R lecture 2022 11 15

Adam Tallman

2022-11-14

## Packages

library(rlist)  
library(tidyverse)  
library(nhstplot)

## Warning: package 'nhstplot' was built under R version 4.2.2

library(reshape)

## Warning: package 'reshape' was built under R version 4.2.2

library(truncnorm)

## Warning: package 'truncnorm' was built under R version 4.2.2

library(BEST)

## Warning: package 'BEST' was built under R version 4.2.2

## Warning: package 'HDInterval' was built under R version 4.2.2

library(BayesFactor)

## Warning: package 'BayesFactor' was built under R version 4.2.2

## Warning: package 'coda' was built under R version 4.2.2

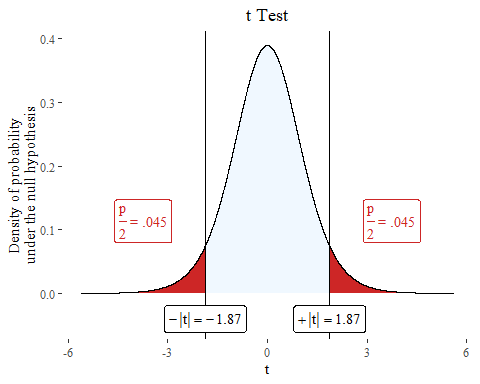
## Simulation of Type I and Type II errors

This is how to simulate a Type II error.

set.seed(123)  
n <- 6  
strong.vowels <- rnorm(n, mean = 85, sd = 10)  
weak.vowels <- rnorm(n, mean = 80, sd = 10)  
strong <- rep("yes", times =n, length.out=n)  
weak <- rep("no", times =n, length.out=n)  
vowel.durations <- list.append(strong.vowels, weak.vowels)  
prominence <- list.append(strong, weak)  
t <- (mean(strong.vowels) - mean(weak.vowels)) / (sqrt((var(strong.vowels)/n) + (var(weak.vowels)/n)))  
t

## [1] 1.873338

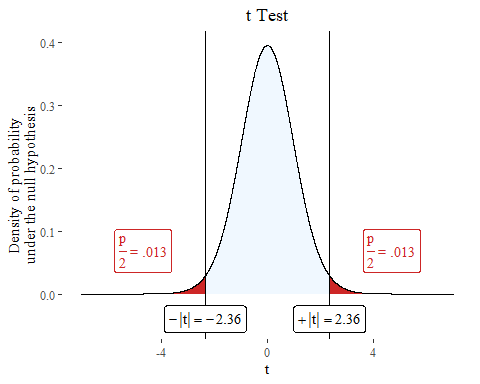
plotttest(t.test(vowel.durations~prominence), tails = "one")



This is how to simulate a Type I error.

set.seed(1234)  
n <- 30  
vowel.durations <- rnorm(n, mean = 80, sd=10)  
h1 <- c("strong","weak")  
h1 <- sample(h1, n, replace=TRUE, prob=c(0.5, 0.5))  
h2 <- c("strong","weak")  
h2 <- sample(h2, n, replace=TRUE, prob=c(0.5, 0.5))  
h3 <- c("strong","weak")  
h3 <- sample(h3, n, replace=TRUE, prob=c(0.5, 0.5))  
h4 <- c("strong","weak")  
h4 <- sample(h4, n, replace=TRUE, prob=c(0.5, 0.5))  
h5 <- c("strong","weak")  
h5 <- sample(h5, n, replace=TRUE, prob=c(0.5, 0.5))  
h6 <- c("strong","weak")  
h6 <- sample(h6, n, replace=TRUE, prob=c(0.5, 0.5))  
df <- data.frame(vowel.durations, h1,h2,h3,h4,h5,h6)

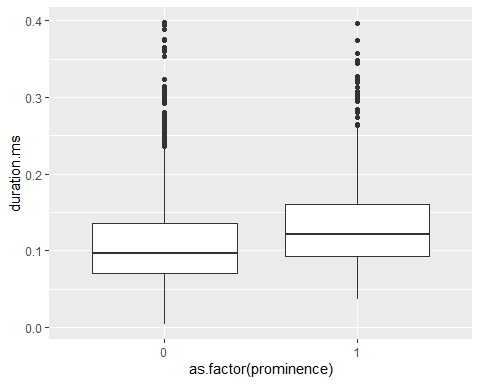
plotttest(t.test(vowel.durations~h4, data=df))



## Confidence interval

df <- read.csv("/Users/Adam/Desktop/araonavowels.csv", header=T)  
df <- subset(df, duration.ms <0.4)  
df$prominence <- ifelse(df$Stress =="y", 1,0)

boxplot <- ggplot(df, aes(as.factor(prominence), duration.ms))+  
 geom\_boxplot()  
boxplot



