t-test

Sebastian Heucke

9/27/2019

Contents

t-test

This document is meant to clarify t-tests. It is mostly based on content of the book http://leanpub.com/dataanalysisforthelifesciences which you can get for free.

Introduction

I will decribe how to obtain a p-value in practice, to understand the concept I will not use the t.test() function which R offers but will construct a t-statistic from "scratch".

The data

The data consist of two populations: female mice on control diets and female mice on high fat diets, with weight being the outcome of interest. To get the dataset:

```
library(downloader)
```

```
##
## Attaching package: 'downloader'
```

```
## The following object is masked from 'package:devtools':
##
##     source_url
```

```
url <- "https://raw.githubusercontent.com/genomicsclass/dagdata/master/inst/extdata/femaleMiceWeights.cs
filename <- "femaleMiceWeights.csv"
if(!file.exists("femaleMiceWeights.csv")) download(url,destfile=filename)
dat <- read.csv(filename)
```

This is how the data look like:

```
head(dat)
```

```
##   Diet Bodyweight
## 1 chow      21.51
## 2 chow      28.14
## 3 chow      24.04
## 4 chow      23.45
## 5 chow      23.68
## 6 chow      19.79
```

The data will now be separated by phenotyp, this is done by creating two vectors, one for the control population and one for the high-fat diet population:

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
control <- filter(dat,Diet=="chow") %>% select(Bodyweight) %>% unlist
treatment <- filter(dat,Diet=="hf") %>% select(Bodyweight) %>% unlist
```

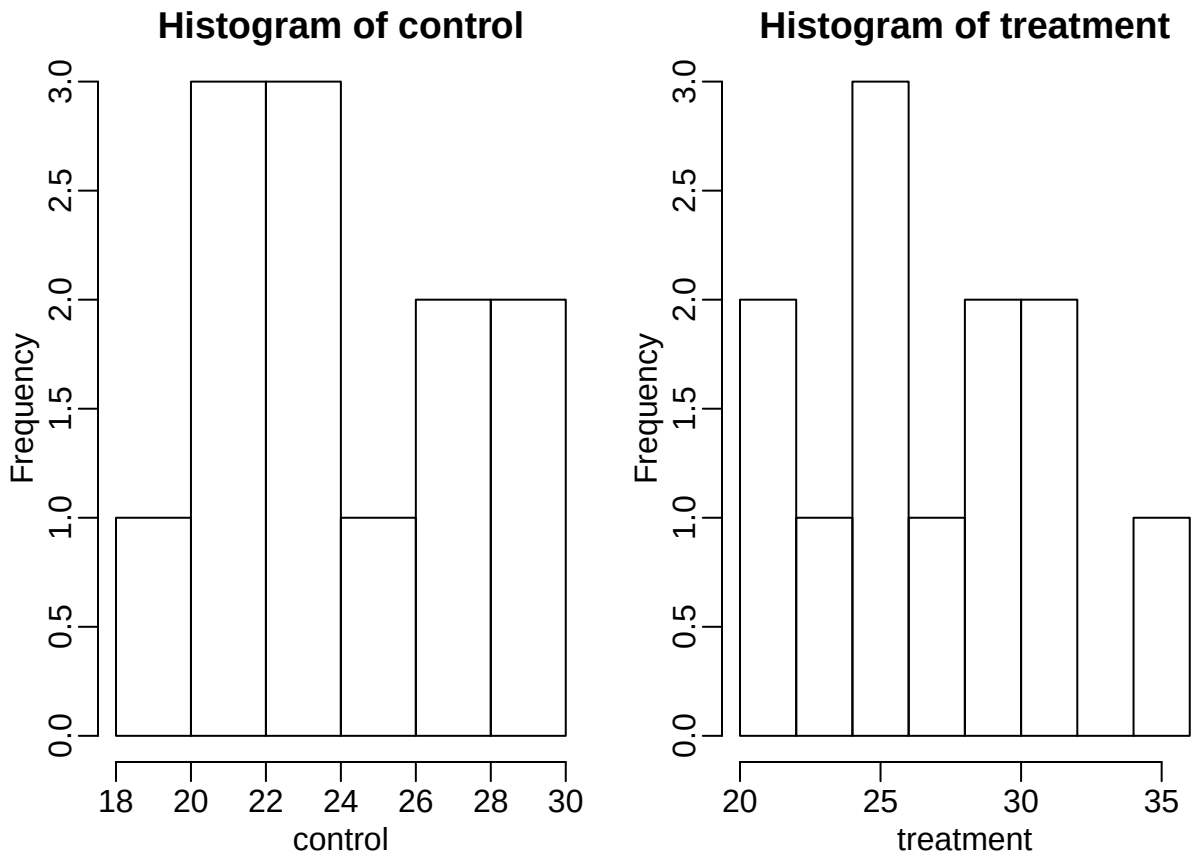
To have a look at the distribution for both controls and high-fat diet mice:

