Ejercicios

Uriel Paluch

21/11/2021

Metodos

```
integracion <- function(funcion, limiteSuperior, limiteInferior, n){
  uniforme <- limiteInferior + (limiteSuperior - limiteInferior) * runif(n = n)

  alturaPromedio <- 1/n * sum(funcion(uniforme))

  anchoBase <- limiteSuperior - limiteInferior

  desvioEstandar <- sqrt( 1/(n-1) * sum((funcion(uniforme) * (limiteSuperior - limiteInferior) - altura

  error <- desvioEstandar/sqrt(n)

  resultados <- list("error" = error, "alfa" =  alturaPromedio * anchoBase)
  return(resultados)
}
```

Ejercicio 1

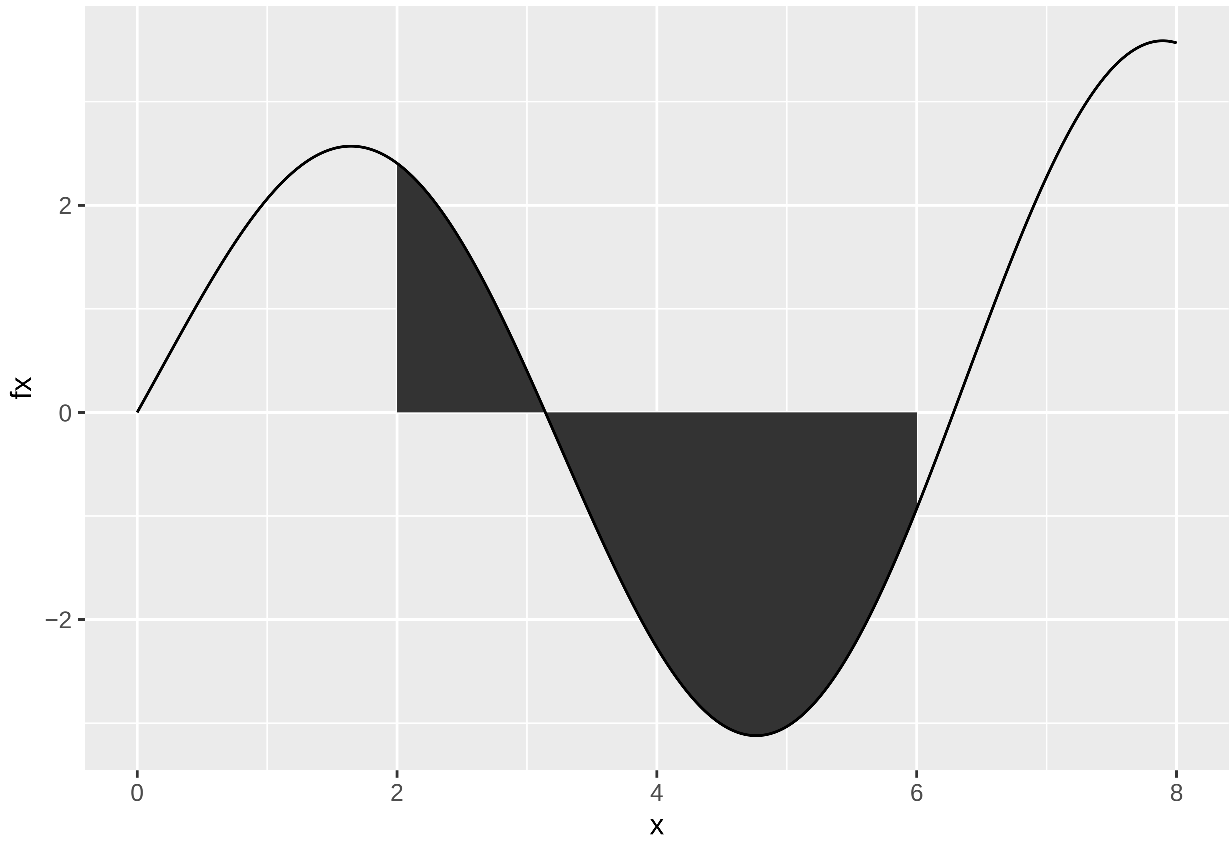
a

```
# Hay que hacerlo así y no con expression porque el resultado es diferente
funcion <- function(x){
  return(sqrt(x+5)*sin(x))
}

# Fijo el seed para que me den los mismos resultados que la guia
set.seed(123)

integracion(funcion = funcion, limiteSuperior = 6, limiteInferior = 2, n = 10000)
```

```
## $error
## [1] 0.07177432
##
## $alfa
## [1] -4.52639
```

```
# Grafico
x <- seq(from = 2, to = 6, by = 0.01)
fx <- funcion(x)

x1 <- seq(from = 0, to = 8, by = 0.01)
fx1 <- funcion(x1)
```

```
ggplot() +
  geom_area(aes(x = x, y = fx )) +
  geom_line(aes(x = x1, y = fx1))
```

