Económicas, UBA. Actuario. Análisis Numérico.

Cuatrimestre 1, 2021. Segundo Examen Parcial.

Schmukler, Lucila 879871.

25/junio/2021

Contents

1 Integral Log-Normal (25 puntos) **2**
1.1 Simpson y Trapecio . . . 2
1.2 Simpson Compuesto . . . 3
1.3 Aproximación Numércica de Esperanza Matemática . . . 5
1.4 Aproximación Numércica de Esperanza Matemática de una función condicional . . . 7

2 Simulación de Precios (25 puntos) **10**
2.1 Simulación de P_T . . . 10
2.2 Simulación de caminos de precios P_t con $t > 0$ . . . 11
2.3 Gráfico de caminos de precios P_t con $t > 0$ (incluyendo $E[P_t]$ e IC) . . . 12
2.4 Estimación de probabilidades a partir de muestras simuladas . . . 14

3 Integración y Derivación Numérica (20 puntos) **16**
3.1 $E(Y|\alpha;\beta)$ . . . 16
3.2 Derivada respecto a α . . . 17
3.3 Derivada respecto a β . . . 17

4 SQL (20 Puntos) **19**
4.1 Consulta Clientes . . . 19
4.2 Consulta Ventas . . . 19

5 Aprendizaje Automático (10 puntos) **20**

```
## Warning: package 'scales' was built under R version 4.1.1

## Warning: package 'flextable' was built under R version 4.1.1

## Warning: package 'tidyverse' was built under R version 4.1.1

## Warning: package 'tibble' was built under R version 4.1.1

## Warning: package 'tidyr' was built under R version 4.1.1

## Warning: package 'readr' was built under R version 4.1.1

## Warning: package 'purrr' was built under R version 4.1.1

## Warning: package 'dplyr' was built under R version 4.1.1

## Warning: package 'stringr' was built under R version 4.1.1

## Warning: package 'forcats' was built under R version 4.1.1
```

1 Integral Log-Normal (25 puntos)

Considere la siguiente función de densidad:

$$f_S(x) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(\ln(x)-\mu)^2}{2\sigma^2}}; \text{con } x > 0.$$

Para su ejercicio particular, considere $\mu = 2.77$ y $\sigma = 0.22\sqrt{0.75}$.

1.1 Simpson y Trapecio

Aproxime la probabilidad de que S esté entre 12.68 y 20.42 usando los métodos de "Trapecio", "Simpson" y "Simpson tres octavos". Indique en cada caso los "nodos" $x_0, x_1, \ldots, x_n$ que se utilizan para la aproximación.

Respuesta:

```
 # Ingrese a continuación, en este bloque, todo el código necesario para resolver el ejercicio 1.1

newtonCotesCerradas <- function(limiteInferior, limiteSuperior, funcion, n){
  #browser()
  h <- (limiteSuperior - limiteInferior)/n

  fx <- rep(NA, times = (n+1))
  for (i in 1:(n+1)) {
    fx[i] <- eval(funcion, list(x = limiteInferior + (i-1)*h))
  }

  # Hay que cambiarlo para que quede solo con un 1
  if (n == 1){
    return((h/2) * (fx[1] + fx[2]))
  }
  else if (n == 2){
    return((h/3) * (fx[1] + 4*fx[2] + fx[3]))
  }
  else if(n == 3){
    return((3/8)*h*(fx[1] + 3*fx[2] + 3*fx[3] + fx[4]))
  }

}

# n = 1. Regla del trapecio.
# n = 2. Regla de Simpson.
# n = 3. Regla de tres octavos de Simpson.
# Poner la funcion con "x" como incognita
mu <- 2.77
sigma <- 0.22*sqrt(0.75)

newtonCotesCerradas(limiteInferior = 12.68, limiteSuperior = 20.42, funcion = expression( (1/(x*sigma*s
```

```
## [1] 0.4802657
```

```
print("Nodos en trapecio: x0, x1")
```

```
## [1] "Nodos en trapecio: x0, x1"
```

```
newtonCotesCerradas(limiteInferior = 12.68, limiteSuperior = 20.42, funcion = expression( (1/(x*sigma*s
```

```
## [1] 0.8011337
```

```
print("Nodos en Simpson: x0, x1, x2")
```

```
## [1] "Nodos en Simpson: x0, x1, x2"
```

