Tutorial 03: Probabilidad.

Atención:

- Este documento pdf lleva adjuntos algunos de los ficheros de datos necesarios. Y está pensado para trabajar con él directamente en tu ordenador. Al usarlo en la pantalla, si es necesario, puedes aumentar alguna de las figuras para ver los detalles. Antes de imprimirlo, piensa si es necesario. Los árboles y nosotros te lo agradeceremos.
- Fecha: 16 de octubre de 2015. Si este fichero tiene más de un año, puede resultar obsoleto. Busca si existe una versión más reciente.

Índice

En la segunda parte del curso estamos aprendiendo el lenguaje matemático de la teoría de la Probabilidad que, ya en la tercera parte, nos va a resultar necesario para poder hacer Inferencia. En esta segunda parte del curso el nivel matemático se eleva, con la aparición de la Combinatoria, y de nociones del Cálculo, como las integrales. ¡Pero que nadie se asuste! En este tutorial vamos a tratar de poner los medios para aprender a calcular, de la forma más simple posible, lo que vamos a necesitar. Entre otras cosas, nos apoyaremos en el ordenador para evitarnos parte de los cálculos más difíciles.

1. Los problemas del Caballero de Méré.

Para poder hacer experimentos relacionado con la idea de probabilidad, necesitamos ser capaces de generar valores al azar. En los dos tutoriales previos, hemos avanzado algunas ideas de la forma en que podemos usar Calc y R para esa tarea. Recordemos brevemente lo que hemos hecho hasta ahora:

- En la página 19 del Tutorial-01 hemos aprendido a usar la función `ALEATORIO.ENTRE` de Calc.
- En la Sección 6.1 del Tutorial-02 (pág. 36) hemos descrito el uso básico de la función `sample` de R. Aquí vamos a profundizar en esa función.

- También, en esa misma sección, hemos aprendido a usar la función `set.seed` de R para que los resultados sean reproducibles. En Calc puede hacerse algo parecido con el *pegado especial*, pero es una herramienta mucho más incómoda que en R[1].
- Hemos insistido en la idea de que, en cualquier caso, los números que se generan con el ordenador son pseudoaleatorios (la propia existencia de `set.seed` es una prueba de esto). Pero podemos tranquilizar al lector: a todos los efectos, la diferencia entre estos números pseudoaleatorios y los números verdaderamente aleatorios es tan sutil, que no va a suponer ningún problema en este curso.

1.1. La función `ALEATORIO.ENTRE` de Calc y los problemas del Caballero de Méré.

La función `ALEATORIO.ENTRE` de Calc es suficiente para los primeros experimentos sencillos sobre la Regla de Laplace y, por ejemplo, los problemas del Caballero de Méré (ver la Sección 3.1, pág. 47 del libro). Para centrar la discusión, hemos aprendido que, por ejemplo,

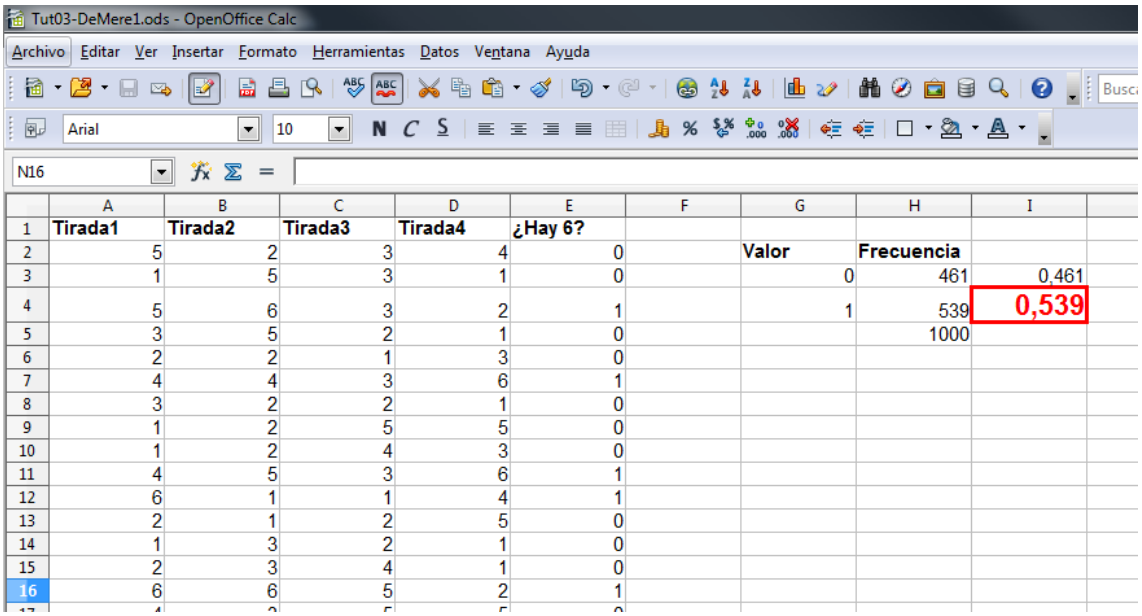
```
ALEATORIO.ENTRE(20;80)
```

produce un número (pseudo)aleatorio entre 20 y 80. Con el lenguaje que estamos desarrollando en el Capítulo 3, podemos añadir que todos los resultados entre 20 y 80 son equiprobables. Si queremos conseguir más de un número, debemos copiar esa fórmula en más celdas de la hoja de cálculo. Cada vez que abras o cierres la hoja de cálculo, los números generados con `ALEATORIO.ENTRE` cambiarán. Y si deseas generar nuevos valores, sin tener que cerrar y abrir la hoja de cálculo, puedes utilizar la combinación de teclas `Ctrl+Mayús+F9`.

Vamos a usar la función para ilustrar los problemas del Caballero de Méré que hemos descrito en la Sección 3.1 (pág. 47). Concretamente, hemos adjuntado a este documentos dos hojas de cálculo,

1. Cap03-DeMere1.ods (para la apuesta (a))
2. Cap03-DeMere2.ods (para la apuesta (b))

en las que hemos *simulado* esas dos apuestas, y hemos jugado 1000 veces cada una de ellas. La Figura 1 muestra el resultado al abrir el primero de esos ficheros. Tus números, al ser aleatorios, serán distintos de los nuestros, pero lo esencial de la discusión seguirá siendo válido. Las primeras cuatro



	A	B	C	D	E	F	G	H	I
1	Tirada1	Tirada2	Tirada3	Tirada4	¿Hay 6?				
2	5	2	3	4	0		Valor	Frecuencia	
3	1	5	3	1	0		0	461	0,461
4	5	6	3	2	1		1	539	0,539
5	3	5	2	1	0			1000	
6	2	2	1	3	0				
7	4	4	3	6	1				
8	3	2	2	1	0				
9	1	2	5	5	0				
10	1	2	4	3	0				
11	4	5	3	6	1				
12	6	1	1	4	1				
13	2	1	2	5	0				
14	1	3	2	1	0				
15	2	3	4	1	0				
16	6	6	5	2	1				

Figura 1: El fichero `Cap03-DeMere1.ods` de Calc

[1] Puedes ver los detalles en este enlace (en inglés):
https://wiki.openoffice.org/wiki/Documentation/How_Tos/Calc:_RAND_function

columnas de la hoja de cálculo (de la `A` a la `D`) se han obtenido usando la función `ALEATORIO.ENTRE`. Cada fila, por tanto se corresponde con una jugada, y si examinas el fichero verás que hemos jugado 1000 veces. Además, cada vez que pulses `Ctrl+Mayús+F9` obtendrás 1000 nuevas partidas de este juego. La columna `E` contiene un 1 si hemos obtenido (al menos) un 6 en las cuatro tiradas, y un 0 si no es así. No queremos entretenernos demasiado en la forma en la que hemos conseguido que Calc haga esto, porque haremos cosas más sofisticadas con R, de una manera muy sencilla, y merece la pena reservar nuestras fuerzas para ese empeño. Pero si sientes curiosidad, puedes hacer clic sobre la celda `E2`, y ver los comandos de Calc que hemos usado para conseguirlo:

```
=SI(O(A2=6;B2=6;C2=6;D2=6);1;0)
```

Esencialmente, lo que hemos dicho, en el lenguaje de Calc, es "Si alguna de las celdas `A2, B2, C2, D2` contiene un 6, entonces el resultado es 1. En caso contrario, es un 0". Y para eso hemos usado dos funciones de Calc, llamadas `SI` y `O`.

Las columnas `G, H` e `I` contienen el análisis de los resultados en forma de tabla de frecuencias y frecuencias relativas. Las frecuencias relativas son las cantidades que debemos comparar con las probabilidades calculadas de forma teórica, para saber si la teoría de las probabilidades que estamos aplicando es una descripción correcta del fenómeno. Y, en este ejemplo en particular, el resultado nos indica que la teoría del Caballero de Méré no está funcionando.

Recuerda que, según hemos visto en la teoría del curso, la ganancia que el Caballero de Méré esperaba era de un 66 % de lo invertido. Y lo que se observa es que la proporción de apuestas perdidas frente a apuestas ganadas es mucho menor de lo que el Cabllero esperaba.