```
library(rafalib)
```

```
##
## Attaching package: 'rafalib'
```

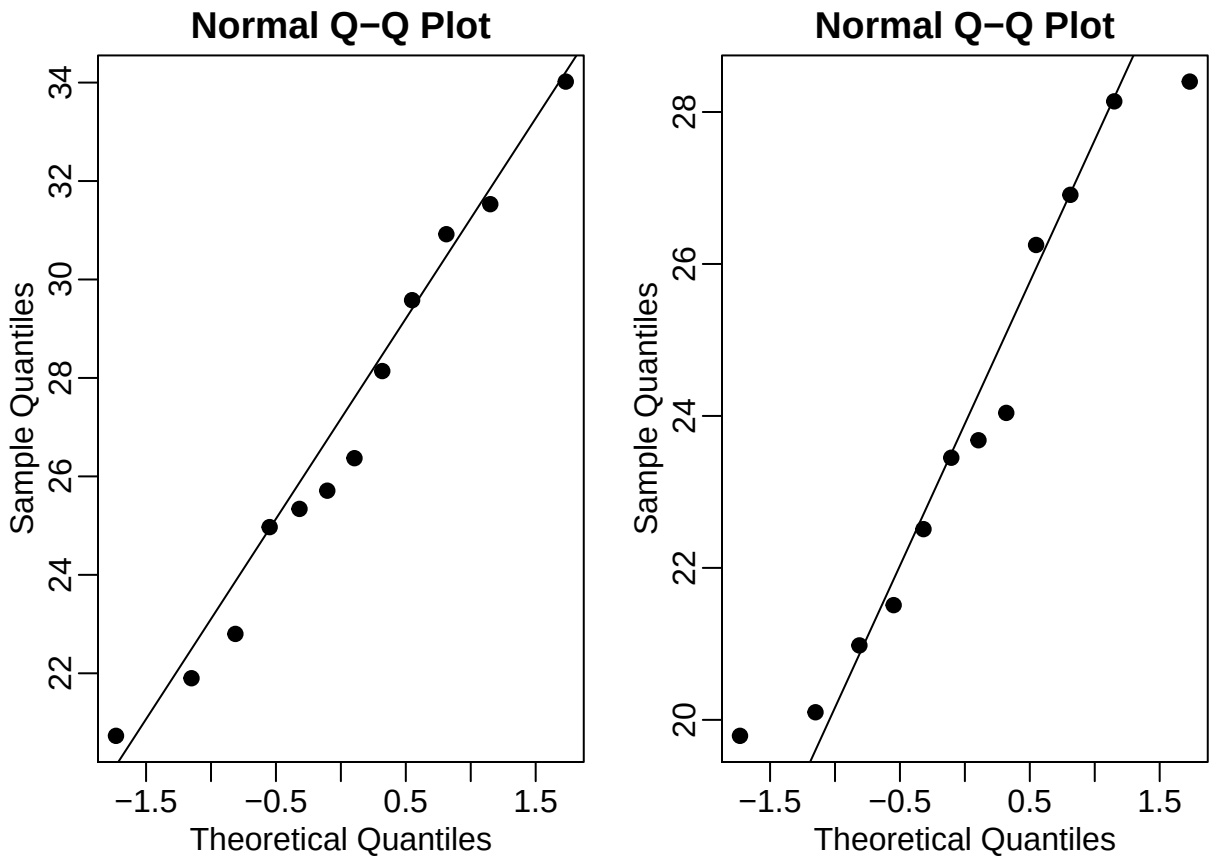
```
## The following object is masked from 'package:devtools':
##
##     install_bioc
```