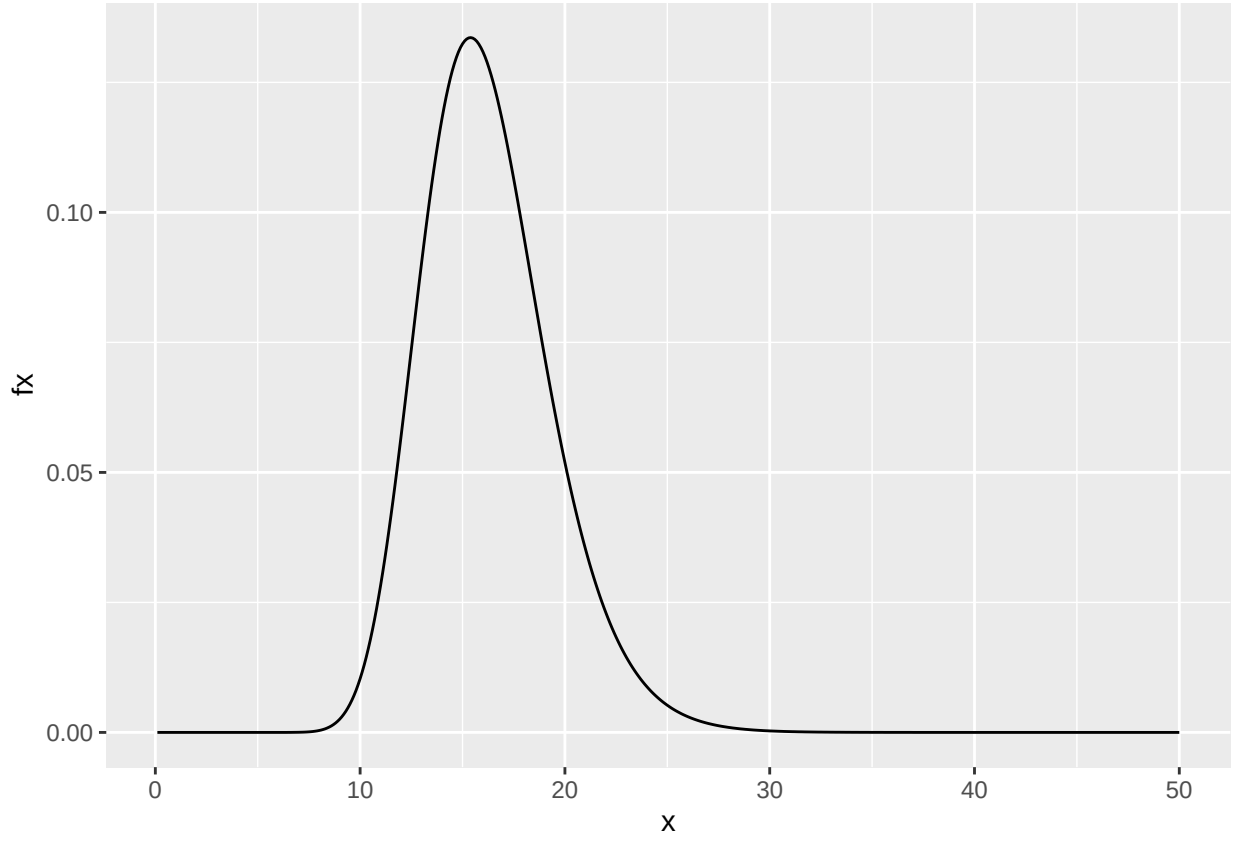
```
newtonCotesCerradas(limiteInferior = 12.68, limiteSuperior = 20.42, funcion = expression( (1/(x*sigma*s
```

```
## [1] 0.7945954
```

```
print("Nodos en Simpson 3/8: x0, x1, x2, x3")
```

```
## [1] "Nodos en Simpson 3/8: x0, x1, x2, x3"
```

```
x <- seq(from = 0.1, to = 50, by = 0.1)
fx <- eval(expression( (1/(x*sigma*sqrt(2*pi))) * exp(- ((log(x)-mu)^2)/(2*sigma^2)) ), list(x = x))

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



1.2 Simpson Compuesto

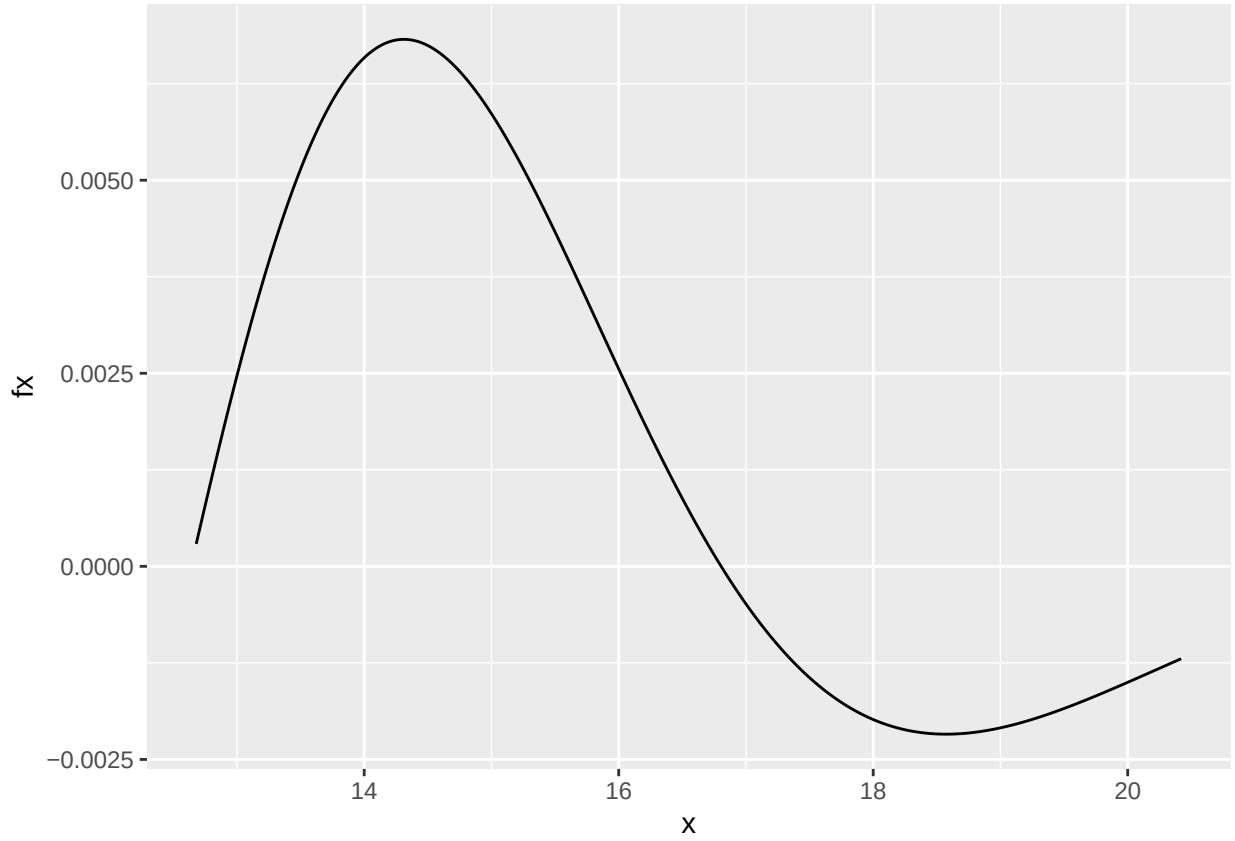
Aproxime el punto anterior usando "Simpson Compuesto", con $n = 1000$. Calcule la cota del error.

Respuesta:

```r
# Ingrese a continuación, en este bloque, todo el código necesario para resolver el ejercicio 1.2

IntegracionCompuesta <- function(limiteInferior, limiteSuperior, funcion, n, cantIntervalos){

  #browser()
  if ((n == 2 || n == 0) && cantIntervalos%%2 != 0){
    return("cantIntervalos debe ser un entero par")
  }

  cantIntervalos <- cantIntervalos/n


  crecimientoIntervalo <- (limiteSuperior-limiteInferior)/cantIntervalos

  fx <- rep(NA, times = (n+1))

  resultado <- 0

  for (i in 1:cantIntervalos) {
    limiteSuperior <- limiteInferior + crecimientoIntervalo

    if (n != 0){
      h <- (limiteSuperior - limiteInferior)/n
    }


    for (i in 1:(n+1)) {
      fx[i] <- eval(funcion, list(x = limiteInferior + (i-1)*h))
    }

    if(n == 2){
      resultado <- resultado + (h/3) * (fx[1] + 4*fx[2] + fx[3])
    }

    limiteInferior <- limiteSuperior
  }

  return(resultado)
}

sigma <- 0.22*sqrt(0.75)