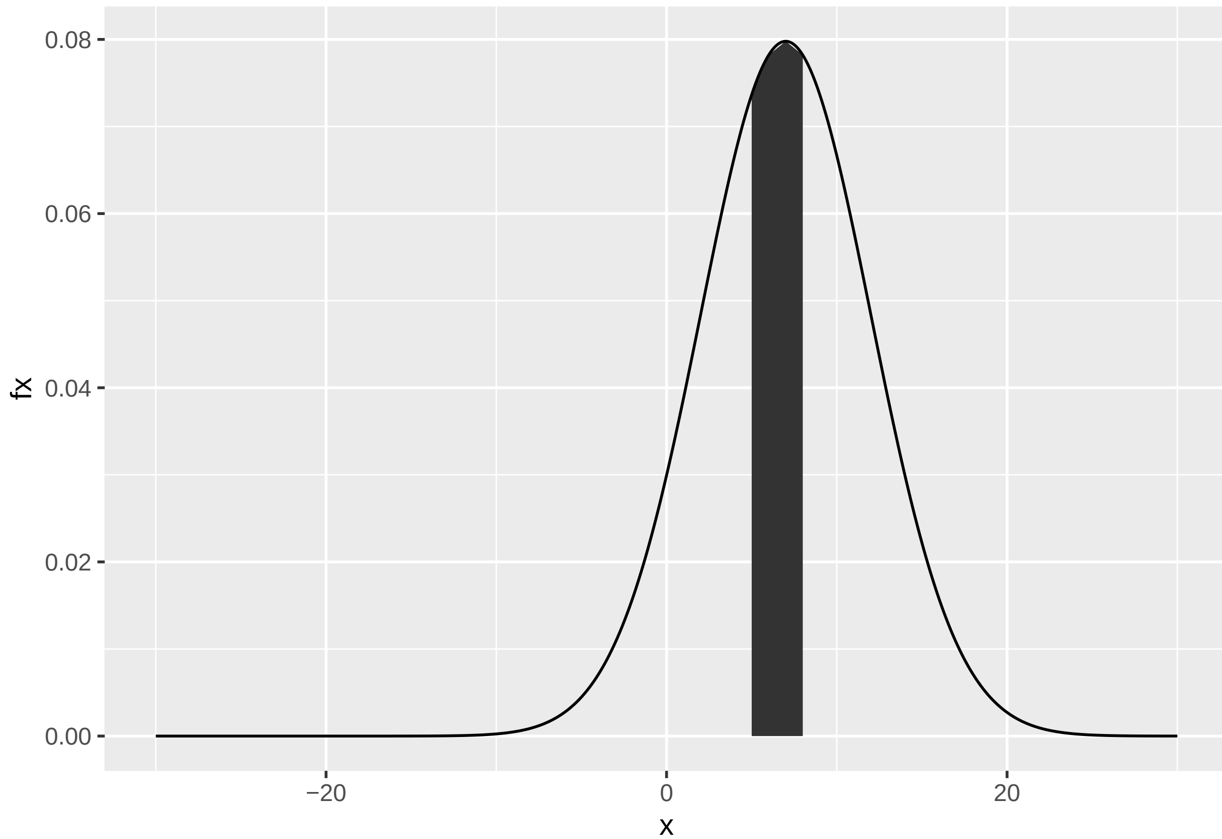


b

```
# Hay que hacerlo así y no con expression porque el resultado es diferente
media <- 7
desvio <- 5
funcion <- function(x){
  return((1/(desvio * sqrt(2*pi))) * exp(-(x-media)^2/(2*desvio^2)))
}


# Fijo el seed para que me den los mismos resultados que la guia
set.seed(123)

integracion(funcion = funcion, limiteSuperior = 8, limiteInferior = 5, n = 10000)
```

```
## $error
## [1] 5.081396e-05
##
## $alfa
## [1] 0.2346953
```

```r
# Grafico
x <- seq(from = 5, to = 8, by = 1)
fx <- funcion(x)

x1 <- seq(from = -30, to = 30, by = 0.1)
fx1 <- funcion(x1)

ggplot() +
  geom_area(aes(x = x, y = fx )) +
  geom_line(aes(x = x1, y = fx1))
```



