CIND-221: Taller 4

Felipe Osorio

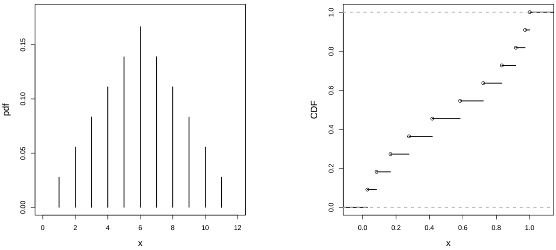
f.osoriosalgado@uandresbello.edu

Facultad de Ingeniería, UNAB

Ejemplo (lanzamiento de dos dados):

Considere el lanzamiento de 2 dados. Sea X la suma de sus caras. Entonces,

x	2	3	4	5	6	7	8	9	10	11	12
$\mathsf{P}(X = x)$	$\frac{1}{36}$	$\frac{2}{36}$	$\frac{3}{36}$	$\frac{4}{36}$	$\frac{5}{36}$	$\frac{6}{36}$	$\frac{5}{36}$	$\frac{4}{36}$	$\frac{3}{36}$	$\frac{2}{36}$	$\frac{1}{36}$



(a) $p(x)$ (b) $F(x)$

Ejemplo:

Suponga

$$f(x) = \frac{1}{2}\exp(-x/2), \qquad x > 0,$$

con $\lambda = 1/2$.

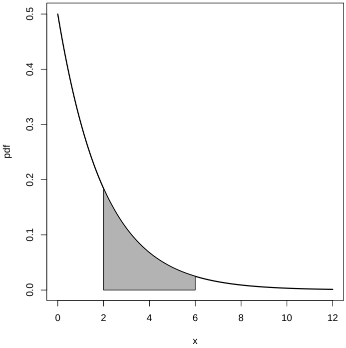
Tenemos que

$$F(x) = 1 - \exp(-x/2), \qquad x > 0,$$

donde $F(x) = \mathsf{P}(X \leq x)$.

Variable aleatoria

Considere P$(2 < X < 6)$, es decir:



De este modo,

$$\begin{aligned} \mathsf{P}(2 < X < 6) &= \frac{1}{2}\int_2^6 \exp(-x/2)\,\mathrm{d}x = F(6) - F(2) \\ &= [1 - \exp(-3)] - [1 - \exp(-1)] = 0.9502 - 0.6321 \\ &= 0.3181 \end{aligned}$$

Adicionalmente,

$$\mathsf{P}(X < 8) = F(8) = 1 - \exp(-4) = 0.9817,$$

y

$$\mathsf{P}(X \geq 8) = 1 - F(8) = 1 - 0.9817 = 0.0183.$$

```
# evaluando la densidad exponencial
> x <- seq(0, 13, length = 300)
> length(x)
[1] 300
> x[1:10]
 [1] 0.00000000 0.04347826 0.08695652 0.13043478 0.17391304
 [6] 0.21739130 0.26086957 0.30434783 0.34782609 0.39130435

> y <- dexp(x, rate = .5)
> plot(x, y, type = "l", ylab = "pdf", lwd = 2)

# calculando probabilidades acumuladas
> pexp(6, rate = .5) # F(6)
[1] 0.9502129
> pexp(2, rate = .5) # F(2)
[1] 0.6321206
> pexp(6, rate = .5) - pexp(2, rate = .5) # F(6) - F(2)
[1] 0.3180924
> pexp(8, rate = .5) # F(8)
[1] 0.9816844
> pexp(8, rate = .5, lower = FALSE) # P(X > 8)
[1] 0.01831564
> 1 - pexp(8, rate = .5)
[1] 0.01831564

```

Cadenas de Markov

```r
matrix.power <- function(a, pow = 2)
{ ## computes the power of a square matrix
  if (is.data.frame(a))
    a <- as.matrix(a)
  if (!is.matrix(a))
    stop("supply a matrix-like 'a'")
  if (!is.numeric(a))
    stop("argument a is not a numeric matrix")

  da <- dim(a)
  n <- da[1]
  p <- da[2]
  if (n != p)
    stop("argument a is not a square matrix")

  if (pow < 0)
    stop("only implemented for positive power")
  z <- a
  k <- pow
  if (k == 0)
    return (diag(n))
  if (k == 1)
    return (z)
  if (k > 1)
    return(z %*% matrix.power(z, pow = k - 1))
}

```

```
# leyendo fuentes R
source("matrix.power.R")

# creando una matriz
> p <- matrix(c(.7, .3, .4, .6), ncol = 2, byrow = TRUE)

# imprimiendo en pantalla
> p
     [,1] [,2]
[1,]  0.7  0.3
[2,]  0.4  0.6

# sumando por filas (MARGIN = 1) o columnas (MARGIN = 2)
> apply(p, 1, sum)
[1] 1 1
> apply(p, 2, sum)
[1] 1.1 0.9

```

Cadenas de Markov

```
> matrix.power(p) # argumento 'pow' por defecto
     [,1] [,2]
[1,] 0.61 0.39
[2,] 0.52 0.48

> matrix.power(p, pow = 0)
     [,1] [,2]
[1,]    1    0
[2,]    0    1

> matrix.power(p, pow = 4)
       [,1]   [,2]
[1,] 0.5749 0.4251
[2,] 0.5668 0.4332

> matrix.power(p, pow = 12)
          [,1]      [,2]
[1,] 0.5714288 0.4285712
[2,] 0.5714283 0.4285717

> matrix.power(p, pow = 20)
          [,1]      [,2]
[1,] 0.5714286 0.4285714
[2,] 0.5714286 0.4285714
```