# n = 2. Simpson
IntegracionCompuesta(limiteInferior = 12.68, limiteSuperior = 20.42, funcion = expression( (1/(x*sigma*s
```

```
## [1] 0.7884381
```

```r
x <- seq(from = 12.68, to = 20.42, by = 0.001)

fx <- eval(D(D(D(D(expression( (1/(x*sigma*sqrt(2*pi))) * exp(- ((log(x)-mu)^2)/(2*sigma^2)) ), "x"),"x

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




```
limiteSuperior <- 20.42
limiteInferior <- 12.68
n <- 1000

h <- (limiteSuperior - limiteInferior)/n

((limiteSuperior - limiteInferior)/180) * (h^4) * max(abs(fx))
```

```
## [1] 1.053109e-12
```

1.3 Aproximación Numércica de Esperanza Matemática

Use "Trapecio Compuesto" con $n = 1000$ para aproximar $E(S)$; es decir, la esperanza matemática de S. Calcule la cota del error.

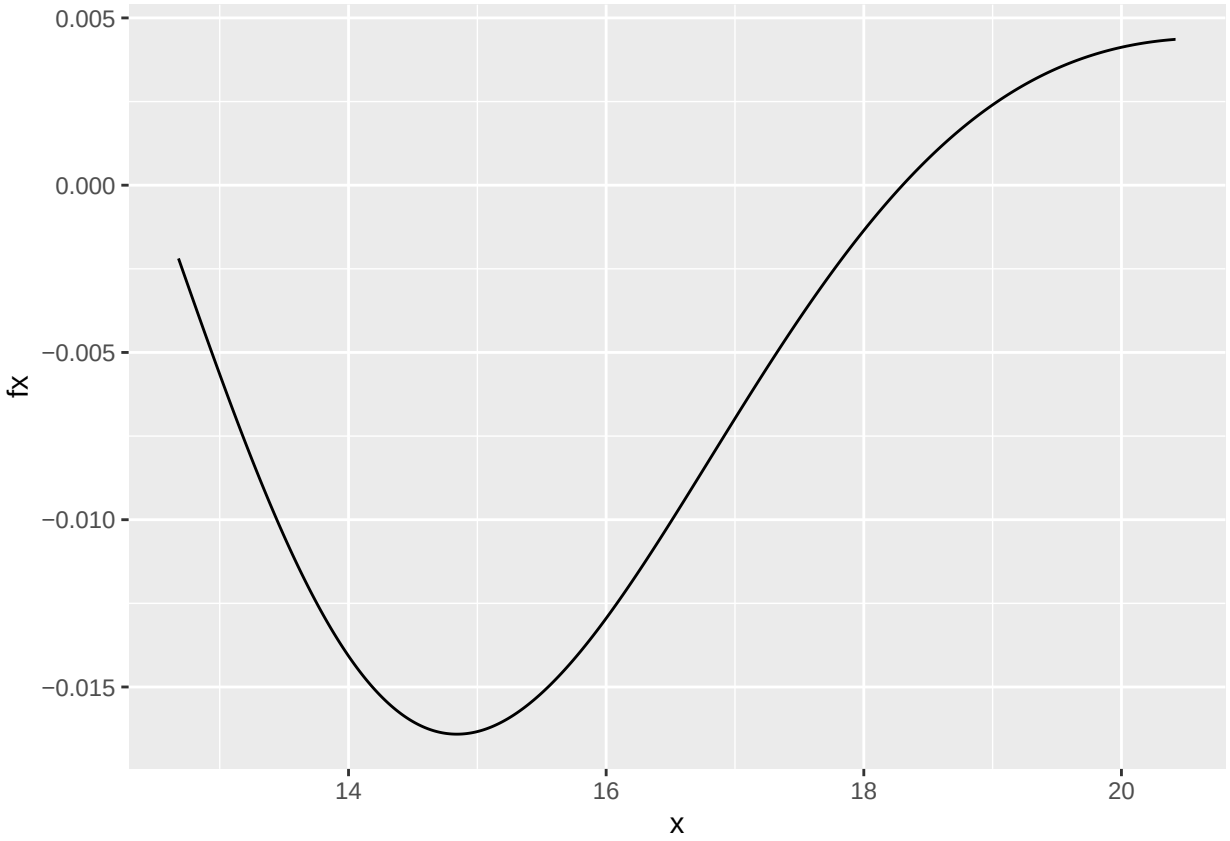
Respuesta:

```
# Ingrese a continuación, en este bloque, todo el código necesario para resolver el ejercicio 1.3
IntegracionCompuesta <- function(limiteInferior, limiteSuperior, funcion, n, cantIntervalos){

  cantIntervalos <- cantIntervalos/n


  crecimientoIntervalo <- (limiteSuperior-limiteInferior)/cantIntervalos

  fx <- rep(NA, times = (n+1))

  resultado <- 0
```

```
  for (i in 1:cantIntervalos) {
    limiteSuperior <- limiteInferior + crecimientoIntervalo

    if (n != 0){
      h <- (limiteSuperior - limiteInferior)/n
    }


    for (i in 1:(n+1)) {
      fx[i] <- eval(funcion, list(x = limiteInferior + (i-1)*h))
    }

    # Trapecio
    if (n == 1){
      resultado <- resultado + (h/2) * (fx[1] + fx[2])
    }

    limiteInferior <- limiteSuperior
  }

  return(resultado)
}

# n = 1. Trapecio
# n = 2. Simpson
IntegracionCompuesta(limiteInferior = 12.68, limiteSuperior = 20.42, funcion = expression(x * (1/(x*sig
```

```
## [1] 12.73976
```

```
x <- seq(from = 12.68, to = 20.42, by = 0.001)

fx <- eval(D(D(expression( (1/(x*sigma*sqrt(2*pi))) * exp(- ((log(x)-mu)^2)/(2*sigma^2)) ), "x"),"x"), l

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