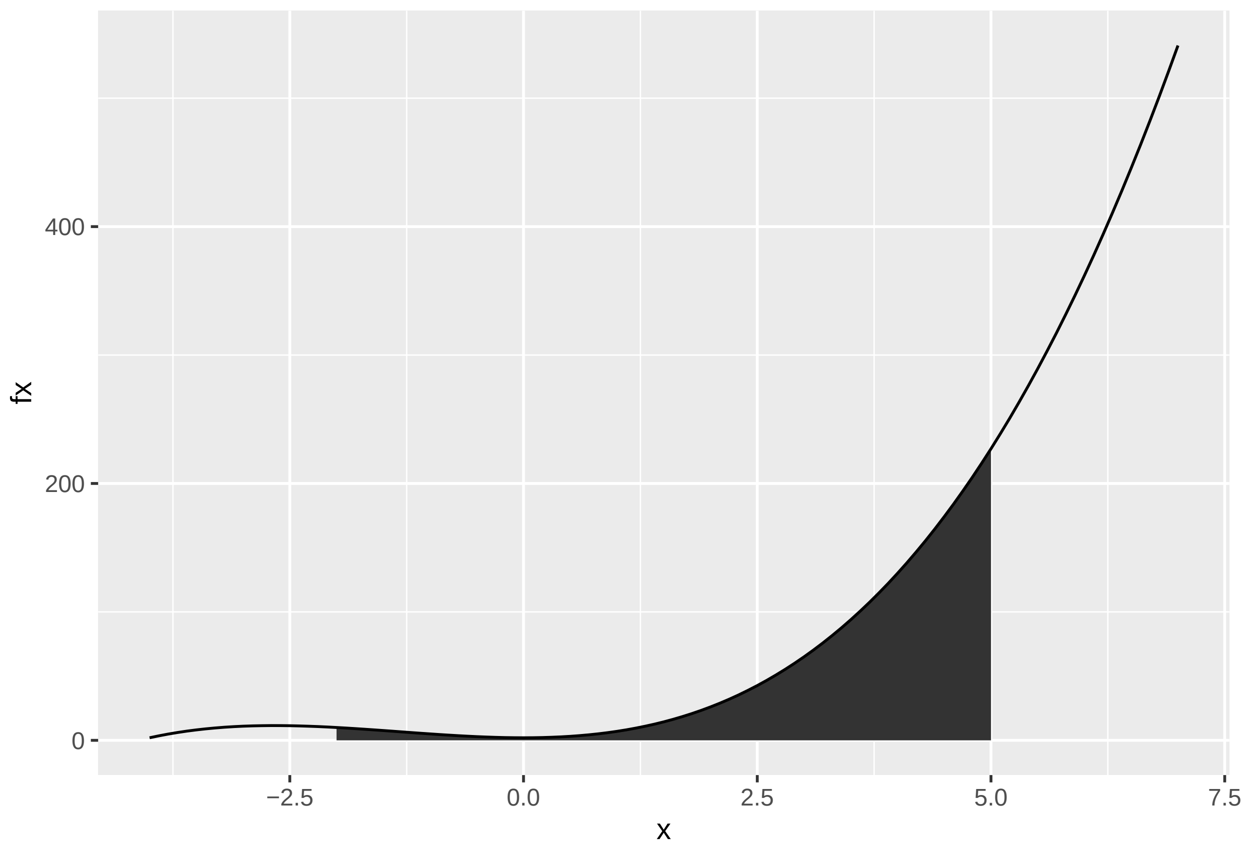
c

```r
# Hay que hacerlo así y no con expression porque el resultado es diferente
funcion <- function(x){
  return(x^3+4*x^2+2)
}

# Fijo el seed para que me den los mismos resultados que la guia
set.seed(123)

integracion(funcion = funcion, limiteSuperior = 5, limiteInferior = -2, n = 10000)
```

```
## $error
## [1] 4.24804
##
```

```
## $alfa
## [1] 336.8058
```

```r
# Grafico
x <- seq(from = -2, to = 5, by = 0.1)
fx <- funcion(x)

x1 <- seq(from = -4, to = 7, by = 0.1)
fx1 <- funcion(x1)

ggplot() +
  geom_area(aes(x = x, y = fx )) +
  geom_line(aes(x = x1, y = fx1))
```