En la Figura 2 puedes ver el comienzo del fichero correspondiente al segundo juego del Caballero de Méré. Sin entrar en muchos detalles (¡explora tú mismo el fichero!), las columnas de la izquierda contienen los resultados de las tiradas, que son números del 1 al 36, donde el 1 corresponde a $(1,1)$ y 36 corresponde a $(6,6)$. Y a la derecha aparece la tabla de frecuencias relativas, que muestra que el resultado es claramente distinto del que hemos obtenido en el otro juego. De hecho, la probabilidad de ganar en este segundo juego es aún más baja que en el otro (es aproximadamente igual a 0.49). Usa `Ctrl+Mayús+F9` unas cuantas veces para comprobarlo. Y recuerda que el Caballero de Méré creía que la probabilidad de ganar era la misma en ambos juegos. En la página 81 del Capítulo 3

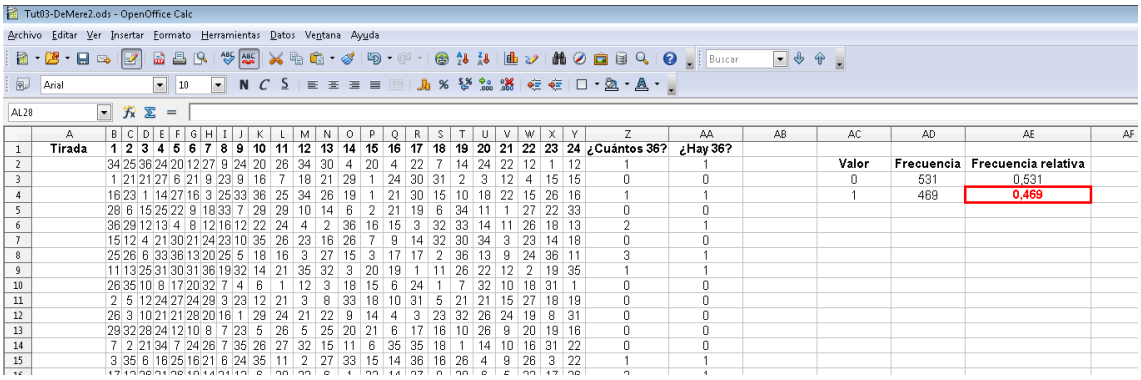


Figura 2: El fichero `Cap03-DeMere2.ods` de Calc

del libro se explica cómo calcular las probabilidades correctas para ambos juegos y, más adelante en este tutorial, daremos los detalles computacionales necesarios.

1.2. Probabilidades y la función `sample` de R.

Vamos a empezar recordando algo de lo que hemos aprendido sobre la función `sample`. Si queremos fabricar 100 números (pseudo)aleatorios entre 20 y 80, usaríamos (como hemos visto en la Sección 6.1 del Tutorial-02) un código similar a este:

```
set.seed(2014)
n=100
(datos = sample(20:80, n, replace=TRUE))
```

```
##   [1] 37 30 58 38 53 25 75 56 26 29 58 23 56 55 59 44 61 23 80 32 72 61 51
##  [24] 70 59 79 46 50 34 80 70 42 46 22 78 60 53 64 65 63 53 33 72 33 76 59
##  [47] 43 70 52 70 80 62 79 31 58 26 26 57 75 31 68 45 55 77 62 65 24 34 46
##  [70] 55 54 65 69 40 62 61 43 27 28 32 28 43 71 78 48 33 43 44 33 48 55 71
##  [93] 22 31 51 55 48 38 31 25
```

(Habrás observado que el formato del código R en este Tutorial es distinto, como ya advertimos al final del Tutorial02). De esta forma los resultados quedan guardados en el vector `datos`. Los valores que hemos obtenido son reproducibles (puedes eliminar o comentar la línea con `set.seed` para evitar esto). Y además, todos los números entre 20 y 80 son equiprobables. Y, puesto que estamos hablando de Probabilidad, es en este último aspecto en el que nos vamos a concentrar.

¿Cómo podemos fabricar valores que no sean equiprobables? En el Tutorial-02 nos las ingeniamos para conseguirlo por el procedimiento de aplicar `sample` a vectores en los que había elementos repetidos. Recuerda el ejemplo:

```
muchosUnos = c(1,1,1,1,1,1,1,1,1,2)
(muestra = sample(muchosUnos, size=100, replace=TRUE) )
```

```
##   [1] 1 1 1 1 1 1 1 1 1 2 1 1 1 2 1 1 1 1 1 1 1 2 1 1 1 2 1 1 1 1 2 1 1 1 1
##  [36] 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 2 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1
##  [71] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 2 1 1 1 1 2
```

Este método, combinado con la función `rep`, nos podría servir para conseguir lo que queremos. Pero no resulta una forma cómoda de trabajar, en cuanto las cosas sean un poco más complicadas.

Ejercicio 1.
Una caja contiene 100 *fichas, idénticas salvo por el número que aparece en ellas. De ellas,* 35 *fichas están marcadas con el número* 1, 15 *con el número* 2, 10 *con el número* 3, 10 *con el número* 4 *y el resto con el número* 5. *Queremos extraer 20 fichas de la caja:*

(a) con reemplazamiento.

(b) sin reemplazamiento.

1. *Escribe el código en R que permite obtener esas* 20 *fichas, usando las funciones* `rep` *y* `sample`. *Los resultados, para los casos (a) y (b), se guardaran en dos vectores, llamados* `sorteo1` *y* `sorteo2`, *respectivamente. Utiliza* `set.seed(2014)` *como primera línea de tu código, para poder comparar tus resultados con los nuestros.*
2. *Obtén las tablas de frecuencias absolutas y relativas de ambos vectores.*

Solución en la página 37. □

Pero la función `sample` nos permite conseguir eso mismo de una manera mucho más sencilla. Podemos añadir a `sample` un argumento opcional `prob`, que es un vector con las probabilidades de cada uno de los elementos entre los que elegimos. Por ejemplo, en el vector `muchosUnos`, teníamos 9 repeticiones de 1 por una sola de 2. Eso significa que la probabilidad de que, al elegir un elemento al azar, sea un 1, es

$$p_1 = \frac{9}{10},$$

mientras que la probabilidad de que sea un 2 es:

$$p_2 = \frac{1}{10}.$$

Teniendo estas probabilidades en cuenta, para elegir 100 elementos al azar podemos hacer, como antes:

```
set.seed(2014)
(muestra = sample(muchosUnos, size=100, replace=TRUE) )
```

```
##   [1] 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 2 1 1 1 2 1 1 1 1 2
##  [36] 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 2 1 2 1 1 1 1 1 2 1 1 1 1 2 1 1 1 1 1 1
##  [71] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
which(muestra==2)
```

```
##  [1]  7 19 26 30 35 45 51 53 59 64 84
```

Hemos incluido la función `which`, para que puedas ver las posiciones que ocupan los doses en este vector. La nueva forma de conseguir esto que vamos a aprender consiste en partir de un vector con una única aparición del 1 y el 2, sobre el que vamos a usar directamente el argumento `prob` de `sample`. Así:

```
set.seed(2014)
(muestra = sample(c(1,2), size=100, replace=TRUE, prob=c(9/10, 1/10) ) )
```

```
##   [1] 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 2 1 1 1 2 1 1 1 1 2
##  [36] 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 2 1 2 1 1 1 1 1 2 1 1 1 1 2 1 1 1 1 1 1
##  [71] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
which(muestra==2)
```

```
##  [1]  7 19 26 30 35 45 51 53 59 64 84
```

Como ves, `which` nos confirma que las posiciones son las mismas en los dos casos, así que los métodos son equivalentes. (Recuerda siempre que `sample` trabaja con las posiciones del vector, y no con sus contenidos.)

1.2.1. De Méré en R, primera visita.

Para cerrar este apartado y preparar la discusión del siguiente, vamos a pedirle al lector que vuelva un momento a pensar en la Figura 1 (pág. 2). La tabla del fichero `Cap03-DeMere1.ods` de Calc representa el resultado de 1000 partidas del primer juego del Caballero de Méré. En ese juego, nosotros agrupamos cuatro tiradas del dado y las llamamos *una partida* del juego. Pero, en el fondo, esa tabla significa que hemos tirado el dado $4 \cdot 1000$ veces en total. Y eso es algo que ya sabemos hacer en R, usando la función `sample`:

```
set.seed(2014)
dado4000 = sample(1:6, 4000, replace=TRUE)
```

Naturalmente, al hacer esto, R fabrica un *vector* de longitud 4000, en el que no resulta evidente la estructura en *partidas* del juego. Los matemáticos se encuentran a menudo con esta situación en la que tenemos un vector de números con alguna estructura adicional. Y han inventado el lenguaje de las matrices para poder trabajar con algunas de esas situaciones. Por ejemplo, los primeros ocho números del vector `dado4000` son:

```
head(dado4000, 8)
```

```
## [1] 2 2 4 2 4 1 6 4
```

pero si queremos verlos como las primeras dos partidas que hemos jugado, lo mejor es colocarlos en forma de matriz, así:

$$\begin{pmatrix} 2 & 2 & 4 & 2 \\ 4 & 1 & 6 & 4 \end{pmatrix}$$

Este objeto es una matriz con dos filas y cuatro columnas. En R podemos hacer algo muy parecido, convirtiendo un vector en una matriz. Sólo tenemos que decirle a R cuál es el vector, cuántas filas o columnas tiene la matriz resultante, y cuál es el orden en que se colocan los elementos (por filas o por columnas). Para fabricar la matriz anterior hacemos:

```
(dosPartidas = matrix( head(dado4000, 8), ncol=4, byrow=TRUE))
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    2    2    4    2
## [2,]    4    1    6    4
```

```
class(dosPartidas)
```

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

La opción `ncol` es el número columnas, mientras que `byrow` debe su nombre al término inglés para fila, que es *row*. Hemos usado la función `class` para que veas que el objeto `dosPartidas` es de tipo `matrix`, una nueva clase de objetos de R. Fíjate también en la forma en la que R muestra esa matriz, indicando entre corchetes el número de fila (a la izquierda) y el número de columna (en la parte superior). En la próxima sección vamos a aprender el manejo básico de las matrices en R, y después retomaremos la discusión de los problemas del Caballero de Méré en R. Despedimos esta sección con un ejercicio.