```
limiteSuperior <- 20.42
limiteInferior <- 12.68
n <- 1000

h <- (limiteSuperior - limiteInferior)/n

((limiteSuperior - limiteInferior)/12) * (h^2) * max(abs(fx))
```

```
## [1] 6.341094e-07
```

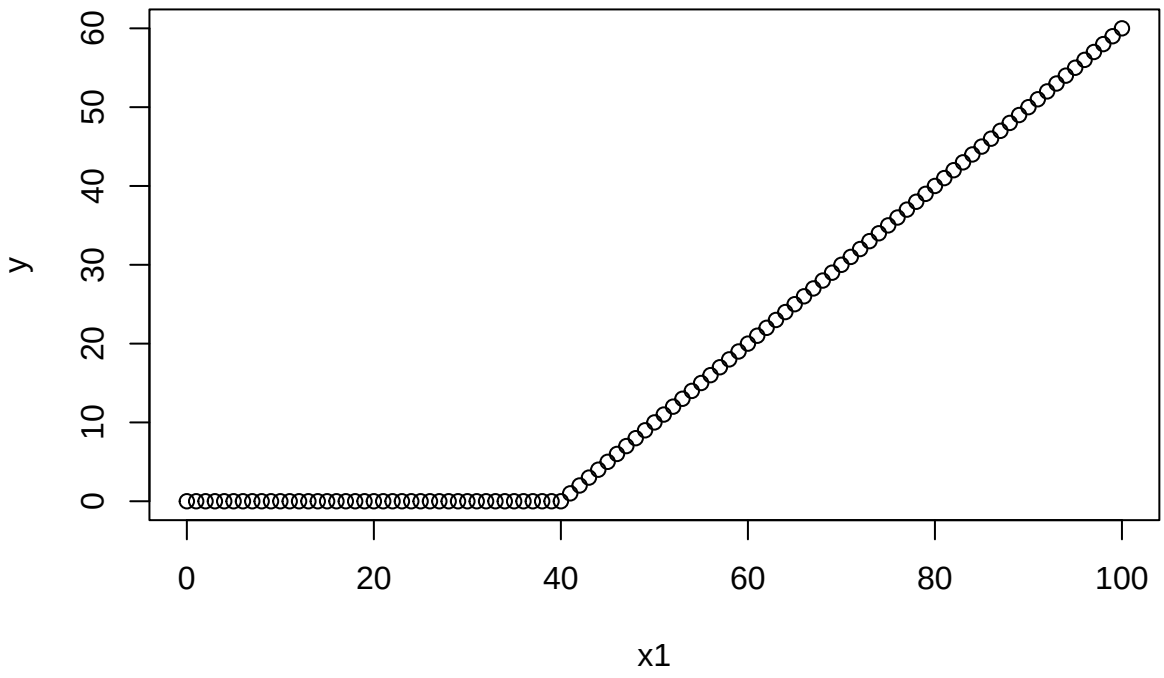
1.4 Aproximación Numércica de Esperanza Matemática de una función condicional

Use "Simpson Compuesto" con $n = 1000$ para aproximar $E[max(S-40, 0)]$; es decir, la esperanza matemática de una función condicional que toma S-40 siempre y cuando sea positivo, y cero en caso contrario.

Respuesta:

```
# Ingrese a continuación, en este bloque, todo el código necesario para resolver el ejercicio 1.4

IntegracionCompuesta <- function(limiteInferior, limiteSuperior, funcion, n, cantIntervalos){

  if ((n == 2 || n == 0) && cantIntervalos%%2 != 0){
    return("cantIntervalos debe ser un entero par")
  }

  cantIntervalos <- cantIntervalos/n
```

```
  crecimientoIntervalo <- (limiteSuperior-limiteInferior)/cantIntervalos

  fx <- rep(NA, times = (n+1))

  resultado <- 0

  for (i in 1:cantIntervalos) {
    limiteSuperior <- limiteInferior + crecimientoIntervalo

    if (n != 0){
      h <- (limiteSuperior - limiteInferior)/n
    }


    for (i in 1:(n+1)) {
      fx[i] <- eval(funcion, list(x = limiteInferior + (i-1)*h))
    }

    if(n == 2){
      resultado <- resultado + (h/3) * (fx[1] + 4*fx[2] + fx[3])
    }

    limiteInferior <- limiteSuperior
  }

  return(resultado)
}

sigma <- 0.22*sqrt(0.75)
mu <- 2.77

# n = 2. Simpson
# esperanza <- if( a <- IntegracionCompuesta(limiteInferior = 12.68, limiteSuperior = 20.42, funcion =
#
# esperanza

# Esto lo hizo dario en clase

f <- function(x){
  if(x>40){
    return(x-40)
  } else {
    return(0)
  }
}

x1 <- seq(0,100)
y <- sapply(x1, f)
plot(x1,y)
```




```
u <- function(x){ ifelse(x>40, x-40, 0)}

IntegracionCompuesta(limiteInferior = 10^-10, limiteSuperior = 100, funcion = expression(u(x) * (1/(x*s
```

```
## [1] 1.076014e-06
```

2 Simulación de Precios (25 puntos)

Considere el siguiente modelo para simular precios:

$$P_{t+\Delta t} = P_t \times \exp\left[(\mu - 0.5\sigma^2)\Delta t + \sigma\sqrt{\Delta t}\epsilon\right]$$

$$P_T = P_0 \times \exp\left[(\mu - 0.5\sigma^2)T + \sigma\sqrt{T}\epsilon\right]$$

donde ϵ es una variable aleatoria normal estándar. Para su ejercicio particular, considere $P_0 = 300$, $T = 0.75$, $\mu = 0.1$ y $\sigma = 0.18$.

2.1 Simulación de P_T

Realice $15,000$ simulaciones de precios en el momento T, P_T. Almacene dicha simulación en una matriz (o *data frame*) llamada `PT`. Grafique un histograma de los precios simulados y calcule el Precio Esperado y el Desvío Estándar.

Nota: Debe simular solamente precios finales (no debe usar "caminos de precios").

Respuesta: Precio esperado : y Desvio Estandar :

```
# Ingrese a continuación, en este bloque, todo el código necesario para resolver el ejercicio 2.1

p0 <- 300
mu <- 0.1
sigma <- 0.18

# 0.75 año
anios <- 0.75

# Simulacion
m <- 15000

# Matriz de camino de precios
pt <- matrix(NA, nrow = m, ncol = 1)

#set.seed(895096)
e <- rnorm(m)

