Testing

Oliver d’Pug

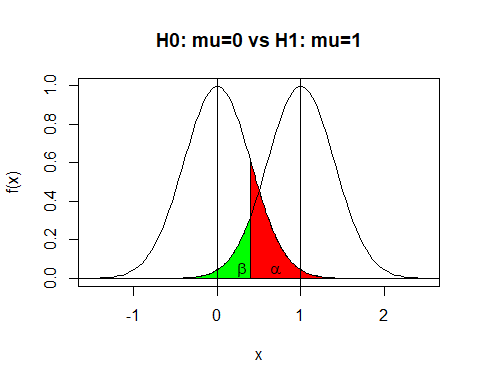
## Simple vs. Simple

Consider with . We wish to test versus . We need to choose so that

and

In pictures, for this looks like:

x\_max = 2.5  
 x\_min = -1.5  
 cord.x <- c(.4,seq(.4,x\_max,0.01),3)   
 cord.y <- c(0,dnorm(seq(.4,x\_max,0.01),0,2/5),0)   
 curve(dnorm(x,0,2/5),xlim=c(x\_min, x\_max),  
 main='H0: mu=0 vs H1: mu=1', ylab="f(x)", xlab="x")   
 polygon(cord.x,cord.y,col='red')  
 cord.x <- c(-2,seq(x\_min,.4,0.01),.4)   
 cord.y <- c(0,dnorm(seq(x\_min,.4,0.01),1,2/5),0)   
 curve(dnorm(x,1,2/5),xlim=c(x\_min, x\_max),  
 add=TRUE)   
 polygon(cord.x,cord.y,col='green')  
  
 abline(v=c(0,1), lty=1)  
 abline(h=0, lty=1)  
 text(0.3,0.05,expression(beta))  
 text(0.7,0.05,expression(alpha))



#lines(c(0.6,0.6),c(0,dnorm(0.6,1,2/5)),lty=3)

For we have

(sigma = 2)

[1] 2

(n = 25)

[1] 25

(se = sigma/sqrt(n))

[1] 0.4

(k = 0.4)

[1] 0.4

(mu = 0) ### H\_0 true

[1] 0

(alpha = pnorm(k, mu, se, lower.tail=FALSE))

[1] 0.1586553

(mu = 1) ### H\_1 true

[1] 1

(beta = pnorm(k, mu, se, lower.tail=TRUE))

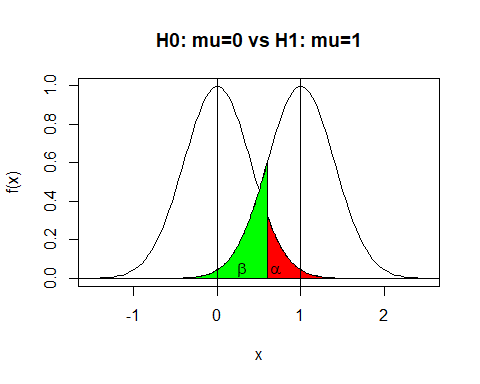
[1] 0.0668072

(power = 1 - beta)

[1] 0.9331928

For we have:

x\_max = 2.5  
 x\_min = -1.5  
 k = 0.6  
 cord.x <- c(k,seq(k,x\_max,0.01),3)   
 cord.y <- c(0,dnorm(seq(k,x\_max,0.01),0,2/5),0)   
 curve(dnorm(x,0,2/5),xlim=c(x\_min, x\_max),  
 main='H0: mu=0 vs H1: mu=1', ylab="f(x)", xlab="x")   
 polygon(cord.x,cord.y,col='red')  
 cord.x <- c(-2,seq(x\_min,k,0.01),k)   
 cord.y <- c(0,dnorm(seq(x\_min,k,0.01),1,2/5),0)   
 curve(dnorm(x,1,2/5),xlim=c(x\_min, x\_max),  
 add=TRUE)   
 polygon(cord.x,cord.y,col='green')  
  
 abline(v=c(0,1), lty=1)  
 abline(h=0, lty=1)  
 text(0.3,0.05,expression(beta))  
 text(0.7,0.05,expression(alpha))



#lines(c(0.6,0.6),c(0,dnorm(0.6,1,2/5)),lty=3)

(k = 0.6)

[1] 0.6

(mu = 0) ### H\_0 true

[1] 0

(alpha = pnorm(k, mu, se, lower.tail=FALSE))

[1] 0.0668072

(mu = 1) ### H\_1 true

[1] 1

(beta = pnorm(k, mu, se, lower.tail=TRUE))

[1] 0.1586553

(power = 1 - beta)

[1] 0.8413447

## Coin “Thump” Example

Suppose that we “thump” a “fair” coin ten times. For testing versus the one-sided alternative at the level we note that

1 - pbinom(7, 10, 0.5)

[1] 0.0546875

pbinom(7, 10, 0.5, lower.tail = FALSE)

[1] 0.0546875

Since 0.0546875 we check

1 - pbinom(8, 10, 0.5)

[1] 0.01074219

pbinom(8, 10, 0.5, lower.tail = FALSE)

[1] 0.01074219

Since 0.0107422 we reject when , “flip a coin” with when , and do not reject when .

To find we note that [ ]

So randomizer coin has probability of causing rejection.

### Power

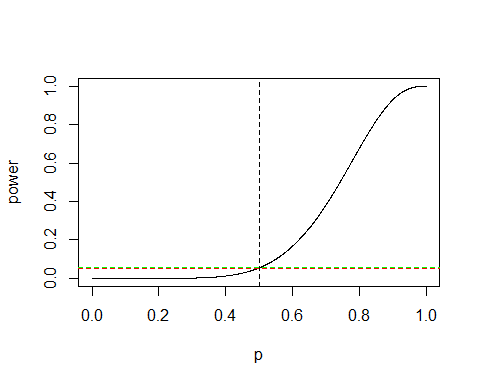
Suppose that we choose — without randomization at 8. Then

pstar <- seq(0, 1, by=0.1)  
 power <- 1 - pbinom(7, 10, pstar)  
 print(data.frame(pstar, power = round(power,4)))

pstar power  
1 0.0 0.0000  
2 0.1 0.0000  
3 0.2 0.0001  
4 0.3 0.0016  
5 0.4 0.0123  
6 0.5 0.0547  
7 0.6 0.1673  
8 0.7 0.3828  
9 0.8 0.6778  
10 0.9 0.9298  
11 1.0 1.0000

We can plot the power curve for various “true” when testing versus .

p <- seq(0,1,by=0.001)  
 power <- 1 - pbinom(7, 10, p)  
 plot(p, power, type="l")  
 abline(v=0.5, lty=2)  
 abline(h=0.05, lty=2, col="red") ### Desired alpha  
 abline(h=1-pbinom(7,10,0.5), lty=2, col="green") ### True alpha



(1-pbinom(7, 10, 0.5)) - 0.05 ### Difference between true and desired alpha

[1] 0.0046875

Note that as increases (moves further away from and into ) our power () increases. This is a good thing.