Cadenas de Markov

```
# ejemplo Slide 15
> p4 <- matrix.power(p, 4)
> p4
       [,1]   [,2]
[1,] 0.5749 0.4251
[2,] 0.5668 0.4332

> init <- c(.4, .6)
> init # distribución inicial
[1] 0.4 0.6

> pred <- init %*% p4
> pred # distribución de X4
        [,1]    [,2]
[1,] 0.57004 0.42996

# chequeo del tipo de datos
> is.matrix(pred)
[1] TRUE
> is.matrix(init)
[1] FALSE

```

```
# ejemplo Slide 3
> ex <- matrix(c(.75,.25,0,.25,.5,.75,0,.25,.25), ncol = 3)

> ex
     [,1] [,2] [,3]
[1,] 0.75 0.25 0.00
[2,] 0.25 0.50 0.25
[3,] 0.00 0.75 0.25

> matrix.power(ex, pow = 10)
          [,1]      [,2]      [,3]
[1,] 0.4320240 0.4265490 0.1414270
[2,] 0.4265490 0.4297562 0.1436949
[3,] 0.4242811 0.4310846 0.1446342

> matrix.power(ex, pow = 20)
          [,1]      [,2]      [,3]
[1,] 0.4285936 0.4285585 0.1428480
[2,] 0.4285585 0.4285790 0.1428625
[3,] 0.4285439 0.4285875 0.1428685

> matrix.power(ex, pow = 40)
          [,1]      [,2]      [,3]
[1,] 0.4285714 0.4285714 0.1428571
[2,] 0.4285714 0.4285714 0.1428571
[3,] 0.4285714 0.4285714 0.1428571

```

```
> p <- matrix(c(.2,.1,.1,.6,.8,.6,.2,.1,.3), ncol = 3)
> p
     [,1] [,2] [,3]
[1,]  0.2  0.6  0.2
[2,]  0.1  0.8  0.1
[3,]  0.1  0.6  0.3

# decomposición espectral de 'p'
> rs <- eigen(p)
> rs
eigen() decomposition
$values
[1] 1.0 0.2 0.1

$vectors
           [,1]       [,2]       [,3]
[1,] -0.5773503  0.6882472 -0.9847319
[2,] -0.5773503 -0.2294157  0.1230915
[3,] -0.5773503  0.6882472  0.1230915

```

```
# vector cuyos elementos son todos 1
> ones <- rep(1, 3)
> ones
[1] 1 1 1

# norma del vector
> sqrt(sum(ones^2)) # sqrt(3)
[1] 1.732051

# normalizando
> ones / sqrt(sum(ones^2))
[1] 0.5773503 0.5773503 0.5773503

# vectores propios
> rs$vectors
           [,1]       [,2]       [,3]
[1,] -0.5773503  0.6882472 -0.9847319
[2,] -0.5773503 -0.2294157  0.1230915
[3,] -0.5773503  0.6882472  0.1230915

```

```r
# vector cuyos elementos son todos 1
> ones <- rep(1, 3)
> ones
[1] 1 1 1

# norma del vector
> sqrt(sum(ones^2)) # sqrt(3)
[1] 1.732051

# normalizando
> ones / sqrt(sum(ones^2))
[1] 0.5773503 0.5773503 0.5773503

# vectores propios
> rs$vectors
           [,1]       [,2]       [,3]
[1,] -0.5773503  0.6882472 -0.9847319
[2,] -0.5773503 -0.2294157  0.1230915
[3,] -0.5773503  0.6882472  0.1230915

```

```
> z <- t(p) # matriz transpuesta
> eigen(z)
eigen() decomposition
$values
[1] 1.0 0.2 0.1

$vectors
          [,1]          [,2]          [,3]
[1,] 0.1441500 -1.345794e-16  7.071068e-01
[2,] 0.9730125 -7.071068e-01 -1.296942e-16
[3,] 0.1801875  7.071068e-01 -7.071068e-01

> eigen(p)
eigen() decomposition
$values
[1] 1.0 0.2 0.1

$vectors
           [,1]       [,2]       [,3]
[1,] -0.5773503  0.6882472 -0.9847319
[2,] -0.5773503 -0.2294157  0.1230915
[3,] -0.5773503  0.6882472  0.1230915

```

```r
stationary <- function(x)
{ ## finds the stationary distribution of a Markov
  ## chain with transition matrix 'x'
  z <- t(x)
  x <- eigen(z)$vectors[,1] # 1st column
  x <- as.vector(x / sum(x))
  x
}

```

```r
# obteniendo la distribución estacionaria
> u <- stationary(p)
> u
[1] 0.1111111 0.7500000 0.1388889

> u > 0
[1] TRUE TRUE TRUE
> sum(u)
[1] 1

```

Usando fastmatrix[1]

```
# otra alternativa, usando 'fastmatrix'
# disponible en: https://github.com/faosorios/fastmatrix
> z <- power.method(t(p))
> z
$value
[1] 1

$vector
[1] 0.1441500 0.9730125 0.1801875

attr(,"iterations")
[1] 13

# distribución debe sumar 1
> st <- z$vector / sum(z$vector)
> st
[1] 0.1111111 0.7500000 0.1388889

```

[1] Disponible en CRAN: https://cran.r-project.org/package=fastmatrix