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

mean(pt)
```

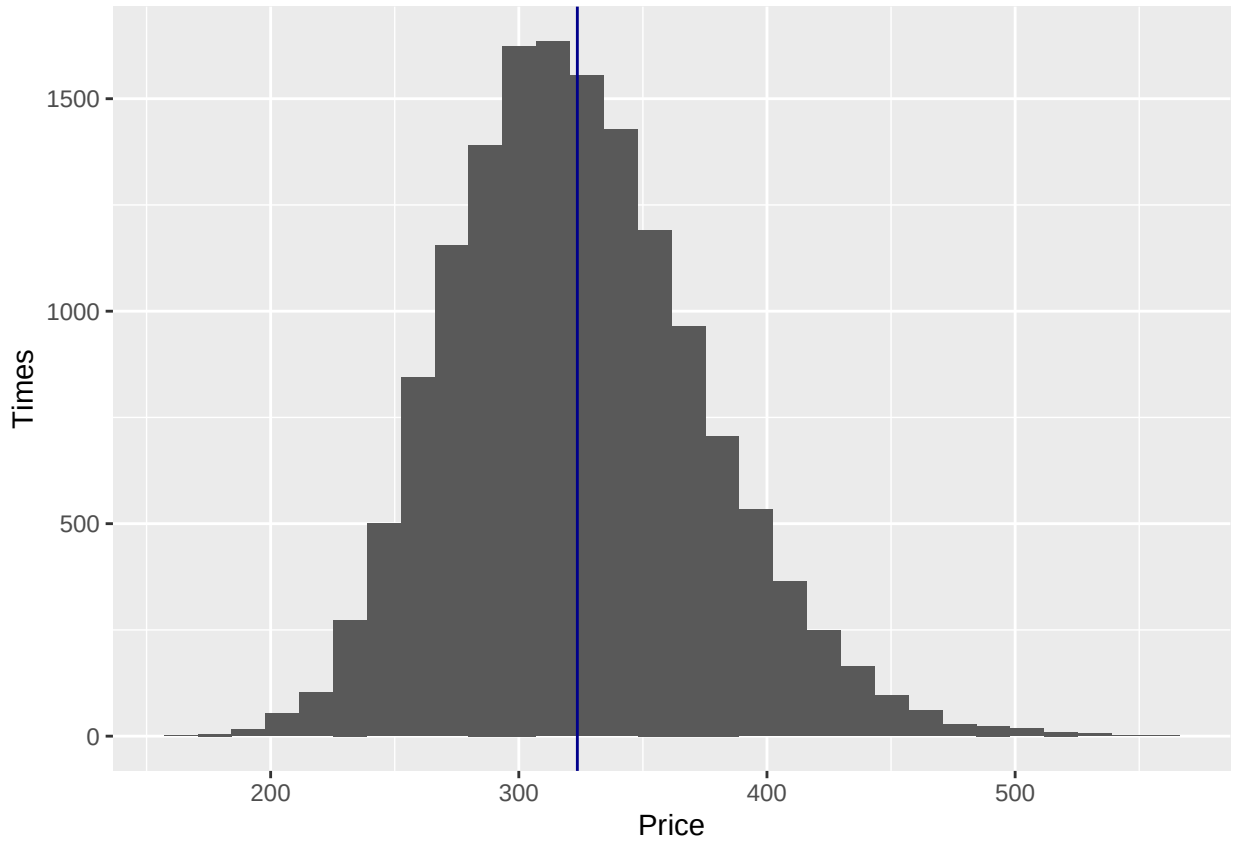
```
## [1] 323.5844
```

```
sd(pt)
```

```
## [1] 50.5384
```

```
ggplot() +
  geom_histogram(aes(pt[,1])) +
  geom_vline(xintercept = mean(pt), colour = "darkblue") +
  xlab("Price") + ylab("Times")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



2.2 Simulación de caminos de precios P_t con $t > 0$

Realice 2000 simulaciones de ***caminos de precios*** desde $t = 0$ hasta $t = T$, con $\Delta t = 1/250$. Almacene los resultados en una matriz (o *data frame*) llamada `P0aT`. Grafique un histograma de los **precios finales** simulados P_T y calcule el $E(P_T)$ y $d.e.(P_T)$ (desvío estándar).

Respuesta: Valor esperdo = 270.7219 y Desvio Estandar = 44.10201

```
# Ingrese a continuación, en este bloque, todo el código necesario para resolver el ejercicio 2.2

p0 <- 300
mu <- 0.1
sigma <- 0.18

# 0.75 año
anios <- 0.75

dt <- 1/250

# Simulacion
m <- 2000

# Time steps
n <- 187.5

# Matriz de camino de precios
P0aT <- matrix(NA, nrow = m, ncol = n+1)
```

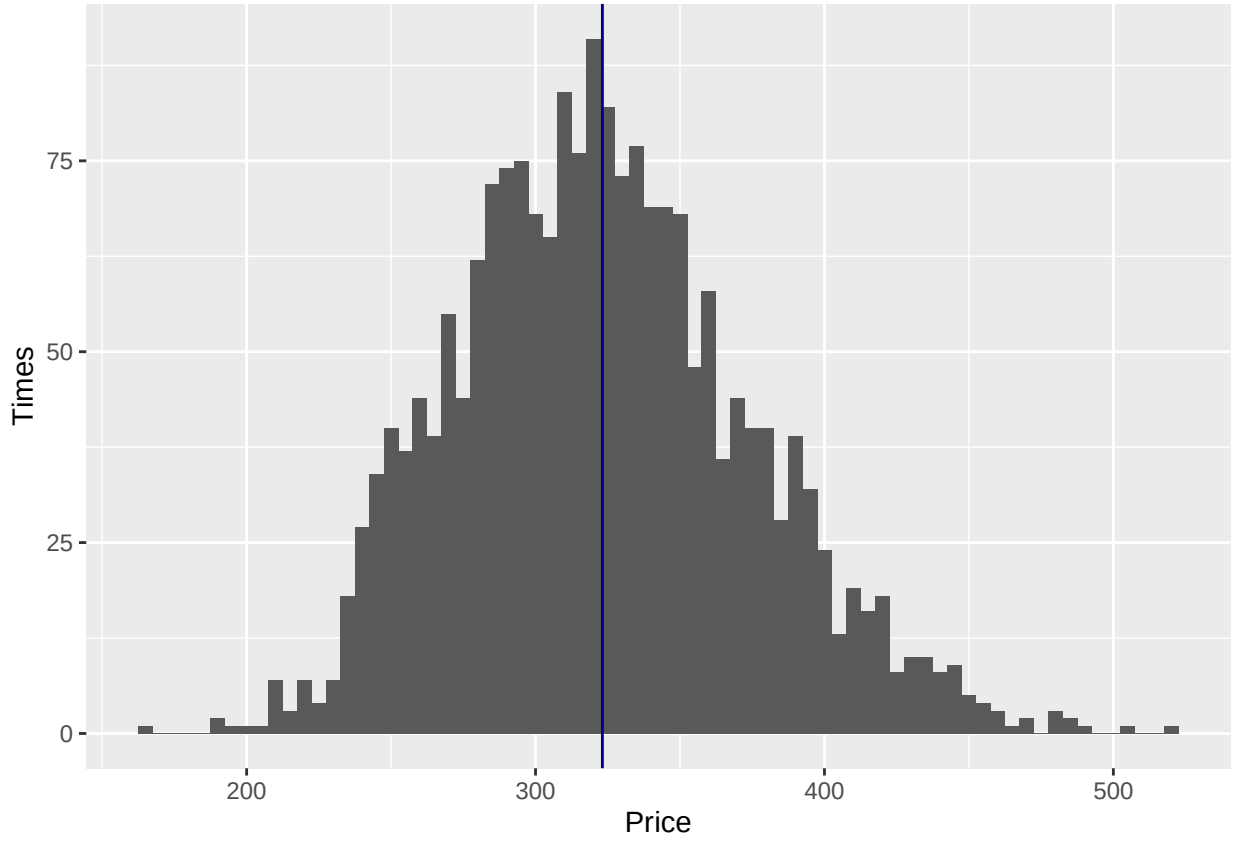
```r
P0aT[,1] <- p0

for (i in 1:m) {
  for(t in 2:(n+1)){
    P0aT[i,t] <- P0aT[i, t-1] * exp((mu-0.5*sigma^2) * dt + sigma * sqrt(dt) * rnorm(1))
  }
}

mean(P0aT[,ncol(P0aT)])
```

```
## [1] 323.1373
```

```r
sd(P0aT[,ncol(P0aT)])
```

```
## [1] 50.74475
```

```r
ggplot() +
  geom_histogram(aes(P0aT[,ncol(P0aT)]), binwidth = 5) +
  geom_vline(xintercept = mean(P0aT[,ncol(P0aT)]), colour = "darkblue") +
  xlab("Price") + ylab("Times")
```