Ejercicio 2.

- *¿Qué sucede si ejecutas el mismo código pero con* `byrow=FALSE`*?*
- *¿Cuál es el valor* por defecto *de* `byrow`*? Es decir, ¿qué sucede si no incluyes un valor de* `byrow`*?*

Solución en la página 38. Recuerda que el ejercicio te resultará mucho más útil si te esfuerzas en resolverlo, sin mirar la solución demasiado pronto. □

2. Matrices en R, un primer contacto

Las matrices son una de las formas que utiliza R para almacenar y organizar la información. A este tipo de objetos (vectores, matrices, tablas, etc.) los informáticos los conocen como Estructuras de Datos. Hemos presentado el primer ejemplo de matriz tratando de hacer visible el hecho de que la diferencia entre un vector y una matriz es, precisamente, una cuestión de *estructura*.

Para trabajar sobre un ejemplo concreto, vamos a colocar los números del 1 al 36 en una matriz de R, llamada `M`, de 4 filas y 9 columnas (diremos que es una matriz 4×9). El código es este:

```
(M = matrix(1:36, nrow=4) )
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
## [1,]    1    5    9   13   17   21   25   29   33
## [2,]    2    6   10   14   18   22   26   30   34
## [3,]    3    7   11   15   19   23   27   31   35
## [4,]    4    8   12   16   20   24   28   32   36
```

Como ya hemos visto antes, R rellena la matriz columna por columna.

Ejercicio 3. *Cambia el código para que la matriz M se rellene fila por fila. Asegúrate, antes de seguir, de que la primera fila de la matriz M la forman los números 1 a 9.* *Solución en la página 38.* □

¡No sigas, si no has hecho este ejercicio!

El resultado de ese ejercicio es que ahora la matriz `M` es así:

```
M
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
## [1,]    1    2    3    4    5    6    7    8    9
## [2,]   10   11   12   13   14   15   16   17   18
## [3,]   19   20   21   22   23   24   25   26   27
## [4,]   28   29   30   31   32   33   34   35   36
```

En R podemos cambiar las dimensiones de una matriz ya definida, usando la función `dim`. Por ejemplo:

```
dim(M)=c(3,12)
M
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
## [1,]    1   28   20   12    4   31   23   15    7    34    26    18
## [2,]   10    2   29   21   13    5   32   24   16     8    35    27
## [3,]   19   11    3   30   22   14    6   33   25    17     9    36
```

El resultado demuestra que R ha redistribuido los 24 elementos de la matriz en 3 filas y 12 columnas, como le habíamos pedido. Dos observaciones sobre ese proceso:

- Para redistribuir los elementos R ha trabajado, de nuevo, por columnas (es su modo por defecto, recuérdalo). Es decir, que la primera columna de la matriz redimensionada la forman los tres primeros elementos de la primera columna de M (antes de la redimensión), que eran el $1, 10$ y el 19. Si avanzas así, por columnas, es fácil ver el esquema que R ha usado para redistribuir los números (y si miras las diagonales verás los restos del orden original).
- Cuidado: aunque el procedimiento es reversible, al hacer esto hemos cambiado la *estructura* de la matriz original.

Ejercicio 4. *Devuelve la matriz M a la forma anterior, en la que la primera fila la forman los números* 1 *a* 9. *Solución en la página 39.* □

¡No sigas, si no has hecho este ejercicio!

Además de cambiar la dimensión, la función `dim` sirve para *obtener* esa dimensión (en forma de vector de dos elementos, número de filas y columnas respectivamente.) Por ejemplo (si has hecho el Ejercicio 3):

```
dim(M)
```

```
## [1] 4 9
```

Ejercicio 5. *¿Cuanto vale la dimensión de un vector? Define un vector con* 10 *elementos (por ejemplo,* `1:10`*) y calcula su dimensión.* *Solución en la página 39.* □

Como ilustra el próximo ejercicio, las matrices no tienen por qué ser numéricas.

Ejercicio 6. *Construye una matriz* 4×3 *con las primeras 12 letras del alfabeto (recuerda el vector* `letters`*), de manera que la primera fila sea* `a b c`*. Comprueba su dimensión, y después cámbiala para obtener una matriz* 3×4. *Solución en la página 39.* □

2.1. Funciones matriciales y operaciones con matrices en R.

La función `dim` es sólo el primer ejemplo que hemos encontrado de una función matricial. Pero hay muchas otras. Por ejemplo, si queremos trasponer una matriz, cambiando filas por columnas, usaremos la función `t` (su nombre es la letra `t`). Al aplicarla a la matriz `M` se obtiene:

```
t(M)
```

```
##       [,1] [,2] [,3] [,4]
##  [1,]    1   10   19   28
##  [2,]    2   11   20   29
##  [3,]    3   12   21   30
##  [4,]    4   13   22   31
##  [5,]    5   14   23   32
##  [6,]    6   15   24   33
##  [7,]    7   16   25   34
##  [8,]    8   17   26   35
##  [9,]    9   18   27   36
```

Al trasponer una matriz $m \times n$ se obtiene una matriz $n \times m$. Pero fíjate en que el efecto de trasponer una matriz $m \times n$ no es lo mismo que redimensionar la matriz para que sus dimensiones sean $n \times m$. El siguiente ejercicio lo demuestra.

Ejercicio 7. *Dada la matriz*

```
(W = matrix(1:15, nrow=3, byrow=TRUE))
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    2    3    4    5
## [2,]    6    7    8    9   10
## [3,]   11   12   13   14   15
```

comprueba que esta operación:

```
t(W)
```

y esta

```
dim(W) = c(5,3)
```

no dan el mismo resultado. Solución en la página 40. □

Otras dos funciones matriciales útiles son `nrow` y `ncol`, que proporcionan, respectivamente, el número de filas y columnas de la matriz. Con la matriz `M`, por ejemplo:

```
nrow(M)
```

```
## [1] 4
```

```
ncol(M)
```

```
## [1] 9
```

Para las matrices numéricas existen, además, cuatro funciones que permiten hacer operaciones por sumas o por columnas. Por ejemplo, la función `rowSums` calcula la suma de cada una de las filas, mientras que `sumCols` hace lo propio con las columnas:

```
rowSums(M)

## [1]  45 126 207 288

colSums(M)

## [1] 58 62 66 70 74 78 82 86 90
```

Las funciones `rowMeans` y `colMeans` proporcionan las medias aritméticas por filas y por columnas, respectivamente.

Hay muchas otras operaciones matriciales y, más adelante, iremos conociendo algunas. En la Sección 2.6 (pág. 14) hemos añadido un apartado opcional con algo más de información, para aquellos lectores que estén algo más familiarizados con el álgebra matricial (multiplicación de matrices, determinantes, inversas y operaciones relacionadas con estas). Pero hay una estrategia general cuando queremos aplicar una función a todas las filas (o columnas) de una matriz, que consiste en utilizar la función `apply`. Esta función usa (al menos) tres argumentos: la matriz, un índice que puede valer 1 o 2 para indicar filas o columnas, respectivamente, y la función que deseamos aplicar. Por ejemplo, dada la matriz `M` que venimos usando, puedes usar `apply` para calcular las sumas de las filas y columnas, mediante estos comandos:

```
# Por filas:
apply(M, 1, sum)

## [1]  45 126 207 288

# Por columnas:
apply(M, 2, sum)

## [1] 58 62 66 70 74 78 82 86 90
```

Y puedes comprobar que estos son los mismos resultados que obtuvimos con `rowSums` y `colSums`, respectivamente.

Ejercicio 8.

