Introducción a Estadística con R
Servicio Social
Rubén Hernández Cid

Ana Cristina Pérez-Gea González
118440

16 de abril de 2015

Índice general

Capítulo 1

Introducción a R

R es un lenguaje de programación y un ambiente de cómputo estadístico. Es muy intuitivo, por lo que es fácil de aprender y también es fácil de manejar para programadores avanzados. R es un software libre y de código abierto, por lo que su uso es gratuito y todo su código está disponible. También forma parte de un proyecto colaborativo y abierto, con un extenso repositorio de paquetes. Los paquetes básicos fueron creados por los desarrolladores de R mientras que los usuarios han también desarrollado paquetes y compartido con los demás usuarios a través de la página web de R. Estos usuarios desarrolladores frecuentemente trabajan en equipos y dan mantenimiento continuo a sus paquetes. R fue desarrollado por Robert Gentleman y Ross Ihaka del Departamento de Estadística de la Universidad de Auckland, Nueva Zelanda en 1993. Su desarrollo actual es responsabilidad de R Development Core Team, quienes mantienen la página de internet al día.

R está disponible para los sistemas operativos Windows, Macintosh, Unix y GNU/Linux. Se puede descargar de la página de internet http://cran.r-project.org. El R Project tiene un manual de instalación y administración de R con instrucciones completas de cómo usar R así como información adicional sobre el lenguaje.

Cuando un usuario instala R, instala la base, es decir un conjunto de paquetes que se consideran básicos para su uso. Gran parte de la funcionalidad adicional está en paquetes que la comunidad contribuye que se tiene que instalar por separado. Los paquetes son mantenidos por voluntarios y se puede ver la información de éstos en la misma página de internet mencionada. La paquetería de R está guardada en diferentes computadoras alrededor del mundo, con base principal en Austria. Para ver los paquetes actualmente en instalados se puede acudir a la función `library()`. Para poder usar un paquete, por ejemplo el paquete foreign, primero se instala con `install.packages("foreign")` y se carga con `library(foreign)`.

1.1. Para entender R

1. Al abrir R se dice que hay una sesión de R corriendo. La consola de R es la interfaz entre R y los usuarios. El signo de $>$ al principio de la línea significa que R está esperando un comando.

2. En la sesión hay objetos. Todo en R es un objeto: vectores, tablas, funciones, etc. Y cada objeto tiene sus propiedades.

3. R opera aplicando funciones a los objetos y creando nuevos objetos. Cada función tiene ciertos tipos de objetos como argumentos y también como resultado.

Las operaciones en R son bastante intuitivas. En este documento vamos a mencionar brevemente la teoría seguida por ejemplos. Estos ejemplos son en gran parte de ??. En las siguientes secciones de este capítulo haremos una introducción al lenguaje de R con ejemplos correspondientes.

1.1.1. Aritmética

Empezaremos con las operaciones básicas de R. En el siguiente ejemplo se muestran operaciones básicas y cómo guardar los resultados en variables con distintos nombres. El signo <- es el usado con mas frecuencia para definir las variables, aunque es también se usa = equivalentemente[1].

```
> x <- 2 + 11 # se guarda una operacion en el objeto x
> x
```

```
[1] 13
```

```
> y <- x^2 - 1 # se llama a x para crear a y
> y
```

```
[1] 168
```

```
> ls() # se listan los objetos en la memoria
```

```
[1] "x" "y"
```

Todo lo que está después de # es comentario y R lo ignora. También se pueden modificar las opciones de R como, por ejemplo, para especificar el número de decimales que imprime R en la consola. Por defecto R imprime 7 decimales y el numero máximo de decimales que R puede mostrar es 22.

```
> options(digits = 16)
> sqrt(2) # raiz cuadrada de 2
```

```
[1] 1.414213562373095
```

```
> exp(1) # e
```

```
[1] 2.718281828459045
```

```
> pi # pi
```

[1] En funciones, se tiene que usar necesariamente = para definir los argumentos cuando el argumento no es implícito

```
[1] 3.141592653589793
```

Hay que tener cuidado porque R opera con métodos numéricos y por lo tanto los dígitos adicionales significativos no son necesariamente confiables.

1.1.2. Tipos de datos

Para nombrar una variable se pueden elegir letras y números, que se pueden separar por puntos y guiones bajos aunque se tiene que comenzar forzosamente con una letra. Por ejemplo, las variable puede ser `x`, `x1`, `x.1` o `x_1`. Los tipos de datos mas usados para definir variables son los siguientes.

- integer: numeros enteros;
- double: numeros reales que pueden ser racionales o irracionales;
- character: caracteres que se representan con " o equivalentemente con ' alrededor de la variable;
- logical: son valores lógicos que pueden ser `TRUE` para verdadero, `FALSE` para falso o `NA` que son las siglas de "not available" y representa valores faltantes;
- complex: numeros complejos.

Se puede determinar el tipo de un objeto dado usando la función `typeof` y se puede cambiar el tipo de la variable con la función `as.tipo`, donde tipo es de dato anteriormente mencionado.

```
> x ← 57
> typeof(x)
```

```
[1] "double"
```

```
> x ← as.integer(x) # se cambia a tipo entero
> typeof(x)
```

```
[1] "integer"
```

```
> y ← 'hola' # es equivalente usar "hola"
> typeof(y)
```

```
[1] "character"
```

```
> sqrt(-2) # no esta definido
```

```
[1] NaN
```

```
> sqrt(as.complex(-2))
```

```
[1] 0+1.4142135623730951i
```

```
> typeof(5 + 3i) # numero complejo
```

```
[1] "complex"
```

```
> 1 > 2
```

```
[1] FALSE
```

El resultado `NaN` representa las siglas en inglés de "not a nomber" que significa que la expresión no está definida.

1.2. Clases de Objetos

Múltiples variables se pueden guardar en diferentes clases de objetos. Un vector es una concatenación de datos y se puede crear de distintas maneras. La función `c` concatena valores dados por el usuario.

```
> vec.1 <- c(1, -1, 0, 2, 0, 3)
> vec.1 # vector numerico
```

```
[1]  1 -1  0  2  0  3
```

```
> class(vec.1) # clase de todo tipo de numero
```

```
[1] "numeric"
```

```
> vec.2 <- c('a', 'b', 'c')
> vec.2
```

```
[1] "a" "b" "c"
```

```
> class(vec.2) # vector de caracteres
```

```
[1] "character"
```

La función `seq` crea un vector a partir de una sucesión de datos especificada por el usuario. La función tiene parámetros `from` y `to` para fijar el primer y ultimo valor de la sucesión así como el parámetro `by` que es el incremento que se tendrá entre cada componente. Si se necesitan incrementos unitarios, se puede usar el operador `:` entre el primer y último dígito.

```
> seq(from = 5, to = 20, by = 3)
```

```
[1]  5  8 11 14 17 20
```

```
> 7:11
```

```
[1]  7  8  9 10 11
```

Con las concatenaciones se pueden definir matrices. Utilizamos `?matrix` para obtener ayuda de la función y ver los parámetros por defecto de la función. Si los parámetros se meten en orden, no es necesario definir qué parametro es. Si queremos una sola fila, el valor por omisión, no necesitamos ponerlo pero hay que definir qué parametros estamos metiendo después de saltarnos uno. En el siguiente ejemplo, usaremos `nrow=1`, definiremos el número de columnas de la matriz `ncol` y `byrow=TRUE` significa que los valores se meterán fila por fila.

```
> mat.1 <- matrix(data = vec.1, ncol = 2, byrow = TRUE)
> mat.1
```

```
     [,1] [,2]
[1,]    1   -1
[2,]    0    2
[3,]    0    3
```

```
> mat.2 <- matrix(vec.1, ncol = 2) # byrow = FALSE
> mat.2
```

```
     [,1] [,2]
[1,]    1    2
[2,]   -1    0
[3,]    0    3
```

```
> class(mat.1) # clase matriz
```

```
[1] "matrix"
```

Las funciones deben tener ciertas clases de objetos y algunas funciones puden hacer cosas dependiendo de la clase de objeto que se le mete como argumento. Por ejemplo, la función `plot` hace gráficas diferentes cuando se aplica a un vector que cuando se aplica a una matriz. Con una matriz se grafica una columna contra la otra mientras que con un vector se grafica el valor de la componente en cada índice.

```
> par(mfrow=c(1,2))
> plot(mat.1)
> plot(vec.1)
```

Una manera de saber acerca de los objetos es usando la función `str`, que muestra la estructura de un objeto.

```
> str(mat.1)
```

```
 num [1:3, 1:2] 1 0 0 -1 2 3
```

```
> str(vec.1)
```

```
 num [1:6] 1 -1 0 2 0 3
```

Para acceder a ciertos elementos de un vector o de una matriz, se pueden especificar los índices que se desean con el operador `[]`. Si se desea obtener múltiples elementos seguidos se utiliza `:` para indicar el primer índice y el último. Si se desea quitar elementos se pone el signo negativo, `-`, seguido por el índice que se desea quitar.

```
> vec.1
```

```
[1]  1 -1  0  2  0  3
```

```
> vec.1[2]
```

```
[1] -1
```

```
> vec.1[2:5]
```

```
[1] -1  0  2  0
```

```
> vec.1[-2]
```

```
[1] 1 0 2 0 3
```

Para subindexar vectores o tablas, el procedimiento es análogo al de vectores, con la diferencia de que en este caso se separa el índice de la fila del índice de la columa por una coma. Si se necesita una columna completa, se deja el lugar del índice de la fila vacío.

```
> mat.1[1,2] # componente fila 1 y columna 2
```

```
[1] -1
```

```
> mat.1[,2] # segunda columna
```

```
[1] -1  2  3
```

Cuidado: La multiplicación usual de matrices se emplea con %*% y * es multiplicación componente a componente.

La clase de objetos mas común en R es la de Data Frame que son tablas de datos. La clase es utilizada para guardar bases de datos y sus funciones facilitan el manejo de las distintas variables. Para importar una base de datos a la consola, se necesita primero cambiar el directorio de trabajo al lugar donde se almacena el archivo. Se puede cargar datos de varias maneras. Archivos separados por comas/tabs de texto se pueden cargar con la función read.csv o read.table. Si la base de datos tiene títulos en el primer renglón, hay que especificarlo con header = TRUE.

```
> mis.datos ← read.table("CASAS.txt", header=TRUE)
```

La función head muestra las primeras seis líneas del archivo, que se puede también especificar cuántas línes mostrar. La función nrow muestra el número de filas (observaciones) y ncol el número de columnas (variables).

```
> head(mis.datos) # muestra 6 observaciones
```

```
  Price BDR  FLR FP RMS ST LOT  TAX BTH CON GAR CDN L1 L2
1    53   2  967  0   5  0  39  652 1.5   1 0.0   0  1  0
2    55   2  815  1   5  0  33 1000 1.0   1 2.0   1  1  0
3    56   3  900  0   5  1  35  897 1.5   1 1.0   0  1  0
4    58   3 1007  0   6  1  24  964 1.5   0 2.0   0  1  0
5    64   3 1100  1   7  0  50 1099 1.5   1 1.5   0  1  0
6    44   4  897  0   7  0  25  960 2.0   0 1.0   0  1  0
```

```
> nrow(mis.datos) # observaciones
```

```
[1] 26
```

```
> ncol(mis.datos) # variables
```

```
[1] 14
```

1.2.1. Subindexar

Por subindexar un objeto se entiende la obtención de un subconjunto de los datos. Las maneras de subindexar objetos son útiles para las necesidades que pueda tener un usuario. Ya se vieron las maneras mas sencillas con índices pero a continuación se presentan otros métodos mas elaborados.

```
> vec.1[c(1,4,5)] # indices que no son consecutivos
```

```
[1] 1 2 0
```

```
> a ← c(1,3,2) # tambien se puede en desorden
> vec.1[a]
```

```
[1]  1  0 -1
```

También se pueden filtrar objetos con valores lógicos.

```
> a.2 ← vec.1 > 0 # vector con TRUE donde cumple y FALSE donde no
> vec.1[a.2] # los elementos cuyo indice tiene TRUE en a.2
```

```
[1] 1 2 3
```

Para subindexar matrices y tablas, se puede hacer lo siguiente.

```
> mat.1[1,2] # un elemento
```

```
[1] -1
```

```
> mat.1[c(1,2), ] # dos renglones, todas las columans
```

```
     [,1] [,2]
[1,]    1   -1
[2,]    0    2
```

```
> mat.1[1:2, ] # lo mismo que el anterior
```

```
     [,1] [,2]
[1,]    1   -1
[2,]    0    2
```

```
> mat.1[1:2,1] # una columna, dos renglones
```

```
[1] 1 0
```

```
> mat.1[1:2,1, drop=FALSE] # para que no convierta a vector
```

```
     [,1]
[1,]    1
[2,]    0
```

```
> mis.datos[1:5, c(2,5,6)]
```

```
  BDR RMS ST
1   2   5  0
2   2   5  0
3   3   5  1
4   3   6  1
5   3   7  0
```

```
> vec.3 ← mis.datos[ , 2] # esto es un vector, la 2a columna
> vec.3[1:20]
```

```
 [1] 2 2 3 3 3 4 5 3 4 4 8 2 3 4 4 2 3 4 2 4
```

También se pueden utilizar valores lógicos para subindexar matrices y tablas. De los datos `mis.datos` se muestra como subindexar los renglores que tengan la medición de la variable `ST` mayor a cero.

```
> indice ← mis.datos[ , 6] > 0 # ST > 0
> sub.datos ← mis.datos[indice, ] # nuevo objeto
> nrow(sub.datos)
```

```
[1] 7
```

Las tablas de clase `Data Frame` son las más usadas, por lo que tiene ciertas ventajas. Por ejemplo, se pueden extraer las columnas que tengan cierto nombre con el signo de $ o extraer observaciones que cumplan ciertas caracterśticas con funciones como `subset`. Se observan las primeras observaciones con la función `head`.

```
> impuesto ← mis.datos$TAX # columna con titulo TAX
> head(impuesto)
```

```
[1]  652 1000  897  964 1099  960
```

```
> aux ← subset(mis.datos, Price > 60 & ST == 1)
> aux
```

```
   Price BDR  FLR FP RMS ST LOT  TAX BTH CON GAR CDN L1 L2
9     72   4 1290  0   8  1  33  800 1.5   1 1.5   0  1  0
11    85   8 2240  1  12  1  50 1200 3.0   0 2.0   0  1  0
17    62   3 1126  0   7  1  30  734 2.0   1 0.0   1  0  1
26    65   3 1023  0   7  1  30  900 2.0   1 1.0   0  1  0
```

1.3. Funciones en R

Cada paquete en `R` tiene funciones propias. Una vez instalado el paquete y cargada la librerá, ya vimos que se puede obtener ayuda con cualquier función tecleando un signo de interrogación seguido por el nombre de la función. Hay que tener cuidado con la clase de objeto que acepta cada función como parámetros.

```
> sum(vec.1) # suma
```

```
[1] 5
```

```
> max(vec.1) # maximo
```

```
[1] 3
```

```
> mean(vec.1) # promedio
```

```
[1] 0.8333333333333334
```

También ya vimos que no es necesario poner parámetros que las funciones tienen por defecto. Por ejemplo, de la ayuda de sort, por defecto, pone `decreasing = TRUE`.

```
> sort(vec.1) # ordena
```

```
[1] -1  0  0  1  2  3
```

```
> sort(vec.1, decreasing = TRUE) # ordena de mayor a menor
```

```
[1]  3  2  1  0  0 -1
```

Como el código de R es libre, se puede acceder al código de cualquier función tecleando el nombre de la función en la consola (sin parámetros ni paréntesis).

```
> sort
```

```
function (x, decreasing = FALSE, ...)
{
    if (!is.logical(decreasing) || length(decreasing) != 1L)
        stop("'decreasing' must be a length-1 logical vector.\nDid you intend to
    set 'partial'?")
    UseMethod("sort")
}
<bytecode: 0x7fe1440d0628>
<environment: namespace:base>
```

Las funciones en R tienen distintos autores dependiendo del paquete que la contiene. Para crear funciones propias, se utliza la función `function`. En el siguiente ejemplo creamos una función que imprime el mayor de los números que se meten de parámetros. Si el usuario solo da un valor, por defecto asignaremos 1 al otro número.

```
> mayor ← function(x, y=1){
+ z ← max(x,y)
+ return(z)
+ }
> mayor(5,3)
```

```
[1] 5
```

```
> mayor(-3)
```

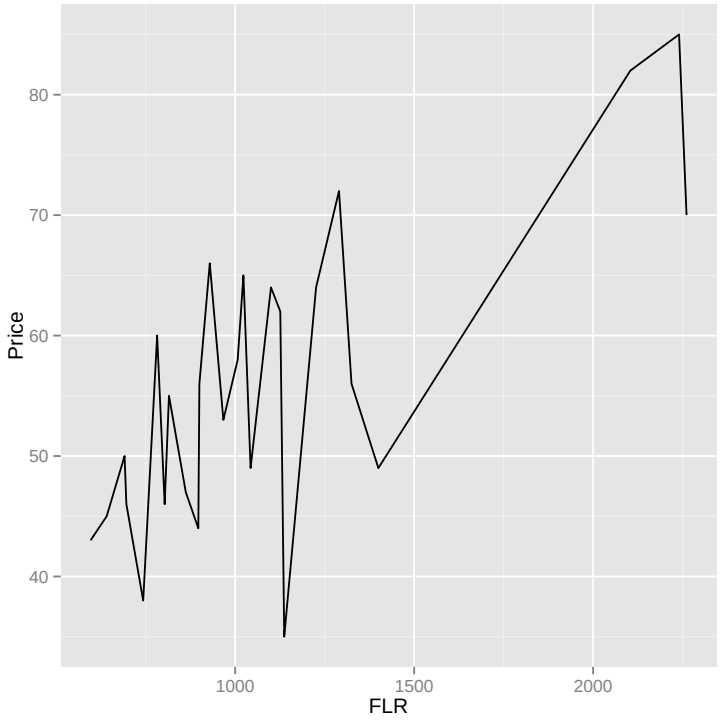
```
[1] 1
```

1.4. Gráficas

En R, el paquete más famoso para graficar es ggplot2. Este paquete fue creado por Hadley Wickham y tiene extensas capacidades para graficar datos univariados y multivariados numéricos y categóricos. Debido a que puede ser un poco complicado aprenderlo, lo iremos usando de poco a poco en este documento. Para otras aplicaciones, ver el anexo del paquete.

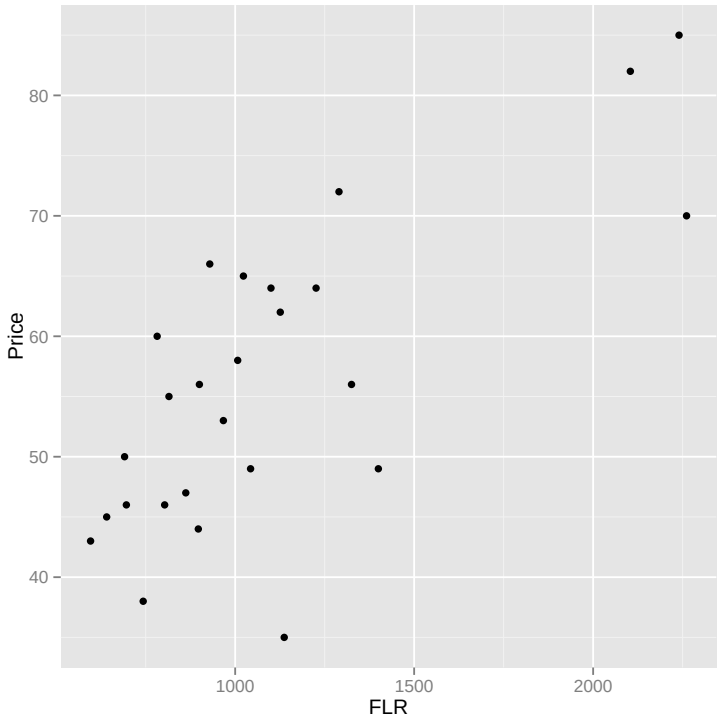
Para generar gráficas, hay que tener distintos comandos separados por un signo $+$. Primero es necesario tener la función `ggplot()` que inicializa un objeto ggplot y toma como argumentos la base de datos así como un objeto aes que toma las variables que se van a usar. La siguiente gráfica representa cómo crece el precio de una casa con base en la variable FLR.

```
> library(ggplot2)
> ggplot(mis.datos, aes(x=FLR,Price)) + geom_line()
```



Para que ggplot sepa que queremos graficar los valores en una línea ponemos `+ geom_line()` y si queremos puntos ponemos `+ geom_point()`.

```
> ggplot(mis.datos, aes(x=FLR,Price)) + geom_point()
```



1.5. Resumen

Las operaciones en R son muy intuitivas y se usa <- para asignar variables.