```
mypar(1,2)
hist(control)
hist(treatment)
```



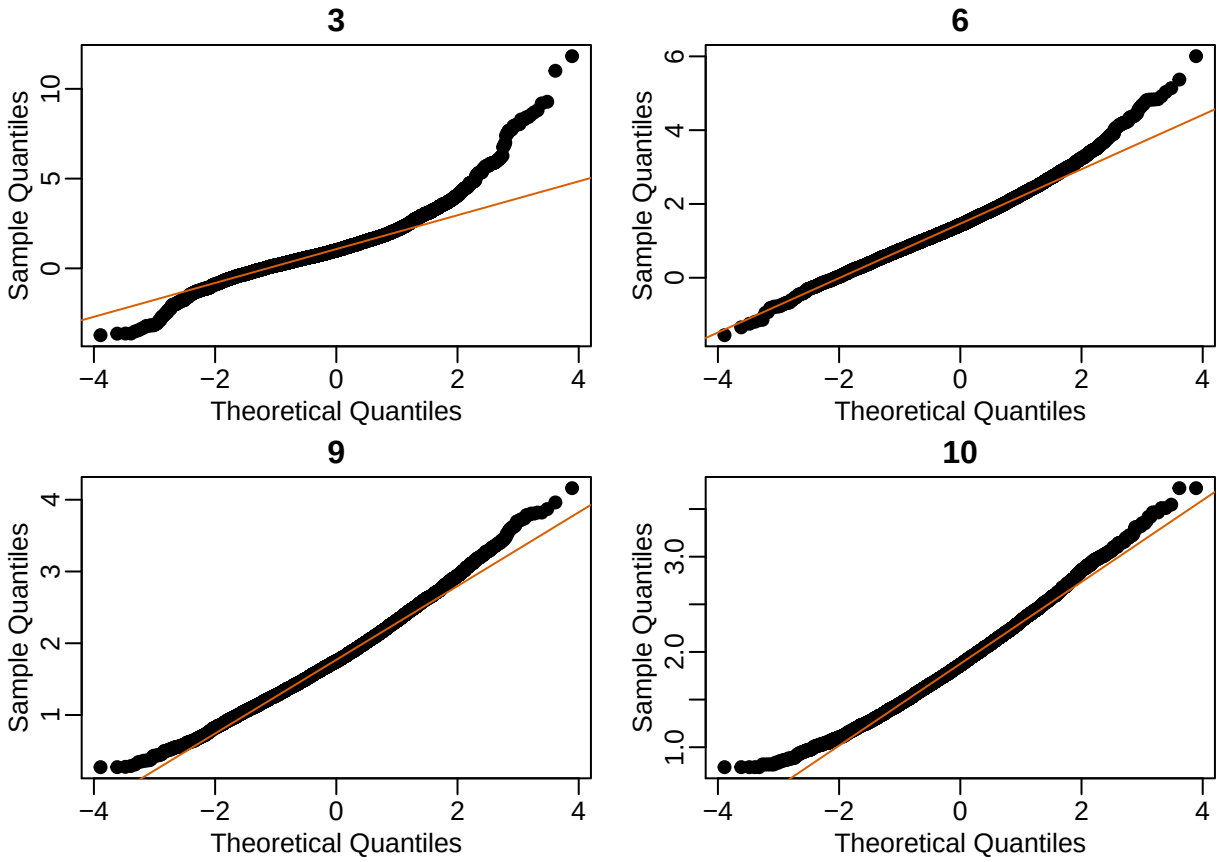
To confirm that the distributions are relatively close to being normally distributed *qq-plots* can be used. This plot compares data on the y-axis against a theoretical distribution on the x-axis. If the points fall on the identity line, then the data is close to the theoretical distribution.

```
mypar(1,2)
qqnorm(treatment)
qqline(treatment)
qqnorm(control)
qqline(control)
```



The larger the sample, the more forgiving the result is to the weakness of this approximation. Below is a simulation with samplesize N=3, 6, 9, 12.

```
Ns <- c(3,6,9,10)
B <- 10000 #number of simulations
##function to compute a t-stat
computetstat <- function(n) {
y <- sample(treatment,n)
x <- sample(control,n)
(mean(y)-mean(x))/sqrt(var(y)/n+var(x)/n)
}
res <- sapply(Ns,function(n) {
replicate(B,computetstat(n))
})
mypar(2,2)
for (i in seq(along=Ns)) {
qqnorm(res[,i],main=Ns[i])
qqline(res[,i],col=2)
}
```



We are asked to report a p-value. What do we do? What you want to know is if the difference of averages, referred to as the *observed effect size*, which is a random variable, is significantly different.

```
control <- filter(dat,Diet=="chow") %>% select(Bodyweight) %>% unlist
treatment <- filter(dat,Diet=="hf") %>% select(Bodyweight) %>% unlist
diff <- mean(treatment) - mean(control)
print(diff)
```

```
## [1] 3.020833
```

What about the standard error? The standard error of this random variable is the population standard deviation divided by the square root of the sample size:

$$SE(\bar{X}) = \sigma/\sqrt{N}$$

The sample standard deviation is used as an estimate of the population standard deviation.