1. *Calcula la cuasidesviación típica muestral de las filas de* `M`.
2. *Calcula la mediana de las columnas de* `M`.
3. *La cuasidesviación típica muestral y la mediana son números. Pero también hay funciones que devuelven vectores en lugar de números. ¿Qué sucede si usas* `apply` *para aplicar la función* `summary` *a las columnas de* `M`*?*

Soluciones en la página 40. □

2.2. Operaciones elemento a elemento.

Otra propiedad interesante de las matrices es que podemos operar con ellas de forma muy parecida a como hemos hecho con los vectores. Para disponer de un ejemplo, este código define un par de matrices `A` y `B`, ambas matrices aleatorias (construidas con sample) de dimensión 3×4.

```
     set.seed(2014)
(A = matrix(sample(-10:10, 12, replace=TRUE), nrow=3))

##      [,1] [,2] [,3] [,4]
## [1,]   -4   -4    9   -7
## [2,]   -7    1    2    3
## [3,]    3   -9   -8   -9
```

```
(B = matrix(sample(-10:10, 12, replace=TRUE), nrow=3))
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    2   -2   10    4
## [2,]    2    4   -6    0
## [3,]    3   -9    8    7
```

Y ahora podemos ilustrar algunas operaciones con estas matrices. Por ejemplo, podemos multiplicar `A` por 2:

```
2 * A
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   -8   -8   18  -14
## [2,]  -14    2    4    6
## [3,]    6  -18  -16  -18
```

Podemos elevar `B` al cuadrado:

```
B^2
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    4    4  100   16
## [2,]    4   16   36    0
## [3,]    9   81   64   49
```

Podemos sumar las matrices:

```
A + B
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   -2   -6   19   -3
## [2,]   -5    5   -4    3
## [3,]    6  -18    0   -2
```

Y también podemos multiplicarlas elemento a elemento.

```
    A * B
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   -8    8   90  -28
## [2,]  -14    4  -12    0
## [3,]    9   81  -64  -63
```

¡Pero cuidado! Lo que R ha hecho aquí es lo mismo que ha hecho en todas las operaciones anteriores: trabajar elemento a elemento, como hacíamos en los vectores. En particular el resultado **no es el producto de matrices** tal como se define habitualmente en Matemáticas. En la Sección 2.6 (pág. 14) veremos cómo se calcula el verdadero producto matricial.

Naturalmente, podemos aplicar una función (como, por poner un ejemplo, el arcotangente, `atan`), a todos los elementos de la matriz:

```
atan(A)
```

```
##       [,1]   [,2]  [,3]  [,4]
## [1,] -1.33 -1.326  1.46 -1.43
## [2,] -1.43  0.785  1.11  1.25
## [3,]  1.25 -1.460 -1.45 -1.46
```

Y no queremos cerrar este apartado sin comentar algo que luego nos será muy útil. Además de las operaciones aritméticas, también podemos hacer operaciones lógicas (booleanas; recuerda el Tutorial02, pág. 42). Si queremos saber qué elementos de la matriz `A` son mayores o iguales que 0, haremos:

```
A >= 0
```

```
##       [,1]  [,2]  [,3]  [,4]
## [1,] FALSE FALSE  TRUE FALSE
## [2,] FALSE  TRUE  TRUE  TRUE
## [3,]  TRUE FALSE FALSE FALSE
```

Y, como ves, la respuesta es una matriz de valores booleanos, con las respuestas a esa pregunta, calculados de nuevo *elemento a elemento*.

Ejercicio 9.

1. *Construye una matriz de la misma dimensión que* `B` *y con elementos* `TRUE/FALSE` *que permita saber si el correspondiente elemento de* `B` *son iguales a* 2.
2. *Cambia la dimensión de la matriz* `A` *para que sea una matriz* 2×6. *Trata de calcular ahora el producto* `A * B`. *¿Qué sucede?*

Soluciones en la página 41. □

2.3. Selección de elementos en las matrices de R.

La selección de elementos de una matriz de R usa técnicas parecidas a las que ya hemos aprendido en el caso de los vectores. Vamos a suponer, por ejemplo, que queremos seleccionar el elemento de la tercera fila, sexta columna de la matriz `M` que venimos usando. En notación matemática, si M es la matriz ese elemento sería $m_{3,6}$ (a menudo se usa una letra mayúscula para la matriz y la misma letra minúscula para sus elementos). Recuerda que la matriz era:

```
M
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
## [1,]    1    2    3    4    5    6    7    8    9
## [2,]   10   11   12   13   14   15   16   17   18
## [3,]   19   20   21   22   23   24   25   26   27
## [4,]   28   29   30   31   32   33   34   35   36
```

El elemento deseado se selecciona usando así los corchetes:

```
M[3, 6]
```

```
## [1] 24
```

¿Y si lo que queremos es seleccionar una fila entera? Por ejemplo, la segunda fila. El propio R, al mostrar la matriz `M`, nos da una pista sobre la forma de conseguir esto con la notación de corchetes. La forma de obtener la segunda fila es esta:

```
M[2, ]
```

```
## [1] 10 11 12 13 14 15 16 17 18
```

mientras que si lo que quieres es la cuarta columna, tendrás que usar:

```
M[ , 4]

## [1]  4 13 22 31
```

Esta notación con corchetes es, como ya vimos con los vectores, muy flexible. Por ejemplo, para seleccionar las columnas de la 5 a la 8 podemos hacer:

```
M[ , 5:8]

##      [,1] [,2] [,3] [,4]
## [1,]    5    6    7    8
## [2,]   14   15   16   17
## [3,]   23   24   25   26
## [4,]   32   33   34   35
```

Y si lo que quieres son las columnas pares, puedes probar con:

```
M[ , seq(from=2, to=ncol(M), by=2)]

##      [,1] [,2] [,3] [,4]
## [1,]    2    4    6    8
## [2,]   11   13   15   17
## [3,]   20   22   24   26
## [4,]   29   31   33   35
```

Fíjate en que, en este ejemplo, como sabemos el número de columnas podríamos haber escrito simplemente

```
     M[ , c(2,4,6,8)]

##      [,1] [,2] [,3] [,4]
## [1,]    2    4    6    8
## [2,]   11   13   15   17
## [3,]   20   22   24   26
## [4,]   29   31   33   35
```

y el resultado es el mismo. Pero el método anterior tiene la ventaja de que no depende del conocimiento previo de las dimensiones de la matriz. Es una buena práctica acostumbrarse a buscar soluciones tan generales como sea posible. Aparte de hacer avanzar por la senda de la sabiduría, esto te facilitaré reutilizar el mismo código en el futuro para otras tareas.

Naturalmente, es posible combinar la selección por filas y por columnas. Supongamos que queremos los elementos de la matriz `M` que ocupan la intersección de las filas 2 y 4 con las columnas 6 y 9. Los podemos obtener de esta manera:

```
M[ c(2, 4), c(6, 9)]

##      [,1] [,2]
## [1,]   15   18
## [2,]   33   36
```

2.4. Matrices por filas y columnas: `cbind` y `rbind`.

Otra manera de fabricar matrices a partir de vectores es utilizar las funciones `cbind` y `rbind` (donde r y c proceden de row (fila) y column (columna), respectivamente). Estas funciones nos permiten escribir las filas o columnas directamente como vectores. Por ejemplo, podemos fabricar una matriz `U` a partir de sus tres filas, definidas todas mediante vectores de la misma longitud (que en este caso, es 4), con el código:

```
(U = rbind(2:5, 1:4, c(-2,1,4,7)))
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    2    3    4    5
## [2,]    1    2    3    4
## [3,]   -2    1    4    7
```

Y para ilustrar la función `cbind` vamos a añadirle una columna más a la matriz `U`, escribiendo:

```
(U = cbind(U, c(11,17,-4)))
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    2    3    4    5   11
## [2,]    1    2    3    4   17
## [3,]   -2    1    4    7   -4
```

Ejercicio 10. *Dada la matriz*

```
     (B = matrix(1:100, nrow=10))
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
##  [1,]    1   11   21   31   41   51   61   71   81    91
##  [2,]    2   12   22   32   42   52   62   72   82    92
##  [3,]    3   13   23   33   43   53   63   73   83    93
##  [4,]    4   14   24   34   44   54   64   74   84    94
##  [5,]    5   15   25   35   45   55   65   75   85    95
##  [6,]    6   16   26   36   46   56   66   76   86    96
##  [7,]    7   17   27   37   47   57   67   77   87    97
##  [8,]    8   18   28   38   48   58   68   78   88    98
##  [9,]    9   19   29   39   49   59   69   79   89    99
## [10,]   10   20   30   40   50   60   70   80   90   100
```

añádele a la derecha una columna que contenga los cuadrados de los elementos de la primera columna de `B`*. ¿Cómo la añadirías a la izquierda? ¿Y en el medio (pongamos, entre la segunda y la tercera)? Solución en la página 41.* □

2.5. El camino de vuelta: de matrices a vectores. El valor `NULL`.

Hemos visto que la función `matrix` sirve para convertir un vector en una matriz. Pero a veces necesitaremos la transformación contraria. Afortunadamente, hay varias maneras de convertir una matriz en un vector. Para trabajar sobre un ejemplo concreto vamos a fabricar con la matriz `A`, que hemos usado en anteriores ejemplos, y que habíamos fabricado así:

```
set.seed(2014)
(A = matrix(sample(-10:10, 12, replace=TRUE), nrow=3))
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   -4   -4    9   -7
## [2,]   -7    1    2    3
## [3,]    3   -9   -8   -9
```

La forma más clara de convertirla en un vector es usando una función matricial llamada `as.vector`:

```
as.vector(A)
```

```
##  [1] -4 -7  3 -4  1 -9  9  2 -8 -7  3 -9
```

Fiel a sus costumbres, R trabaja por columnas, de manera que los primeros tres elementos del vector son los de la primera columna de `A`. Si queremos fabricar el vector avanzando por filas, podemos usar la función `t` para trasponer primero la matriz, y luego usar `as.vector`:

```
as.vector(t(A))
```

```
##  [1] -4 -4  9 -7 -7  1  2  3  3 -9 -8 -9
```

Hay otra manera muy fácil de convertir matrices en vectores, usando

```
     c(A)
```

```
##  [1] -4 -7  3 -4  1 -9  9  2 -8 -7  3 -9
```

El resultado es el mismo que con `as.vector`, así que podrías pensar que es mejor (por más breve) usar la función `c`. Pero a menudo, cuando escribimos programas, la claridad es más importante que la brevedad.