Para concatenar objetos, se puede usar `c`, `:` o `seq`. Concatenando enteros o valores lógicos, se puede subindexar otros objetos como matrices.

La clase de objetos más común en R es la de Data Frame, donde se guarda una tabla de datos con columnas como variables y filas observaciones. Las operaciones en R se facilitan para esta clase de objeto.

En R, las funciones:

- Reciben parámetros por nombre y por posición.
- No siempre es necesario pasar TODOS los parámetros ya que típicamente existen valores por defecto que hacen más fácil la rutina de programar.
- Pueden hacer distintas cosas dependiendo de la clase del parámetro.

Capítulo 2

Variables Aleatorias Discretas

Muchas veces nos interesa, más allá de saber resultados de algún evento, conocer el comportamiento general de estos posibles resultados. Una variable aleatoria es esta función de interés que asigna eventos a números reales. En este capítulo vamos a mencionar brevemente la teoría de variables aleatorias discretas intuitivamente y con ejemplos.

Para hacer un análisis de una variable aleatoria discreta, se necesita conocer ciertas funciones básicas en R. Se ilustran algunas funciones en el siguiente ejemplo.

Ejemplo. Se definen las probabilidades de cuatro variables aleatorias discretas, donde $x = (x_1, x_2, ...)$ representa el valor de la variable y f su probabilidad $P(X = x_i)$.

```
> x ← c(0,1,2,3)
> f ← c(1/8, 3/8, 3/8, 1/8)
```

La esperanza matemática, o media μ, es el valor esperado (a largo plazo) de un evento. Para obtener la media, se necesita tomar cada uno de los valores de x, multiplicarlos por su probabilidad f y sumarlos.

```
> mu ← sum(x * f)
> mu
```

```
[1] 1.5
```

La varianza σ^2 es una medida de dispersión de la media de la variable aleatoria. Para poder obtener la varianza, se debe tomar el valor de x, restarle la media μ, elevarlo al cuadrado y multiplicar por la probabilidad f.

```
> sigma2 ← sum((x-mu)^ 2 * f)
> sigma2
```

```
[1] 0.75
```

Ahora se calcula la desviación estándar, sacando la raíz del valor anterior.

```
> sigma ← sqrt(sigma2)
> sigma
```

```
[1] 0.8660254037844386
```

Ahora vamos a definir F la función de probabilidad acumulada como la probabilidad de que X tome un valor menor o igual a x. Para calcular F, se necesita sumar la probabilidad del evento $X = x$ con las probabilidades de los eventos anteriores a x.

```
> F ← cumsum(f)
> F
```

```
[1] 0.125 0.500 0.875 1.000
```

El paquete `distrEx` nos facilita hacer el análisis anterior. El soporte de F es los posibles valores que puede tomar X y se tiene que especificar en la función `DiscreteDistribution` para definir la variable aleatoria X.

```
> library(distrEx)
> X ← DiscreteDistribution(supp = 0:3, prob = c(1,3,3,1)/8)
> E(X); var(X); sd(X)
```

```
[1] 1.5
```

```
[1] 0.75
```

```
[1] 0.8660254037844386
```

Aquí notamos que primero cargamos el paquete, luego definimos el objeto X y finalmente utilizamos funciones que están dentro del paquete. Hay paquetes que pueden contener distintas funciones con el mismo nombre (como por ejemplo la función `var`), pero `R` analiza el tipo de objeto que le metemos para saber qué función usar.

2.1. Distribución Uniforme Discreta

Una variable aleatoria X con distribución uniforme tiene función de masa de probabilidad

$$f_X(x) = \frac{1}{m}$$

para toda $x = x_1, x_2, x_3, \ldots, x_m$ la cual tiene media

$$\mu = \frac{1}{m} \sum_{i=1}^{m} x_i$$

y varianza

$$\sigma^2 = \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu)^2.$$

Una función práctica para manejar variables uniformes discretas es `sample`, que toma como argumentos la m y la cantidad de repeticiones del experimento. Entre mas veces repitamos el experimento, mas nos acercamos a los valores teóricos.

Ejemplo. Se tira un dado justo 3,000 veces, calculemos el número de veces que sale cada cara. Observamos que cada una de las 6 caras sale aproximadamente el mismo número de veces. Vamos a usar un histograma para visualizar el resultado. Un histograma es una gráfica de barras que representa las frecuencias de distintos valores de X.

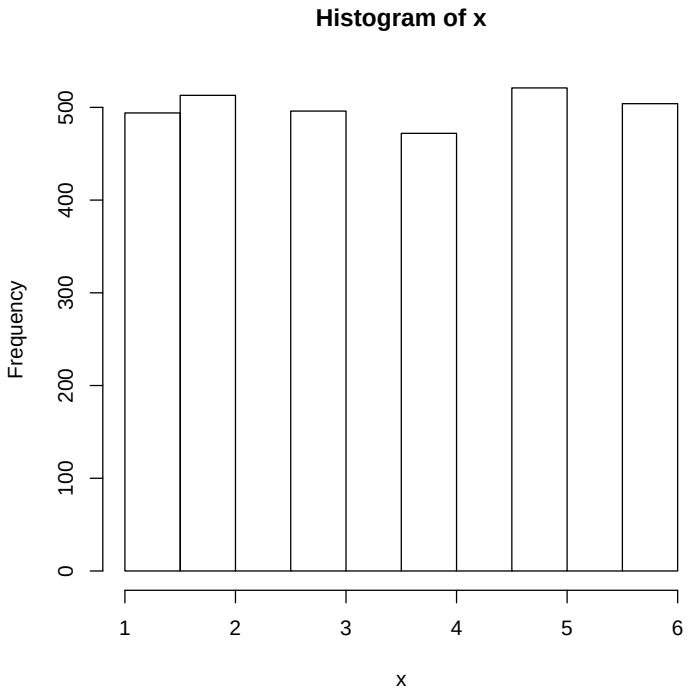
```
> x ← sample(6, size = 3000, replace = TRUE) # m = 6, 3000 veces
> mean(x) # mu = (6 + 1)/2 = 3.5
```

```
[1] 3.508333333333333
```

```
> var(x) # sigma2 = (36 - 1)/12 = 2.92
```

```
[1] 2.936242636434367
```

```
> hist(x)
```



Ejemplo. Queremos elegir 270 números aleatorios entre 30 y 70. Si a la función `sample` le damos un vector en su primer argumento, se toma eso como el soporte.

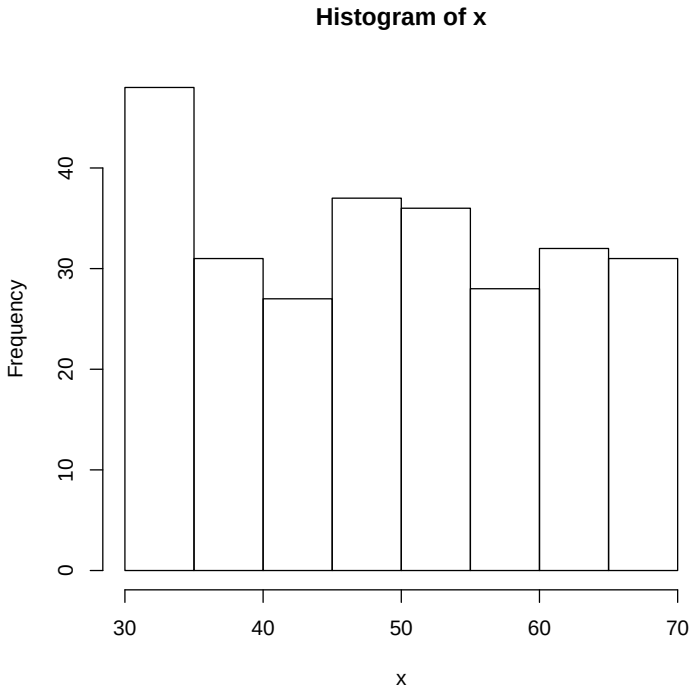
```
> x ← sample(30:70, size = 270, replace = TRUE)
> mean(x) # mu = (70 - 30 + 1)/2 + 30 = 50.5
```

```
[1] 49.42222222222222
```

```
> var(x) # sigma2 = ((70 - 30)^2 + 1)/12 = 133.42
```

```
[1] 144.9586121437422
```

```
> hist(x)
```



Los métodos que R utiliza para hacer los cálculos son numéricos y por lo tanto puede ser que no sean muy precisos, hay que tener cuidado aunque por lo general sí son confiables.

2.2. Distribución Binomial

Si tenemos una prueba con solo resultados de éxito o fracaso, decimos que la prueba tiene distribución Bernoulli. Ahora queremos realizar n pruebas Bernoulli independientes, cada una teniendo probabilidad p de éxito y probabilidad $1-p$ de fracaso. Si X representa el número de éxitos en la prueba, se dice que X tiene distribución Binomial.

La función de masa de probabilidad de una variable aleatoria Binomial es

$$f_X(x) = \binom{n}{x} p^x (1-p)^{n-x}$$

con $x = 0, 1, 2, \ldots, n$ que tiene media

$$\mu = np$$

y varianza

$$\sigma^2 = np(1-p)$$

Podemos utilizar la función `pbinom` y derivados para simular variables aleatorias con distribución Binomial. Esta función está incluida en uno de los paquetes que se carga automaticamente[1] cuando iniciamos la sesión de R.

```
> dbinom(2, size = 4, prob = 1/2) # funcion de densidad P(X = 2)
```

```
[1] 0.375
```

```
> pbinom(2, size = 4, prob = 1/2) # funcion de distribucion P(X ≤ 2)
```

```
[1] 0.6875
```

```
> pbinom(9, size = 12, prob = 1/6) -
+ pbinom(6, size = 12, prob = 1/6) # P(6 ≤ X < 9)
```

```
[1] 0.00129175754208255
```

```
> diff(pbinom(c(6,9), size = 12, prob = 1/6)) # lo mismo
```

```
[1] 0.00129175754208255
```

El paquete `distr` permite definir objetos en la memoria como variable aleatoria.

```
> library(distr)
> X ← Binom(size = 3, prob = 1/2)
> X
```

```
Distribution Object of Class: Binom
 size: 3
 prob: 0.5
```

En este paquete, la función análoga a `dbinom` es `d(X)` y la función análoga a `pbin` es `p(X)`. Ahora comparamos las funciones de este paquete con los resultados anteriores.

```
> d(X)(1) # funcion de densidad P(X = 1)
```

```
[1] 0.3750000000000001
```

```
> p(X)(2) # funcion de distribucion P(X < 2)
```

```
[1] 0.875
```

Para simular k variables aleatorias binomiales utilizamos `rbinom`.

```
> rbinom(10, size = 4, prob = 1/2)
```

[1] En las opciones de R, se puede modificar qué paquetes cargar cuando se inicia la sesión.

```
[1] 1 2 4 3 4 2 4 2 3 1
```

Las diferentes probabilidades de la distribución Binomial se pueden generar recursivamente. La siguiente es una función que genera sus probabilidades en distintos valores.

```
> bin ← function(n,p){
+ return(binr(n,p,n))
+ }
> binr ← function(n,p,k){
+ if(k≥ 1){
+ # La entrada k+1 representa la probabilidad en k.
+ # Se concatena la respuesta conforme sale,
+ # para obtener todas la probabilidades
+ B ← c(binr(n, p, k-1), p/(1-p) *
+ (n-(k-1))/((k-1)+1) * binr(n, p, k-1)[k])
+ }
+ else{
+ B ← (1 - p)^ n # Caso particular para la probabilidad en cero
+ }
+ return(B)
+ }
```

Ejemplo. Sea X una variable aleatoria Binomial con parámetros $n = 6$ y $p = 0.4$.

```
> n = 6
> p = 0.4
> bin(n,p) # llamamos a la funcion definida
```

```
[1] 0.046655999999999989 0.186623999999999957 0.311039999999999983
[4] 0.276480000000000004 0.138240000000000002 0.036864000000000008
[7] 0.004096000000000002
```

Se muestran los valores de $P(X = 0)$, $P(X = 1)$, $P(X = 2)$, $P(X = 3)$, $P(X = 4)$, $P(X = 5)$ y $P(X = 6)$ respectivamente.

2.3. Función Generadora de Momentos

La función generadora de momentos de una variable aleatoria discreta X está definida[2] por la fórmula

$$M_X(t) = E(e^{tX}) = \sum_{x \in S} e^{tx} f_X(x)$$

en particular, tenemos

$$M_X(0) = E(X).$$

[2]No hay garantía de que exista pues la suma puede no convergir, a diferencia de la función característica que determina totalmente a X

Ejemplo. Para ilustrar el comportamiento de la distribución Binomial, utilizamos una prueba con $n = 10$ pruebas independientes, teniendo una probabilidad de $p = 0.3$ y repetimos para 10,000 casos. Para obtener los momentos, existe un paquete llamado `moments`.

```
> library(moments) # paqueteria para obtener momentos de las variables
> vars <- rbinom(10000, 10, .3) # parametros n, repeticiones y p
    respectivamente
> hist(vars)
> mean(vars) # el valor esperado es n * p
```

```
[1] 3.0148
```

```
> var(vars) # la varianza es np(1-p)
```

```
[1] 2.125793539353935
```

```
> moment(vars,1) # el primer momento es la esperanza E[X]
```

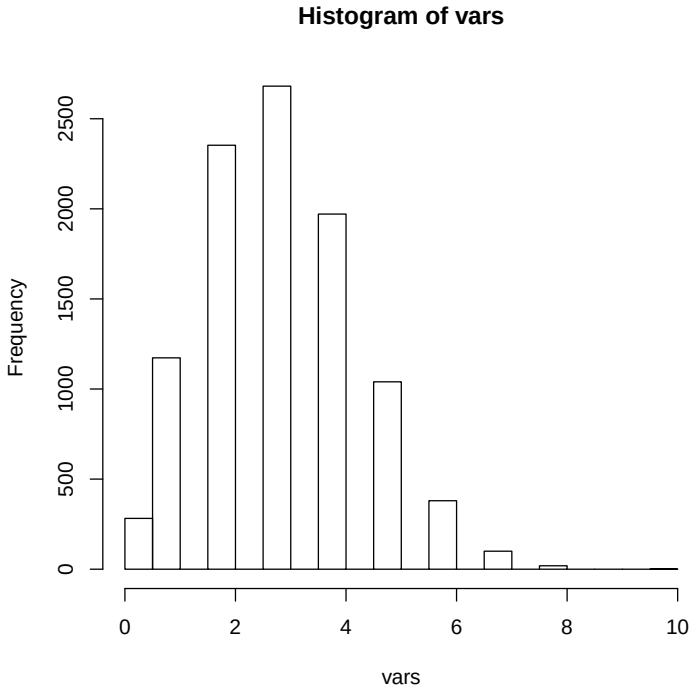
```
[1] 3.0148
```

```
> moment(vars,2) # el segundo momento es E[X^2] = E[X]^2 + V[X]
```

```
[1] 11.2146
```

```
> mean(vars)^2 + var(vars) # se puede ver la equivalencia
```

```
[1] 11.21481257935394
```



2.4. Distribución Poisson

Una variable aleatoria X que toma valores 0,1,2,... es Poisson parámetro λ si para alguna $\lambda > 0$ tenemos

$$f_X(x) = e^{-\lambda}\frac{\lambda^x}{x!}$$

lo que tiene una media y varianza

$$\mu = \sigma^2 = \lambda$$

Para las variables aleatorias comunes, tenemos las funciones definidas que hemos estado usando pero también podemos crear funciones propias. Creando una función para la distribución Poisson, tenemos el siguiente algoritmo para calcular $P(X = n)$ con parámetro l.

```
> poi ← function(n,l){ #Se necesitan n y l
+ return(poir(l,n))
+ }
> poir ← function(l,k){
+ if(k≥ 1){
+ P ← c(poir(l,k-1), l/k * poir(l,k-1)[k])
+ }
+ else{
+ P ← exp(1)^ (-l) # Caso para P(X=0)
+ }
+ return(P)
+ }
```

Por ejemplo, si queremos calcular $P(X \leq 2)$, teniendo una media en 3.2, utilizamos el algoritmo anterior para imprimir $P(X = 0)$, $P(X = 1)$, y $P(X = 2)$. Si sumamos estas cantidades, obtenemos el resultado deseado.

```
> l ← 3.2; n ← 2; poi(n,l); sum(poi(n,l))
```

```
[1] 0.04076220397836622 0.13043905273077191 0.20870248436923508
```

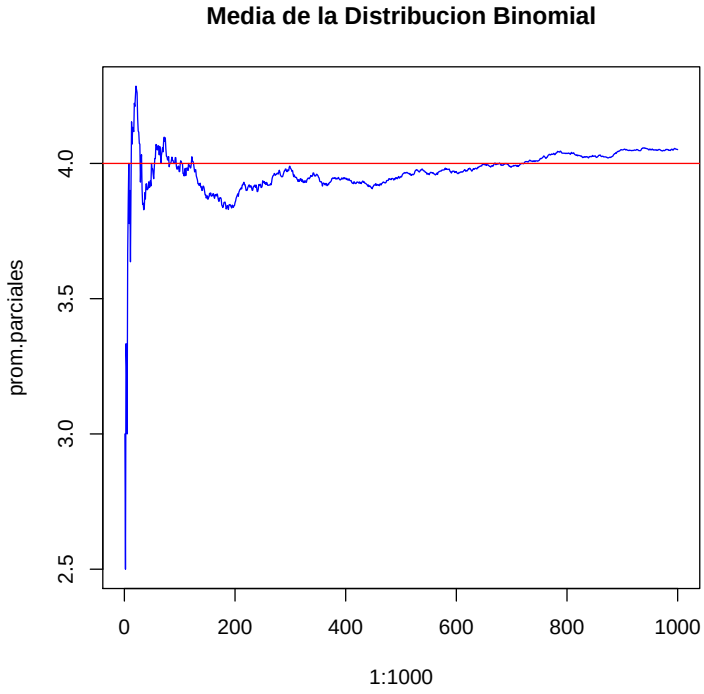
```
[1] 0.3799037410783732
```

2.4.1. Modelando Eventos Raros

Supongamos que tenemos 1,000 variables de una distribución dada. En el primer paso destapamos la primera variable y graficamos su valor. En el segundo paso destapamos la segunda y graficamos el promedio de los dos valores. Continuando de esta manera, en en el n-ésimo paso destapamos la n-ésima variable y graficamos el promeido de los n valores. Con esto, obtenemos la siguiente gráfica y su correspondiente código.

```
> n = 10
> prob = 0.4
> X ← rbinom(1000, n, prob)
```

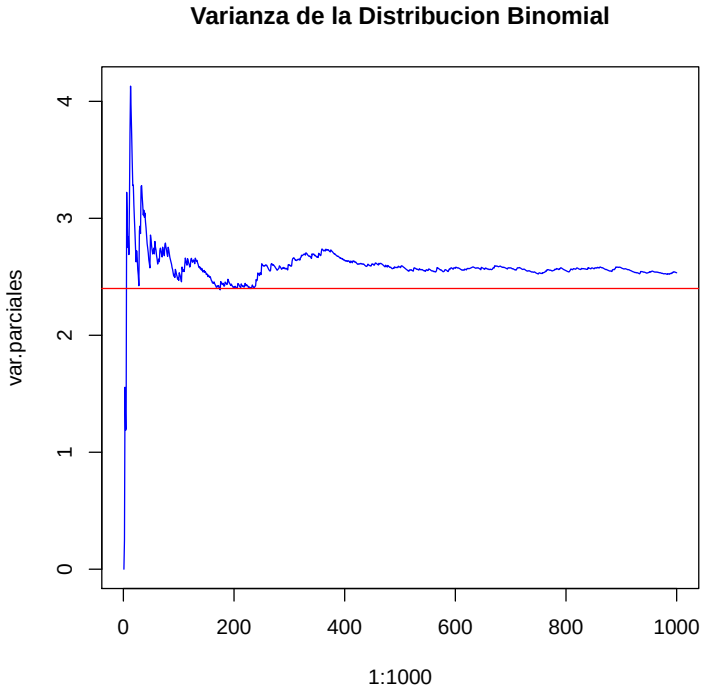
```
> prom.parciales ← cumsum(X)/1:1000
> plot(1:1000, prom.parciales, type='l', col='blue')
> title('Media de la Distribucion Binomial')
> abline(h=n*prob, col='red')
```



En el mismo experimento, podemos en vez de promediar los valores, promediar los cuadrados de sus valores menos el promedio parcial (i.e., la varianza muestral), y obtenemos lo siguiente.