##Variability of sample means and resampling (a brief review of quantiles)

There’s a function called sample() that samples randomly from a vector.

sample(df$duration.ms, size =10, replace=TRUE)

## [1] 0.062 0.110 0.160 0.137 0.095 0.096 0.081 0.175 0.181 0.072

The values don’t have to be numeric

sample(c("juice", "coke"), size=5, replace=TRUE)

## [1] "juice" "juice" "juice" "juice" "juice"

mean( sample(df$duration.ms[ df$prominence == 0 ],size=40,replace=TRUE))

## [1] 0.104575

mean( sample(df$duration.ms[ df$prominence == 1 ],size=30,replace=TRUE))

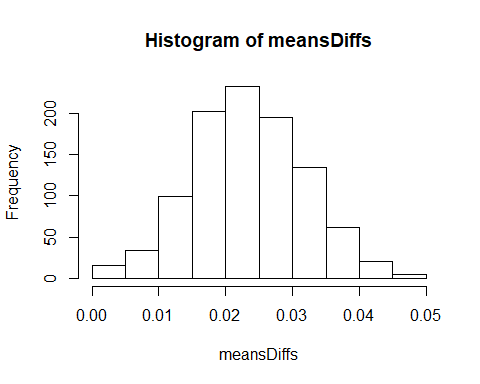
## [1] 0.1447667

set.seed(2)  
mean(sample(df$duration.ms[ df$prominence == 1 ],size=40,replace=TRUE) ) - mean(sample(df$duration.ms[ df$prominence == 0 ],size=40,replace=TRUE) )

## [1] 0.0232

meansDiffs<- replicate(1000, mean( sample(df$duration.ms[ df$prominence == 1 ],size=100,replace=TRUE) ) - mean( sample(df$duration.ms[ df$prominence == 0 ],size=100,replace=TRUE) ))

hist(meansDiffs, col="white")



quantile(meansDiffs, c(0.025, 0.975))

## 2.5% 97.5%   
## 0.00673275 0.03991800

95% of the time the difference between the two groups is between 0.003111875 and 0.045692500.

When we produce p values or confidence intervals (next), the idea behind it is that we are sampling from a population over and over again.

## Confidence intervals

The confidence interval is a measurement of uncertainty. As the interval is larger the uncertainty is higher. The “interval” spans over an interval of values in a probability distribution. So when we say 95% confidence interval, we mean 95% of the probability values.

## An inferential test

Let’s just do a t.test.

t.test(df$duration.ms~df$Stress)

##   
## Welch Two Sample t-test  
##   
## data: df$duration.ms by df$Stress  
## t = -8.0082, df = 1318, p-value = 2.538e-15  
## alternative hypothesis: true difference in means between group n and group y is not equal to 0  
## 95 percent confidence interval:  
## -0.02885846 -0.01750164  
## sample estimates:  
## mean in group n mean in group y   
## 0.1116558 0.1348359

By default its 95% but note that we can change that.

t.test(df$duration.ms~df$Stress, conf.level = 0.88)

##   
## Welch Two Sample t-test  
##   
## data: df$duration.ms by df$Stress  
## t = -8.0082, df = 1318, p-value = 2.538e-15  
## alternative hypothesis: true difference in means between group n and group y is not equal to 0  
## 88 percent confidence interval:  
## -0.02768333 -0.01867678  
## sample estimates:  
## mean in group n mean in group y   
## 0.1116558 0.1348359

## ANOVA based on a simulation

Maybe in school you learnt this

Where m is the intercept, and b is the intercept.

Here’s the form for a typical linear model in statistics.

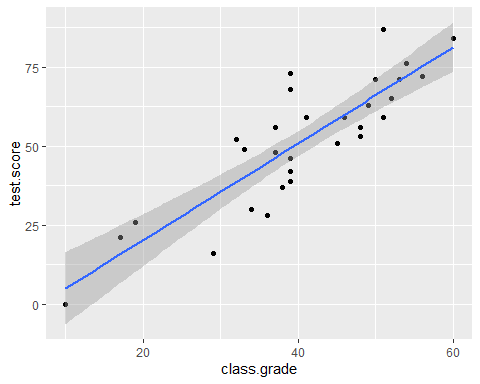
y is the thing you are predicting. a is your intercept, beta is your coefficient, that’s how much a change in x will affect a change in y. We can add more betas in the equation as well. Everytime you see a B or a beta it is referring to a coefficient in a linear model - the slope - how fast does a change x change y.

