Hypothesis Testing

Part 1: One and two sample t-tests

QantArch Week 5 | 07-03-2022

Universiteit Leiden

One sample t test

$$t = \frac{\bar{x} - \mu}{s_x} \qquad s_x = \frac{s}{\sqrt{n}}$$

One sample t test

Let's test the `Area` variable against a population mean.

Wait… have you uploaded your data?

```r
pits_data <- readr::read_csv(here("data/derived_data/pits-data.csv"))
```

One sample t test

Let's test the `Area` variable against a population mean.

We will randomly generate the true population mean μ from a normal distribution.

```r
set.seed(42) # make sure we generate the same number every time
(pop_mean <- rnorm(1, mean = mean(pits_data$Area, na.rm = T), 1))
```

```
## [1] 238.9387
```

The mean of Area is `mean(pits_data$Area, na.rm = T)` = **237.5677778**

The standard deviation is `sd(pits_data$Area, na.rm = T)` = **94.4995673**

The sample size is `length(!is.na(pits_data$Area))` = **91**

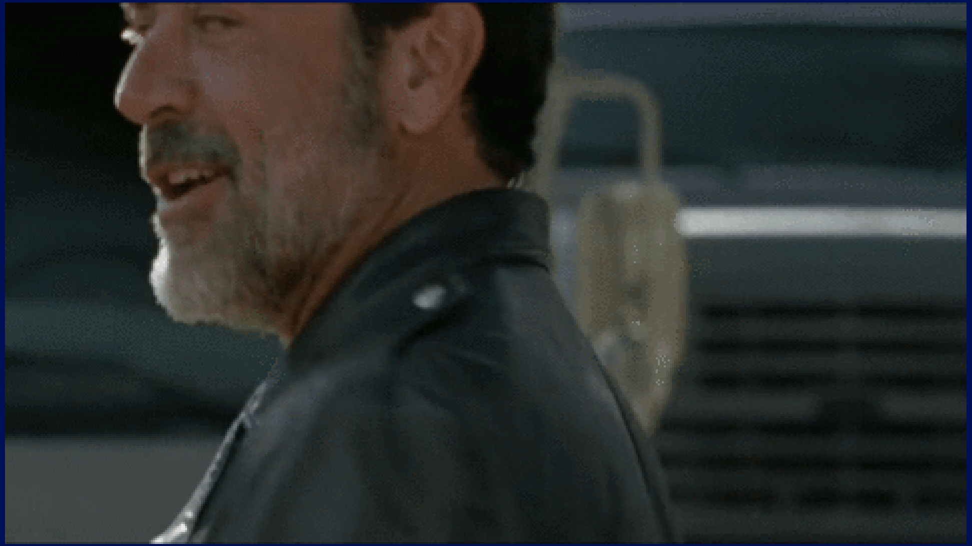
One sample t-test

$$t = \frac{\bar{x} - \mu}{s_x} \qquad s_x = \frac{s}{\sqrt{n}}$$

Now we can just run that through our calculatoR…

```r
t_stat <- (237.568 - 238.939) / (94.500 / sqrt(91))
```

Our t-statistic is **-0.1383969**



$$t = \frac{\bar{x} - \mu}{s_x}$$

One sample t-test: A better option

We can translate the t-test equation to code

First the nominator (or difference in means)

```
t_nom <- mean(pits_data$Area, na.rm = T) - pop_mean
```

then the sample standard deviation

```
sd_sample <- sd(pits_data$Area, na.rm = T)
```

and the sample size… why `sum` not `length`?

```
n_sample <- sum(!is.na(pits_data$Area))
```

Put it together

```
t_stat <- t_nom / (sd_sample / sqrt(n_sample))
```

Our t-statistic is **-0.1376308**.

Now how do we evaluate the t-statistic?