Finalmente, hay un método algo más esotérico para convertir una matriz en un vector, que consiste en "borrarle" sus dimensiones. Recuerda que para obtener las dimensiones de la matriz hacemos:

```
dim(A)
```

```
## [1] 3 4
```

Pero también podemos usar la función `dim` para *cambiar* esas dimensiones. Vamos a hacer esto, pero cambiando las dimensiones por el valor `NULL`. Este símbolo representa un valor especial R, como otros valores especiales que ya hemos encontrado: `NA, NaN, Inf`. El valor `NULL` se utiliza, principalmente, para "vaciar" una variable. Es una forma de decirle a R "olvídate del contenido de esta variable". Así que si hacemos

```
     dim(A) = NULL
```

le estamos diciendo a R "olvídate de que `A` tenía dimensiones". Y el resultado es que si ahora preguntamos por `A` obtenemos:

```
A
```

```
##  [1] -4 -7  3 -4  1 -9  9  2 -8 -7  3 -9
```

```
class(A)
```

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

Es decir, que `A` ha pasado a ser un vector de tipo `integer`. Los vectores, en R, no tienen dimensión. O, mejor dicho, su dimensión es `NULL`.

2.6. Introducción al álgebra matricial en R.

Esta sección es opcional.

Esta sección no pretende ser, en modo alguno, una introducción siquiera mínimamente completa al Álgebra Lineal con R, porque eso nos ocuparía mucho más espacio del que razonablemente podemos dedicarle en este tutorial. Pero sí queremos que sirva de aperitivo para aquellos lectores con conocimientos más avanzados sobre matrices.

Para empezar, en R, el producto matricial se representa con el operador `%*%`. Por ejemplo, si definimos las matrices:

```r
(U = matrix(1:8, nrow=2, byrow=TRUE))

##      [,1] [,2] [,3] [,4]
## [1,]    1    2    3    4
## [2,]    5    6    7    8

(V = matrix(rep(c(1,-1),12), nrow=4, byrow=TRUE))

##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1   -1    1   -1    1   -1
## [2,]    1   -1    1   -1    1   -1
## [3,]    1   -1    1   -1    1   -1
## [4,]    1   -1    1   -1    1   -1
```

En este caso, U es 2×4, mientras que V es 4×6, así que el producto es esta matriz 2×6:

```r
U %*% V

##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]   10  -10   10  -10   10  -10
## [2,]   26  -26   26  -26   26  -26
```

Naturalmente, si tratamos de hacer un producto no definido, R nos reprende con un error:

```r
V %*% U

## Error in V%*% U: argumentos no compatibles
```

Para matrices cuadradas, como la matriz `M`, podemos calcular el determinante:

```r
(K = matrix(c(3,1,7,3), nrow=2))

##      [,1] [,2]
## [1,]    3    7
## [2,]    1    3

det(K)

## [1] 2
```

Si la matriz no es cuadrada, se producirá un error. Y si, como ha sucedido en este ejemplo, el determinante es distinto de 0, entonces podemos usar la función `solve` para calcular la inversa de la matriz:

```r
solve(K)

##      [,1] [,2]
## [1,]  1.5 -3.5
## [2,] -0.5  1.5

K %*% solve(K)

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

Hemos incluido el producto matricial para que resulte evidente que, de hecho, hemos obtenido la inversa.

Como hemos dicho, apenas hemos arañado la superficie de lo que es posible hacer con R. Existen muchas más funciones (y paquetes) para calcular autovalores (ver la función `eigen()`, descomposiciones de Cholesky (ver la función `chol`), o en valores singulares (función `svd`), etc. El enlace

http://www.statmethods.net/advstats/matrix.html

contiene algo más de información sobre este tema.

3. Los juegos del Caballero de Méré en R

Ya va siendo hora de poner a trabajar a las matrices de R en un problema que nos interesa. Más adelante en el curso, cuando manejemos R con más soltura, conoceremos otros objetos (en especial las listas y los `data.frames`) que harán más fácil esta tarea. Pero por el momento nos las podemos arreglar con lo que hemos aprendido para obtener resultados sobre los juegos del Caballero de Méré muy parecidos a los que hemos obtenido con Calc.

3.1. El primer juego

Recuerda que ya habíamos empezado el trabajo, en la Sección 1.2.1 (pág. 5), en la que habíamos construido un vector de 4000 tiradas del dado, y descubrimos que podíamos agruparlas en partidas de cuatro tiradas cada una usando una matriz de cuatro columnas. El código necesario es:

```
set.seed(2014)
```

Al hacer esto, la matriz `deMere1` es una matriz 1000×4 que contiene, en cada fila, una partida del primer juego. Es decir, esa matriz contiene el equivalente a las primeras cuatro columnas del fichero `Cap03-DeMere1.ods` de Calc que aparecía en la Figura 1. Por ejemplo, las primeras 10 partidas son:

```
head(deMere1, 10)
```

```
##       [,1] [,2] [,3] [,4]
##  [1,]    2    2    4    2
##  [2,]    4    1    6    4
##  [3,]    1    1    4    1
##  [4,]    4    4    4    3
##  [5,]    5    1    6    2
##  [6,]    6    5    4    5
##  [7,]    4    6    3    3
##  [8,]    2    6    5    3
##  [9,]    3    1    6    5
## [10,]    4    5    5    5
```

Para saber si ha ganado en una partida concreta, el Caballero de Méré tiene que preguntarse si alguno de los cuatro números de esa partida es un 6. Ya hemos visto que eso se puede hacer en R de una manera muy fácil, con el código `deMere1 == 6`. Vamos a guardar el resultado en una matriz de valores booleanos, y le echaremos un vistazo a las 10 primeras filas de esa matriz.

```
     esSeis = (deMere1 == 6)
head(esSeis, 10)
```

```
##        [,1]  [,2]  [,3]  [,4]
##  [1,] FALSE FALSE FALSE FALSE
```

```
##  [2,] FALSE FALSE  TRUE FALSE
##  [3,] FALSE FALSE FALSE FALSE
##  [4,] FALSE FALSE FALSE FALSE
##  [5,] FALSE FALSE  TRUE FALSE
##  [6,]  TRUE FALSE FALSE FALSE
##  [7,] FALSE  TRUE FALSE FALSE
##  [8,] FALSE  TRUE FALSE FALSE
##  [9,] FALSE FALSE  TRUE FALSE
## [10,] FALSE FALSE FALSE FALSE
```

El siguiente paso es fácil de describir: tenemos que ver cuáles son las filas que contienen al menos un valor `TRUE`. El truco consiste en usar la equivalencia de los valores `TRUE` y `FALSE` con 1 y 0 que aprendimos en el Tutorial02 (ver la página 44). Vamos a fijarnos, por ejemplo, en la primera partida, en la que no hay ningún 6, y sumemos esa fila de la matriz `esSeis`:

```
esSeis[1, ]
```

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

```
## [1] 0
```

En cambio en la segunda partida hay un seis, y la suma de la segunda fila de `esSeis` así lo indica:

```
esSeis[2, ]
```

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

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

```
## [1] 602
```

En la partida número 602 el Caballero de Méré tuvo mucha suerte, y obtuvo 6 cuatro veces [2]:

```
esSeis[602, ]
```

```
## [1] TRUE TRUE TRUE TRUE
```

```
## [1] 4
```

Ejercicio 11. *¿Cómo hemos averiguado que en esa fila aparecían cuatro seises? Tienes todos los ingredientes para responder, pero si no lo ves claro sigue leyendo y luego vuelve a este ejercicio. Una pista: las funciones* `which` *y* `max` *ayudarán a obtener la respuesta, junto con el ingrediente que vamos a ver a continuación. Solución en la página 41.* □

[2] ¿Es un milagro? ¿Cómo de probable crees que es que suceda esto en 1000 partidas? Aprenderemos a contestar esa pregunta en el Capítulo 5 del libro, y veremos que es aproximadamente igual a 0.0008. No es, desde luego, un milagro...

Todo eso está muy bien, pero no podemos ir fila for fila sumando unos y ceros en cada una de las 1000 filas de la matriz `esSeis`. Afortunadamente, hay una función que nos ayuda a hacer exactamente lo que necesitamos: la función `rowSums`:

```
cuantosSeis = rowSums(esSeis)


##  [1] 0 1 0 0 1 1 1 1 1 0
```

El vector `cuantosSeis` nos dice cuántas veces se ha obtenido 6 en cada una de las 1000 partidas. Para saber las partidas ganadoras basta saber el número de elementos de `cuantosSeis` que son mayores que 0. A partir de ahí es muy fácil calcular la proporción de partidas ganadoras:

```
partidasGanadoras = (cuantosSeis > 0)


## partidasGanadoras
## FALSE  TRUE
##  0.48  0.52
```

Y, como puede verse, el resultado es coincidente con el que apuntaba Calc. Naturalmente, esto es sólo un ejemplo. Pero basta con volver a ejecutar el código (sin usar `set.seed`, desde luego) para obtener otro conjunto de partidas y seguir el experimento, constatando, como hicimos con Calc, que la probabilidad estimada por el Caballero de Méré era errónea. Para facilitar esa tarea, hemos agrupado el código en el fichero adjunto Tut-03-DeMere1.R, cuyo listado aparece en la Tabla 1. Prueba a ejecutarlo unas cuantas veces.

