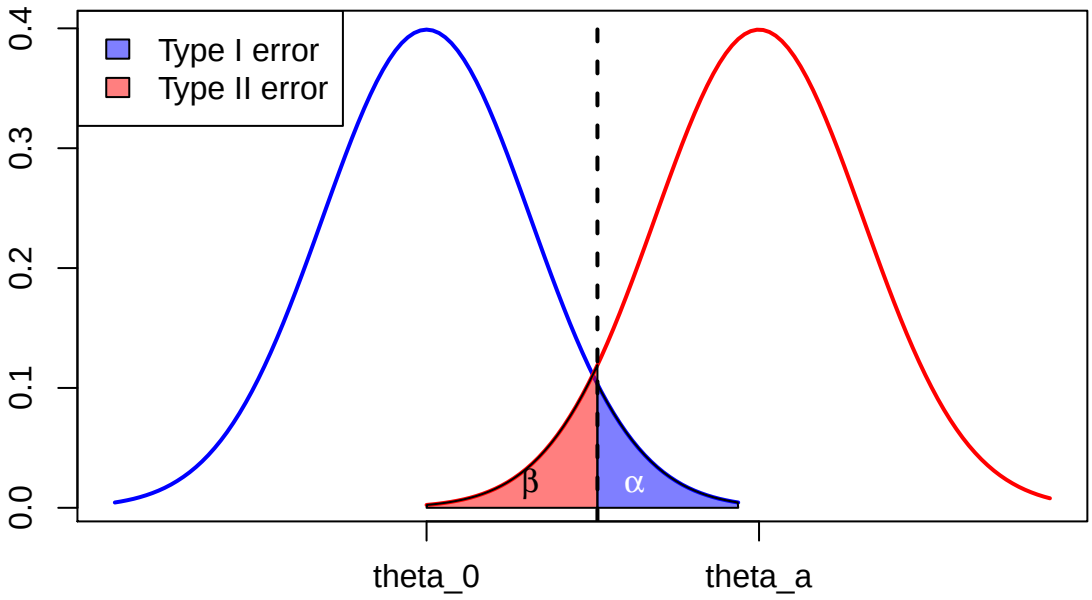
Lab4

Yingtong Lyu

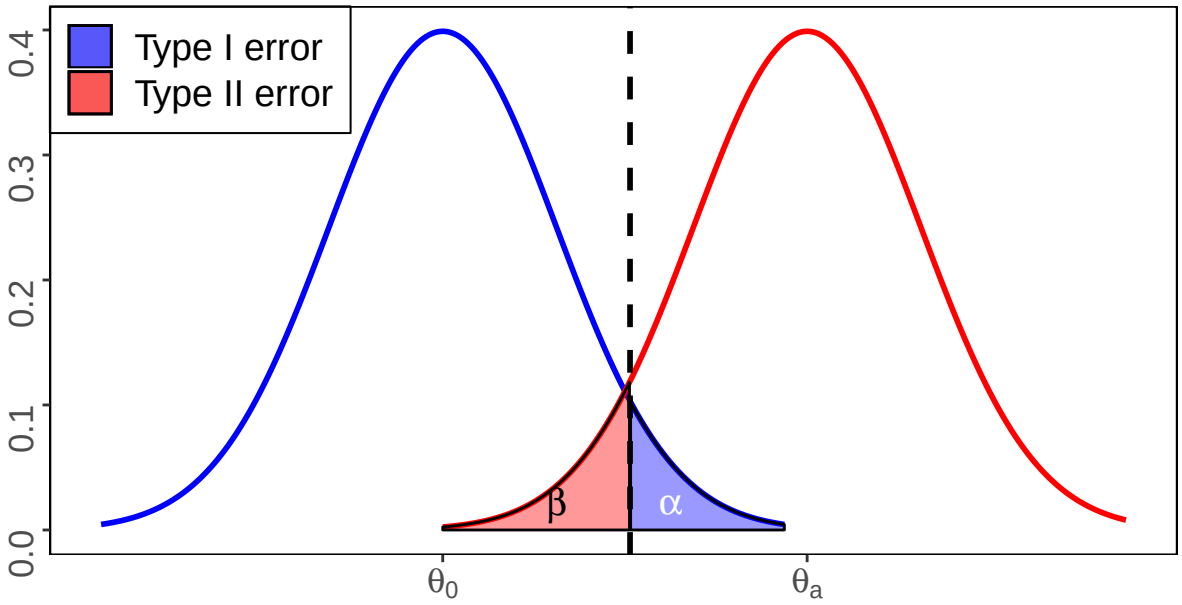
The graph from the Lecture 8 notes

```
curve(dnorm, from=-3, to=3, xlim=c(-3, 6), lwd=2, col=4, axes=FALSE, xlab=NA, ylab=NA)
curve(dnorm(x, mean=3.2), add=TRUE, col=2, lwd=2, from=0, to=6)
coord.x <- c(qnorm(0.95),seq(qnorm(0.95), 0, by=-0.01), 0)
coord.y <- c(0, dnorm(coord.x[-c(1, length(coord.x))], mean=3.2), 0)
polygon(coord.x, coord.y, col=rgb(1, 0, 0, 0.5))
coord.x <- c(qnorm(0.95), seq(qnorm(0.95), 3, by=0.01), 3)
coord.y <- c(0, dnorm(coord.x[-c(1, length(coord.x))]), 0)
polygon(coord.x, coord.y, col=rgb(0, 0, 1, 0.5))
abline(v = qnorm(0.95), lty=2, lwd=2)
text(1, 0.02, labels = expression(beta))
text(2, 0.02, labels = expression(alpha), col = "white")
axis(1, at = c(0, 3.2), labels = c(expression(theta_0), expression(theta_a)))
axis(2)
box()
legend("topleft", c("Type I error", "Type II error"), fill=c(rgb(0, 0, 1, 0.5), rgb(1, 0, 0, 0.5)))
```