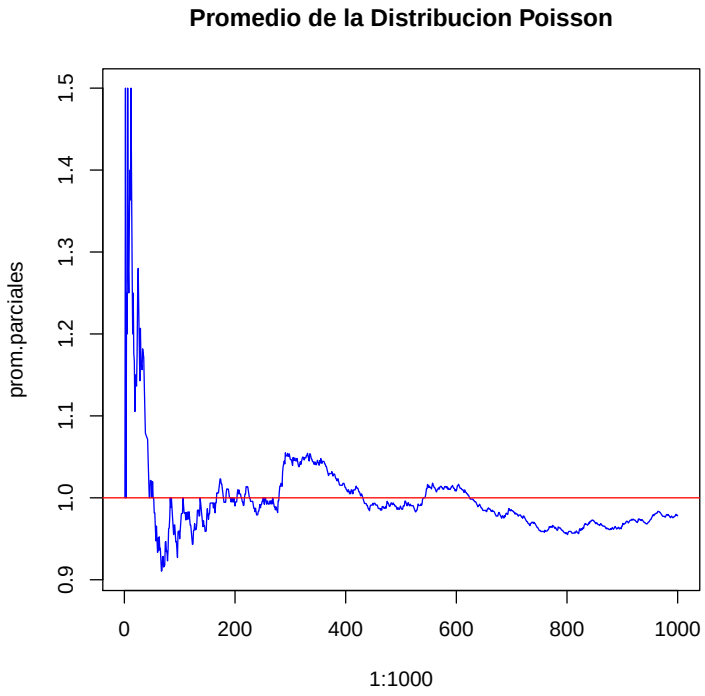
```
> var.parciales ← vector("numeric",1000)
> for (i in 1:1000){
+ var.parciales[i] ← sum((X[1:i] - prom.parciales[i])^ 2)/i
+ }
```

```
> plot(1:1000, var.parciales, type='l', col='blue')
> title('Varianza de la Distribucion Binomial')
> abline(h=n*prob*(1 - prob), col='red')
```



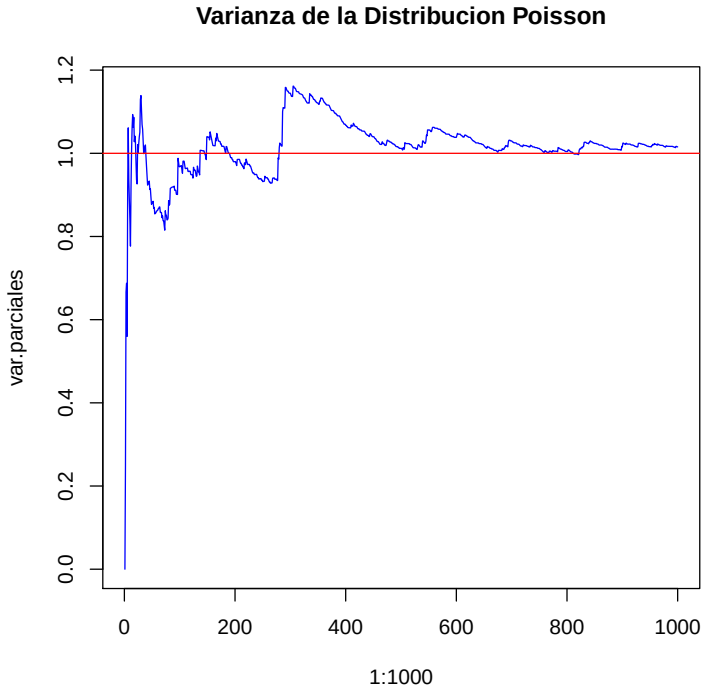

```
> # POISSON
> lambda ← 1
> X ← rpois(1000,lambda)
```

```
> prom.parciales ← cumsum(X)/1:1000
> plot(1:1000, prom.parciales, type='l',col='blue')
> title('Promedio de la Distribucion Poisson')
> abline(h=lambda, col='red')
```




```
> var.parciales ← vector("numeric",1000)
> for (i in 1:1000)
+ var.parciales[i] ← sum((X[1:i] - prom.parciales[i])^ 2)/i
```

```
> plot(1:1000, var.parciales, type='l', col='blue')
> title('Varianza de la Distribucion Poisson')
> abline(h=lambda, col='red')
```



Ejemplo en la vida práctica de una Poisson (La ley de los eventos raros)

Supongamos que la aseguradora ABC recibe datos de la frecuencia de accidentes automovilísticos de alto impacto. Supongamos también que la población del pueblo PUEBLANDIA es de 250 mil habitantes con automóvil y que se sabe que en promedio cada año hay 75 accidentes de esta naturaleza.

El siguiente código en R simula si la persona i con $i = 1, \ldots, 250000$ tendrá un accidente de alto impacto en el siguiente año. X es un vector aleatorio de 250 mil coordenadas, si la i-ésima coordenada es 1, entonces la i-ésima persona tendrá un accidente.

```
> prob ← 75/250000 # estimacion de la probabilidad
> X ← sample(c(0,1), 250000, prob=c(1-prob,prob), replace=TRUE)
```

Ahora la aseguradora ABC tiene contrato con 10,000 personas pero desconoce de qué tipo son. Repitamos el siguiente experimento muchas veces, seleccionemos al azar 10,000 personas y veamos cuántas personas con accidentes habrá.

```
> accidentes ← vector("numeric", 1000)
> for (j in 1:1000) {
+ aseguradora ← sample(X,10000)
+ accidentes[j] ← sum(aseguradora) # total de accidentes
+ }
```

```
> Y ← rpois(1000, lambda = 10000*prob)
> table(accidentes)
```

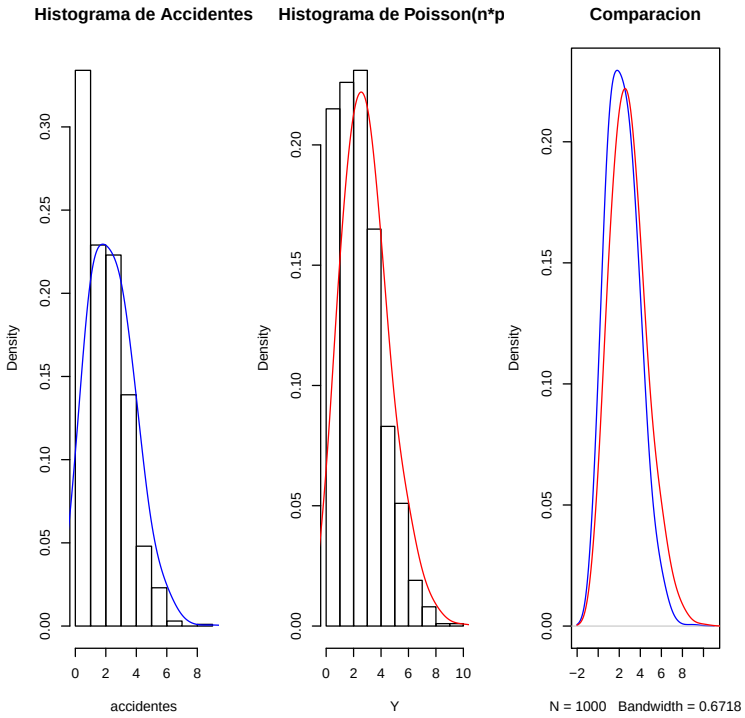
```
accidentes
  0   1   2   3   4   5   6   7   9
```

```
 93 241 229 223 139  48  23   3   1
```

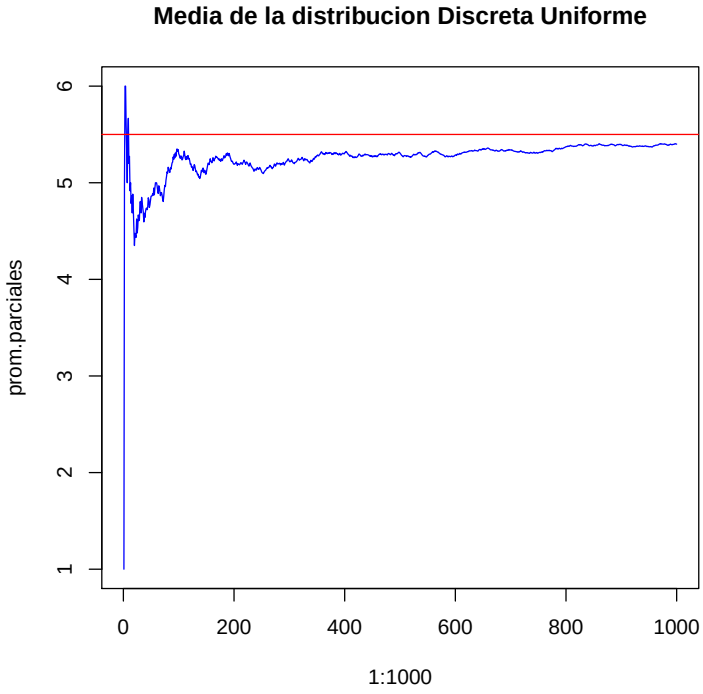
```
> table(Y)
```

```
Y
  0   1   2   3   4   5   6   7   8   9  10
 54 161 226 231 165  83  51  19   8   1   1
```

```
> par(mfrow = c(1,3)) # comando para mostrar las 3 graficas en 1 linea
> hist(accidentes, prob=TRUE, main='Histograma de Accidentes')
> lines(density(accidentes, adjust=2), col='blue')
> hist(Y, prob=TRUE, main='Histograma de Poisson(n*p)')
> lines(density(Y, adjust=2), col='red')
> plot(density(accidentes, adjust=2), type='l', col='blue', main='
    Comparacion')
> lines(density(Y, adjust=2), col='red')
```

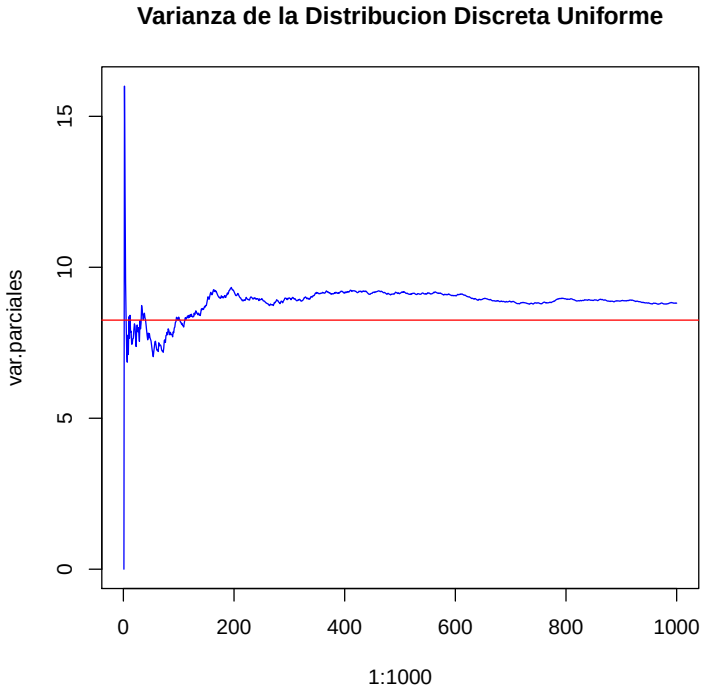



```
> # DISCRETA UNIFORME
> n ← 10
> X ← sample(n,1000,replace=TRUE)
> prom.parciales ← cumsum(X)/1:1000
> plot(1:1000,prom.parciales,type='l', col='blue')
> title('Media de la distribucion Discreta Uniforme')
> abline(h=(n+1)/2, col='red')
```




```
> var.parciales ← vector("numeric",1000)
> for (i in 1:1000)
+ var.parciales[i] ← sum((X[1:i]-prom.parciales[i])^ 2)/i
```

```
> plot(1:1000, var.parciales,type='l',col='blue')
> title('Varianza de la Distribucion Discreta Uniforme')
> abline(h=(n^ 2-1)/12, col='red')
```



2.5. Distribución geométrica

Ahora queremos saber cuántas pruebas independientes se necesitan para obtener un primer éxito de un experimento, con una probabilidad p de éxito con $0 < p < 1$. Sea X el número de fallos necesarios antes de obtener ese éxito, entonces X es una variable aleatoria Geométrica con

$$f_X(x) = (1-p)^x p$$

para $x = 0, 1, 2, \ldots$ con media

$$\mu = \frac{1-p}{p}$$

y varianza

$$\sigma^2 = \frac{1-p}{p^2}.$$

Ejemplo. Si sabemos que un futbolista ha metido un gol para el 81.2 % de sus tiros, ¿cuál es la probabilidad de que el futbolista falle al menos 5 tiros antes de anotar su primer gol? Sea X el número de tiros fallados antes de su primer gol. Se necesita $P(X \geq 5)$ que, como la distribución es discreta, equivale a calcular el valor de $P(X > 4)$.

```
> pgeom(4, prob = 0.812, lower.tail = FALSE) # P(X > x)
```

```
[1] 0.0002348492871679997
```

```
> 1 - pgeom(4, prob = 0.812) # Default: lower.tail = TRUE, P(X ≤ 4)
```

```
[1] 0.000234849287167993
```

2.6. Distribución Binomial Negativa

Podemos generalizar el caso anterior con ahora la probabilidad de obtener un número específico de éxitos. Se hacen pruebas independientes, cada una teniendo probabilidad p con $0 < p < 1$ de tener éxito, hasta que se obtienen r éxitos acumulados. Si X representa la cantidad de fracasos antes de obtener esos r éxitos, entonces

$$f_X(x) = \binom{r+x-1}{r-1} p^r (1-p)^x$$

con media

$$\mu = \frac{r}{p}$$

y varianza

$$\sigma^2 = \frac{r(1-p)}{p^2}$$

Ejemplo. Tiramos una moneda y sea X el número de águilas hasta obtener siete soles, calcular el valor de $P(X = 5)$ la probabilidad de que obtengamos 5 fracasos antes de obtener 7 éxitos.

```
> dnbinom(5, size = 7, prob = 0.5)
```

```
[1] 0.11279296875
```

2.7. Distribución Hipergeométrica

La distribución hipergeométrica es usada cuando tenemos, por ejemplo, una urna con pelotas blancas y negras. Si hay m pelotas blancas y n negras y se sacan k pelotas de la urna sin remplazo, podemos definir X como la cantidad de pelotas blancas que sacaremos de la urna. Con esto, la distribución de X está dada por

$$f_X(x) = \frac{\binom{m}{x}\binom{n}{k-x}}{\binom{m+n}{k}}$$

para $x = 0, 1, \ldots, k$ con media

$$\mu = kp$$

y varianza

$$\sigma^2 = \frac{n+m-k}{n+m-1} kp(1-p).$$

Ejemplo. Un coprador de componentes eléctricos tiene indicación de revisar los paquetes de 10 componentes cada uno. De cada paquete, selecciona 3 componentes al azar y solo lo acepta si ninguno de los 3 está defectuoso. Si el 30 % de los paquetes tiene 4 componentes defectuosos y el 70 % tiene solo 1 defectuoso, ¿qué proporción de paquetes aceptará el comprador?

En este ejemplo, necesitamos que el comprador saque 0 componentes defectuosas, que diremos que son nuestras bolas blancas. De 10 componentes en cada paquete, nuestra urna, una parte tiene 4 defectuosos (blancas) y 10-4=6 no defectuosos (negras) mientras que la otra parte tiene 1 defectuoso y 9 no defectuosos. Vamos a sumar las probabilidades de que no salga ningún defectuoso si sacamos 3 componentes (bolas) al azar.

```
> 3/10*dhyper(0, 4, 10-4, 3) + 7/10*dhyper(0, 1, 10-1, 3)
```

```
[1] 0.5399999999999999
```

2.8. Resumen

En la siguiente tabla podemos ver un resumen de las variables aleatorias discretas. En la segunda columna ponemos la función en `R` de la densidad de cada una de las distribuciones. Excepto por el caso de la distribución uniforme, para la función de distribución cambiamos la primera letra por una `p`, para los cuantiles por una `q` y para la generarción aleatoria por una `r`.

Distribución	R	$f_X(x)$	μ	σ^2
Uniforme	`sample`	$\frac{1}{m}$	$\frac{1}{m}\sum_{i=1}^{m} x_i$	$\frac{1}{m}\sum_{i=1}^{m}(x_i-\mu)^2$
Binomial	`dbinom`	$\binom{n}{x}p^x(1-p)^{n-x}$	np	$np(1-p)$
Poisson	`dpois`	$e^{-\lambda}\frac{\lambda^x}{x!}$	λ	λ
Geométrica	`dgeom`	$(1-p)^x p$	$\frac{1-p}{p}$	$\frac{1-p}{p^2}$
Binomial Negativa	`dnbinom`	$\binom{r+x-1}{r-1}p^r\,(1-p)^x$	$\frac{r}{p}$	$\frac{r(1-p)}{p^2}$
Hipergeométrica	`dhyper`	$\frac{\binom{m}{x}\binom{n}{k-x}}{\binom{m+n}{k}}$	kp	$\frac{n+m-k}{n+m-1}kp\,(1-p)$

Capítulo 3

Variables Aleatorias Continuas

En el capitulo anterior nos interesó estudiar variables aleatorias discretas, cuyos resultados toman un número finito o infinito numerable de valores. En este capítulo se usarán variables aleatorias que toman como valores un conjunto no discreto, sus valores no son aislados. Esto quiere decir que ahora tomaremos intervalos de valores para nuestro dominio.

Toda variable aleatoria continua X tiene una función de densidad de probabilidad denotada f_X asociada. La función f_X cumple las siguientes tres propiedades:

- $f_X(x) > 0$ para todo $x \in S_X$
- $\int_{x \in S_X} f_X(x)\, dx = 1$
- $\mathbb{P}(X \in A) = \int_{x \in A} f_X(x)\, dx$, para un evento $A \subset S_X$

La esperanza de una variable aleatoria continua se puede calcular con

$$\mu = \mathbb{E}(X) = \int_{x \in S} x f_X(x)\, dx$$

y la varianza con

$$\sigma^2 = \mathbb{E}(X - \mu)^2 = \int_{x \in S} (x - \mu)^2 f_X(x)\, dx.$$

Y para la función generadora de momentos ahora usaremos la definición

$$M_X(t) = \mathbb{E}\left(e^{tX}\right) = \int_{-\infty}^{\infty} e^{tx} f_X(x)\, dx$$

Ejemplo. Sea X una variable aleatoria con función de densidad $f(x) = 3x^2$ con soporte $0 < x < 1$. Encontrar $\mathbb{P}(0.14 \leq X \leq 0.71)$.

Solución. Analíticamente, sabemos que

$$\begin{aligned} \mathbb{P}(0.14 \leq X \leq 0.71) &= \int_{0.14}^{0.71} 3x^2 dx \\ &= x^3 \big|_{0.14}^{0.71} \\ &\approx 0.355167 \end{aligned}$$

.

Ahora usando R, se define una función que calcula la función de densidad de probabilidad. después se utiliza la función `integrate` con parámetros un objeto definido como función, límite inferior a integrar y límite superior para sacar la probabilidad deseada.

```
> # funcion de densidad de probabilidad
> f ← function(x) 3 * x^2
> # se integra para sacar la probabilidad deseada
> integrate(f, lower = 0.14, upper = 0.71)
```

```
0.3551669999999999 with absolute error < 3.9e-15
```

Lo anterior también se puede obtener usando el paquete `distr`. La función de densidad se define igual al caso anterior. El paquete permite generar un objeto de clase `AbscontDistribution` que reconoce a la variable como una variable aleatoria continua y permite funcionalidades extensas. Los parámetros que se utilizan son `d` que pide la función de densidad, `low1` es el límite inferior y `up1` es el límite superior. El objeto de la clase tiene varios métodos integrados, incluyendo el método `p` que calcula la distribución acumulada de la variable. La sintaxis para obtener la solución al problema es la siguiente.

```
> library(distr)
> f ← function(x) 3 * x^2
> X ← AbscontDistribution(d = f, low1 = 0, up1 = 1)
> p(X)(0.71) - p(X)(0.14)
```

```
[1] 0.3551670172065496
```

Usando la misma variable aleatoria definida previamente, la media y la varianza se obtienen sencillamente. Se calcula la media con

$$\begin{aligned}\mu &= \int_{-\infty}^{\infty} x \cdot 3x^2 dx \\ &= \frac{3}{4}\, x^4\Big|_0^1 \\ &= \frac{3}{4}\end{aligned}$$

y para calcular la varianza se necesita

$$\begin{aligned}\mathbb{E}\left(X^2\right) &= \int_{-\infty}^{\infty} x^2 3x^2 dx \\ &= \frac{3}{5}x^5\Big|_0^1 \\ &= \frac{3}{5}\end{aligned}$$

con lo que ahora

$$\sigma^2 = \frac{3}{5} - \left(\frac{3}{4}\right)^2 = \frac{3}{80}$$

.