We can calculate the critical t-value

First we set out α. Let's say 0.047212… it's just as arbitrary as 0.05

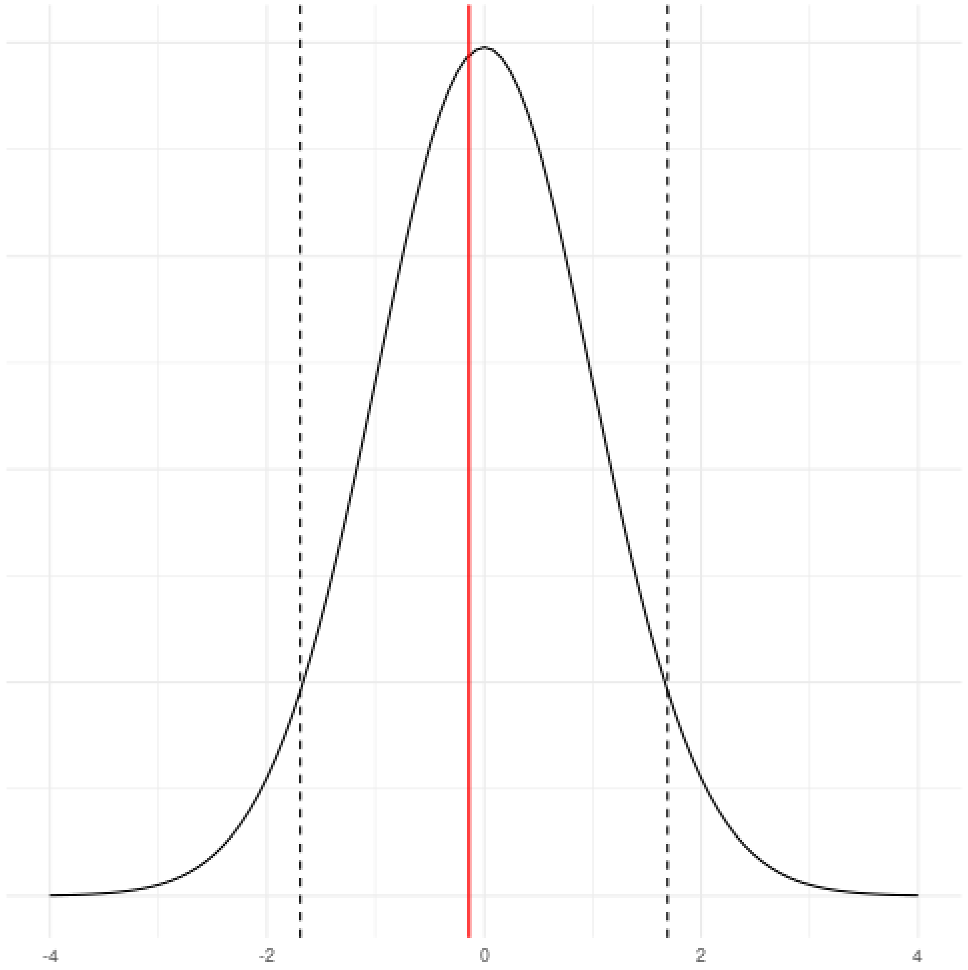
```
alpha <- 0.047212
```

Now we use the alpha and the degrees of freedom (df) to calculate the critical t-value

```
qt(1 - alpha, n_sample - 1)
```

```
## [1] 1.690541
```

Let's see where our t-value falls within the t-distribution with the calculate critical t-values.



Can we use this method to calculate t for other variables, too?

AIN'T NOBODY GOT TIME FOR THAT
THE DAILY SHOW with TREVOR NOAH

One sample t-test: A betterer option

Functions

We can create a function!

Starting with a name

```r
t_stat <- function(){
  # body of the function
}
```

Then we can take the code we used earlier,

```r
 t_nom <- mean(pits_data$Area, na.rm = T) - pop_mean
 sd_sample <- sd(pits_data$Area, na.rm = T)
 n_sample <- sum(!is.na(pits_data$Area))
 t_stat <- t_nom / (sd_sample / sqrt(n_sample))
```

One sample t-test: A betterer option

Functions

We can create a function!

Starting with a name

```r
t_stat <- function(){
  # body of the function
}
```

Then we can take the code we used earlier, and put it inside the body of the function

```r
t_calc <- function(){
  t_nom <- mean(pits_data$Area, na.rm = T) - pop_mean
  sd_sample <- sd(pits_data$Area, na.rm = T)
  n_sample <- sum(!is.na(pits_data$Area))
  t_stat <- t_nom / (sd_sample / sqrt(n_sample))
}
```

One sample t-test: A betterer option

Functions

and now we just need the value we want the function to `return`.

```r
t_calc <- function(){
  t_nom <- mean(pits_data$Area, na.rm = T) - pop_mean
  sd_sample <- sd(pits_data$Area, na.rm = T)
  n_sample <- sum(!is.na(pits_data$Area))
  t_stat <- t_nom / (sd_sample / sqrt(n_sample))
}
```

One sample t-test: A betterer option

Functions

we need to specify which object we want the function to `return`.

```
t_calc <- function(){
  t_nom <- mean(pits_data$Area, na.rm = T) - pop_mean
  sd_sample <- sd(pits_data$Area, na.rm = T)
  n_sample <- sum(!is.na(pits_data$Area))
  t_stat <- t_nom / (sd_sample / sqrt(n_sample))
  return(t_stat)
}
```

We can run the function by calling its name

```
t_calc()
```

```
## [1] -0.1376308
```

One sample t-test: A betterer option

Functions

To make sure we can use the same function for different variables, we need to give it arguments and remove `Area`-specific objects

```r
t_calc <- function(){
  t_nom <- mean(pits_data$Area, na.rm = T) - pop_mean
  sd_sample <- sd(pits_data$Area, na.rm = T)
  n_sample <- sum(!is.na(pits_data$Area))
  t_stat <- t_nom / (sd_sample / sqrt(n_sample))
  return(t_stat)
}
```