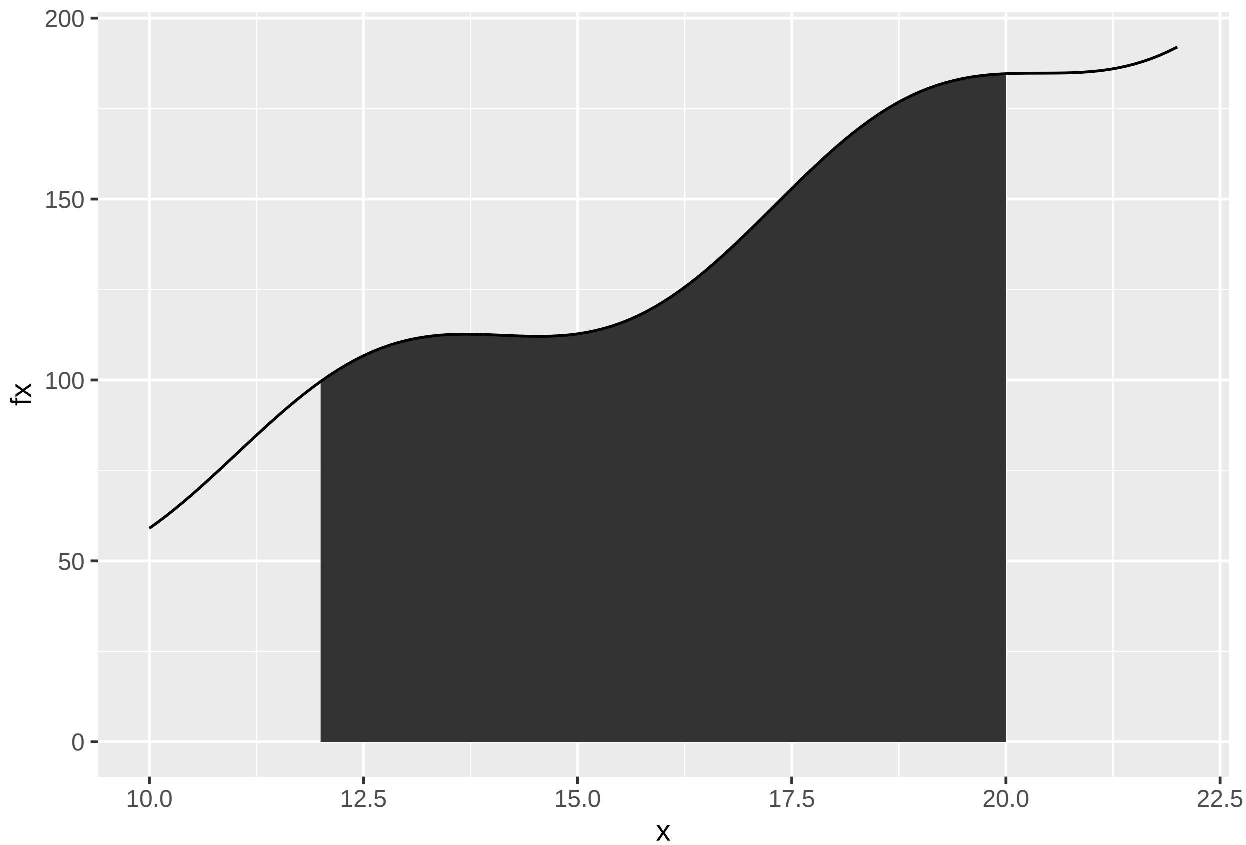
d

```r
# Hay que hacerlo así y no con expression porque el resultado es diferente
funcion <- function(x){
  return(x*log(x^3)+12*cos(x))
}

# Fijo el seed para que me den los mismos resultados que la guia
set.seed(123)

integracion(funcion = funcion, limiteSuperior = 20, limiteInferior = 12, n = 10000)
```

```
## $error
```

```
## [1] 2.245877
##
## $alfa
## [1] 1083.306
```

```
# Grafico
x <- seq(from = 12, to = 20, by = 0.1)
fx <- funcion(x)

x1 <- seq(from = 10, to = 22, by = 0.1)
fx1 <- funcion(x1)

ggplot() +
  geom_area(aes(x = x, y = fx )) +
  geom_line(aes(x = x1, y = fx1))
```