El paquete `distrEx` es una extensión del paquete `distr`. Tiene funcionalidades adicionales para el objeto ya definido como variable aleatoria continua. Dos funciones de interés son la de la esperanza `E` y la de la varianza `Var` que se implementan a continuación.

```
> library(distrEx)
> E(X); var(X); 3/80 # media, varianza y varianza teorica
```

```
[1] 0.7496336769758919
```

```
[1] 0.03768305003062067
```

```
[1] 0.0375
```

3.1. Distribución Uniforme

Se dice que una variable aleatoria X tiene distribución uniforme en el intervalo (a, b) si tiene función de densidad de probabilidad

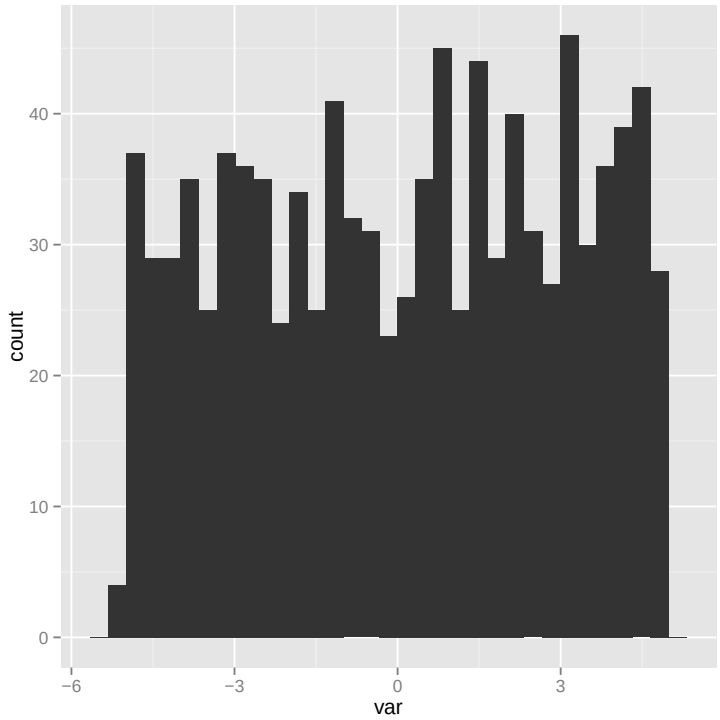
$$f(x) = \begin{cases} \frac{1}{b-a}, & a \leq x < b \\ 0, & x < a \text{ ó } x \geq b \end{cases}$$

lo que significa que la probabilidad que X esté en un subintervalo del soporte es la logitud del subintervalo. Con esto, la función de distribución acumulada es

$$F(x) = \begin{cases} 0, & x < a \\ \frac{x-a}{b-a}, & a \leq x < b \\ 1, & x \geq b \end{cases}$$

lo que es análogo a la distribución uniforme discreta ya que cualquier intervalo de la misma longitud tiene la misma probabilidad. Se muestra a continuación un histograma para mostrar las frecuencias de una variable aleatoria con soporte de -5 a 5. Se toma una muestra de 1,000 variables para poder ilustrar claramente el comportamiento de la distribución. Usamos el paquete `ggplot2` para graficar, pero para esto tenemos que convertir los datos en `data frame`.

```
> x ← data.frame(var = runif(1000,-5,5))
> ggplot(x, aes(var)) + geom_histogram()
```



Se puede demostrar facilmente que

$$\mathbb{E}(X) = \frac{a+b}{2}$$

y también que

$$\mathbb{V}(X) = \frac{(b-a)^2}{12}$$

Si se generan mil variables aleatorias uniformes en el intervalo de 0 a 10, la esperanza y la varianza coinciden con las fórmulas anteriores.

```
> x ← runif(1000,0,10); mean(x); var(x) # E(X) = 5 V(X) = 100/12
```

```
[1] 4.989099790591281
```

```
[1] 8.54763786900434
```

Ejemplo. Camiones llegan a una parada de autobús en intervalos de 15 minutos empezando a las 7 AM. Si un pasajero llega a la parada a un tiempo distribuido uniformemente entre las 7 y las 7:30, encontrar la probabilidad de que espere menos de cinco minutos a que llegue el camión.

Solución. Sea X los minutos después de las 7 a los cuales llega el pasajero. Como X es uniforme en el intervalo $(0, 30)$ entonces el pasajero esperará menos de cinco minutos si llega entre las 7:10 y las 7:15 o entre las 7:25 y 7:30. La probabilidad se calcula como sigue.

$$P\{10 < X < 15\} + P\{25 < X < 30\} = \int_{10}^{15} \frac{1}{30} dx + \int_{25}^{30} \frac{1}{30} dx = \frac{1}{3}$$

```
> punif(15, min = 0, max = 30) - punif(10, min = 0, max = 30) +
+ punif(30, min = 0, max = 30) - punif(25, min = 0, max = 30)
```

```
[1] 0.3333333333333334
```

3.2. Distribución Normal

Se dice que X tiene distribución normal con parámetros μ y σ^2 si tiene como función de densidad

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-\mu)^2/2\sigma^2}$$

con $-\infty < x < \infty$. La distribución de X tiene forma de campana y es simétrica con centro en μ. Los parámetros μ y σ^2 representan la media y la varianza respectivamente. La distribución normal también es conocida como la distribución Gaussiana ya que el matemático alemán Karl F. Gauss contribuyó a su desarrollo. Esta distribución es la más importante ya que tiene un amplio espectro de aplicaciones y ocurrencias en fenómenos naturales.

Cuando $\mu = 0$ y $\sigma = 1$ se dice que la distribución es normal estándar y su densidad de probabilidad se denota ϕ así como su función de densidad acumulada se denota Φ. Como no existe una forma analítica de expresar la integral de la función de densidad de una normal, se define una nueva variable aleatoria $Z = \frac{X-\mu}{\sigma}$ que es normal estándar y se encontra el valor de $\Phi(Z)$. En general, los valores necesitan calcularse usando métodos de integración numérica. Sin embargo, existen tablas que contienen los valores para una normal estándar, por lo que una vez hecha la transformación, únicamente tiene que consultarse una tabla. La función generadora de momentos para Z es

$$M(t) = e^{-t^2/2}$$

con $-\infty < t < \infty$.

Ejemplo. Sea X una variable aleatoria normal con parámetros $\mu = 3$ y $\sigma^2 = 9$, encontremos $P(2 < X < 5)$ y $P(|X - 3| > 6)$.

Solución. Sabemos que

$$P(2 < X < 5) = P(2 > X) - P(X > 5)$$

entonces la primera probabilidad se puede calcular como sigue.

```
> pnorm(2, mean = 3, sd = sqrt(9), lower.tail = FALSE) - pnorm(5, mean =
    3, sd = sqrt(9), lower.tail = FALSE)
```

```
[1] 0.3780661222713134
```

Ahora

$$P(|X - 3| > 6) = P(X - 3 > 6) + P(-X + 3 > 6) = P(X > 9) + P(X < -3)$$

entonces encontramos la segunda probabilidad con el siguiente código.

```
> pnorm(9, mean = 3, sd = sqrt(9), lower.tail = FALSE) + pnorm(-3, mean =
    3, sd = sqrt(9), lower.tail = TRUE)
```

```
[1] 0.04550026389635842
```

3.2.1. La aproximación normal a la distribución Binomial

Un resultado importante de la distribución normal es el teorema de DeMoivre-Laplace, el que dice que cuando n es grande, una variable binomial con parámetros n y p tendrá aproximadamente la misma distribución que una normal con la misma media y varianza que la binomial. Es decir,

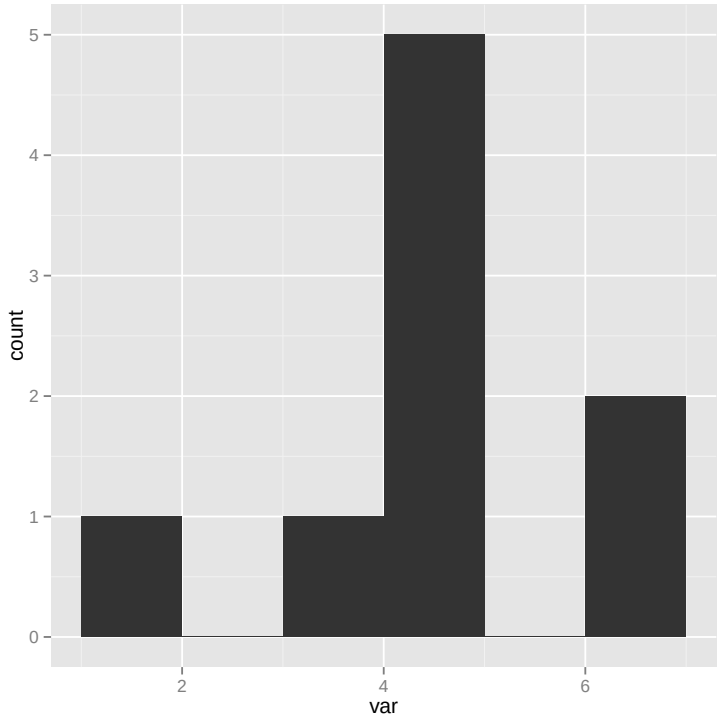
Teorema 1. *Si S_n denota la cantidad de éxitos que ocurren cuando se realizan n pruebas independientes, cada una con probabilidad p de ser éxito, entonces para cualquier $a < b$ se tiene*

$$P\left(a \leq \frac{S_n - np}{\sqrt{np(1-p)}} \leq b\right) \to \Phi(b) - \Phi(a)$$

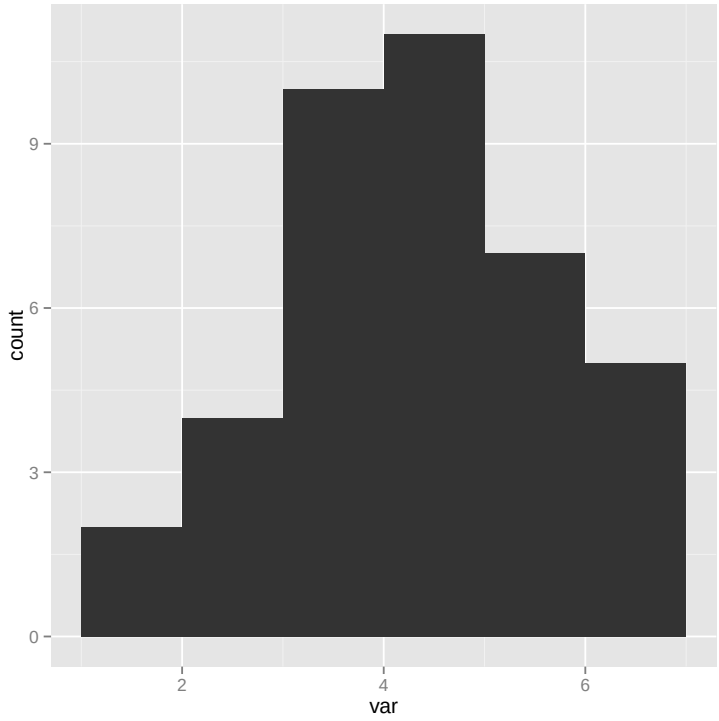
cuando $n \to \infty$.

Para visualizar el teorema podemos generar variables aleatorias con $p = 0.4$ fija pero incrementaremos la n en cada una de las siguientes gráficas. Tomaremos el tamaño de pruebas igual a diez en todos los casos.

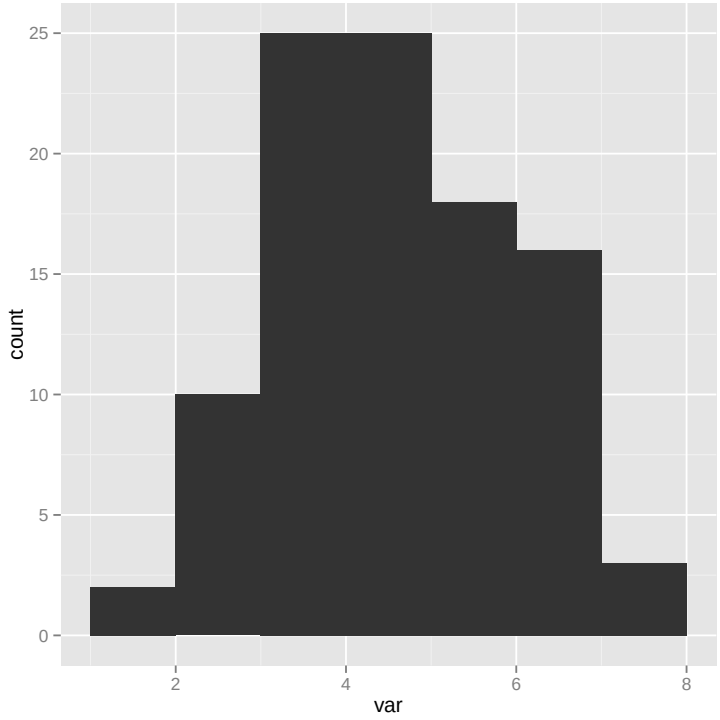
```
> x ← data.frame(var = rbinom(10, 10, .4))
> ggplot(x, aes(var)) + geom_histogram(binwidth=1) + xlim(min(x$var), max(
    x$var))
```



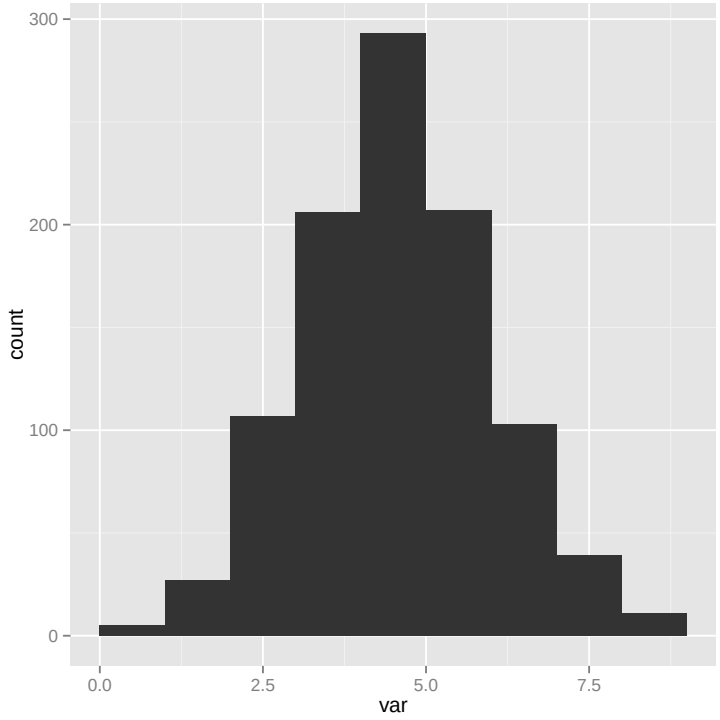
```
> x ← data.frame(var = rbinom(40, 10, .4))
> ggplot(x, aes(var)) + geom_histogram(binwidth=1) + xlim(min(x$var), max(
    x$var))
```




```
> x ← data.frame(var = rbinom(100, 10, .4))
> ggplot(x, aes(var)) + geom_histogram(binwidth=1) + xlim(min(x$var), max(
    x$var))
```




```
> x ← data.frame(var = rbinom(1000, 10, .4))
> ggplot(x, aes(var)) + geom_histogram(binwidth=1) + xlim(min(x$var), max(
    x$var))
```



Se puede observar que entre mayor es el número de observaciones, la distribución de las

observaciones se aproxima más a aquella de una variable aleatoria normal, con la forma de una campana.

Ejemplo. Sea X el número de veces que una moneda justa cae en sol cuando se tira 40 veces. Encuentre la probabilidad de que $X = 20$, aproximándola con la distribución normal.

Solución. Para poder usar la aproximación normal, se recomienda hacer una corrección de continuidad debido a que la distribución binomial es discreta y la normal es continua. Lo que hacemos es tomar

$$P(x = i) = P(i - 1/2 < X < i + 1/2)$$

y con esto ya le podemos aplicar la transformación.

$$\begin{aligned} P(X = 20) &= P(19.5 < X < 20.5) \\ &= P\left(\frac{19.5 - 20}{\sqrt{10}} < \frac{X - 20}{\sqrt{10}} < \frac{20.5 - 20}{\sqrt{10}}\right) \\ &\approx P\left(-.16 < \frac{X - 20}{\sqrt{10}} < .16\right) \\ &\approx \Psi(.16) - \Psi(-.16) \approx 0.1272 \end{aligned}$$

```
> pnorm(.16) - pnorm(-.16)
```

```
[1] 0.1271189257828657
```

La solución analítica al problema usando la distrbución binomial es

$$P(X = 20) = \binom{40}{20}\left(\frac{1}{2}\right)^{40} \approx 0.1254$$

lo que coincide bastante con la aproximación con la distribución normal.

3.3. Distribución exponencial

Una variable aleatoria continua se dice que es exponencial parámetro λ si tiene la función de densidad

$$f(x) = \begin{cases} \lambda e^{-\lambda x}, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

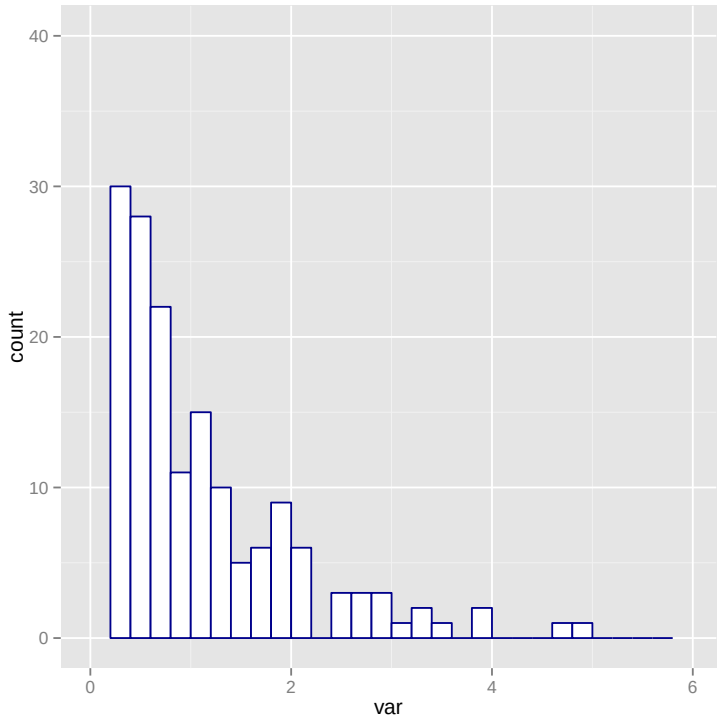
con $\lambda > 0$. La función de distribución acumulada $F(a)$ de una variable que se distribuye exponencial está dada por

$$\begin{aligned} F(a) &= P(X \leq a) \\ &= \int_0^a \lambda e^{-\lambda x} dx \\ &= -e^{-\lambda x}|_0^a \\ &= 1 - e^{-\lambda a} \end{aligned}$$

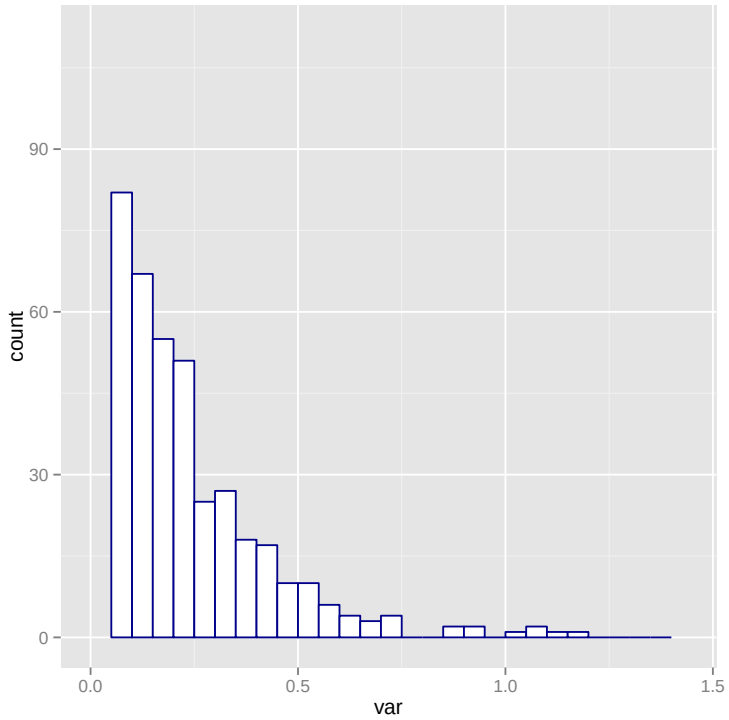
donde $a \geq 0$. Notemos que, abusando de notación, tenemos $F(\infty) = \int_0^\infty \lambda e^{-\lambda x} dx = 1$.