```
# Limpieza inicial
rm(list=ls())

# Generamos las 4000 tiradas del dado
dado4000 = sample(1:6, 4000, replace = TRUE)

# Los agrupamos en partidas cada cuatro tiradas.
deMere1 = matrix(dado4000, ncol = 4, byrow = TRUE)

# Localizamos las apariciones del 6.
esSeis = (deMere1 == 6)

# Contamos cuantas apariciones del seis hay en cada partida.
cuantosSeis = rowSums(esSeis)

# Localizamos aquellas partidas con al menos un 6.
partidasGanadoras = (cuantosSeis > 0)

# Y medimos la proporcion de partidas ganadoras.
(proporcion = table(partidasGanadoras)/length(partidasGanadoras))
```

Tabla 1: Comandos R para el primer juego del Caballero de Méré. Fichero `Tut-03-DeMere1.R`

3.2. El segundo juego

En lugar de hacer una descripción igual de detallada del segundo juego, vamos a proponer al lector que lo aborde mediante un ejercicio.

Ejercicio 12.

1. *Construye un vector* `dosDados24000` *con* $24 \cdot 1000$ *tiradas de un "dado" de 36 caras. El número* 36 *corresponde al* 6 *doble.*
2. *Conviértelo en una matriz, agrupando las tiradas por partidas (cada partida consta de* 24 *tiradas).*
3. *Identifica las partidas ganadoras.*

4. *Calcula la proporción de partidas ganadoras sobre el total de* 1000 *partidas.*

Solución en la página 42. □

3.2.1. Comprobando experimentalmente la regla de Laplace

Uno de nuestros primeros ejemplos usando la Regla de Laplace, al final de la Sección 3.2 del libro, se refiere al juego en el que tiramos dos veces un dado y queremos calcular la probabilidad del suceso

$$A = \textit{obtener al menos un seis en las dos tiradas}.$$

La probabilidad ingenua predice $\frac{12}{36} \approx 0.33$ para la probabilidad $P(A)$. En cambio, la regla de Laplace predice $\frac{11}{36}$. Para comprobarlo experimentalmente hemos incluido aquí una hoja de cálculo de Calc, llamada

Tut03-DeMere1a.ods

en la que se ha simulado ese lanzamiento. Recarga los valores unas cuantas veces (con `Ctrl+Mayús+F9`), para ver lo que sucede.

Ejercicio 13. *Escribe el código R necesario para comprobar estos resultados. Debería resultar muy fácil, ahora que hemos visto como hacerlo para los dos juegos del Caballero de Méré.* *Solución en la página 42.* □

4. Ejemplos de Probabilidad Geométrica con GeoGebra.

En la Sección 3.3 (pág. 51) hemos incluido varios ejemplos de un tipo de problemas que se denominan de Probabilidad Geométrica. Para visualizar esos problemas, ahora vamos a utilizar los primeros ficheros GeoGebra del curso.

4.1. Probabilidad geométrica. Método de Montecarlo.

4.1.1. Un punto al azar en el segmento $[0, 1]$.

Vamos a tratar de usar R para ilustrar las ideas que han aparecido en el Ejemplo 3.3.2 del libro (pág. 52). En ese ejemplo hemos tomado un número n_0 muy grande, por ejemplo $n_0 = 100000$, para definir $n_0 + 1$ puntos homogéneamente repartidos en el intervalo $[0, 1]$. Vamos a usar R para construir esos puntos. Mostramos los primeros puntos con `head` y los últimos con `tail`. Los últimos están tan cerca de 1 que R los redondea a 1 (al mostrarlos, pero no en las operaciones internas; ejecuta si quieres `puntos[n0-1]==1` para comprobarlo):

```
n0 = 100000
puntos = (0:n0)/n0
head(puntos)

## [1] 0.00000 0.00001 0.00002 0.00003 0.00004 0.00005

tail(puntos)

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

Ahora vamos a elegir

```
     k = 10000
```

de esos puntos, y vamos a ver que proporción de ellos pertenecen al intervalo $A = [2/3, 1]$. Usaremos ideas que ya conocemos, como la equivalencia de `TRUE` y `FALSE` con `1` y `0`, respectivamente.

```
     set.seed(2014)
elegidos = sample(puntos, k)
enA = (elegidos >= 2/3)
(proporcion = sum(enA) / k)

## [1] 0.329
```

La proporción 0.329, como ves, es cercana al valor $\frac{1}{3}$ =0.333 que pronosticábamos.

4.1.2. Lanzando dardos. El método de Montecarlo para calcular áreas.

Para ilustrar el Ejemplo 3.3.3 del libro (pág. 54), vamos a utilizar el fichero GeoGebra:

Cap03-MonteCarloAreaCirculo.ggb

La parte (a) de la Figura 3 muestra lo que verás cuando lo abras con GeoGebra. El cuadrado inicial, al que llamaremos C_1, es de lado 4, así que su área es 16. El interior del cuadrado contiene cinco puntos, porque el valor del deslizador inicialmente es $n = 5$. Arrastra el deslizador con el ratón para ir viendo como se generan más puntos en el interior del cuadrado. Por ejemplo, la parte (b) de la Figura 3 muestra un ejemplo en el que se han generado $n = 2800$ puntos. Si ahora marcas con el ratón la casilla rotulada *"Área del cuadrado pequeño"*, verás aparecer un cuadrado más pequeño, de color rojo y centrado en el cuadrado más grande, al que vamos a llamar C_2. El lado de este cuadrado más pequeño es 2, así que su área es 4. Pero vamos a *fingir* que no sabemos cuánto vale el área, y vamos a tratar de calcular ese valor lanzando dardos, y contando la proporción de dardos que hacen blanco en el cuadrado pequeño. La idea subyacente es que ese número de dardos que aciertan en C_2 es proporcional al área de C_2, así que tenemos:

$$\frac{\text{Dardos en } C_2}{\text{Total de dardos (en } C_1)} = \frac{\text{Área de } C_2}{\text{Área de } C_1}$$

En la Figura 4 se muestra ese proceso. Puedes ver que hemos lanzado $n = 3000$ dardos, y que la proporción de aciertos en C_2 es 0.265. Así que usando la ecuación anterior, y sabiendo que el área de C_1 es 16, estimamos que:

$$\text{Área de } C_1 \approx 16 \cdot 0.265 \approx 4.24$$

Que, sin ser impresionante, es una primera aproximación. Prueba a mover el deslizador para ver como, a medida que el número de puntos aumenta, las estimaciones son cada vez mejores. Puedes moverlos hasta $n = 5000$. Y una vez que llegues a ese valor, pulsa `Ctrl + R`. Cada vez que lo hagas, GeoGebra volverá a lanzar 5000 nuevos dardos, y podrás ver una nueva estimación del área de C_2 (también puedes mover el deslizador ligeramente para conseguir lo mismo).

Todo eso puede estar muy bien, pero el lector estará pensando que ya sabemos calcular (y de forma exacta) el área de un cuadrado. Al fin y al cabo de ahí hemos sacado el área 16 de C_1, para empezar. Es cierto. Pero lo interesante empieza ahora, el cuadrado era sólo calentamiento. Desmarca la casilla del cuadrado, y marca la del círculo. Vamos a llamar C_3 al círculo que aparece, y cuyo radio es 1. Imagínate de nuevo que no conocemos la fórmula para el área del círculo. El razonamiento es el mismo, y nos conduce a la relación:

$$\frac{\text{Dardos en } C_3}{\text{Total de dardos (en } C_1)} = \frac{\text{Área de } C_3}{\text{Área de } C_1 \text{ (es 16)}}$$