set.seed(1)  
school <- sample(c("good", "bad"), size=30, replace=30)  
school.binary <- ifelse(school == "good", 1, 0)  
class.grade <- as.integer(rtruncnorm(a=0, b=100, m = 40, sd=15, n =30))  
intercept <- 0  
b1 <- 15  
b2 <- 1  
error <- rnorm(m=0, sd=10, n=30)  
test.score <- intercept + b1\*school.binary +b2\*class.grade + error  
test.score <- ifelse(test.score < 0, 0, test.score)  
test.score <- ifelse(test.score> 100, 100, test.score)  
test.score <- as.integer(test.score)  
data <- data.frame(test.score, school, class.grade)

Let’s look at the relationship between class grade and test score.

ggplot(data,aes(class.grade, test.score)) +  
 geom\_point()+  
 geom\_smooth(method='lm')

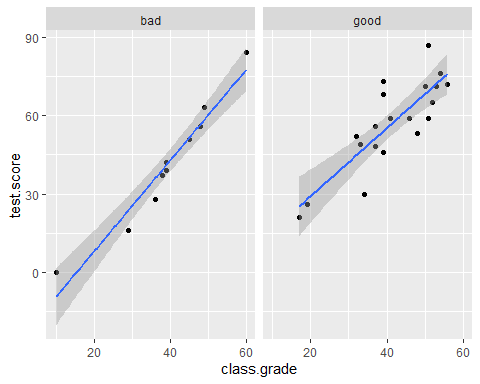
## `geom\_smooth()` using formula 'y ~ x'



If we use facet\_wrap we can look at the relationship across groups. The way we simulated the data, they should be roughly the same.

ggplot(data,aes(class.grade, test.score)) +  
 geom\_point()+  
 geom\_smooth(method='lm')+  
 facet\_wrap(~school)

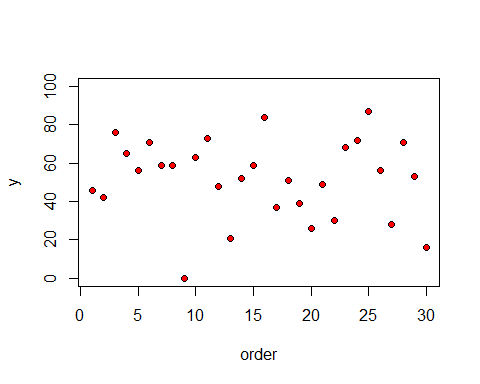
## `geom\_smooth()` using formula 'y ~ x'



attach(data)

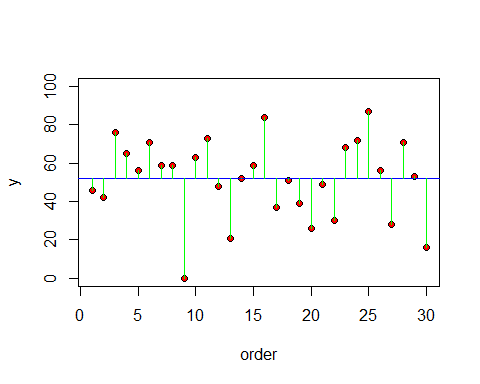
## The following objects are masked \_by\_ .GlobalEnv:  
##   
## class.grade, school, test.score

plot(1:30,test.score,ylim=c(0,100),ylab="y",xlab="order",pch=21,bg="red")



There’s lots of scatter, showing that the variance is large. What would the overall variance look like.

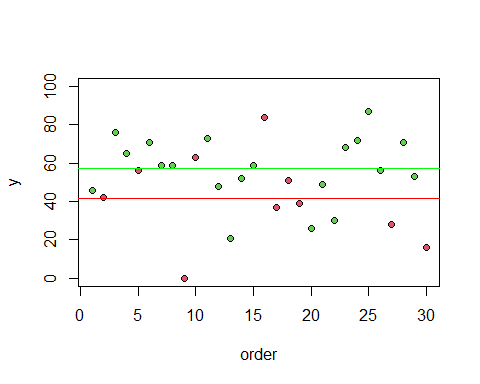
plot(1:30,test.score,ylim=c(0,100),ylab="y",xlab="order",pch=21,bg="red")  
abline(h=mean(test.score),col="blue")  
for(i in 1:100) lines(c(i,i),c(mean(test.score),test.score[i]),col="green")



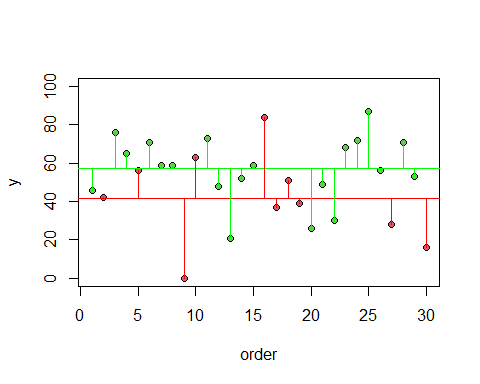
We calculate the total sum squares (SSY). Note that this is the top part of the equation for calculating the variance.