En los siguientes histogramas se pueden observar dos simulaciones de variables aleatorias exponenciales. La primera tiene $\lambda = 1$ con $n = 200$ observaciones y la segunda tiene $\lambda = 5$ con $n = 500$ observaciones.

```
> x ← data.frame(var = rexp(200))
> ggplot(x, aes(var)) + geom_histogram(binwidth=.2, colour = "darkblue",
    fill = "white") + xlim(min(x$var), max(x$var))
```




```
> x ← data.frame(var = rexp(500, 5))
> ggplot(x, aes(var)) + geom_histogram(binwidth=.05, colour = "darkblue",
    fill = "white") + xlim(min(x$var), max(x$var))
```



Se pueden calcular fácilmente la esperanza y la varianza de la distribución exponencial y obtenemos

$$E(X) = \frac{1}{\lambda}$$

$$V(X) = \frac{1}{\lambda^2}.$$

Notemos que la esperanza es el recíproco del parámetro y la varianza es el cuadrado de la esperanza. Comprobemos los resultados anteriores con los siguientes dos ejemplos.

```
> x ← rexp(200); mean(x); mean(x)^ 2; var(x)
```

```
[1] 0.9319460955199482
```

```
[1] 0.8685235249548765
```

```
[1] 1.070947648989875
```

```
> x ← rexp(500,5); mean(x); mean(x)^ 2; var(x)
```

```
[1] 0.1874665687959434
```

```
[1] 0.03514371441612418
```

```
[1] 0.03443236940659097
```

La distribución exponencial se utiliza comunmente para calcular el tiempo que se necesita para que suceda algún evento particular como el tiempo necesario para que llegue el tren o para que haya un terremoto.

Ejemplo. Supongamos que la duración de una llamada por teléfono está dada en minutos y se distribuye exponencialmente con media 10. Si alguien llega al teléfono inmediatamente antes que una segunda persona, encuentra la probabilidad de que la segunda persona tenga que esperar más de diez minutos y entre 10 y 20 minutos.

Solución. Sea X la duración de la llamada en minutos. Dado que la media es 10, sabemos que $\lambda = 1/10$. Necesitamos $P(X > 10)$ que se puede calcular de la siguiente manera.

```
> pexp(10, rate = 1/10, lower.tail=FALSE)
```

```
[1] 0.3678794411714423
```

Ahora, para que espere entre 10 y 20 minutos, podemos calcular $P(X > 10) - P(X > 20)$.

```
> pexp(10, rate = 1/10, lower.tail=FALSE) - pexp(20, rate = 1/10,
    lower.tail=FALSE)
```

```
[1] 0.2325441579348296
```

3.4. Distribución gamma

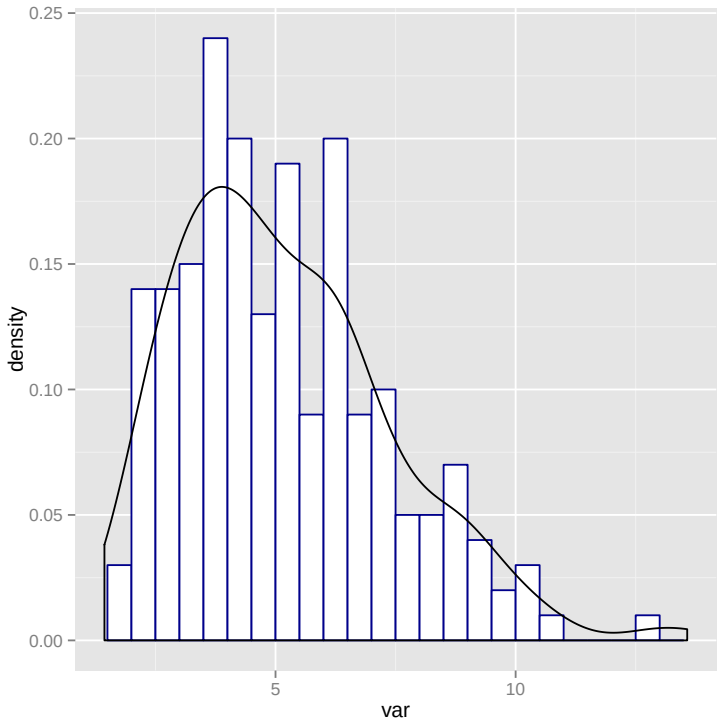
Una variable aleatoria tiene distribución gamma parámetros (α, Γ) si su función de densidad está dada por

$$f(x) = \begin{cases} \frac{\lambda e^{-\lambda x}(\lambda x)^{\alpha-1}}{\Gamma(\alpha)} & x \geq 0 \\ 0 & x < 0 \end{cases}$$

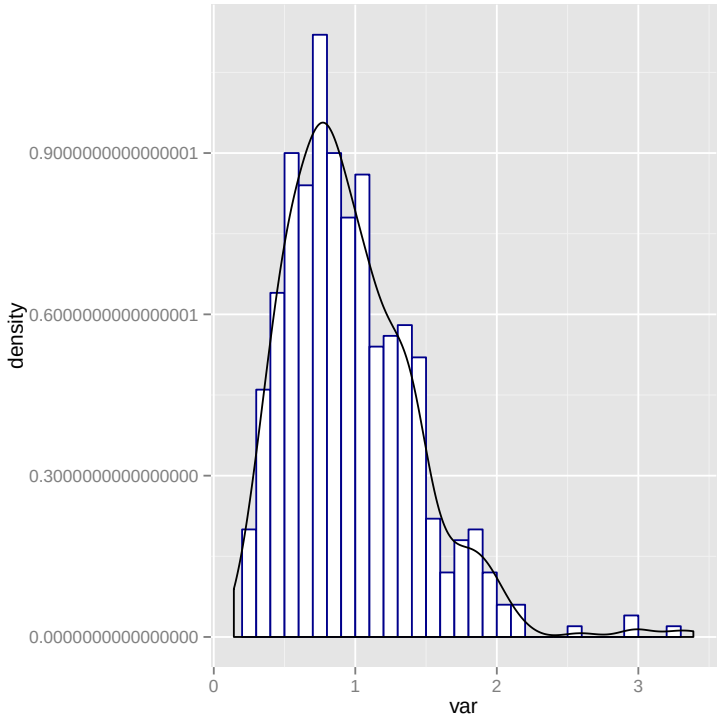
donde $\lambda \geq 0$, $\alpha \geq 0$ y $\Gamma(\alpha)$ es la función gamma definida como

$$\Gamma(\alpha) = \int_0^\infty e^{-y} y^{\alpha-1} dy.$$

```
> x ← data.frame(var = rgamma(200, 5))
> ggplot(x, aes(var)) + geom_histogram(aes(y = ..density..), binwidth=.5,
    colour = "darkblue", fill = "white") + xlim(min(x$var), max(x$var)) +
    geom_density()
```




```
> x ← data.frame(var = rgamma(500, 5, 5))
> ggplot(x, aes(var)) + geom_histogram(aes(y = ..density..), binwidth=.1,
    colour = "darkblue", fill = "white") + xlim(min(x$var), max(x$var)) +
    geom_density()
```



Para valores enteros de α, se puede demostrar facilmente que $\Gamma(\alpha) = (n-1)!$. En R, existe la función gamma que usaremos para comporbar el resultado.

```
> n ← 5
> gamma(n); factorial(n-1)
```

```
[1] 24
```

```
[1] 24
```

```
> n ← 10
> gamma(n); factorial(n-1)
```

```
[1] 362880
```

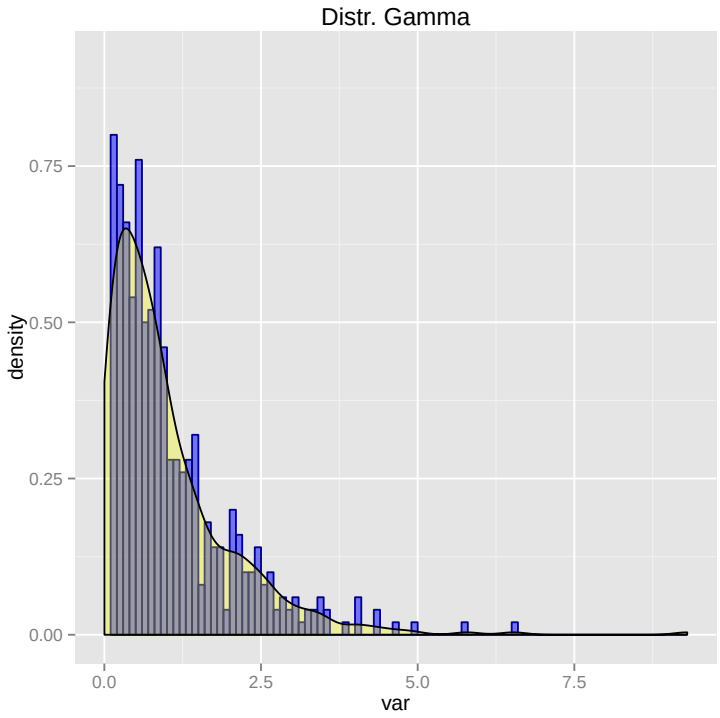
```
[1] 362880
```

Una distribución gamma con $\alpha = 1$ y $\lambda > 0$ es también exponencial parámetro λ. Simulando una variable aleatoria gamma con $\alpha = 1$ y $\lambda = 1$ y posteriormente una exponencial parámetro $\lambda = 1$, se obtienen los siguientes resultados.

```
> x ← data.frame(var = rgamma(500, 1))
> ggplot(x, aes(var)) + geom_histogram(aes(y = ..density..), binwidth=.1,
    colour = "darkblue", fill = "blue", alpha = 0.5) + xlim(min(x$var), max
    (x$var)) + geom_density(fill = 'yellow', alpha = 0.3) + ggtitle('Distr.
     Gamma')
> mean(x$var); var(x$var)
```

```
[1] 1.009545098226431
```

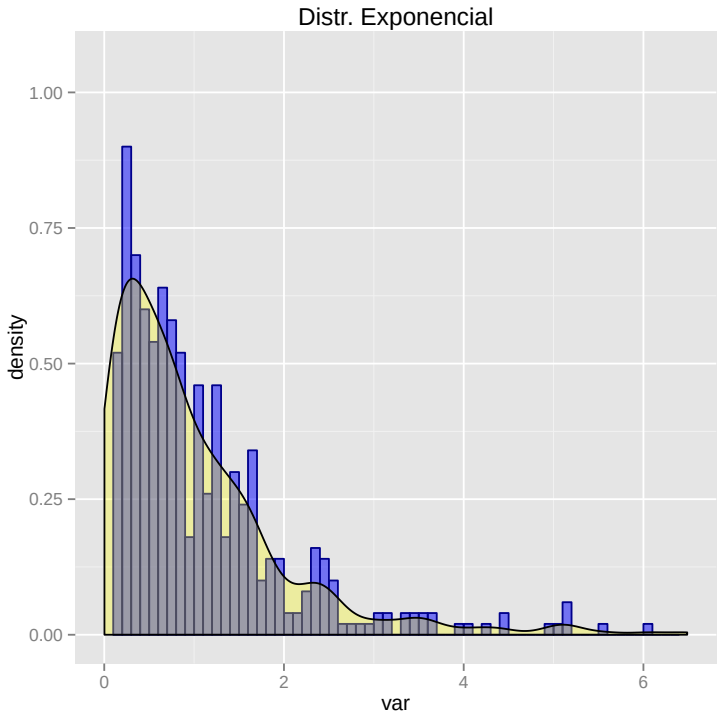
```
[1] 1.057202182136149
```




```
> x ← data.frame(var = rexp(500))
> ggplot(x, aes(var)) + geom_histogram(aes(y = ..density..), binwidth=.1,
    colour = "darkblue", fill = "blue", alpha = 0.5) + xlim(min(x$var), max
    (x$var)) + geom_density(fill = 'yellow', alpha = 0.3) + ggtitle('Distr.
     Exponencial')
> mean(x$var); var(x$var)
```

```
[1] 0.9946763961365823
```

```
[1] 1.011471520857803
```



La esperanza y la varianza de una variable aleatoria gamma están dadas por

$$E(X) = \frac{\alpha}{\lambda}$$

$$V(X) = \frac{\alpha}{\lambda^2}.$$

3.5. Resumen

La función de densidad de probabilidad, la media y la varianza de una variable aleatoria continua se pueden calcular usando los paquetes `distr` y `distrEx`. Se define la variable como objeto `AbscontDistribution` y se utilizan los métodos `p`, `E` y `Var`.

Si tenemos una variable aleatoria continua X tiene función de densidad de probabilidad, f, que cumple que para cualquier conjunto B,

$$P(x \in B) = \int_B f(x)dx$$

donde si X es continua, entonces la función de distribución F también es diferenciable y

$$\frac{d}{dx}F(x) = f(x).$$

El valor esperado de una variable aleatoria continua X está definida por

$$E(X) = \int_{-\infty}^{\infty} xf(x)dx$$

En la siguiente tabla podemos ver un resumen de las variables aleatorias continuas. En la segunda columna ponemos la función en `R` que genera variables aleatorias de cada una de las distribuciones. Para la función de distribución cambiamos la primera letra por una `p`, para los cuantiles por una `q` y para la función de densidad por una `d`.

Distribución	`R`	$f_X(x)$	Dominio	μ	σ^2
uniforme	`runif`	$\frac{x-a}{b-a}$	$a < x < b$	$\frac{a+b}{2}$	$\frac{(b-a)^2}{12}$
normal	`rnorm`	$\frac{1}{\sqrt{2\pi}\sigma} e^{-(x-\mu)^2/2\sigma^2}$	$-\infty < x < \infty$		
exponencial	`rexp`	$\lambda e^{-\lambda x}$	$x \geq 0$	$\frac{1}{\lambda}$	$\frac{1}{\lambda^2}$
gamma	`rgamma`	$\frac{\lambda e^{-\lambda x}(\lambda x)^{\alpha-1}}{\Gamma(\alpha)}$	$x \geq 0$	$\frac{\alpha}{\lambda}$	$\frac{\alpha}{\lambda^2}$

Capítulo 4

Variables aleatorias con distribución conjunta

4.1. Funciones de distribución conjunta

Muchas veces nos interesa el comportamiento de más de una variable aleatoria a la vez. Para dos variables aleatorias X y Y, la función de distribución de probabilidad acumulada conjunta está definida por

$$F(a,b) = P(X \leq a, Y \leq b)$$

para $-\infty < a, b < \infty$.

Las distribuciones marginales de X y Y se pueden obtener de la siguiente manera:

$$F_X(a) = \lim_{b\to\infty} F(a,b)$$

$$F_Y(b) = \lim_{a\to\infty} F(a,b)$$

Con lo anterior, cualquier probabilidad de X y Y se puede expresar en términos de su función de distribución conjunta y sus funciones marginales. Si se necesita que $X > a$ y $Y > b$, por ejemplo, se puede escribir

$$P(X > a, Y > b) = 1 - F_X(a) - F_Y(b) + F(a,b)$$

En el caso de variables aleatorias discretas, la función de masa de probabilidad de X y Y sería

$$p(x,y) = P(X = x, Y = y)$$

La probabilidad marginal de X se obtiene de su conjunta a través de

$$p_X(x) = P(X = x) = \sum_{y:p(x,y)>0} p(x,y)$$

y de la misma forma

$$p_Y(y) = P(Y = y) = \sum_{x:p(x,y)>0} p(x,y).$$

Ejemplo. Supongamos que 3 bolas se seleccionan aleatoriamente de una urna que contiene 3 bolas rojas, 4 blancas y 5 azules. Si X representa la cantidad de bolas rojas que se eligen y Y

la cantidad de bolas blancas, entonces encuentre su función de masa de probabilidad dada por $p(i,j) = P(X = i, Y = j)$.

Solución. Podemos encontrar todas las probabilidades y construir una tabla de las probabilidades que representa la función de masa de probabilidad para los distintos valores de X y Y.

Primero podemos crear una función en R que calcule la masa de probabilidad para (i,j) dado. Sabemos que el dominio de la función es $i = 0,\ldots,3$ y $j = 0,\ldots,3$, por lo que hay que tener cuidado al definir y usar la función.

```
> masa ← function(i,j){
+ if(i + j < 4)
+ m ← choose(3,i)*choose(5,3-i-j)*choose(4,j)/choose(12,3)
+ else
+ m ← 0
+ return(m)
+ }
```

Ahora podemos probar la función y checar que efectivamente sea una función de masa de probabilidad. Para esto, chequemos que la suma de todos los valores posibles de probabilidades sea igual a uno.

```
> options(digits=2)
> m ← matrix(nrow = 4, ncol = 4)
> for(i in 0:3){
+ for(j in 0:3){
+ m[i+1, j+1] ← masa(i,j)
+ }
+ }
> m
```

```
      [,1]  [,2]  [,3]  [,4]
[1,] 0.0455 0.182 0.136 0.018
[2,] 0.1364 0.273 0.082 0.000
[3,] 0.0682 0.055 0.000 0.000
[4,] 0.0045 0.000 0.000 0.000
```

La distribución marginal de X es la suma de las filas y está dada por el siguiente vector, con $X \in \{0,1,2,3\}$.

```
> rowSums(m)
```

```
[1] 0.3818 0.4909 0.1227 0.0045
```

Y la distribución marginal de Y es la suma de las columnas.

```
> colSums(m)
```

```
[1] 0.255 0.509 0.218 0.018
```

Decimos que X y Y son continuamente conjuntos si existe una función $f(x,y)$, definida para todo x y y reales si tiene la propiedad que

$$P\{(X,Y) \in C\} = \iint\limits_{(x,y)\in C} f(x,y)dxdy$$

con C cualquier subconjunto de $\mathbb{R}^2$. La función $f(x,y)$ es la función de densidad de probabilidad conjunta de X y Y.

Como

$$F(a,b) = P\{X \in (-\infty, a], Y \in (-\infty, b]\} = \int_{-\infty}^{b}\int_{-\infty}^{a} f(x,y)dx\,dy$$

entonces también se tiene que mientras F tenga sus derivadas parciales bien definidas

$$f(x,y) = \frac{\partial^2}{\partial x \partial y}F(x,y).$$

Con lo anterior también se pueden definir las funciones de densidad de probabilidad marginales

$$f_X(x) = \int_{-\infty}^{\infty} f(x,y)dy$$

$$f_Y(y) = \int_{-\infty}^{\infty} f(x,y)dx$$

Ejemplo. Supongamos que se tiene un círculo de radio $R = 1$ y que se elige un punto dentro del círculo con la misma probabilidad de elegir cualquier punto de coordenadas (X,Y). Con lo anterior, sabemos que la distribución conjunta de X y Y está dada por

$$f(x,y) = \begin{cases} c, & x^2 + y^2 \leq R^2 \\ 0, & x^2 + y^2 > R^2 \end{cases}$$

para alguna constante c. Determine c, encuentre las funciones de densidad marginal de X y Y, la probabilidad de el punto tenga una distancia al origen menor o igual a una consatante a y encuentre la esperanza de esta última variable.

Solución. Para encontrar c, sabemos que

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)dy\,dx = 1$$

entonces integrando, obtenemos que

$$c = \frac{1}{\pi R^2}.$$

Ahora podemos encontrar las distribuciones marginales

$$f_X(x) = \frac{2}{\pi R^2}\sqrt{R^2 - x^2}, \quad x^2 \leq R^2$$

y 0 si $x^2 > R^2$. Así como

$$f_Y(y) = \frac{2}{\pi R^2}\sqrt{R^2 - y^2}, \quad y^2 \leq R^2$$

y 0 si $y^2 > R^2$.