Write the R code using ggplot function to recreate

```
library(ggplot2)
x1=seq(-3, 3, by=0.01)
x2=seq( 0, 6, by=0.01)
y1=dnorm(x1)
y2=dnorm(x2, mean=3.2)
pdf1=data.frame(x1,y1)
pdf2=data.frame(x2,y2)
p<-ggplot() +
  #draw two density line
  geom_line(data = pdf1, aes(x1, y1), color = "blue", size=1) +
  geom_line(data = pdf2, aes(x2, y2), color = "red",  size=1) +
  #draw shawdow
  geom_ribbon(data = pdf1[pdf1$x1 > qnorm(0.95),],
              aes(x = x1, ymin = 0, ymax = y1, fill = "a"), alpha=0.4, colour="black") +
  geom_ribbon(data = pdf2[pdf2$x2 < qnorm(0.95),],
              aes(x = x2, ymin = 0, max = y2, fill = "b"),  alpha=0.4, colour="black")+
  #draw vertical line
  geom_vline(xintercept = qnorm(0.95),linetype = 2,lwd = 1)+
  #add annotation
  annotate("text", x = 1, y = 0.02, label = paste0(expression(beta)),
           color = 'black', size = 5,parse=TRUE) +
  annotate("text", x = 2, y = 0.02, label = paste0(expression(alpha)),
           color = 'white', size = 5,parse=TRUE) +
  theme(panel.background =element_blank(),
        panel.border = element_rect(color = 'black',fill=NA),
        axis.text.y = element_text(size = 13, angle = 90),
        axis.text.x = element_text(size = 13),
        axis.title = element_blank(),
        legend.title = element_blank())
#legend adjustment
p+theme(legend.justification = c(0,1),
        legend.box.background = element_rect(color='black',fill=NA),
        legend.position = c(0,1),
        legend.box.margin = margin(c(1,1,1,1)),
        legend.text = element_text(size = 14,margin = margin(l=8)))+
  scale_x_continuous(breaks = c(0,3.2),labels = c(expression(theta[0],theta[a])))+
  scale_fill_manual(values = c("a"="blue", "b"="red"),
                    label= c("Type I error","Type II error"))+
  theme(plot.margin = unit(c(1.5,0.5,2,0.5),"cm"))
```



4. Implement a function that will check if a given positive integer is a prime number.

else if(is.integer(x) == F){ stop("The input should be a integer") }

```
# Firstly, write a function to check if the input is integer
is.wholenumber <-
    function(x, tol = .Machine$double.eps^0.5)  abs(x - round(x)) < tol

is.prime <- function (a) {
  # Check if the input is numeric.
  if (is.numeric(a) == F) {
    return("The input should be a numeric value.")
  }
  if(is.wholenumber(a) == F){
    return("The input should be a integer")}
  if (a <= 1) {
        return(F)
      }
  else {
        for(i in 2:(a-1)) {
      if ((a %% i) == 0) {
        return(F)
      }
        }
    return(T)
```

```
  }
}
```

```
#Test Sets
#4.1 If the input is not numeric,
is.prime("hello")
```

```
## [1] "The input should be a numeric value."
```

```
#4.2 If the input is not integer,
is.prime(123.123)
```

```
## [1] "The input should be a integer"
```

```
#4.3 If the input is negative,
is.prime(-1)
```

```
## [1] FALSE
```

```
for (i in 0:100) {
  if (is.prime(i) == T) {
    print(i)
  }
}
```

```
## [1] 3
## [1] 5
## [1] 7
## [1] 11
## [1] 13
## [1] 17
## [1] 19
## [1] 23
## [1] 29
## [1] 31
## [1] 37
## [1] 41
## [1] 43
## [1] 47
## [1] 53
## [1] 59
## [1] 61
## [1] 67
## [1] 71
## [1] 73
## [1] 79
## [1] 83
## [1] 89
## [1] 97
```