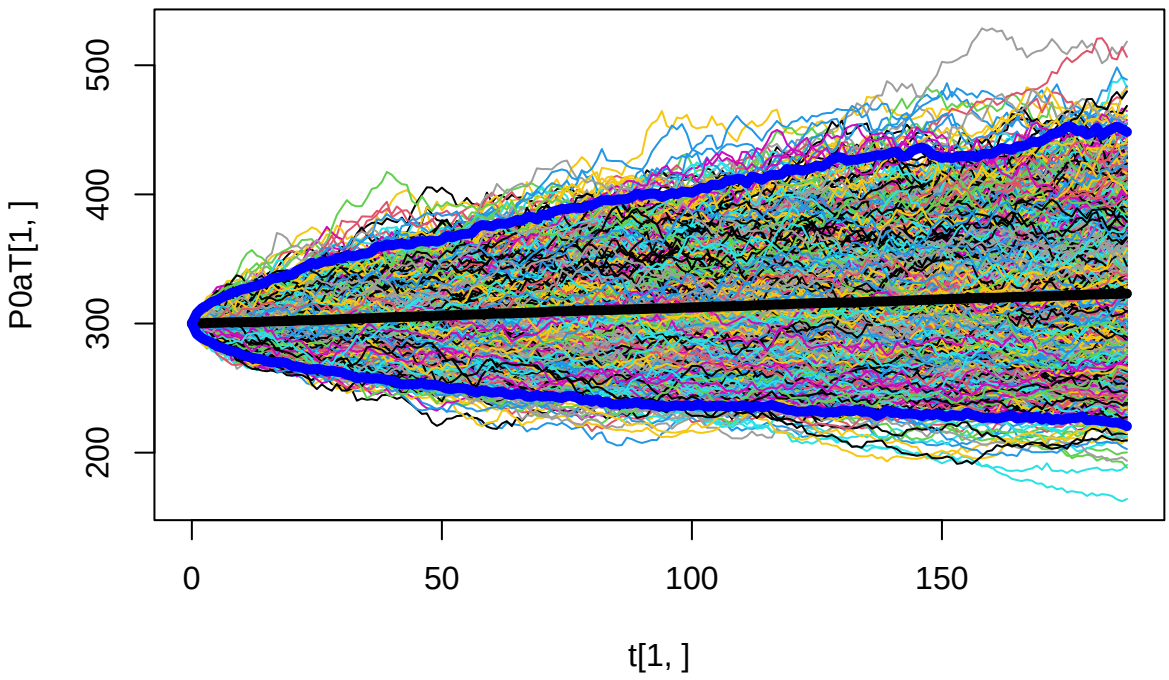


2.3 Gráfico de caminos de precios P_t con $t > 0$ (incluyendo $E[P_t]$ e IC)

Con los resultados del punto anterior, grafique todos los caminos simulados con colores al azar. Incluya con línea gruesa de color **negro** el camino del valor esperado y con líneas gruesas de color **azul** el camino de un Intervalo de Confianza con 99% de probabilidad.

Respuesta:

```r
# Ingrese a continuación, en este bloque, todo el código necesario para resolver el ejercicio 2.3
# Gráfico

t <- rep(0:n, m)
t <- matrix(t, nrow = m, ncol = n+1, byrow = T)

plot(t[1,], POaT[1,], type = "l", ylim = c(min(POaT), max(POaT)))
for (i in 2:m) {
  lines(t[i,], POaT[i,], col = trunc(runif(1)*m))
}

# Medias
m <- matrix(NA, nrow = 1, ncol = n+1)
for (i in 1:(n+1)) {
  m[i] <- mean(POaT[,i])
}

# IC
prob <- 0.99
ls <- matrix(NA, nrow = 1, ncol = n+1)
for (i in 1:(n+1)) {
  ls[i] <- quantile(POaT[,i], prob)
}

li <- matrix(NA, nrow = 1, ncol = n+1)
for (i in 1:(n+1)) {
  li[i] <- quantile(POaT[,i], 1-prob)
}

lines(t[1,], m, col = 'black', lwd = 5)
lines(t[1,], ls, col = 'blue', lwd = 5)
lines(t[1,], li, col = 'blue', lwd = 5)
```



2.4 Estimación de probabilidades a partir de muestras simuladas

Usando las matrices `PT` (del punto 2.1) y `P0aT` (del punto 2.2), realice dos estimaciones de la probabilidad de que el precio final P_T (con $T = 0.75$) sea menor a 268.95 y dos estimaciones de la probabilidad de que el precio sea mayor a 364.79. Compare los resultados obtenidos usando `PT` (del punto 2.1) con los resultados usando `P0aT` (del punto 2.2)