Sea $D = \sqrt{X^2 + Y^2}$ la distancia del punto (X, Y) al origen. Entonces la función de distribución para $0 \leq a \leq R$ es

$$\begin{aligned} F_D(a) &= P\{X^2 + Y^2 \leq a^2\} \\ &= \frac{1}{\pi R^2} \iint\limits_{x^2+y^2 \leq a^2} dy\, dx \\ &= \frac{a^2}{R^2} \end{aligned}$$

Derivando la función anterior se obtiene

$$f_D(a) = \frac{2a}{R^2}, \quad 0 \leq a \leq \mathtt{R}$$

y por lo tanto

$$E(D) = \frac{2}{R^2}\int_0^R a^2 da = \frac{2R}{3}.$$

4.1.1. Simulación: calculando π

En esta sección vamos a calcular el valor de π usando valores simulados de la distribución uniforme. Sabemos que el área de un círculo de radio r está dado por

$$A_1 = \pi r^2$$

y que el área de un cuadrado de lado l está dado por

$$A_2 = l^2$$

entonces la razón de área del círculo contra el cuadrado es

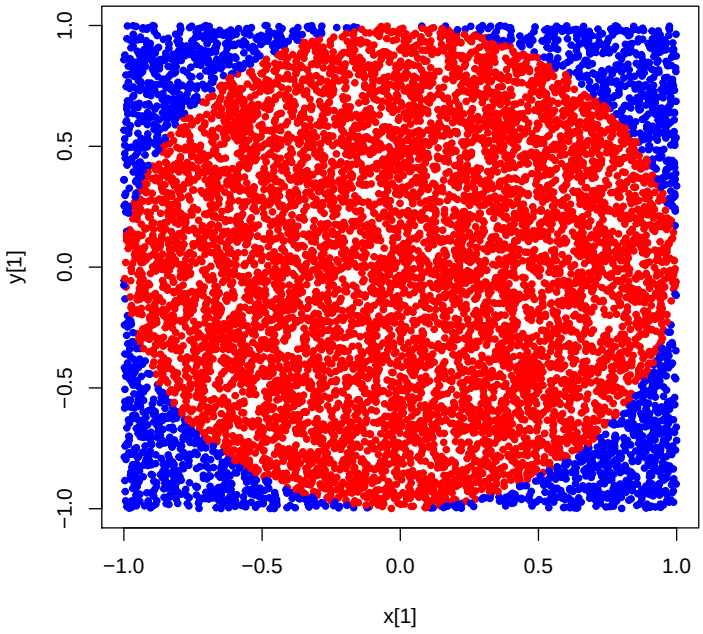
$$R = \frac{A_1}{A_2} = \frac{\pi r^2}{l^2}$$

y eligiremos $r = 1$ y $l = 2$ para nuestro ejemplo. Con esto, usaremos 10,000 puntos aleatorios uniformemente distribuidos en el cuadrado de 2×2 centrado en el origen para aproximar la razón que buscamos.

```
> x ← runif(10000, min=-1, max=1)
> y ← runif(10000, min=-1, max=1)
```

Tendremos un contador de los puntos que caen dentro del círculo y un contador para los que caen en el cuadrado pero fuera del círculo. Los puntos dentro del círculo los graficaremos en rojo y los demás en azul.

```
> circulo ← 0 # veces que los puntos caen dentro del circulo
> cuadrado ← 0 # veces que los puntos caen fuera del circulo
> plot(x[1], y[1], type="n", xlim=range(x), ylim=range(y))
> for(i in 1:10000){ # para todos los puntos
+ if(x[i]^2 + y[i]^2 < 1){ # si dentro del circulo, entra
+ circulo ← circulo + 1 # suma el punto dentro del circulo
+ points(x[i], y[i], pch=20, col="red") # punto en rojo
+ }else{
+ cuadrado ← cuadrado + 1 # si fuera del circulo, entra
+ points(x[i], y[i], pch=20, col="blue") # punto en azul
+ }
+ }
```



Con la ecuación anterior, tenemos que

$$\frac{A_1}{A_2} = \frac{\pi 1^2}{2^2}$$

por lo que

$$\pi = 4\frac{A_1}{A_2}$$

Por lo que obtenemos el valor de π tomando los puntos que caen en el círculo, dividiendo entre los que caen dentro o fuera del círculo y multiplicando por 4.

```
> 4 * circulo/(circulo+cuadrado)
```

```
[1] 3.1
```

4.2. Variables aleatorias independientes

Decimos que dos variables aleatorias X y Y son independientes si, para cualesquiera dos eventos A y B se tiene que

$$P(X \in A, Y \in B) = P(X \in A)P(Y \in B)$$

lo que significa que, cumpliendo con los axiomas de probabilidad, la función de distribución conjunta F debe cumplir

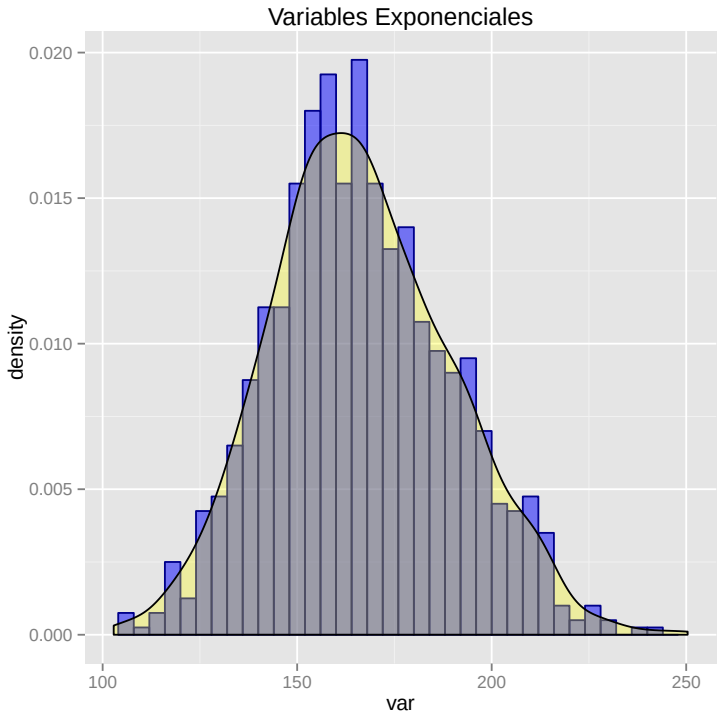
$$F(a,b) = F_X(a)F_Y(b) \quad \forall\, a, b$$

Proposición 1. *Si X y Y son variables aleatorias independientes gamma con parámetros (s,λ) y (t,λ) respectivamente, entonces $X+Y$ es una variable aleatoria gamma con parámetro $(s+t,\lambda)$.*

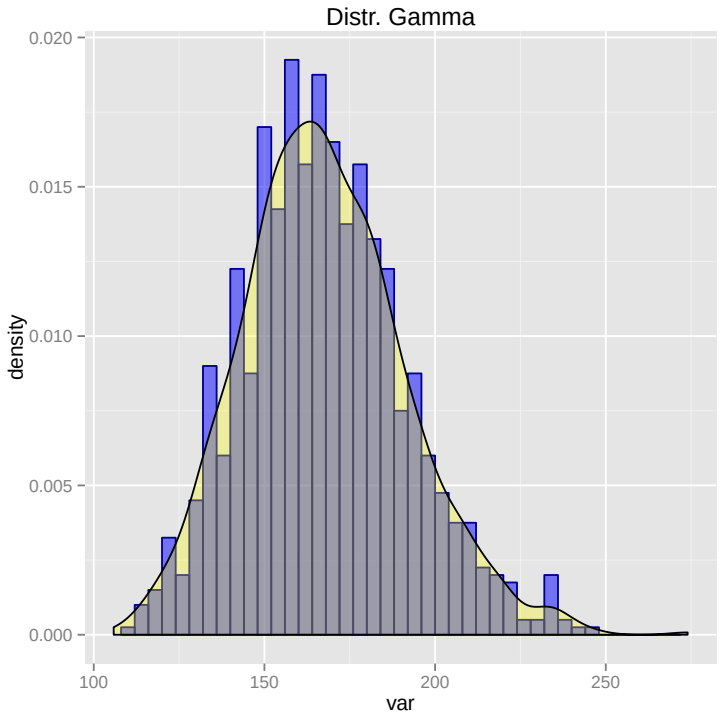
Ejemplo. Sean $X_1, X_2, \ldots, X_n$ n variables aleatorias independientes exponenciales, con parámetro λ cada una. Sabemos que una variable aleatoria exponencial parámetro λ es lo mismo que una gamma parámetro $(1,\lambda)$ y por lo tanto, de la proposición anterior, $X_1, X_2, \ldots, X_n$ tiene distribución gamma parámetro (n,λ).

A continuación haremos el ejemplo con $\lambda = 0.3$ para las n variables aleatorias exponenciales. Para poder graficar cómo se distribuye la suma de las variables, hacemos m experimentos y graficamos. Graficando una gamma con parámetro (n,λ), podemos ver que las dos gráficas son similares.

```
> n ← 50 # cantidad de exponenciales
> lambda ← .3
> x ← 1:50
> m ← 1000 # replicaciones del experimento
> xx ← 1:m
> for(j in 1:m){
+ for(i in 1:n)
+ x[i] ← rexp(1,lambda)
+ xx[j] ← sum(x)
+ }
> ggplot(data.frame(var = xx), aes(var)) + geom_histogram(aes(y =
    ..density..), binwidth=4, colour = "darkblue", fill = "blue", alpha = 0
    .5) + xlim(min(xx), max(xx)) + geom_density(fill = 'yellow', alpha = 0
    .3) + ggtitle('Variables Exponenciales')
```

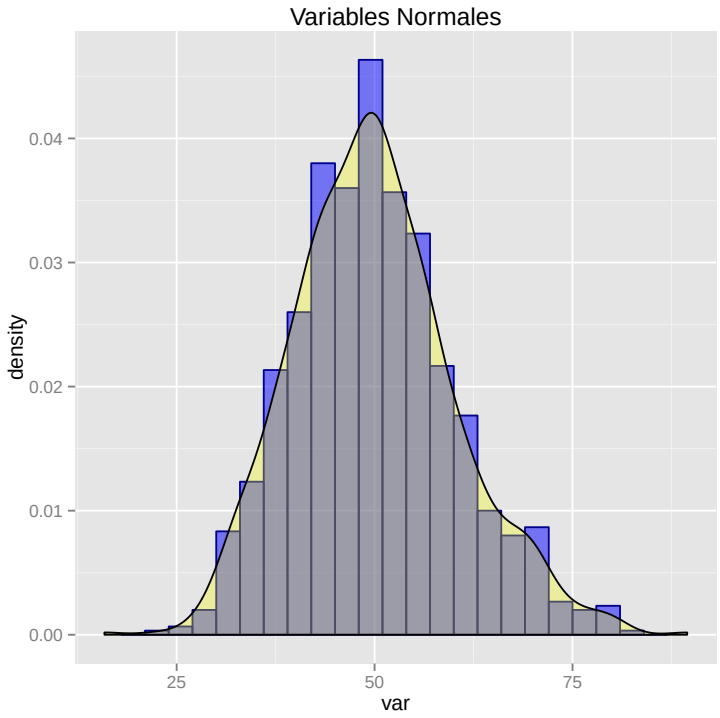



```
> x ← data.frame(var = rgamma(m,n,lambda))
> ggplot(x, aes(var)) + geom_histogram(aes(y = ..density..), binwidth=4,
    colour = "darkblue", fill = "blue", alpha = 0.5) + xlim(min(x$var), max
    (x$var)) + geom_density(fill = 'yellow', alpha = 0.3) + ggtitle('Distr.
     Gamma')
```

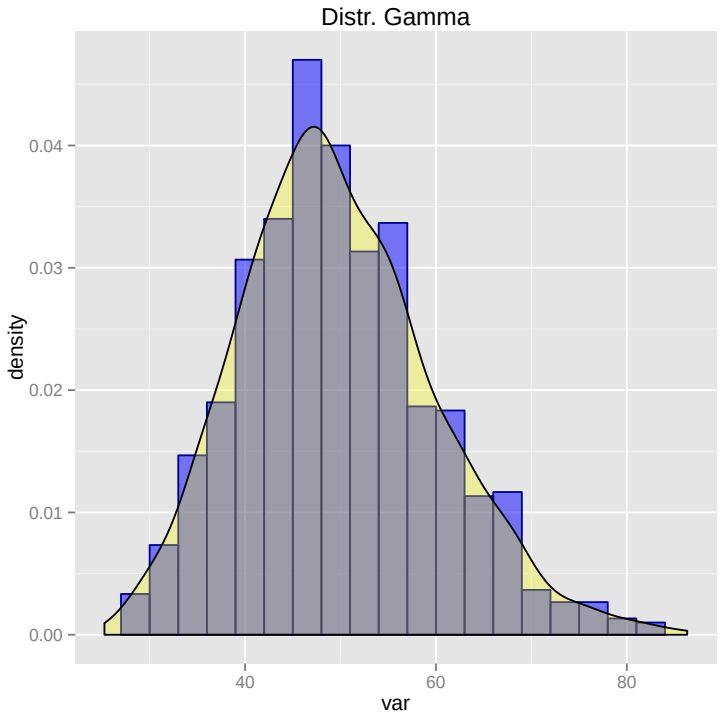


Ejemplo. Ahora sean $Z_1, Z_2, \ldots Z_n$ variables aleatorias independientes normal estándar, entonces $Y = \sum_{i=1}^{n} Z_i^2$ tiene una distribución chi-cuadrada con n grados de libertad. Sabemos que cuando $n = 1$, la chi-cuadrada es una gamma parámetro $(1/2, 1/2)$ y por lo tanto la suma de chi-cuadradas es gamma y tiene parámetro $(n/2, 1/2)$. A continuación se toman variables aleatorias normales con media cero y varianza uno para poder consecuentemente compararlo con la distribución gamma.

```
> n ← 50 # cantidad de xi
> x ← 1:50
> m ← 1000 # replicaciones del experimento
> xx ← 1:m
> for(j in 1:m){
+ for(i in 1:n)
+ x[i] ← rnorm(1,0,1)
+ xx[j] ← sum(x^ 2)
+ }
> ggplot(data.frame(var = xx), aes(var)) + geom_histogram(aes(y =
    ..density..), binwidth=3, colour = "darkblue", fill = "blue", alpha = 0
    .5) + xlim(min(xx), max(xx)) + geom_density(fill = 'yellow', alpha = 0
    .3) + ggtitle('Variables Normales')
```




```
> x ← data.frame(var = rgamma(m,n/2,1/2))
> ggplot(x, aes(var)) + geom_histogram(aes(y = ..density..), binwidth=3,
    colour = "darkblue", fill = "blue", alpha = 0.5) + xlim(min(x$var), max
    (x$var)) + geom_density(fill = 'yellow', alpha = 0.3) + ggtitle('Distr.
     Gamma')
```



Proposición 2. *Sean* $\{X_i \mid i = 1, \ldots, n\}$ *variables aleatorias independientes que se distribuyen normal con parámetros* μ_i, σ_i^2 *para* $i = 1, \ldots, n$*, entonces* $\sum_{i=1}^n X_i$ *se distribuye normal con media* $\sum_{i=1}^n \mu_i$ *y varianza* $\sum_{i=1}^n \sigma_i^2$.

Ejemplo. Supongamos que se tiene un equipo de fútbol que tiene que jugar un total de 44 partidos en la temporada. El equipo jugará 26 partidos contra equipos de primera división y 18 contra segunda. La probabilidad de que el equipo gane en primera división es de 0.4 mientras que en segunda es de 0.7, asumiendo que las victorias son independientes. Calcule la probabilidad de que el equipo gane 25 juegos o más y la probabilidad de que el equipo gane más juegos en primera división que en segunda.

Solución. Sean X_A y X_B el número de juegos que ganará el equipo en primera y segunda división respectivamente. Sabemos que X_A y X_B son variables aleatorias independientes binomiales y

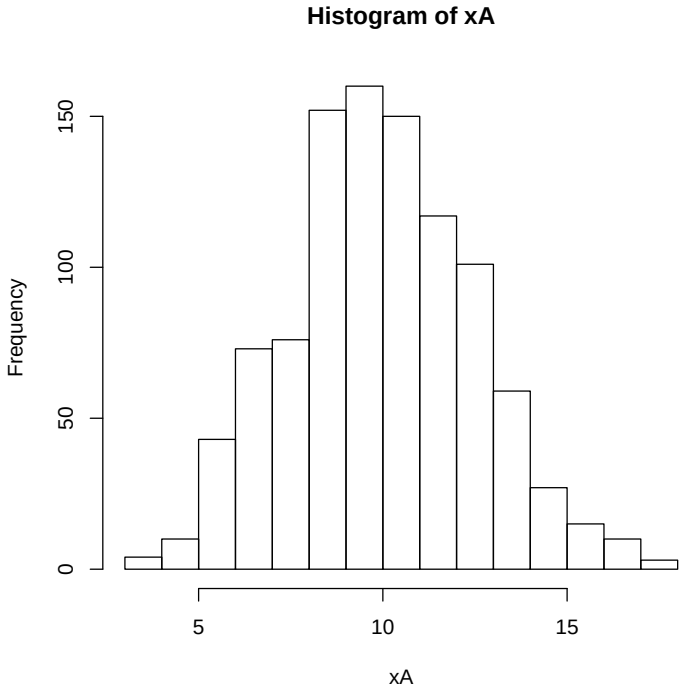
$$E[X_A] = 26(.4) = 10.4 \quad V(X_A) = 26(.4)(.6) = 6.24$$
$$E[X_B] = 18(.7) = 12.6 \quad V(X_B) = 18(.7)(.3) = 3.78$$

```
> xA ← rbinom(1000, 26, 0.4)
> mean(xA); var(xA)
```

```
[1] 10
```

```
[1] 6.4
```

```
> hist(xA)
```

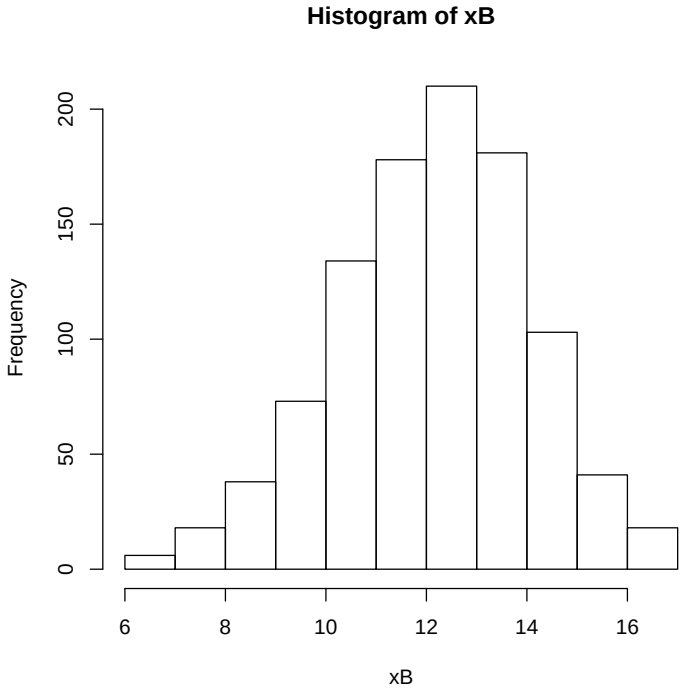



```
> xB ← rbinom(1000, 18, 0.7)
> mean(xB); var(xB)
```

```
[1] 13
```

```
[1] 3.8
```

```
> hist(xB)
```



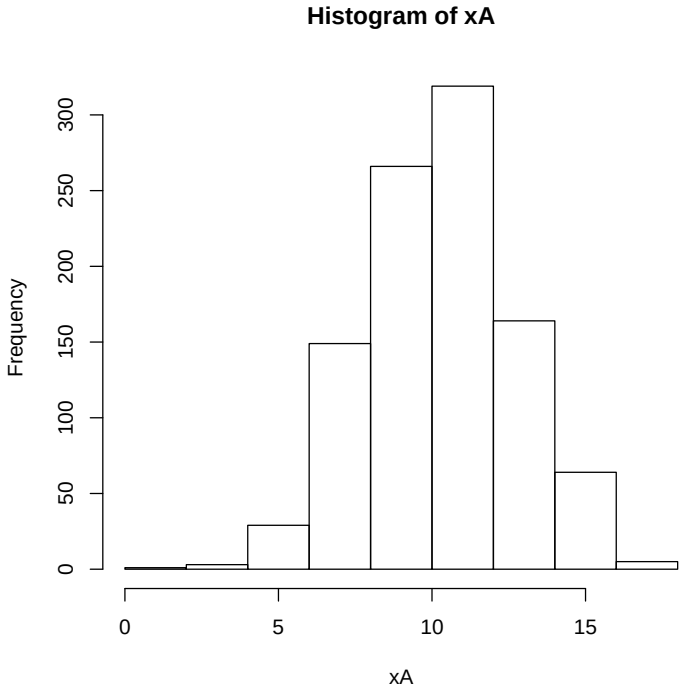
Por la aproximación normal a la binomial, tenemos que X_A y X_B tendrán aproximadamente la misma distribución que variables aleatorias normales independientes con las medias y varianzas anteriores.

```
> xA ← rnorm(1000, 10.4, sqrt(6.24))
> mean(xA); var(xA)
```

```
[1] 10
```

```
[1] 5.9
```

```
> hist(xA)
```