```
sd(control)/sqrt(length(control))
```

```
## [1] 0.8725323
```

This is the SE of the sample average, but we actually want the SE of the difference. The variance of the difference of two random variables is the sum of its variances. Compute the variance and take the square root.

```
se <- sqrt(
var(treatment)/length(treatment) +
var(control)/length(control)
)
```

The t-statistic is the ratio of the random variable *diff* by its SE.

```
tstat <- diff/se
tstat
```

```
## [1] 2.055174
```

So now to calculate a p-value all we need to do is ask: how often does a normally distributed random variable exceed *diff*? R has a built-in function, *pnorm*, to answer this specific question, pnorm(a) returns the probability that a random variable following the standard normal distribution falls below a. To obtain the probability that is larger than a, we use 1-pnorm(a). We want to know the probability of seeing something as extreme as *diff*: either smaller (more negative) or larger than diff. We call these two regions "tails".

```
righttail <- 1 - pnorm(abs(tstat))
lefttail <- pnorm(-abs(tstat))
pval <- lefttail + righttail
print(pval)
```

```
## [1] 0.0398622
```

In this case, the p-value is smaller than 0.05 and using the conventional cutoff 0.05, we would call the difference *statistically significant*.

The t-distribution

The Central Limit Theorem relies on large samples. When the CLT does not apply, there is another option that does not rely on *asymptotic results*. If the distribution of the population is normal, then we can work out the exact distribution of the t-statistic. With small samples, it is hard to check if the population is normal. If we use this approximation, then statistical theory tells us that the distribution of the random variable *ttstat* follows a t-distribution. The t-distribution has a location parameter like the normal and another parameter called *degrees of freedom*.

```
result <- t.test(treatment, control)
result
```

```
##
##  Welch Two Sample t-test
##
## data:  treatment and control
## t = 2.0552, df = 20.236, p-value = 0.053
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.04296563  6.08463229
## sample estimates:
## mean of x mean of y
##  26.83417  23.81333
```

The p-value is slighly bigger now. This is to be expected because the Central Limit Theorem approximation considers the denominator of tstat practically fixed (for large samples sizes), while the t-distribution approximation takes into accout that the denominator (standard error of the difference) is a random variable. The smaller the sample size, the more the denominator varies.

Confidence Intervals

Additionally to reporting p-values a confidence interval can provide a very important piece of information: the effect size. Sometimes the effect size is divided by the mean of the control group and so expressed as a percent increase. A confidence interval includes information about your estimated effect size and the uncertainty associated with this estimate.

Confidence Interval for Population Mean

First construct a confidence interval for the population mean of control female mice.

```
dat <- read.csv("mice_pheno.csv")
chowPopulation <- dat[dat$Sex=="F" & dat$Diet=="chow",3]
```

The population average μ_x is our parameter of interest here:

```
mu_chow <- mean(chowPopulation)
print(mu_chow)
```

```
## [1] 23.89338
```

In real life you rarely have the entire population, therefore we estimate the popolation mean, as we did for p-values.

```
N <- 30
chow <- sample(chowPopulation,N)
print(mean(chow))
```

```
## [1] 24.64767
```

Because this is a random variable, the sample average will not be a perfect estimate. In fact, because in this example we know the value of the parameter, we can see that they are not exactly the same. A confidence interval is a statistical way of reporting our finding, the sample average, in a way that explictly summarizes the variability of our random variable.

Defining the Interval A 95% confidence interval is a random interval with a 95% probability of falling on the parameter we are estimating. Again, the definition of the confidence interval is that 95% of *random intervals will contain the true, fixed value μ_x.

```
# standard error
se <- sd(chow)/sqrt(N)
print(se)
```

```
## [1] 0.6408885
```

```
Q <- qnorm(1- 0.05/2)
interval <- c(mean(chow)-Q*se, mean(chow)+Q*se )
interval
```