Ejercicio 2

```
N <- 10000
resultado <- matrix(NA, nrow = N, ncol = 2)
set.seed(123)

for (i in 1:N) {
  # Genera un número aleatorio con distribucion binomial
  n <- rbinom(n = 1, size = 1200, prob = 0.7984)
```

```r
  resultado[i, 1] <- n

  xi <- rchisq(df = 2, n = n)

  suma <- sum(xi)

  resultado[i,2] <- suma
}

n_esperanza <- mean(resultado[,1])
n_varianza <- var(resultado[,1])

suma_esperanza <- mean(resultado[,2])
suma_varianza <- var(resultado[,2])
```

a

```r
n_esperanza
```

```
## [1] 958.1095
```

b

```r
n_varianza
```

```
## [1] 195.8719
```

c

```r
suma_esperanza
```

```
## [1] 1916.74
```

d

```r
sqrt(suma_varianza)
```

```
## [1] 67.81974
```

Ejercicio 3

a

```r
mu <- c(0.15, 0.12)
sigma <- c(0.2, 0.19)

p0 <- rep(NA, times = 2)

getSymbols("YPFD.BA", auto.assign = TRUE, src = "yahoo")
```

```
## 'getSymbols' currently uses auto.assign=TRUE by default, but will
## use auto.assign=FALSE in 0.5-0. You will still be able to use
## 'loadSymbols' to automatically load data. getOption("getSymbols.env")
## and getOption("getSymbols.auto.assign") will still be checked for
```

```
## alternate defaults.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.
```

```
## Warning: YPFD.BA contains missing values. Some functions will not work if
## objects contain missing values in the middle of the series. Consider using
## na.omit(), na.approx(), na.fill(), etc to remove or replace them.
```

```
## [1] "YPFD.BA"
```

```
aux <- data.frame(YPFD.BA)
rm(YPFD.BA)

p0[1] <- aux['2020-11-06',]$YPFD.BA.Adjusted
rm(aux)

getSymbols("MELI.BA", auto.assign = TRUE, src = "yahoo")
```

```
## Warning: MELI.BA contains missing values. Some functions will not work if
## objects contain missing values in the middle of the series. Consider using
## na.omit(), na.approx(), na.fill(), etc to remove or replace them.
```

```
## [1] "MELI.BA"
```

```
aux <- data.frame(MELI.BA)
rm(MELI.BA)

p0[2] <- aux['2020-11-06',]$MELI.BA.Adjusted
rm(aux)

# 1 año
anios <- 1

# Simulacion
m <- 10000

# Matriz de camino de precios
pt <- matrix(NA, nrow = m, ncol = 2)

set.seed(123)
e <- rnorm(m)

pt[,1] <- p0[1] * exp((mu[1]-0.5*sigma[1]^2) * anios + sigma[1]*sqrt(anios)*e)

prob <- 0.975

# YPF
mean(pt[,1])
```

```
## [1] 644.948
```

```
YPF_mean <- mean(pt[,1])
quantile(pt[,1], prob)
```

```
##    97.5%
## 934.4096
```

```r
quantile(pt[,1], 1-prob)
```

```
##     2.5%
## 425.9127
```

```r
e <- rnorm(m)
pt[,2] <- p0[2] * exp((mu[2]-0.5*sigma[2]^2) * anios + sigma[2]*sqrt(anios)*e)

# MELI
mean(pt[,2])
```

```
## [1] 4169.362
```

```r
MELI_mean <- mean(pt[,2])
quantile(pt[,2], prob)
```

```
##   97.5%
## 5950.82
```

```r
quantile(pt[,2], 1-prob)
```

```
##     2.5%
## 2829.561
```