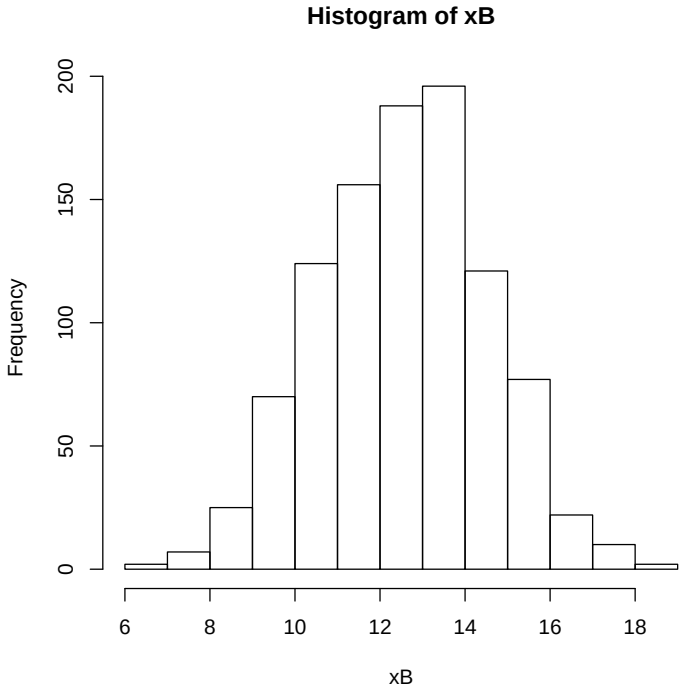



```
> xB ← rnorm(1000, 12.6, sqrt(3.78))
> mean(xB); var(xB)
```

```
[1] 13
```

```
[1] 3.9
```

```
> hist(xB)
```



Con esto, por la proposición anterior, $X_A + X_B$ tienen distribución normal con media 23 y varianza 10.02.

```
> xZ ← rnorm(1000, 23, sqrt(10.02))
> mean(xZ); mean(xA + xB)
```

```
[1] 23
```

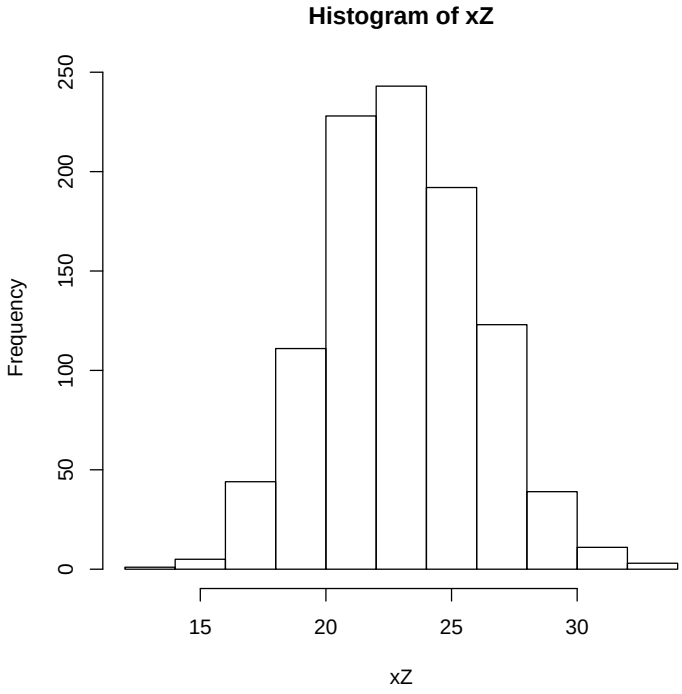
```
[1] 23
```

```
> var(xZ); var(xA + xB)
```

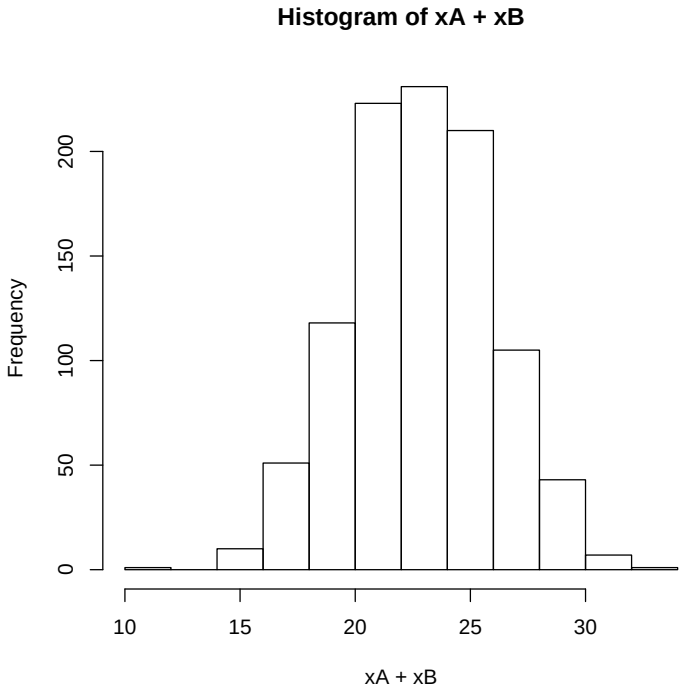
```
[1] 9.6
```

```
[1] 9.8
```

```
> hist(xZ)
```




```
> hist(xA + xB)
```



Proposición 3. *Sean $X_1, \ldots, X_n$ variables aleatorias independientes geométricas, con X_i de pará-*

metro p_i para cada $i = 1, \ldots, n$. Si todas las p_i son distintas, entonces, para $k \geq n$,

$$P(S_n = k) = \sum_{i=1}^{n} p_i q_i^{k-1} \prod_{j \neq i} \frac{p_j}{p_j - p_i}$$

Capítulo 5

Teoremas de Límite

5.1. La Desigualdad de Chebyshev y la Ley Débil de Grandes Números

Proposición 4. *(Desigualdad de Markov). Si X es una variable aleatoria que toma únicamente valores no negativos, entonces para cualquier valor $a > 0$ tenemos*

$$P\{X \geq a\} \leq \frac{E[X]}{a}.$$

Ejemplo. Si tiramos 200 veces una moneda no justa que cae en sol con probabilidad $1/10$, encuentre una cota superior de que la moneda caiga en sol al menos 120 veces.

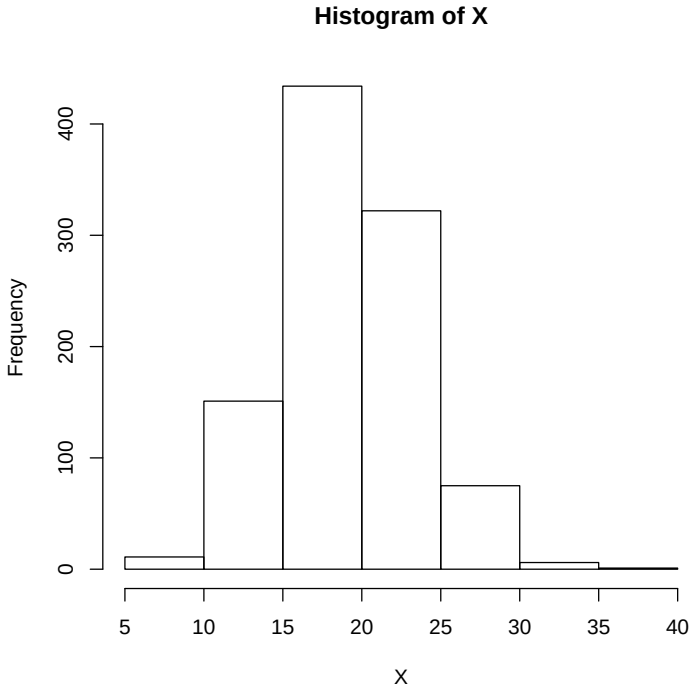
Mostramos una simulación de 1000 observaciones de una variable binomial con las características anteriores.

```
> X ← rbinom(1000, 200, 1/10)
> hist(X)
> mean(X) # E(X)
```

```
[1] 20
```

```
> mean(X)/120 # cota
```

```
[1] 0.16
```



Sabemos que $E(X) = 20$ y $a = 120$, por lo que

$$P\{X \geq 120\} \leq \frac{20}{120} = \frac{1}{6}$$

Podemos sacar la probabilidad teórica de que la moneda haya caído en sol al menos 120 veces.

```
> pbinom(120, 200, 1/10, lower.tail = FALSE) # P(X > 120)
```

```
[1] 2.8e-68
```

Lo cual es casi cero y podemos concluir que la desigualdad de Markov no es muy fuerte aunque puede ser bastante útil cuando únicamente se conoce la media de la variable.

Proposición 5. *(Desigualdad de Chebyshev). Si X es una variable aleatoria (puede ser positiva o negativa) con media μ y varianza σ^2 finitas, entonces, para cualquier valor $k > 0$,*

$$P\{|X - \mu| \geq k\} \leq \frac{\sigma^2}{k^2}$$

Lo que quiere decir que, independientemente de la distribución de la variable aleatoria, se puede asegurar que cierto porcentaje de observaciones será dentro de k desviaciones estándar de la media.

Ejemplo. Si se tiene el mismo problema de la moneda anterior, se puede calcular la probabilidad de que caigan 120 o más soles de la siguiente manera. Sabemos que $\mu = 20$ y $V(X) = 200 \times 1/10 \times 9/10 = 18$, entonces

$$P(X \geq 120) = P(X - 20 \geq 100) \leq \frac{18}{10000} = 0.0018$$

```
> sd(X)^2 / (120 - mean(X))^2
```

```
[1] 0.0018
```

Con lo que se puede concluir que, si podemos conocer la varianza de la variable aleatoria, se puede tener una mejor cota.

La importancia de las desigualdades anteriores es la habilidad de acotar distintas probabilidades cundo se conoce únicamente la media o ambas la media y la varianza.

Ejemplo. Supongamos que la cantidad de artículos producidos en una fábrica durante una semana es una variable aleatoria con media 50. ¿Qué se puede decir de la probabilidad de que la producción de la próxima semana exceda 75? Si la varianza de la producción de una semana se sabe que es de 25 artículos, ¿qué se puede decir de la probabilidad de que la producción de la próxima semana sea entre 40 y 60?

Solución. Sea X el número de artículos que se producirán en una semana. Por la desigualdad de Markov,

$$P(X > 75) \leq \frac{E(X)}{75} = \frac{50}{75} = \frac{2}{3}$$

Y ahora por la desigualdad de Chebyshev,

$$P(|X - 50| \geq 10) \leq \frac{\sigma^2}{10^2} = \frac{1}{4}$$

Por lo tanto,

$$P(|X - 50| < 10) \geq 1 - \frac{1}{4} = \frac{3}{4}$$

y entonces la probabilidad de que la producción de la próxima semana sea de entre 40 y 60 artículos es por lo menos 0.75.

La desigualdad de Chebyshev es válida para cualquier distribución de la variable aleatoria, no podemos esperar que la cota se encuentre siempre próxima al valor real de la probabilidad.

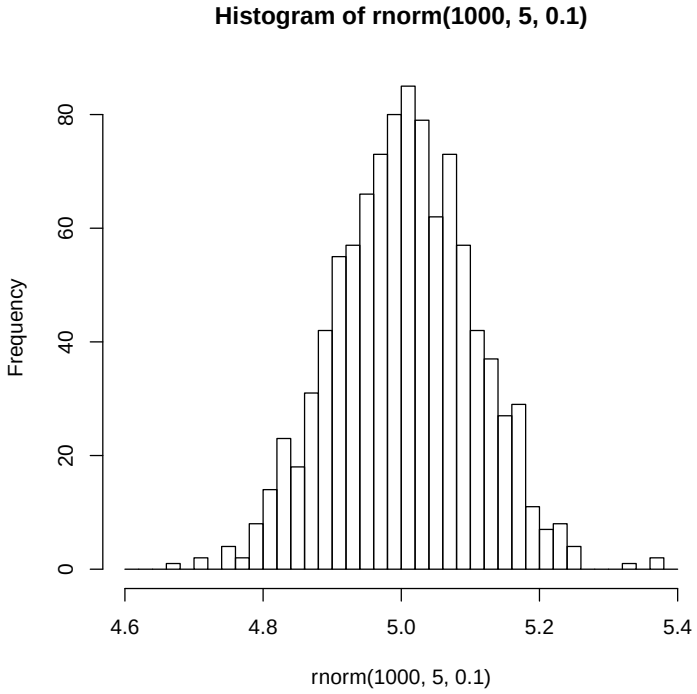
Proposición 6. *Si* $V(X) = 0$, *entonces*

$$P(X = E(X)) = 1$$

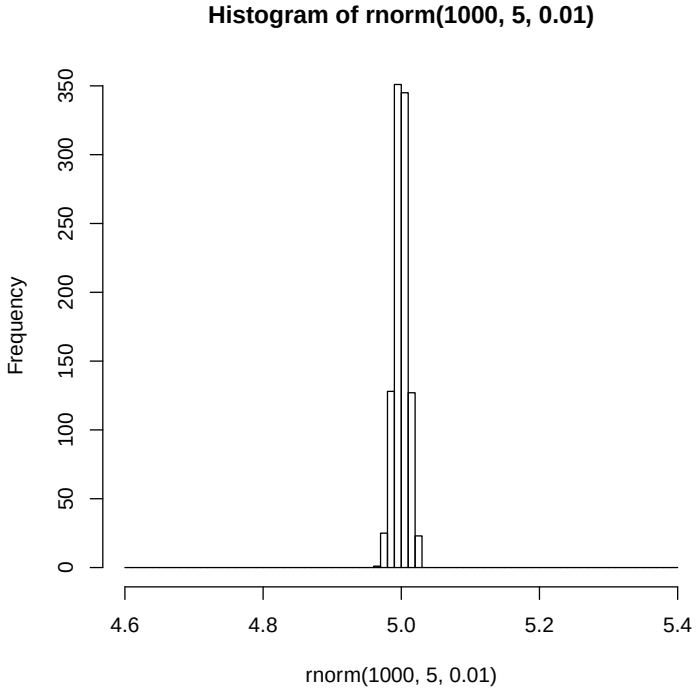
Lo que significa que las únicas variables aleatorias que tienen varianzas igual a cero son las que son constantes con probabilidad 1.

Hagamos una prueba generando variables aleatorias normales con media cinco y varianza cada vez menor. Conforme la varianza disminuye, los valores se acercan más a la media.

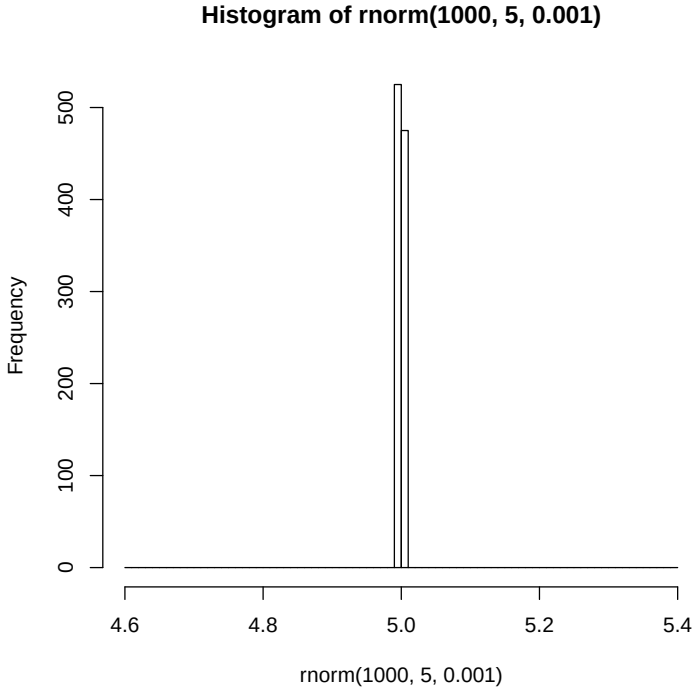
```
> hist(rnorm(1000, 5, 0.1), breaks = seq(from = 4.6, to = 5.4, by = 0.02))
```




```
> hist(rnorm(1000, 5, 0.01), breaks = seq(from = 4.6, to = 5.4, by = 0.01)
    )
```




```
> hist(rnorm(1000, 5, 0.001), breaks = seq(from = 4.6, to = 5.4, by = 0.01
    ))
```



Teorema 2. *(Ley Débil de los Grandes Números). Sean $X_1, X_2, \ldots$ una secuencia de variables aleatorias independientes e idénticamente distribuidas, cada una con media finita $E(X_i) = \mu$. Entonces, para cualquier $\varepsilon > 0$,*

$$P\left\{\left|\frac{X_1 + \ldots + X_n}{n} - \mu\right| \geq \varepsilon\right\} \to 0 \quad \textit{conforme} \quad n \to \infty$$

5.2. Teorema Central del Límite

El Teorema Central del Límite dice que la suma de una gran cantidad de variables aleatorias independientes se distribuye aproximadamente normal.

Teorema 3. *Teorema Central del Límite Sea $X_1, X_2, \ldots$ una sucesión de variables aleatorias independientes e idénticamente distribuidas, cada una con media μ y varianza σ^2. Entonces la distribución de*

$$\frac{X_1 + \ldots + X_n - n\mu}{\sigma\sqrt{n}}$$

tiende a la distribución normal estándar conforme $n \to \infty$. Eso significa que, para $-\infty < a < \infty$,

$$P\left\{\frac{X_1 + \ldots + X_n - n\mu}{\sigma\sqrt{n}} \leq a\right\} \to \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{a} e^{-x^2/2} dx \quad \textit{conforme} \quad n \to \infty$$

Ejemplo. Un astrónomo está interesado en medir la distancia, en años luz, de su observatorio a una estrella. El astrónomo tiene una manera de medir la distancia, pero a causa de condiciones

variables de la atmásfera y error normal, cada vez que mide una observación será una distancia aproximada. Considerando esto, el astrónomo quiere hacer varias mediciones y después tomar el promedio de las mediciones como su estimador de la distancia. Si el astrónomo piensa que los valores de las mediciones son variables aleatorias independientes e idénticamente distribuidas con media d (la distancia real) y varianza 4 (años luz), ¿cuántas mediciones necesitan hacerse para que el astronomo esté suficientemente seguro de que la distancia es precisa con entre ± 0.5 años luz?

Solución. Supongamos que el astrónomo decide hacer n observaciones distintas. Si $X_1, X_2, \ldots, X_n$ son las n mediciones, entonces se tiene que por el teorema central del límite

$$Z_n = \frac{\sum_{i=1}^{n} X_i - nd}{2\sqrt{n}}$$

tiene una distribución aproximadamente normal. Entonces,

$$\begin{aligned} P\left\{-.5 \leq \frac{\sum_{i=1}^{n} X_i}{n} - d \leq .5\right\} &= P\left\{-.5\frac{\sqrt{n}}{2} \leq Z_n \leq .5\frac{\sqrt{n}}{2}\right\} \\ &\approx \Phi\left(\frac{\sqrt{n}}{4}\right) - \Phi\left(-\frac{\sqrt{n}}{4}\right) \\ &= 2\Phi\left(\frac{\sqrt{n}}{4}\right) - 1 \end{aligned}$$

Por lo tanto, si el astrónomo quiere estar $95\,\%$ seguro de que el valor estimado es preciso dentro de $.5$ años luz de diferencia, debería de hacer n^* mediciones tal que

$$2\Phi\left(\frac{\sqrt{n^*}}{4}\right) - 1 = .95 \quad \Rightarrow \quad \Phi\left(\frac{\sqrt{n^*}}{4}\right) = .975$$

Y con una tabla de probabilidad normal,

$$\frac{\sqrt{n^*}}{4} = 1.96 \quad \Rightarrow \quad n^* = (7.84)^2 \approx 61.47$$

```
> n ← 5
> pnorm(.5*sqrt(n)/2) - pnorm(-.5*sqrt(n)/2)
```

```
[1] 0.42
```

```
> n ← 15
> pnorm(.5*sqrt(n)/2) - pnorm(-.5*sqrt(n)/2)
```

```
[1] 0.67
```

```
> n ← 62
> pnorm(.5*sqrt(n)/2) - pnorm(-.5*sqrt(n)/2)
```

```
[1] 0.95
```

Por lo que el astrónomo debe hacer 62 observaciones para obtener el valor que desea.

Notemos que el procedimiento anterior se hizo con la hipótesis de que con $n = 62$, la distribución de Z_n se distribuye normal. Por lo general, este es el caso aunque la pregunta de qué tan grade debe ser n depende de las distribuciones desconocidas de las X_i. Si el astrónomo no quiere arriesgarse, también puede usar la desigualdad de Chebyshev. Sabemos que

$$E\left[\sum_{i=1}^{n}\frac{X_i}{n}\right] = d \quad V\left(\sum_{i=1}^{n}\frac{X_i}{n}\right) = \frac{4}{n}$$

entonces por la desigualdad de Chebyshev tenemos

$$P\left\{\left|\sum_{i=1}^{n}\frac{X_i}{n} - d\right| > .5\right\} \leq \frac{4}{n(.5)^2} = \frac{16}{n}$$

Por lo tanto, si el astrónomo hace $n = 16/.05 = 320$ observaciones, puede estar 95 % seguro que su estimador será preciso dentro de 0.5 años luz.

```
> n ← 320
> pnorm(.5*sqrt(n)/2) - pnorm(-.5*sqrt(n)/2)
```

```
[1] 1
```

```
> pcauchy(.5*sqrt(n)/2) - pcauchy(-.5*sqrt(n)/2) # checar xq q raro!!
```

```
[1] 0.86
```

El teorema central del lmite también se puede enunciar para variableas aleatorias independientes, aunque no necesariamente identicamente distribuidas.