```
## [1] 23.39155 25.90379
```

The interval edges should be smaller on the left und greater on the right side then the mean.

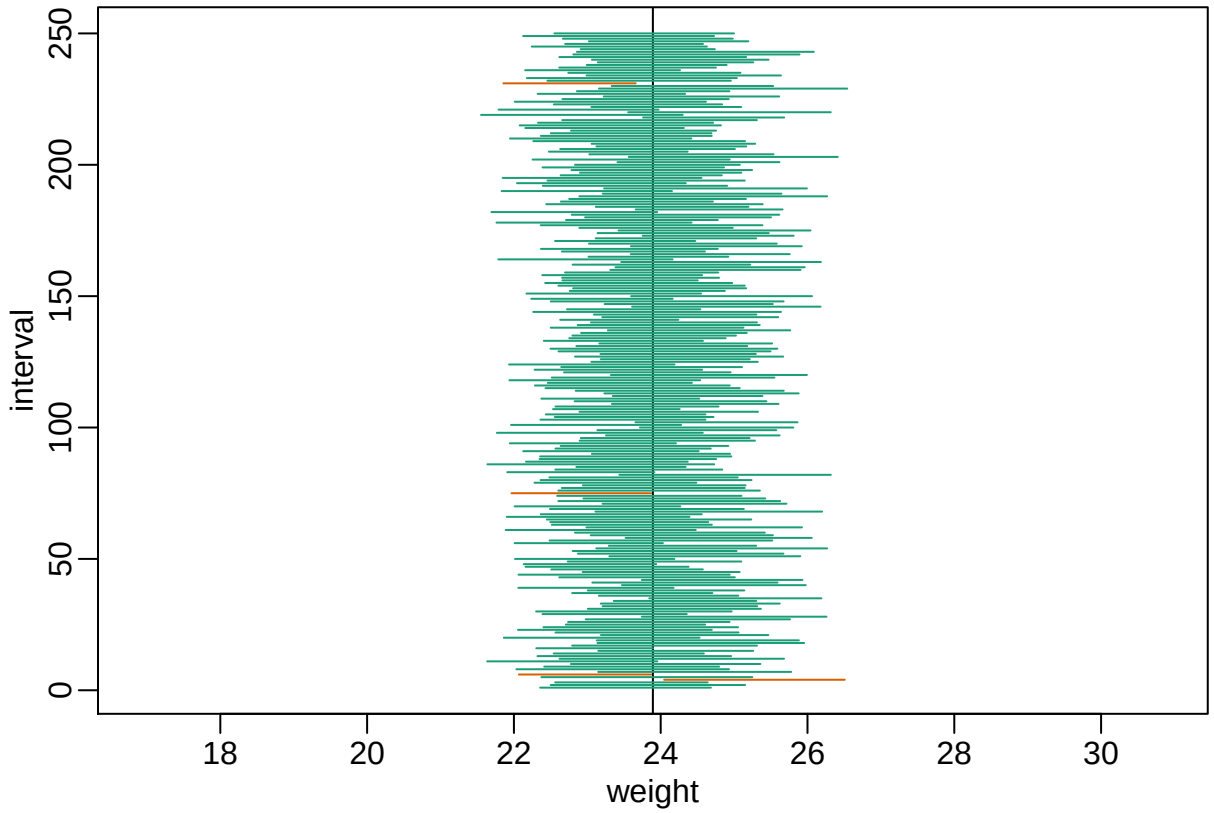
```
interval[1] < mu_chow & interval[2] > mu_chow
```