The way an ANOVA works is by fitting means across two difference groups and looking at the variance within groups versus variance across groups. Its easy to understand through visualizations.

plot(1:30,test.score,ylim=c(0,100),ylab="y",xlab="order", pch=21,bg= as.numeric(as.factor(school))+1 )  
abline(h=mean(test.score[school=="good"]),col="green")  
abline(h=mean(test.score[school=="bad"]), col="red")



plot(1:30,test.score,ylim=c(0,100),ylab="y",xlab="order", pch=21,bg= as.numeric(as.factor(school))+1 )  
abline(h=mean(test.score[school=="good"]),col="green")  
abline(h=mean(test.score[school=="bad"]), col="red")  
index <- 1:length(test.score)  
for (i in 1:length(index)){  
if (school[i] == "good")  
lines(c(index[i],index[i]),c(mean(test.score[school=="good"]),test.score  
[i]), col="green")  
else  
lines(c(index[i],index[i]),c(mean(test.score[school=="bad"]),test.score  
[i]), col="red")  
}



The intuition behind the ANOVA is that when the means of the two groups are meaningfully different, then the sum of squares of each group added together will be smaller than the sum of squares computed from the overall mean. So we are interested in whether the variance of the test scores calculated for each group is smaller than that for the test scores of the group as a whole.

To formalize the analysis we need to talk about a new sum of squares: the sum of the squares differences between the individual y values and the mean of the reference group (could be either school - sometimes called the “treatment” in experimental studies) (Crawley 201: 154).

This is the sum of the squares of the lengths of the vertical red lines and the sum of squares of the lengths of the vertical green lines.

We can state this more generally for k levels: j is an index for the level.

The total SSY / total sum squared is broken up into two parts. The explained part of the variation SSG (sum of squared values from adding the distinction between the good and bad schools) and the unexplained part of the variation SSE.

To get SSY we just

SSY= sum(( test.score - mean(test.score) )^2)

Just as an aside we get the variance by dividing by the sample size minus 1 In case you are wondering where the “variance” aspect of analysis of variance comes from).

SSY/(30-1)

## [1] 426.4379

var(test.score)

## [1] 426.4379

The SSE is the sum of the sqaures residuals calculated for each group.

SSEgood <- sum((test.score[school=="good"]-mean(test.score[school=="good"]))^2)  
SSEbad <- sum((test.score[school=="bad"]-mean(test.score[school=="bad"]))^2)  
SSE = SSEgood + SSEbad

SSG = SSY - SSE

The F ratio is the variance of the error of the group divided by the variance of error. The variance for the error for the group treatment is calculated by taking the number the squared sums divided by the number of levels.

F.ratio <- (SSG / 1) / (SSE / 28)

F.ratio

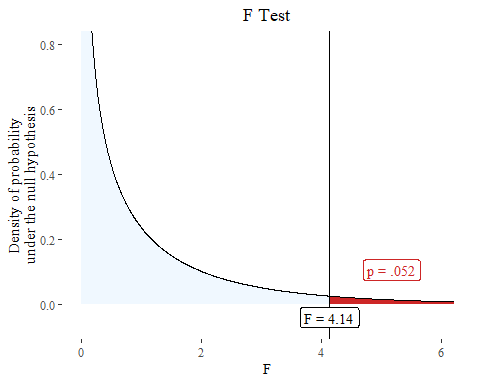
## [1] 4.13516

You can calculate the p value like so.

1-pf(8.269,1,28)

## [1] 0.007623596

plotftest(f = 4.13516, dfnum = 1, dfdenom = 28)



summary(aov(data$test.score~data$school))

## Df Sum Sq Mean Sq F value Pr(>F)   
## data$school 1 1591 1591.3 4.135 0.0516 .  
## Residuals 28 10775 384.8   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

You can display the confidence interfvals of an anova table with TukeyHSD().

TukeyHSD(aov(data$test.score~data$school))

## Tukey multiple comparisons of means  
## 95% family-wise confidence level  
##   
## Fit: aov(formula = data$test.score ~ data$school)  
##   
## $`data$school`  
## diff lwr upr p adj  
## good-bad 15.45 -0.1131889 31.01319 0.051577