Teorema 4. *(Teorema Central del Límite para variables aleatorias independientes). Sean* $X_1, X_2, \ldots$ *una sucesión de variables aleatorias independientes con medias* $\mu_i = E(X_i)$ *y varianzas* $\sigma_i^2 = V(X_i)$. *Si las* X_i *estén acotadas uniformemente (para alguna* M, $P(|X_i| < M) = 1$ *para toda* i*) y* $\sum_{i=1}^{\infty}\sigma_i^2 = \infty$, *entonces*

$$P\left\{\frac{\sum_{i=1}^{n}(X_i - \mu_i)}{\sqrt{\sum_{i=1}^{n}\sigma_i^2}} \leq a\right\} \to \Phi(a) \quad \textit{conforme} \;\; n \to \infty$$

5.3. La Ley Fuerte de los Grandes Números

La ley fuerte de los grandes números es uno de los resultados más conocidos en la probabilidad. Lo que dice es que si tenemos una sucesión de variables aleatorias independientes con la misma distribución, con probabilidad 1 la sucesión convergirá a la media de tal distribución.

Teorema 5. *(Ley fuerte de los grandes números). Sean $X_1, X_2, \ldots$ una sucesión de variables aleatorias independientes e idénticamente distribuidas, cada una con media finita $\mu = E(X_i)$. Entonces, con probabilidad 1, se tiene*

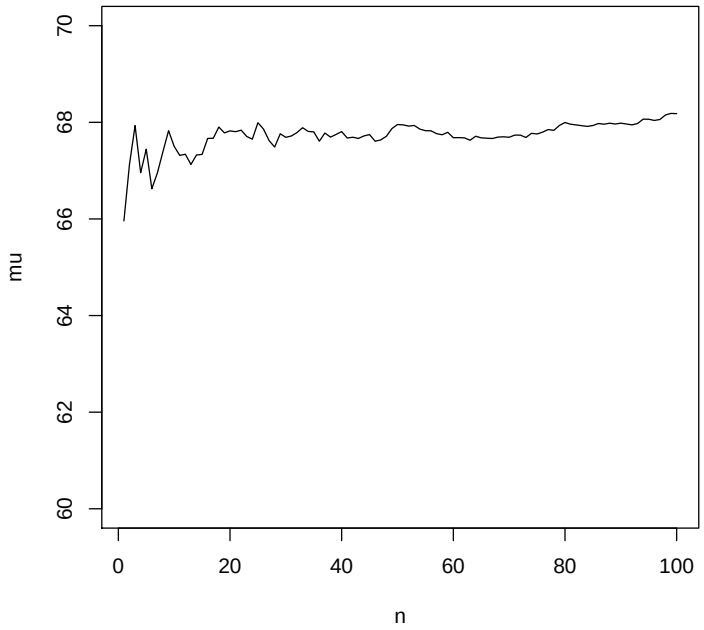
$$\frac{X_1 + X_2 + \ldots + X_n}{n} \to \mu \quad \textit{conforme} \ \ n \to \infty$$

Equivalentemente,

$$P\left\{ \lim_{n\to\infty} \frac{X_1 + X_2 + \ldots + X_n}{n} = \mu \right\} = 1$$

Para visualizar el teorema, podemos hacer una simulación de una variable aleatoria con media 68 y varianza 3.5, haciendo n observaciones. Se crea un vector que para calcular con la suma acumulada de los valores de la variable aleatoria entre n, la cantidad de observaciones en ese punto, para obtener la media para cada n. Se grafican las medias de la variable aleatoria conforme n crece y se puede observar cómo la media converge.

```
> n ← 1:100
> x ← rnorm(100, 68, 3.5)
> s ← cumsum(x)
> mu ← s/n
> plot(mu, xlab="n", ylim = c(60,70), type="l")
```



5.4. Otras Desigualdades

Teorema 6. *(Desigualdad de Chebyshev de un lado). Si X es una variable aleatoria con media 0 y varianza finita σ^2, entonces, para cualquiera $a > 0$,*

$$P(X > a) \leq \frac{\sigma^2}{\sigma^2 + a^2}$$

Ejemplo. Si la cantiad de artículos producidos en una fábrica durante una semana es una variable aleatoria con media 100 y varianza 400, construya una cota superior de la probabilidad de que la producción de la semana acual sea por lo menos 120.

Solución. Se sigue de la desigualdad de Chebyshev de un lado que

$$P(X \geq 120) = P(X - 100 \geq 20) \leq \frac{400}{400 + (20)^2} = \frac{1}{2}$$

Por lo tanto, la probabilidad de que en la semana actual se produzcan 120 o más artículos es a lo más $1/2$. Si se obtuviera la cota a partir de la desigualdad de Markov se tendría

$$P(X \geq 120) \leq \frac{E(X)}{120} = \frac{5}{6}$$

lo cual es una cota más débil que la anterior.

Corolario 1. *Si $E(X) = \mu$ y $V(X) = \sigma^2$, entonces, para $a > 0$,*

$$P(X \geq \mu + a) \leq \frac{\sigma^2}{\sigma^2 + a^2}$$

$$P(X \leq \mu - a) \leq \frac{\sigma^2}{\sigma^2 + a^2}$$

Proposición 7. *(Cotas de Chernoff).*

$$P(X \geq a) \leq e^{-ta} M(t) \quad \textit{para toda } \ t > 0$$

$$P(X \leq a) \leq e^{-ta} M(t) \quad \textit{para toda } \ t < 0$$

Dado que las cotas de Chernoff son para todo valor que obtenga t, se crea la mejor cota para $P(X \geq a)$ usando la t que minimize $e^{-ta}M(t)$.

Ejemplo. Si Z es una variable aleatoria normal estándar, entonces su función generadora de momentos es $M(t) = e^{t^2/2}$, por lo que la cota de Chernoff para $P(Z \geq a)$ está dada por

$$P(Z \geq a) \leq e^{-ta} e^{t^2/2} \quad \text{para toda } \ t > 0$$

Ahora, como el valor de t que minimiza $e^{t^2/2 - ta}$ es el valor que minimiza $t^2/2 - ta$, lo cual es $t = a$, se obtiene

$$P(Z \geq a) \leq e^{-a^2/2} \quad \text{para } \ a > 0$$

Proposición 8. *(Desigualdad de Jensen). Si $f(x)$ es una función convexa, entonces*

$$E(f(X)) \geq f(E(X))$$

suponiendo que las esperanzas existen y son finitas.

Capítulo 6

Anexos

6.1. ggplot2

El paquete ggplot2 facilita hacer gráficas en R. El siguiente ejemplo es de la página de internet http://www.statmethods.net/advgraphs/ggplot2.html. Primero se necesita cargar el paquete.

```
> library(ggplot2)
```

El paquete tiene bases de datos incluidas como es el caso de los automóviles de la revista *1974 Motor Trend* en Estados Unidos. La base contiene el consumo de gasolina de los autos así como 10 variables de diseño y rendimiento. Las primeras seis observaciones se muestran a continuación.

```
> head(mtcars)
```

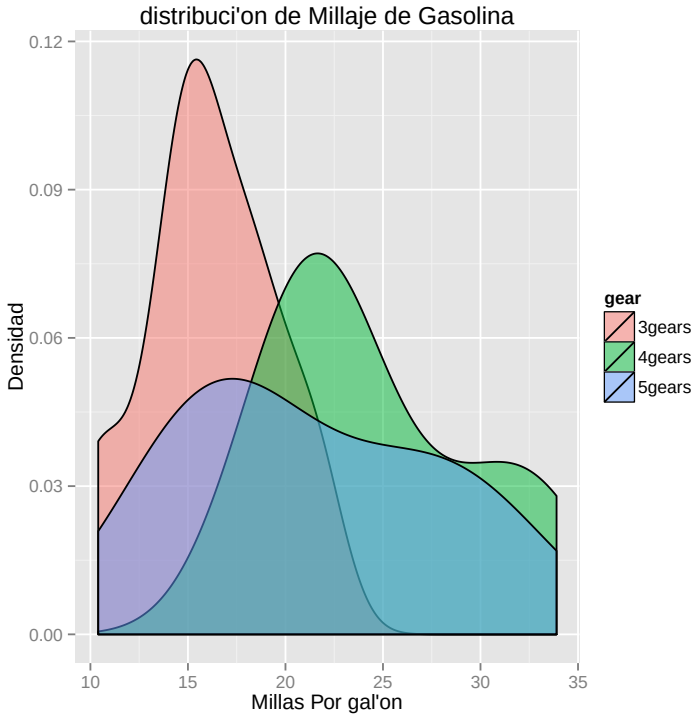
```
                  mpg cyl disp  hp drat  wt qsec vs am gear carb
Mazda RX4          21   6  160 110  3.9 2.6   16  0  1    4    4
Mazda RX4 Wag      21   6  160 110  3.9 2.9   17  0  1    4    4
Datsun 710         23   4  108  93  3.9 2.3   19  1  1    4    1
Hornet 4 Drive     21   6  258 110  3.1 3.2   19  1  0    3    1
Hornet Sportabout  19   8  360 175  3.1 3.4   17  0  0    3    2
Valiant            18   6  225 105  2.8 3.5   20  1  0    3    1
```

La variable de engranaje del auto sabemos que tiene un cierto orden donde 3, 4 y 5 engranajes estén en orden creciente y por lo mismo hay que convertirlos en factor para que conserven el orden al graficar. Lo mismo aplica para la variable de cilindros donde 4, 6 y 8 cilindros estén en orden creciente. La variable que indica si el auto es automático o estándar puede verse como booleana, tomando valores 0 si es automático y 1 si es estándar.

```
> mtcars$gear <- factor(mtcars$gear,levels=c(3,4,5), labels=c("3gears","4
    gears","5gears"))
> mtcars$am <- factor(mtcars$am,levels=c(0,1), labels=c("Automatic","Manual
    "))
> mtcars$cyl <- factor(mtcars$cyl,levels=c(4,6,8), labels=c("4cyl","6cyl","
    8cyl"))
```

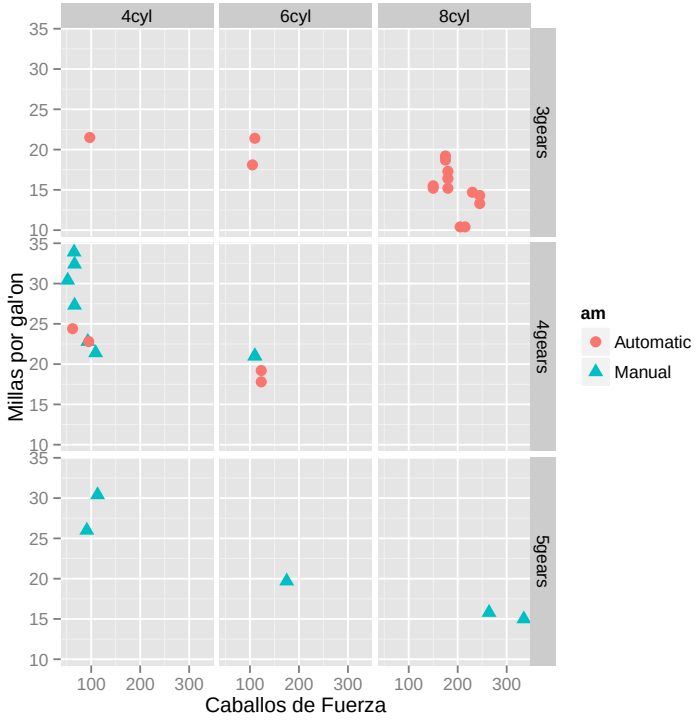
La densidad del kernel para el millaje por galón agrupada por la cantidad de engranajes, indicado por color, se grafica de la siguiente manera.

```
> qplot(mpg, data=mtcars, geom="density", fill=gear, alpha=I(.5), main="
    distribuci\'on de Millaje de Gasolina", xlab="Millas Por gal\'on", ylab
    ="Densidad")
```



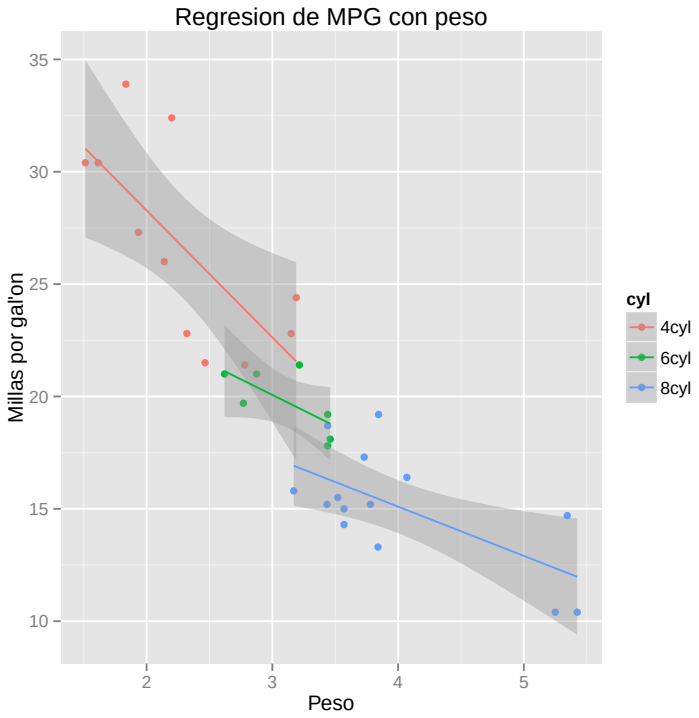
El siguiente es un diagrama de dispersión de millaje por galón contra los caballos de fuerza brutos. Para cada combinación de engranajes y cilindros en cada faceta, el tipo de transmisión está representado por forma y color.

```
> qplot(hp, mpg, data=mtcars, shape=am, color=am, facets=gear~cyl, size=I
    (3), xlab="Caballos de Fuerza", ylab="Millas por gal\'on")
```



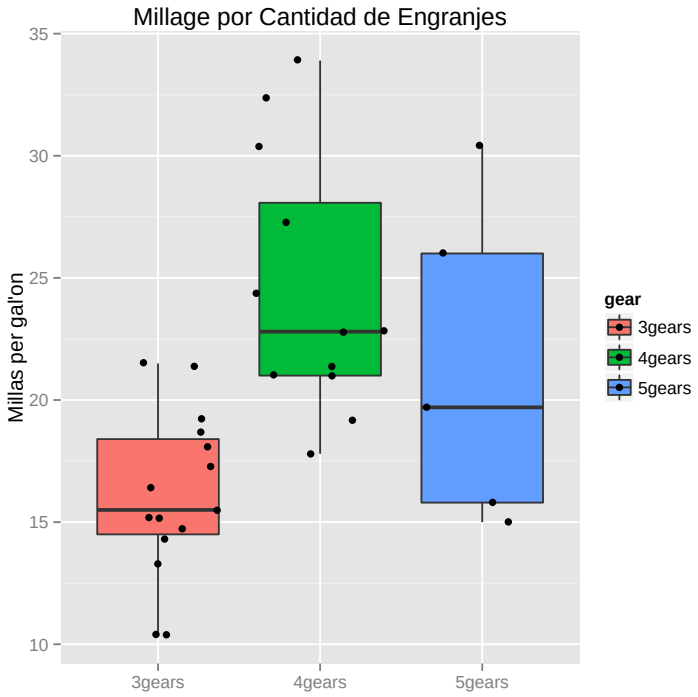
Se puede hacer distintas regresiones de millaje por galón contra el peso del automóvil, separándolas por la cantidad de cilindros que tiene.

```
> qplot(wt, mpg, data=mtcars, geom=c("point", "smooth"), method="lm",
    formula=y~ x, color=cyl, main="Regresion de MPG con peso", xlab="Peso",
     ylab="Millas por gal\'on")
```



Por último se muestra una gráfica de caja y brazos del millaje por galón por número de engranjes. Las observaciones se muestran vibradas para que aparezcan dispersas y con ello se observen más claramente.

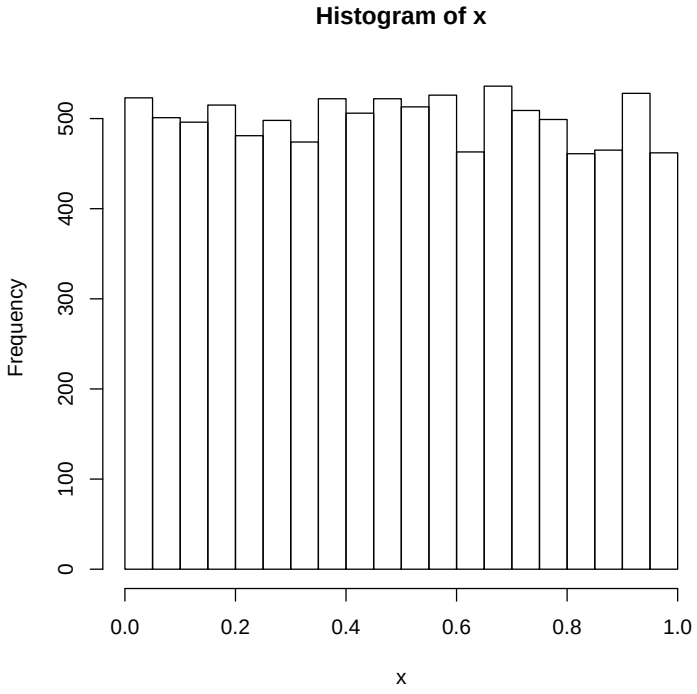
```
> qplot(gear, mpg, data=mtcars, geom=c("boxplot", "jitter"), fill=gear,
    main="Millage por Cantidad de Engranjes", xlab="", ylab="Millas per gal
    \'on")
```



6.2. Variables pseudo-aleatorias

Como una computadora no puede generar variables completamente aleatorias, se utilizan métodos pseudoaleatorios para generar tales números. En R ya vienen incluidos algoritmos que generan números aleatorios. La máquina utiliza semillas para comenzar los algoritmos. Estas semillas se toman de números de la máquina como por ejemplo la hora del día.

```
> x ← runif(10000) # se generan 1000 variables pseudo-aleatorias uniformes
> hist(x) # histograma de las variables aleatorias
```



Una manera común que se utiliza para generar variables pseudoaleatorias uniformes es usando la función módulo. A continuación se crea una función en R para generar números aleatorios. Se toman cuatro semillas a, c, M y el punto inicial X_0 para generar los números con la función

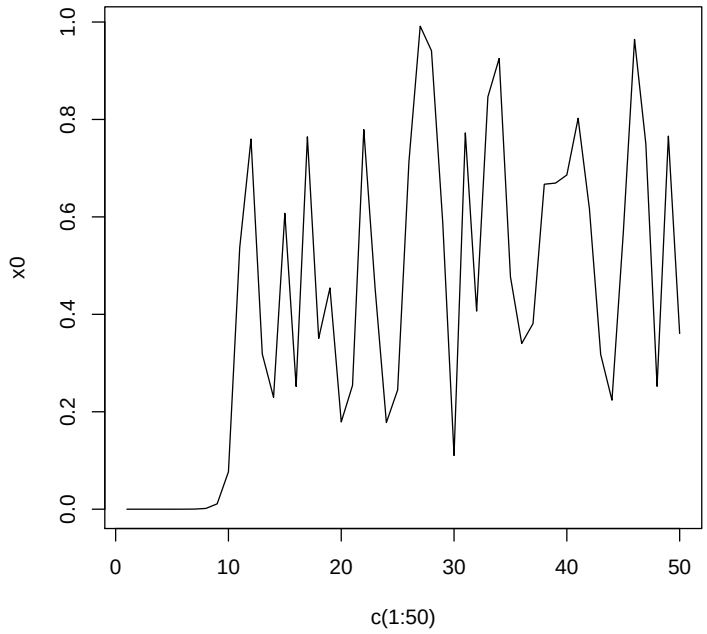
$$X_{i+1} = (aX_i + c) \mathrm{mod} M$$

pero como M puede ser un múmero muy grande, los números quedan entre cero y $M-1$ entonces se dividen entre $M-1$ para poder simular una distribución uniforme.

```
> m1 ← function(n,x,a,c,M){
+ for(i in 1:n){
+ x[i+1] ← (a*x[i]+c) %%M
+ }
+ x ← x/(M-1)
+ return(x[-1])
+ }
```

Con este método, si M es una semilla muy grande y a, c, X_0 muy chicos, los primeros números no serán pseudo-aleatorios ya que el módulo no entra en efecto para ellos.

```
> x0 ← m1(50,1e-83,7,7,2^ 32-1)
> plot(c(1:50),x0,type="l")
```



Para una buena selección de semillas, por lo general se escoge M un primo muy grande (el mas grande que tenga la máquina) y X_0 una semilla grande. Cuidando estos detalles, escogemos las semillas adecuadas y corremos las funciones mientras graficamos los histogramas.