One sample t-test: A betterer option

Functions

To make sure we can use the same function for different variables, we need to give it arguments and remove `Area`-specific objects

```r
t_calc <- function(variable, pop_mean){
  t_nom <- mean(variable, na.rm = T) - pop_mean
  sd_sample <- sd(pits_data$Area, na.rm = T)
  n_sample <- sum(!is.na(pits_data$Area))
  t_stat <- t_nom / (sd_sample / sqrt(n_sample))
  return(t_stat)
}
```

One sample t-test: A betterer option

Functions

To make sure we can use the same function for different variables, we need to give it arguments and remove `Area`-specific objects

```r
t_calc <- function(variable, pop_mean){
  t_nom <- mean(variable, na.rm = T) - pop_mean
  sd_sample <- sd(variable, na.rm = T)
  n_sample <- sum(!is.na(variable))
  t_stat <- t_nom / (sd_sample / sqrt(n_sample))
  return(t_stat)
}
```

One sample t-test: A betterer option

Functions

To make sure we can use the same function for different variables, we need to give it arguments and remove `Area`-specific objects

```r
t_calc <- function(variable, pop_mean){
  t_nom <- mean(variable, na.rm = T) - pop_mean
  sd_sample <- sd(variable, na.rm = T)
  n_sample <- sum(!is.na(variable))
  t_stat <- t_nom / (sd_sample / sqrt(n_sample))
  return(t_stat)
}
```

The last two lines don't need changing, because all the objects are calculated within the function.

Exercise

Use the function to calculate the t-statistic and critical t-value for `Length`

We'll pretend that the true population mean is **1.72**, and we'll set our α at **0.05**.

Solution

```r
# t-statistic
t_stat <- t_calc(pits_data$Points, 1.72)
t_stat
```

```
## [1] 2.159668
```

```r
# p-value
alpha <- 0.05
df <- sum(!is.na(pits_data$Points)) - 1
qt(1 - alpha, df)
```

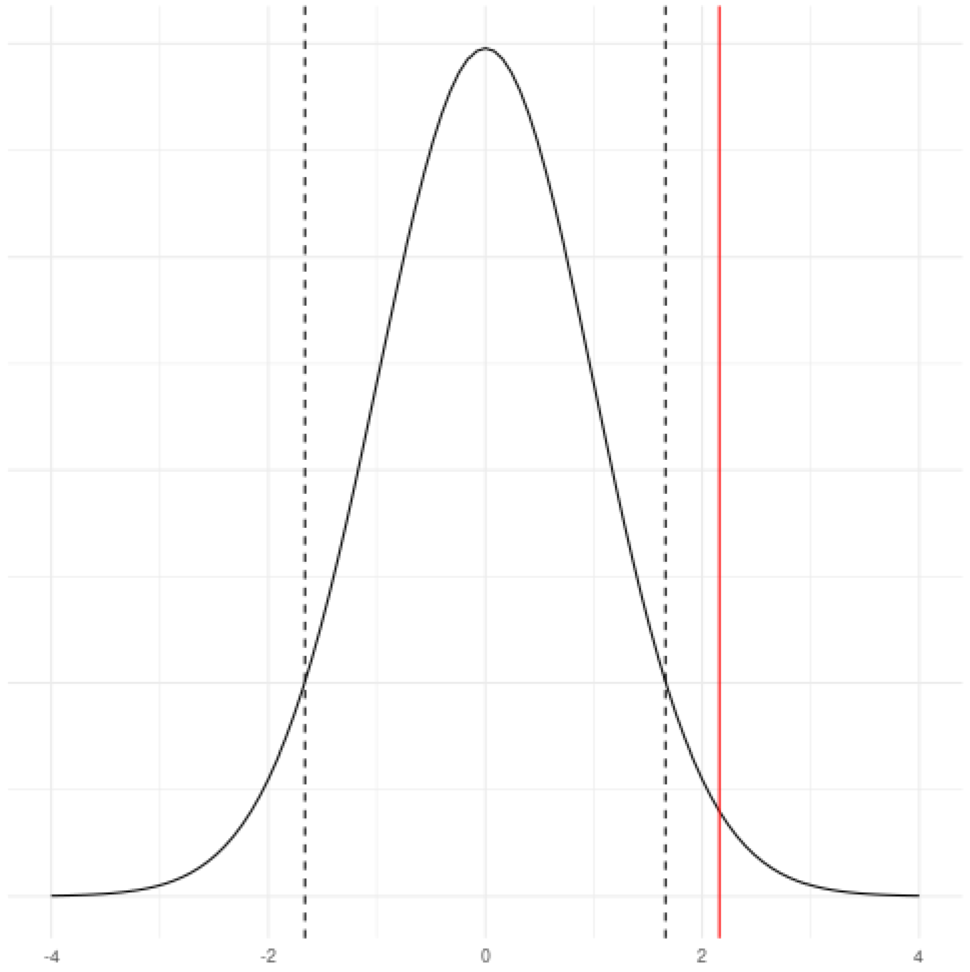
```
## [1] 1.662354
```

Break

Evaluating the t-statistic

Let's look at the critical values for our `Points` statistic.