Respuesta:

```
# Ingrese a continuación, en este bloque, todo el código necesario para resolver el ejercicio 2.4

# Hacemos el supuesto de que los precios no pueden ser menores a cero

# Tengo dos ideas:
# 1) A partir de las matrices genero un polinomio mediante interpolación (Ej: Spline) y con eso calculo
# 2) Calculo la proporción sobre el total


df_pt <- as.data.frame(pt)

punto1 <- rep(NA, times = 2)

a <- df_pt %>% filter(V1 < 268.95)

punto1[1] <- nrow(a) / nrow(df_pt)

df_p0aT <- as.tibble(P0aT[,ncol(P0aT)])
```

```
## Warning: `as.tibble()` was deprecated in tibble 2.0.0.
## Please use `as_tibble()` instead.
## The signature and semantics have changed, see `?as_tibble`.
```

```
a <- df_p0aT %>% filter(value < 268.95)

punto1[2] <- nrow(a) / nrow(df_p0aT)

# Mayor a 364.79

df_pt <- as.data.frame(pt)

punto2 <- rep(NA, times = 2)

a <- df_pt %>% filter(V1 > 364.79)

punto2[1] <- nrow(a) / nrow(df_pt)

df_p0aT <- as.tibble(P0aT[,ncol(P0aT)])

a <- df_p0aT %>% filter(value > 364.79)

punto2[2] <- nrow(a) / nrow(df_p0aT)

df_resultado <- data.frame("menor a 268.95" = punto1, "mayor a 364.79" = punto2)

rownames(df_resultado) <- c("pt", "p0aT")


df_resultado
```

```
##      menor.a.268.95 mayor.a.364.79
## pt           0.1344         0.1998
## p0aT         0.1425         0.1985
```

3 Integración y Derivación Numérica (20 puntos)

Considere la siguiente función de densidad de la variable aleatoria Y, con dominio en el intervalo $Y \in (0;1)$, y parámetros $\alpha = 3$ y $\beta = 1$:

$$f_Y(x|\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1}$$

La función $\Gamma(z)$ está definida por la siguiente integral: $\Gamma(z) = \int_0^\infty t^{z-1}e^{-t}dt$.

3.1 $E(Y|\alpha;\beta)$

Calcule mediante una integral numérica, usando Simpson Compuesto con $n = 100$, la esperanza matemática de la variable aleatoria Y. Los valores de la función $\Gamma(z)$ que se incluyen en la función de densidad también debe estimarlos numéricamente usando Simpson Compuesto con $n = 1000$ y un valor de límite superior que considere adecuado.

```
# Ingrese a continuación, en este bloque, todo el código necesario para resolver el ejercicio 3.1

IntegracionCompuesta <- function(limiteInferior, limiteSuperior, funcion, n, cantIntervalos){

  #browser()
  if ((n == 2 || n == 0) && cantIntervalos%%2 != 0){
    return("cantIntervalos debe ser un entero par")
  }

  cantIntervalos <- cantIntervalos/n


  crecimientoIntervalo <- (limiteSuperior-limiteInferior)/cantIntervalos

  fx <- rep(NA, times = (n+1))

  resultado <- 0

  for (i in 1:cantIntervalos) {
    limiteSuperior <- limiteInferior + crecimientoIntervalo

    if (n != 0){
      h <- (limiteSuperior - limiteInferior)/n
    }


    for (i in 1:(n+1)) {
      fx[i] <- eval(funcion, list(x = limiteInferior + (i-1)*h))
    }

    if(n == 2){
      resultado <- resultado + (h/3) * (fx[1] + 4*fx[2] + fx[3])
    }

    limiteInferior <- limiteSuperior
  }
```

```
  return(resultado)
}


alfa <- 3
beta <- 1

gamma_alfa_beta <- IntegracionCompuesta(limiteInferior = 0, limiteSuperior = 1000, funcion = expression

gamma_alfa <- IntegracionCompuesta(limiteInferior = 0, limiteSuperior = 1000, funcion = expression( x^(a

gamma_beta <- IntegracionCompuesta(limiteInferior = 0, limiteSuperior = 1000, funcion = expression( x^(b

IntegracionCompuesta(limiteInferior = 0, limiteSuperior = 1, funcion = expression( x * (gamma_alfa_beta/
```

```
## [1] 0.7373213
```

3.2 Derivada respecto a α

Estime numéricamente la derivada de $E(Y|\alpha;\beta)$ respecto del parámetro α.

Vuelvo a la definición clásica de derivada:

$$\lim_{x\to\infty} f'(x) = \frac{f(x+h) - f(x)}{h}$$

```
# Ingrese a continuación, en este bloque, todo el código necesario para resolver el ejercicio 3.2

alfa <- 3
beta <- 1

gamma_alfa_beta <- IntegracionCompuesta(limiteInferior = 0, limiteSuperior = 1000, funcion = expression

gamma_alfa <- IntegracionCompuesta(limiteInferior = 0, limiteSuperior = 1000, funcion = expression( x^(a

gamma_beta <- IntegracionCompuesta(limiteInferior = 0, limiteSuperior = 1000, funcion = expression( x^(b

fx <- IntegracionCompuesta(limiteInferior = 0, limiteSuperior = 1, funcion = expression( x * (gamma_alfa

alfa <- 3 + 10^-10

gamma_alfa_beta <- IntegracionCompuesta(limiteInferior = 0, limiteSuperior = 1000, funcion = expression

gamma_alfa <- IntegracionCompuesta(limiteInferior = 0, limiteSuperior = 1000, funcion = expression( x^(a

gamma_beta <- IntegracionCompuesta(limiteInferior = 0, limiteSuperior = 1000, funcion = expression( x^(b

fx_h <- IntegracionCompuesta(limiteInferior = 0, limiteSuperior = 1, funcion = expression( x * (gamma_al

(fx_h - fx) / (10^-10)
```

```
## [1] 0.06812217
```