```
## [1] TRUE
```

We will cover 95% of the time. This can be simulated. 5% of the cases will fail to cover μ_x.

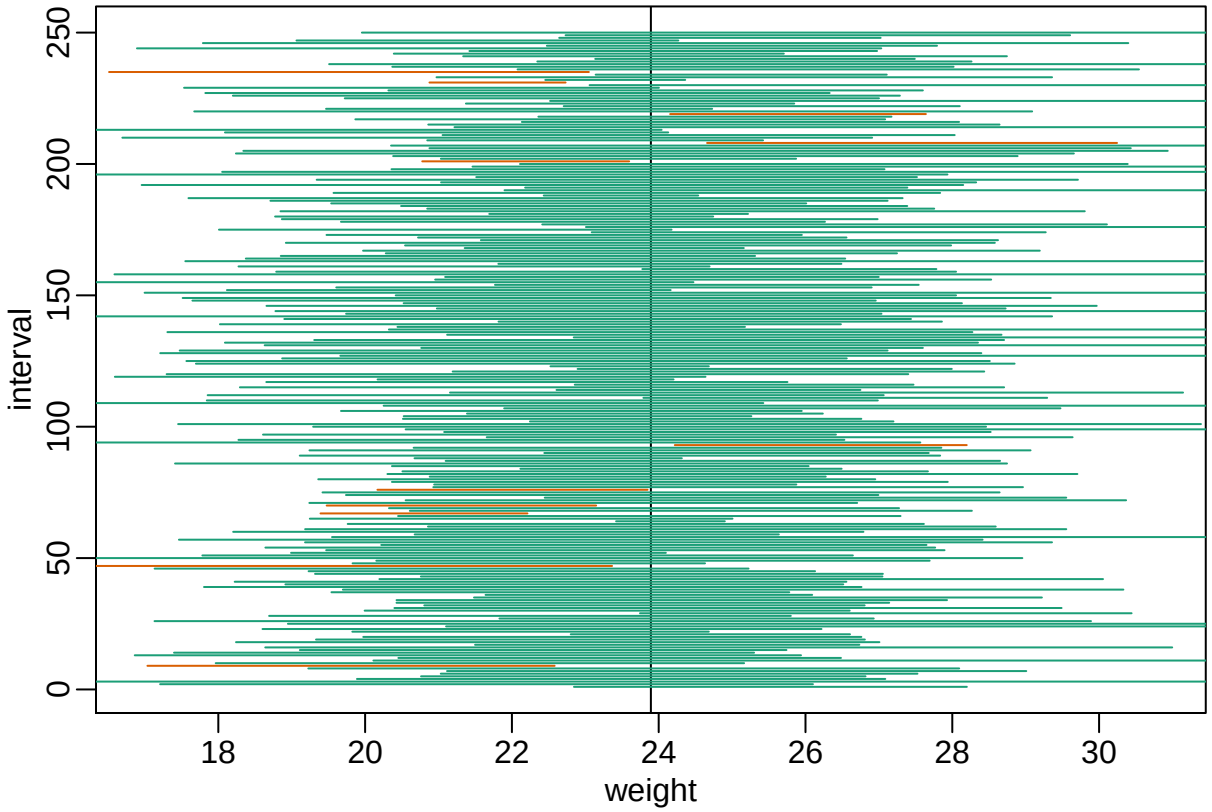
```
library(rafalib)
B <- 250
mypar()
plot(mean(chowPopulation)+c(-7,7),c(1,1),type="n",
xlab="weight",ylab="interval",ylim=c(1,B))
abline(v=mean(chowPopulation))
for (i in 1:B) {
chow <- sample(chowPopulation,N)
se <- sd(chow)/sqrt(N)
interval <- c(mean(chow)-Q*se, mean(chow)+Q*se)
```

```
covered <-
mean(chowPopulation) <= interval[2] & mean(chowPopulation) >= interval[1]
color <- ifelse(covered,1,2)
lines(interval, c(i,i),col=color)
}
```



Small Sample Size With small sample sizes the confidence interval becomes larger, to be able to cover 95% of the time.

```
mypar()
plot(mean(chowPopulation) + c(-7,7), c(1,1), type="n",
xlab="weight", ylab="interval", ylim=c(1,B))
abline(v=mean(chowPopulation))
##Q <- qnorm(1- 0.05/2) ##no longer normal so use:
Q <- qt(1- 0.05/2, df=4)
N <- 5
for (i in 1:B) {
chow <- sample(chowPopulation, N)
se <- sd(chow)/sqrt(N)
interval <- c(mean(chow)-Q*se, mean(chow)+Q*se )
covered <- mean(chowPopulation) <= interval[2] & mean(chowPopulation) >= interval[1]
color <- ifelse(covered,1,2)
lines(interval, c(i,i),col=color)
}
```




```
t.test(treatment,control,conf.level=0.9)$conf.int
```

```
## [1] 0.4871597 5.5545070
## attr(,"conf.level")
## [1] 0.9
```

Power Calculations

For low amount of samples we might fail to see a p-value smaller than 0.05.

```
set.seed(1)
# sample size 12
N <- 12
hf <- sample(treatment,N)
control <- sample(control,N)
t.test(hf,control)$p.value
```

```
## [1] 0.05299888
```

```
# sample size 3
N <- 3
hf <- sample(treatment,N)
control <- sample(control,N)
t.test(hf,control)$p.value
```

```
## [1] 0.9824993
```

Did we make a mistake, by not rejecting the null hypothesis, are we saying the diet has no effect? The answer

to this question is no. All we say i s that we did not reject the null hypothesis. But this does not necessarily imply that the null is true. The problem is that, in this particular instance we don't have enough *power*.

Types of Error

Whenever a statistival test is performed, there will be a chance of making a mistake. This is why our p-values are not 0. Under the null, there is always a positive, perhaps very small chance that we will reject the null when it is true. If the p-value is 0.05, it will happen 1 out of 20 times. This *error* is called *type I error* (false positive).

A type I error is defined as rejecting the null when we should not. The reason you dont use infinitesimal cut-offs to avoid type I errors at all cost is that there is another error, to not reject the null when we should. This is called *type II error* or false negative.

Power Calculation

Power is the probability of rejecting the null when the null is false.

Suppose the sample size is:

```
N <- 12

dat <- read.csv("mice_pheno.csv") #Previously downloaded
controlPopulation <- filter(dat,Sex == "F" & Diet == "chow") %>%
select(Bodyweight) %>% unlist
hfPopulation <- filter(dat,Sex == "F" & Diet == "hf") %>%
select(Bodyweight) %>% unlist
```

and we will reject the null hypothesis at:

```
alpha <- 0.05
```

What is the power with this particular data? To compute the probability we simulate taking a sample of size N from both control and treatment groups and perform a t-test comparing these two, and report if the p-value is less than *alpha* or not.

```
# number of simulations
B <- 2000
reject <- function(N, alpha=0.05){
hf <- sample(hfPopulation,N)
control <- sample(controlPopulation,N)
pval <- t.test(hf,control)$p.value
pval < alpha
}

# use replictate to run B times
rejections <- replicate(B,reject(N))
mean(rejections)
```

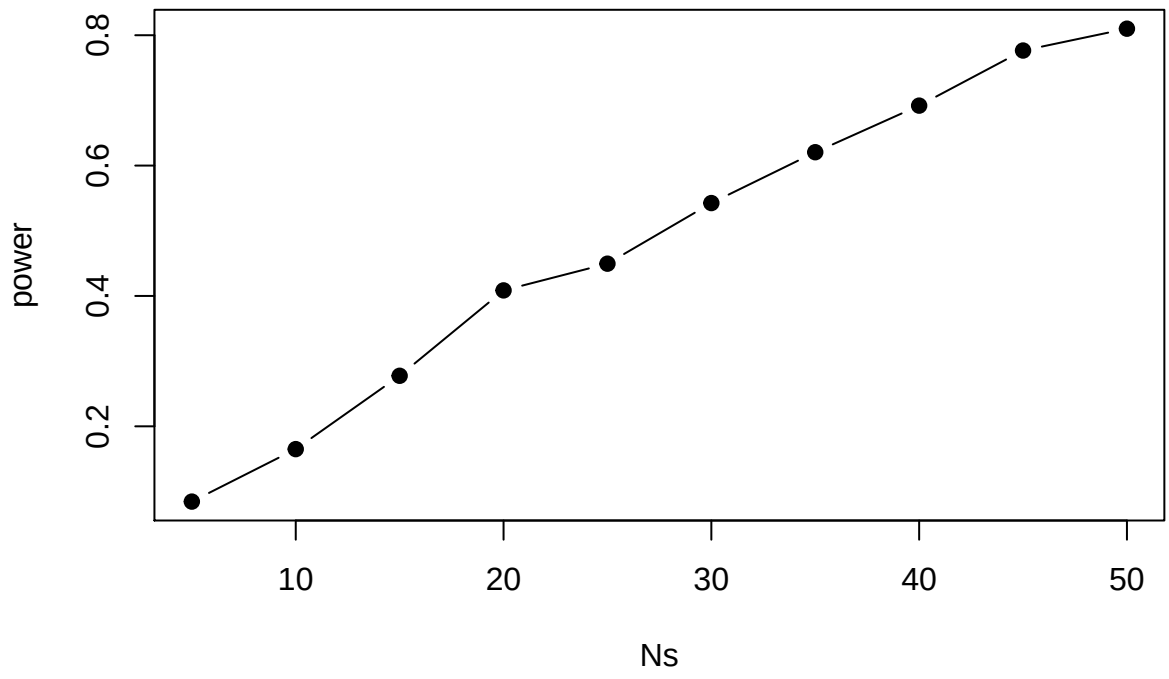
```
## [1] 0.2135
```

The power is the proportion of times we correctly reject. So with $N = 12$ the power is only about 21%. Let's improve with N. We will use the function *sapply*, which applies a function to each of the elements of a vector. We want to repeat the following sample size:

```
# sample size from 5, 10, 15 ...up to 50 by step 5
Ns <- seq(5, 50, 5)
```

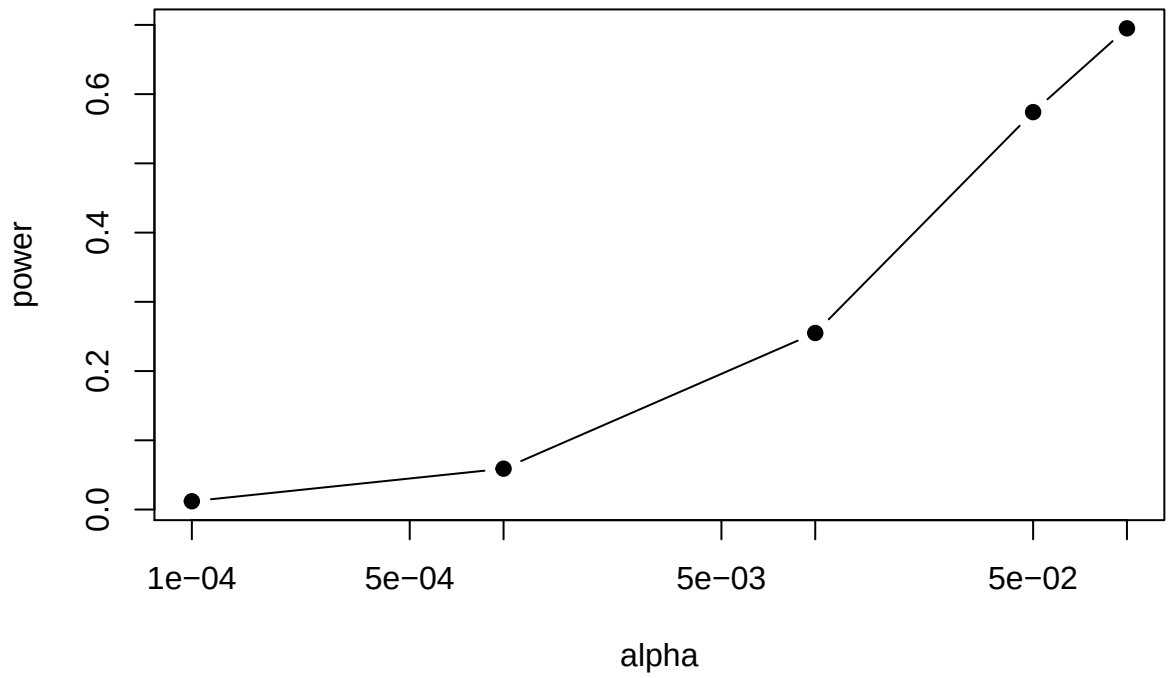
```
power <- sapply(Ns,function(N){
rejections <- replicate(B, reject(N))
mean(rejections)
})

plot(Ns, power, type="b")
```



Similarily, if we chance the level *alpha* at which we reject, power changes. The smaller the chance of type I error to be, the less power you get. You trade of between the two types of error.

```
N <- 30
alphas <- c(0.1,0.05,0.01,0.001,0.0001)
power <- sapply(alphas,function(alpha){
rejections <- replicate(B,reject(N,alpha=alpha))
mean(rejections)
})
plot(alphas, power, xlab="alpha", type="b", log="x")
```



Session information

```
## R version 3.6.1 (2019-07-05)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 18.04.3 LTS
##
## Matrix products: default
## BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.7.1
## LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.7.1
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] rafalib_1.0.0  dplyr_0.8.3    downloader_0.4 knitr_1.23
## [5] devtools_2.1.0 usethis_1.5.1
##
```

```
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.1         RColorBrewer_1.1-2 compiler_3.6.1
##  [4] pillar_1.4.2       prettyunits_1.0.2  remotes_2.1.0
##  [7] tools_3.6.1        testthat_2.1.1     digest_0.6.20
## [10] pkgbuild_1.0.5     pkgload_1.0.2      evaluate_0.14
## [13] memoise_1.1.0      tibble_2.1.3       pkgconfig_2.0.2
## [16] rlang_0.4.0        cli_1.1.0          yaml_2.2.0
## [19] xfun_0.8           withr_2.1.2        stringr_1.4.0
## [22] desc_1.2.0         fs_1.3.1           tidyselect_0.2.5
## [25] rprojroot_1.3-2    glue_1.3.1         R6_2.4.0
## [28] processx_3.4.1     rmarkdown_1.14     sessioninfo_1.1.1
## [31] purrr_0.3.2        callr_3.3.1        magrittr_1.5
## [34] backports_1.1.4    ps_1.3.0           htmltools_0.3.6
## [37] assertthat_0.2.1   stringi_1.4.3      crayon_1.3.4
```

The document was processed on 2019-09-30.