As you may be able to tell, the overall process is not a dichotomy.



Evaluating the t-statistic

As you may be able to tell, the overall process is not a dichotomy.

We can produce a probability value of the data/estimate in relation to the 'null hypothesis'.

p-value

> *Informally, a p-value is the probability under a specified statistical model that a statistical summary of the data (e.g., the sample mean difference between two compared groups) would be equal to or more extreme than its observed value.* (American Statistical Association)

The p-value is NOT the probability of your hypothesis. Or any hypothesis for that matter…

Evaluating the t-statistic

Our α determines the critical threshold for our p-value in order to reject the null hypothesis.

Our α from the previous example is 0.05,

which means we need a p-value equal to, or lower than 0.05.

Calculating the p-value

```r
2 * pt(-abs(t_stat), n_sample - 1)
```

```
## [1] 0.03348712
```

Two sample t-test

Exercise

Create a function to calculate the t-statistic of a two sample t-test

$$t = \frac{\bar{x} - \bar{y}}{s_{x-y}}$$

$$s_{x-y} = \sqrt{\frac{(n_x - 1)s_x^2 + (n_y - 1)s_y^2}{n_x + n_y - 2}(\frac{1}{n_x} + \frac{1}{n_y})}$$

💡 **Hint:** Code > Rainbow Parentheses

Solution

```r
t_calc2 <- function(x, y){
  mean_x <- mean(x, na.rm = T)
  mean_y <- mean(y, na.rm = T)
  n_x <- sum(!is.na(x))
  n_y <- sum(!is.na(y))
  s_x <- sd(x, na.rm = T)
  s_y <- sd(y, na.rm = T)

  s_xy <- sqrt(
    (((n_x-1)*s_x^2 + (n_y-1)*s_y^2) /
      (n_x + n_y -2)) *
      (n_x^-1 + n_y^-1)
    )
  t_stat <- (mean_x - mean_y) / s_xy
  return(t_stat)
}
```

Two sample t-test

Now let's compare Areas of each `Location`.

```r
inside <- filter(pits_data, Location == "Inside")
outside <- filter(pits_data, Location == "Outside")
t_stat2 <- t_calc2(inside$Area, outside$Area)
t_stat2
```

```
## [1] 9.901939
```

```r
df <- sum(!is.na(pits_data$Area)) - 1
p_value <- 2 * pt(-abs(t_stat2), df = df)
p_value
```

```
## [1] 5.104513e-16
```

We can check our results from the one and two sample t-tests with the built-in function `t.test`

For a one sample test we need to specify `mu`

```
pop_mean <- 238.9387
t.test(pits_data$Area, mu = pop_mean)
```

```
##
##      One Sample t-test
##
## data:  pits_data$Area
## t = -0.13763, df = 89, p-value = 0.8908
## alternative hypothesis: true mean is not equal to 238.9387
## 95 percent confidence interval:
##  217.7752 257.3603
## sample estimates:
## mean of x
##  237.5678
```

For a two sample test we need the two variables we are comparing (and `var.equal = T` for now).

```r
t.test(inside$Area, outside$Area, var.equal = T)
```

```
##
##      Two Sample t-test
##
## data:  inside$Area and outside$Area
## t = 9.9019, df = 88, p-value = 5.715e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  110.4006 165.8415
## sample estimates:
## mean of x mean of y
##  317.3711  179.2500
```

Yes, we could have done that all along…

But you learned something along the way, right? …right??

Reporting results

The mean area of house pits Inside (m = 317) the "white wall" is 138 sqft larger than the mean area of houses outside (m = 179) the "white wall", $t(89) = 9.9$, $p < 0.001$, 95%CI(110, 166).

Note: Report exact p-value if $p > 0.001$

Or a table

```
t.test(inside$Area, outside$Area, var.equal = T) %>%
  tidy()
```

```
## # A tibble: 1 × 10
##   estimate estimate1 estimate2 statistic  p.value parameter conf.low conf.high
##      <dbl>     <dbl>     <dbl>     <dbl>    <dbl>     <dbl>    <dbl>     <dbl>
## 1     138.      317.      179.      9.90 5.72e-16        88     110.      166.
## # … with 2 more variables: method <chr>, alternative <chr>
```