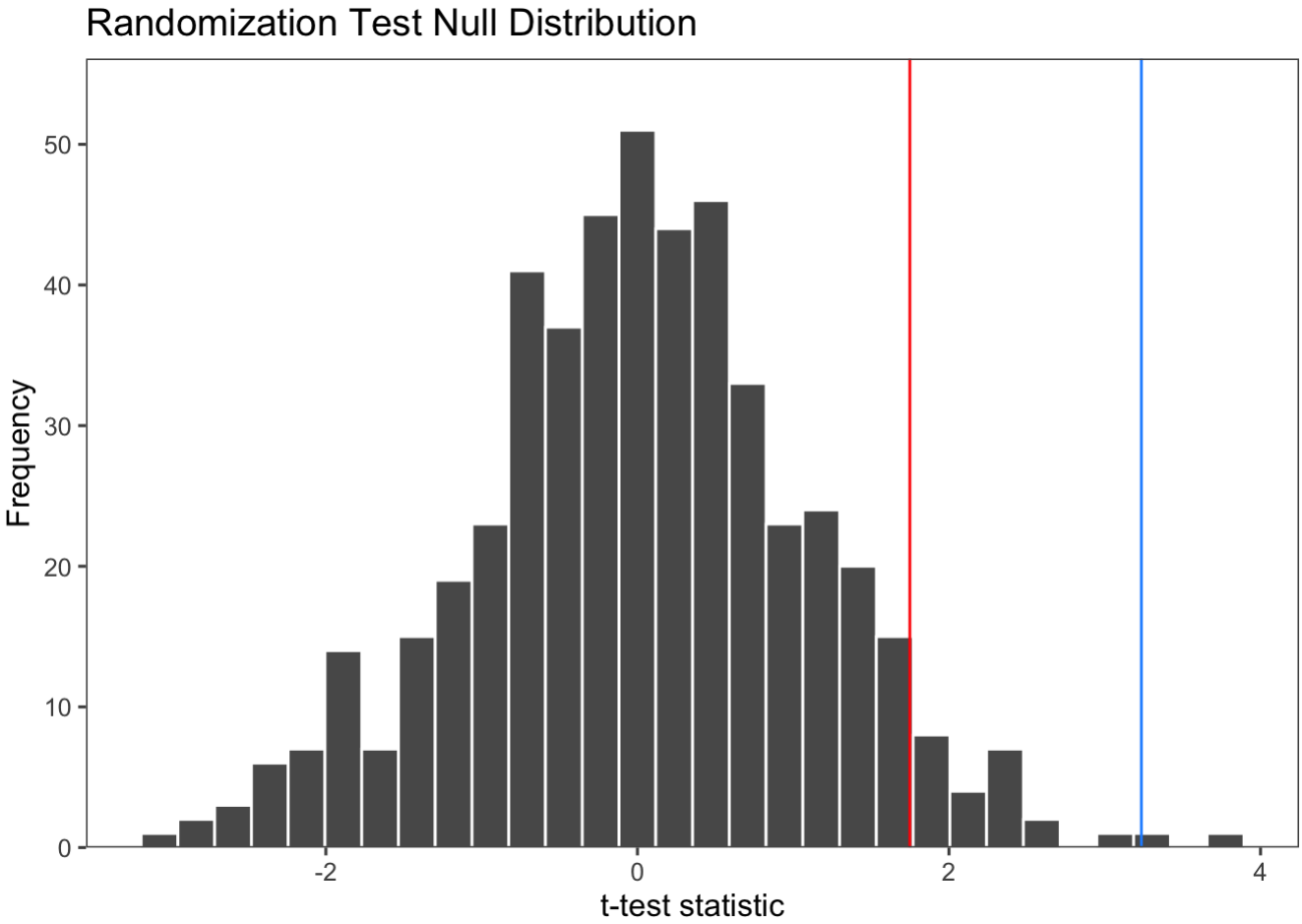
Next we visualize randomization test null distribution

```
#tidying up the t-test statistic

tidt_ts <- tibble(Statistic = permTs)

tidt_ts %>%
  ggplot(aes(x = Statistic)) +
  geom_histogram(color = "white") +
  labs(title = "Randomization Test Null Distribution",
       x = "t-test statistic",
       y = "Frequency") +
  scale_y_continuous(expand = expansion(mult = c(0, 0.1)))
```



Add vertical lines to the plot to indicate where the 95th percentile is (a red dotted line), and where our observed test statistic is (solid blue line).

```
tidt_ts <- tibble(Statistic = permTs)

tidt_ts %>%
  ggplot(aes(x = Statistic)) +
  geom_histogram(color = "white") +
  labs(title = "Randomization Test Null Distribution",
       x = "t-test statistic",
       y = "Frequency") +
  scale_y_continuous(expand = expansion(mult = c(0, 0.1))) +
  geom_vline(xintercept = quantile(permTs, probs = 0.95), color = "red") +
  geom_vline(xintercept = tResult$statistic, color = "dodgerblue")
```



Calculate the p-value for the randomization test. In this case, the p-value can be calculated as the proportion of randomization test statistics greater than or equal to our observed t-test statistic due to our stated hypotheses.

```
#calculating --value for randomization test
mean(permTs >= tResult$statistic)
```

```
## [1] 0.002
```

```
janitor::tabyl(permTs >=tResult$statistic)
```

```
##  permTs >= tResult$statistic   n percent
##                        FALSE 499   0.998
##                         TRUE   1   0.002
```

We reject the null hypothesis at the 5% significance level since our p-value (0.002) is less than 0.05 **Interpretation in context**: We have sufficient evidance that the average propotion reduction in the tumor size of patients after two months of treatment is greator for the T-cell therapy compared to the standard of care at 5% significant