b

```r
# Rendimiento logaritmico esperado
rl <- log(mean(pt)/p0)
p0_m <- matrix(rep(p0,m), ncol = 2, byrow = T)
rl_matrix <- log(pt/p0_m)
# YPF
mean(rl_matrix[,1])
```

```
## [1] 0.1295257
```

```r
# MELI
mean(rl_matrix[,2])
```

```
## [1] 0.1002198
```

c

```r
mu <- c(0.15, 0.12, 0.3)
sigma <- c(0.2, 0.19, 0.42)

p0 <- rep(NA, times = 3)

getSymbols("YPFD.BA", auto.assign = TRUE, src = "yahoo")
```

```
## Warning: YPFD.BA contains missing values. Some functions will not work if
## objects contain missing values in the middle of the series. Consider using
## na.omit(), na.approx(), na.fill(), etc to remove or replace them.
```

```
## [1] "YPFD.BA"
```

```r
aux <- data.frame(YPFD.BA)
rm(YPFD.BA)
```

```r
p0[1] <- aux['2020-11-06',]$YPFD.BA.Adjusted
rm(aux)

getSymbols("MELI.BA", auto.assign = TRUE, src = "yahoo")
```

```
## Warning: MELI.BA contains missing values. Some functions will not work if
## objects contain missing values in the middle of the series. Consider using
## na.omit(), na.approx(), na.fill(), etc to remove or replace them.
```

```
## [1] "MELI.BA"
```

```r
aux <- data.frame(MELI.BA)
rm(MELI.BA)

p0[2] <- aux['2020-11-06',]$MELI.BA.Adjusted
rm(aux)

p0[3] <- 54

# 1 año
anios <- 1

# Simulacion
m <- 10000

# Matriz de camino de precios
pt <- matrix(NA, nrow = m, ncol = 3)

# Matriz de covarianzas
rho <- diag(3)
rho[1,2] <- rho[2,1] <- 0.9
rho[1,3] <- rho[3,1] <- 0.7
rho[2,3] <- rho[3,2] <- 0.6

# Matriz de cholesky
ch <- chol(rho)

set.seed(123)
z <- matrix(rnorm(3*m),nrow=m,ncol=3)

e <- z %*% ch

pt_cor <- matrix(NA, nrow = m, ncol = 3)

for (i in 1:m) {
  for (k in 1:3) {
    pt_cor[i,k] <- p0[k] * exp((mu[k] - 0.5 * sigma[k]^2) * anios + sigma[k] * sqrt(anios) * e[i,k])
  }
}

# YPF
mean(pt_cor[,1])
```

```
## [1] 644.948
```

```r
quantile(pt_cor[,1], 0.975)
```

```
##    97.5%
## 934.4096
```

```r
quantile(pt_cor[,1], 0.025)
```

```
##     2.5%
## 425.9127
```

```r
# MELI
mean(pt_cor[,2])
```

```
## [1] 4171.71
```

```r
quantile(pt_cor[,2], 0.975)
```

```
##  97.5%
## 5917.4
```

```r
quantile(pt_cor[,2], 0.025)
```

```
##    2.5%
## 2837.18
```

```r
# LOMA
mean(pt_cor[,3])
```

```
## [1] 72.85403
```

```r
quantile(pt_cor[,3], 0.975)
```

```
##    97.5%
## 155.0048
```

```r
quantile(pt_cor[,3], 0.025)
```

```
##     2.5%
## 29.47365
```

d

```r
p0_m <- matrix(rep(p0,m), nrow = m, ncol = 3, byrow = T)

rl_c <- log(pt_cor/p0_m)

mean(rl_c[,1])
```

```
## [1] 0.1295257
```

```r
mean(rl_c[,2])
```

```
## [1] 0.1007903
```

```r
mean(rl_c[,3])
```

```
## [1] 0.2092465
```

e

f

```
# YPF
mean(pt_cor[,1]) * 200
```

```
## [1] 128989.6
```

```
YPF_mean * 200
```

```
## [1] 128989.6
```

```
# MELI
mean(pt_cor[,2]) * 120
```

```
## [1] 500605.2
```

```
MELI_mean * 120
```

```
## [1] 500323.5
```