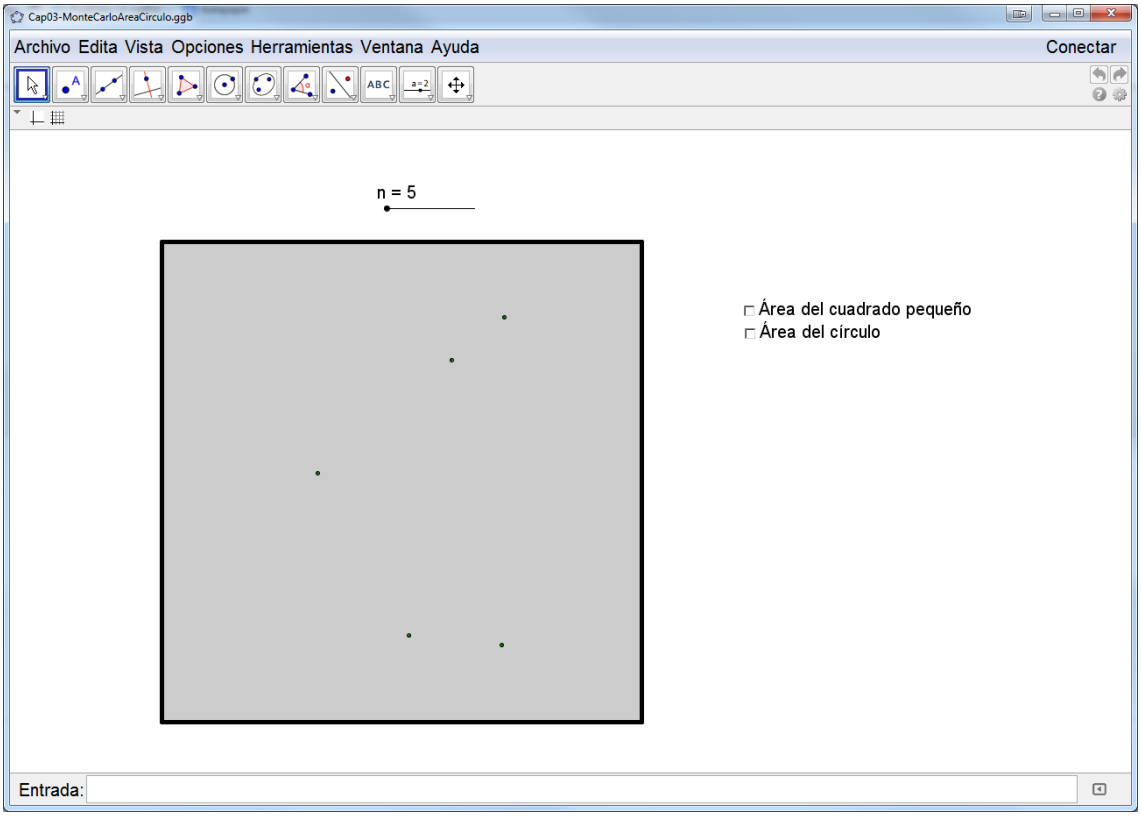
Por lo tanto,

$$\text{Área de } C_3 \approx 16 \cdot \frac{\text{Dardos en } C_3}{\text{Total de dardos (en } C_1)}.$$

Así que esto nos proporciona un procedimiento para aproximar el área del círculo (o de cualquier otra figura, por cierto) lanzando dardos. Eso empieza a parecer más interesante, ¿verdad?

Vuelve a repetir los pasos que dimos con el cuadrado, moviendo el deslizador hacia la derecha para ver como cambia la aproximación. Y cuando llegues a 5000 dardos, usa `Ctrl + R` para hacer varios

(a)



(b)

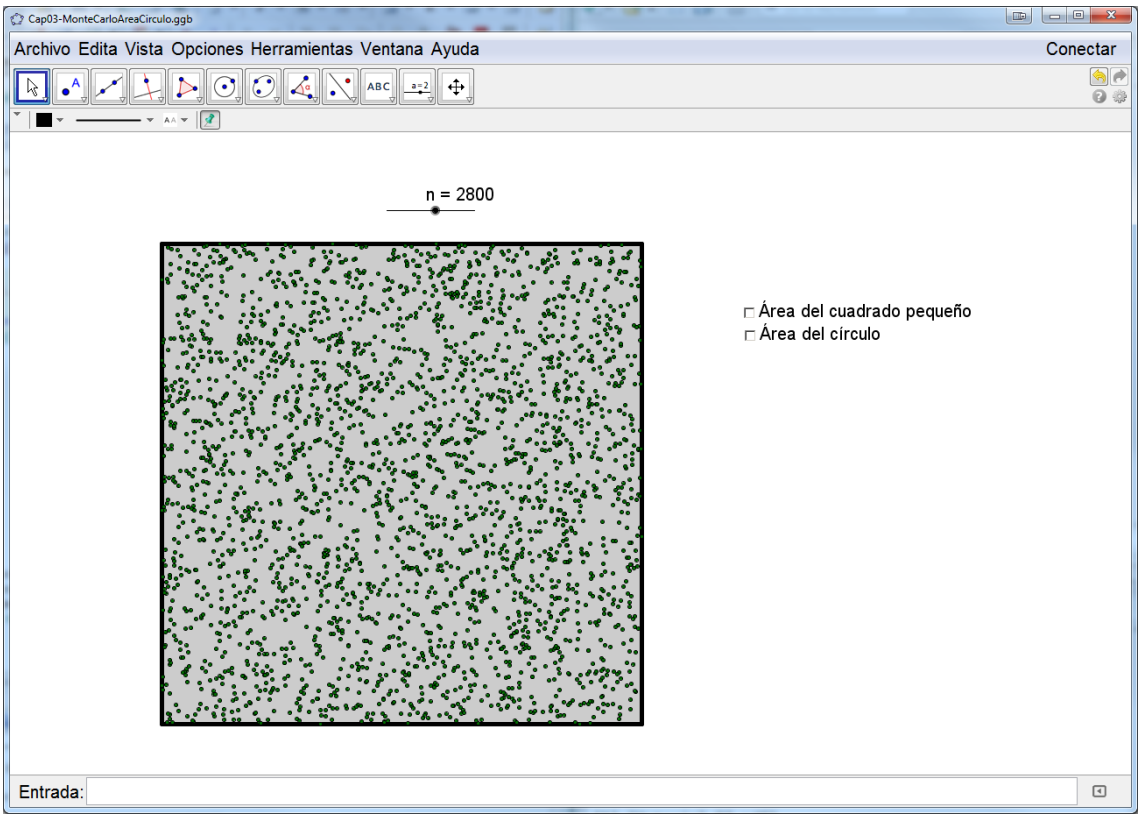


Figura 3: (a) Abriendo el fichero `Cap03-MonteCarloAreaCirculo.ggb` con GeoGebra (b) Más puntos al mover el deslizador...

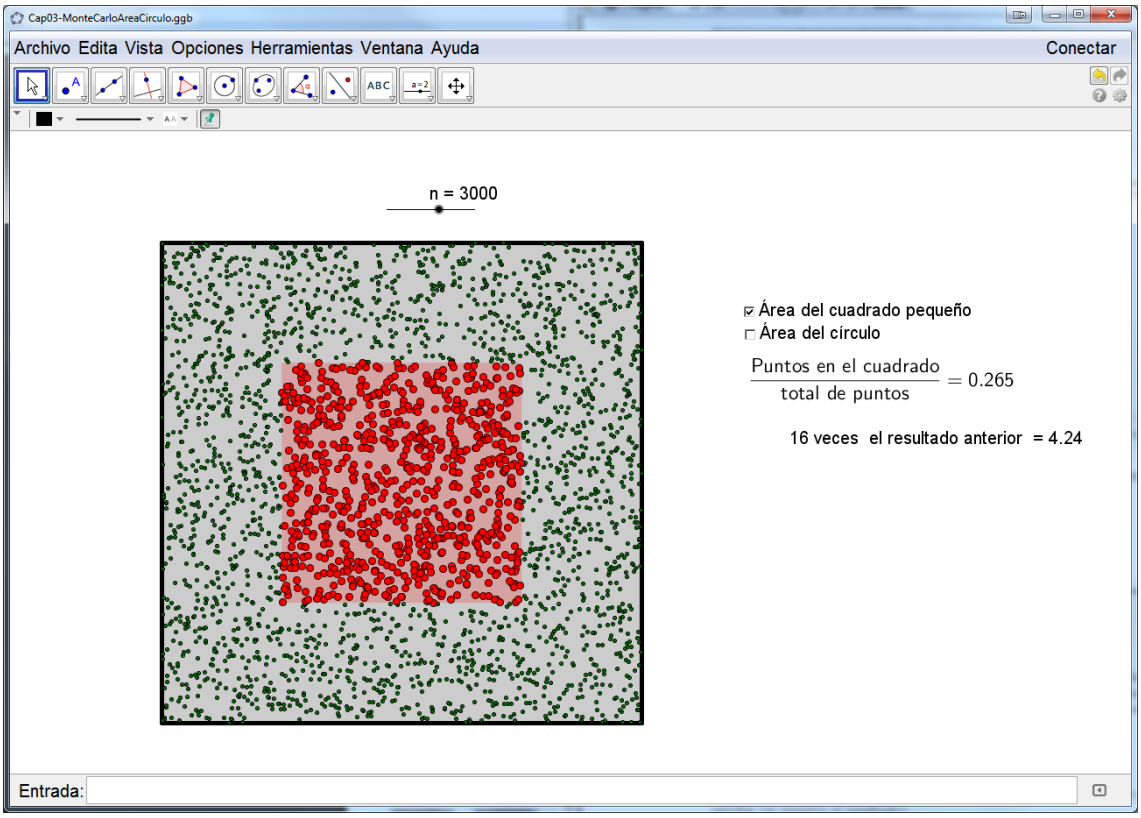


Figura 4: Lanzando dardos para calcular el área del cuadrado pequeño.