3.3 Derivada respecto a β

Estime numéricamente la derivada de $E(Y|\alpha;\beta)$ respecto del parámetro β.

```r
# Ingrese a continuación, en este bloque, todo el código necesario para resolver el ejercicio 3.3

alfa <- 3
beta <- 1

gamma_alfa_beta <- IntegracionCompuesta(limiteInferior = 0, limiteSuperior = 1000, funcion = expression

gamma_alfa <- IntegracionCompuesta(limiteInferior = 0, limiteSuperior = 1000, funcion = expression( x^(a

gamma_beta <- IntegracionCompuesta(limiteInferior = 0, limiteSuperior = 1000, funcion = expression( x^(b

fx <- IntegracionCompuesta(limiteInferior = 0, limiteSuperior = 1, funcion = expression( x * (gamma_alfa

beta <- 1 + 10^-10

gamma_alfa_beta <- IntegracionCompuesta(limiteInferior = 0, limiteSuperior = 1000, funcion = expression

gamma_alfa <- IntegracionCompuesta(limiteInferior = 0, limiteSuperior = 1000, funcion = expression( x^(a

gamma_beta <- IntegracionCompuesta(limiteInferior = 0, limiteSuperior = 1000, funcion = expression( x^(b

fx_h <- IntegracionCompuesta(limiteInferior = 0, limiteSuperior = 1, funcion = expression( x * (gamma_al

(fx_h - fx) / (10^-10)
```

```
## [1] NaN
```

```r
print("gamma_beta no puede ser < 1 porque si no la función no converge a ningún número.")
```

```
## [1] "gamma_beta no puede ser < 1 porque si no la función no converge a ningún número."
```

4 SQL (20 Puntos)

Utilice la base de datos de la siguiente web para preparar sus códigos de SQL: https://www.w3schools.com/sql/trysql.asp?filename=trysql_select_all

4.1 Consulta Clientes

Escriba una consulta SQL que contenga los clientes de Estados Unidos y Alemania que gastaron más de $10.000 en el total de todas sus compras. La salida debe contener los siguientes campos: Nombre del Cliente, ciudad y país del cliente, Cantidad de compras, Total (calculado como la suma total del Precio*Cantidad).

Respuesta (SELECT c.CustomerName as NombreCliente, c.City as Ciudad, c.Country as Pais, count(DISTINCT o.OrderID) as CantidadCompra, ROUND(sum(p.Price*d.Quantity),2) as Total FROM Customers c INNER JOIN Orders o ON c.CustomerID=o.CustomerID INNER JOIN OrderDetails d ON o.OrderID=d.OrderID INNER JOIN Products p ON d.ProductID=p.ProductID WHERE Country = 'USA' GROUP BY c.CustomerName HAVING sum(p.Price*d.Quantity) > 10000 UNION SELECT c.CustomerName as NombreCliente, c.City as Ciudad, c.Country as Pais, count(DISTINCT o.OrderID) as CantidadCompra, ROUND(sum(p.Price*d.Quantity),2) as Total FROM Customers c INNER JOIN Orders o ON c.CustomerID=o.CustomerID INNER JOIN OrderDetails d ON o.OrderID=d.OrderID INNER JOIN Products p ON d.ProductID=p.ProductID WHERE Country = 'Alemania' GROUP BY c.CustomerName HAVING sum(p.Price*d.Quantity) > 10000):

4.2 Consulta Ventas

Escriba una consulta SQL que contenga todas las ventas realizadas por Robert King a los clientes de España, México y Canadá. La salida debe contener los siguientes campos: Nombre y Apellido del Vendedor, Nombre del Cliente, Ciudad y País del Cliente, ID de la Orden de Compra (tabla Orders), Nombre del Producto (tabla Products), Presentación (Unit, de tabla Products), Cantidad (tabla OrderDetails), Precio (tabla Products) y Total (calculado como Precio*Cantidad).

Respuesta (SELECT E.LastName as Apellido, FirstName as Nombre, U.CustomerName as NombreCliente,U.City as Ciudad, U.Country as Pais, R.OrderID as IDOrder, P.ProductName as NombreProducto, P.Unit as Presentacion, O.Quantity as Cantidad, P.price as Precio, round(P.Price*(O.Quantity),2) as Total FROM Products P INNER JOIN OrderDetails O ON P.ProductID = O.ProductID INNER JOIN Orders R ON O.OrderID = R.OrderID INNER JOIN Employees E ON R.EmployeeID = E.EmployeeID INNER JOIN Customers U ON R.CustomerID = U.CustomerID WHERE LastName = 'King' AND Country = 'Spain' UNION SELECT E.LastName as Apellido, FirstName as Nombre, U.CustomerName as NombreCliente,U.City as Ciudad, U.Country as Pais, R.OrderID as IDOrder, P.ProductName as NombreProducto, P.Unit as Presentacion, O.Quantity as Cantidad, P.price as Precio, round(P.Price*(O.Quantity),2) as Total FROM Products P INNER JOIN OrderDetails O ON P.ProductID = O.ProductID INNER JOIN Orders R ON O.OrderID = R.OrderID INNER JOIN Employees E ON R.EmployeeID = E.EmployeeID INNER JOIN Customers U ON R.CustomerID = U.CustomerID WHERE LastName = 'King' AND Country = 'Canada' UNION SELECT E.LastName as Apellido, FirstName as Nombre, U.CustomerName as NombreCliente,U.City as Ciudad, U.Country as Pais, R.OrderID as IDOrder, P.ProductName as NombreProducto, P.Unit as Presentacion, O.Quantity as Cantidad, P.price as Precio, round(P.Price*(O.Quantity),2) as Total FROM Products P INNER JOIN OrderDetails O ON P.ProductID = O.ProductID INNER JOIN Orders R ON O.OrderID = R.OrderID INNER JOIN Employees E ON R.EmployeeID = E.EmployeeID INNER JOIN Customers U ON R.CustomerID = U.CustomerID WHERE LastName = 'King' AND Country = 'Mexico'):

5 Aprendizaje Automático (10 puntos)

Defina el aprendizaje automático y mencione qué ventajas se pueden obtener de su utilización. Explique brevemente los dos tipos de métodos más utilizados en la actualidad.

Respuesta (ML es un conjunto de técnicas de programación y métodos, que permiten el análisis de grandes volúmenes de datos y detectar automáticamente patrones sin intervención humana, basados en experiencia pasada. Debido al Big Data, las organizaciones precisan del ML para aprovechar los datos que se obtienen de las fuentes externas, como por ej redes sociales, datos del supermercado, etc, para poder mejorar, optimizar y automatizar procesos para una mejor toma de decisiones. Algunas de las utilidades cotidianas se aplican justamente en las compras en el supermercado en lo que llamamos segmentacion de clientes, y por ejemplo en el correo electronico cuando distinguimos entre spam y no spam. Esto lo podemos lograr gracias a los metodos que se utilizan hoy en dia :el aprendizaje supervisado y el aprendizaje no supervisado. En el supervisado se le asigna a cada salida una etiqueta (tenemos 1000 mails, separados en dos grupos a los cuales les asigno la etiqueta de spam o no spam). En el no supervisado los datos no tienen etiquetas asociadas (tenemos los dos grupos de mails , pero no poseen la etiqueta de cual es spam y cual no)):