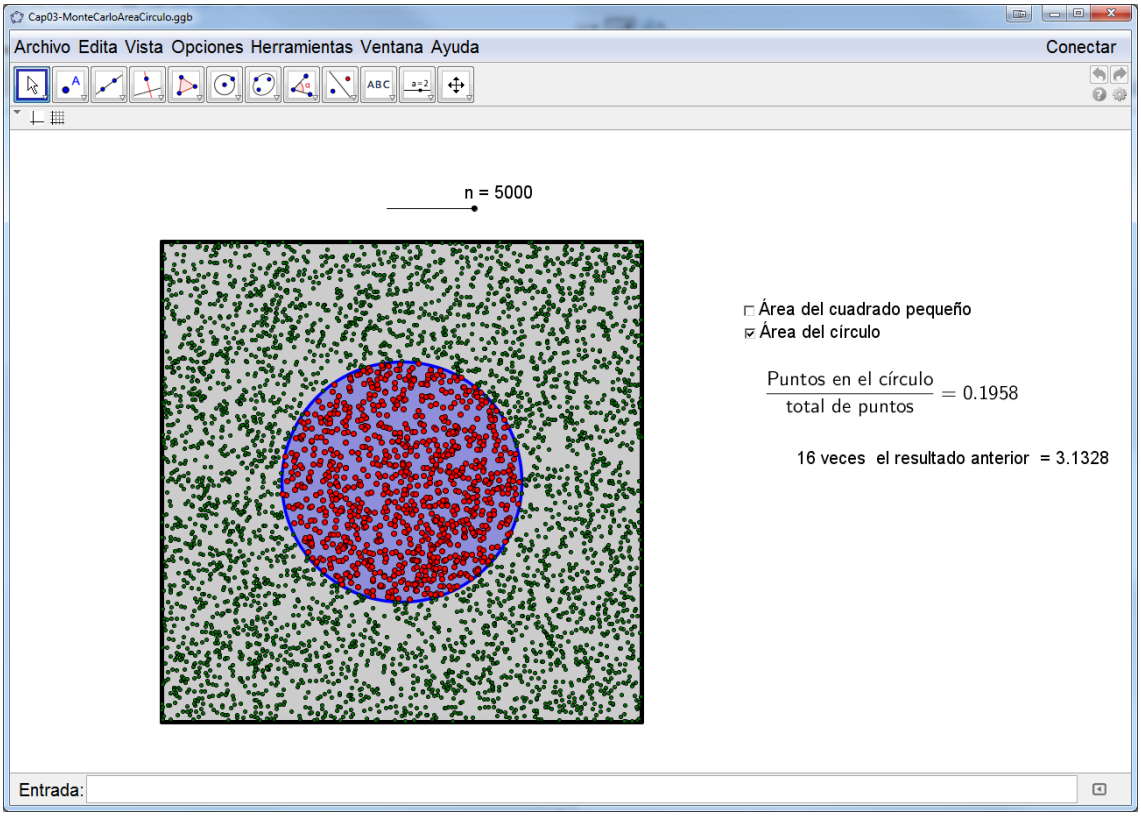


Figura 5: Lanzando dardos para calcular el área del círculo, que es precisamente π.

experimentos. Al cabo de unos cuantos intentos espero que te hayas convencido de que el área del círculo tiene un valor cercano a 3.1. El valor real, naturalmente, es π. Así que es posible calcular el

valor de π mientras juegas a los dardos (como ilustra la Figura 5)... siempre que estés dispuesto a jugar durante un buen rato, y no te esfuerces lo más mínimo en apuntar. Recuerda que lo esencial de estos experimentos es observar el vínculo que existe entre áreas y probabilidades. Cuanto mayor es el área de una figura, mayor es la probabilidad de acertarle con un dardo lanzado al azar.

5. Operaciones simbólicas. Wiris, Wolfram Alpha.

Cuando empezamos a trabajar con probabilidades, usando al principio la Regla de Laplace, y luego con la Combinatoria, a menudo nos surge la necesidad de operar con fracciones, como en el cálculo de esta fracción (que procede del Ejemplo 3.5.2, pág. 69, de nuestro curso):

$$\frac{\frac{3}{5}\cdot\frac{2}{6}}{\frac{3}{5}\cdot\frac{2}{6}+\frac{4}{5}\cdot\frac{4}{6}}=\frac{3}{11}$$

Si intentas hacer esta cuenta en R, para empezar tienes que ser cuidadoso con los paréntesis (se aplica la regla de "más vale que sobren"), y escribir:

```
( (3/5) * (2/6) ) /( (3/5) * (2/6) + (4/5) * (4/6) )

## [1] 0.273
```

Fíjate en los espacios que hemos dejado para hacer la expresión más legible. La respuesta 0.273 no es exactamente lo que queríamos. Para un problema como este, es muy posible que queramos ver el resultado en forma de fracción. El problema es que R es un programa esencialmente numérico, que trabaja con la representación de los números en forma decimal (podemos *obligarle*, hasta cierto punto, a trabajar con fracciones, pero no es su lenguaje natural). Mientras que 0.273 es una versión numérica de la respuesta, en el sentido de *redondeada a unas cuantas cifras significativas* y, por lo tanto, es una respuesta aproximada. En cambio la respuesta en forma de fracción

$$\frac{3}{11}$$

es una respuesta simbólica, y es absolutamente exacta: no hay ninguna pérdida de precisión. Usamos la dualidad *numerico/simbólico* en el sentido habitual en las matemáticas contemporáneas. En ese sentido, las cantidades

$$\sqrt{2}, \qquad \pi$$

son cantidades simbólicas, mientras que sus contrapartes numéricas (con cinco cifras significativas) son:

$$1.4142, \qquad 3.1416$$

Y conviene insistir en que R es un programa numérico, no simbólico. Por eso, para algunas operaciones del curso vamos a tener que recurrir a la ayuda de programas simbólicos. Veamos algunos.

5.1. Wiris.

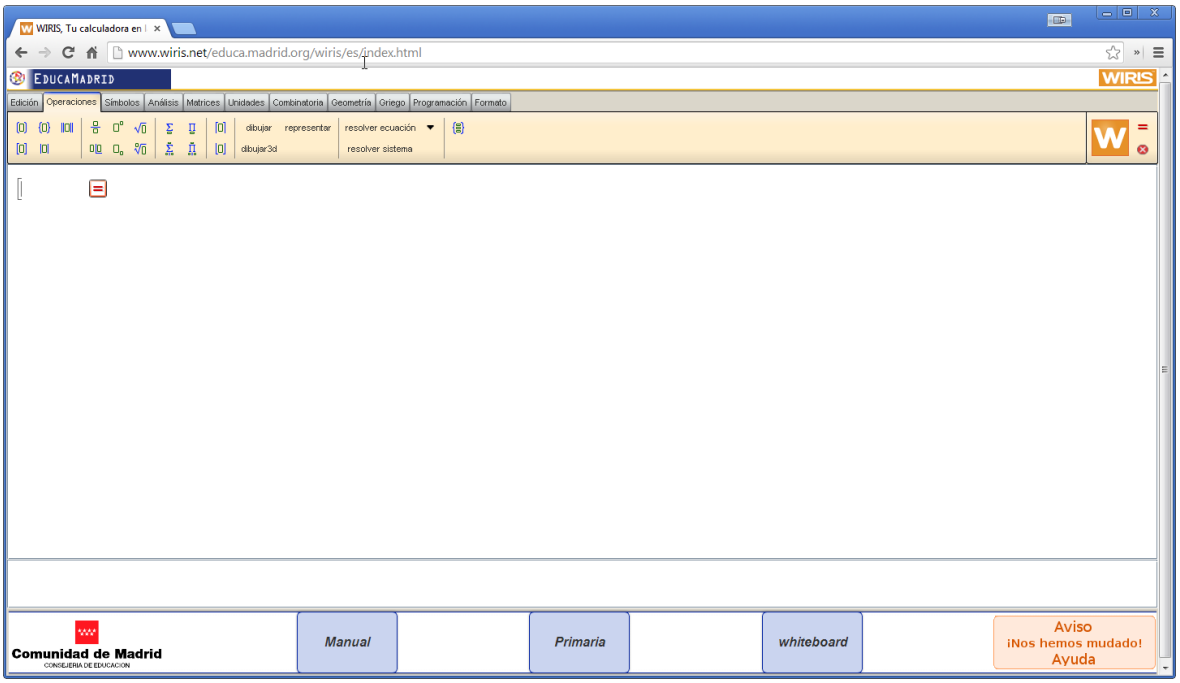
El programa Wiris CAS (de Computer Algebra System) es una creación de la empresa de software matemático *Maths for More*, con sede en Barcelona, y fundada por profesores y antiguos estudiantes de la Universitat Politècnica de Catalunya. Para utilizar el programa debemos estar conectados a internet. El programa se usa en el navegador, a través de la Web, en la página de la propia empresa (que incluye publicidad insertada en la página):

http://www.wiris.net/demo/wiris/es/index.html

o a través de las páginas Wiris que algunas Consejerías de Educación de distintas comunidades autónomas españolas ponen a libre disposición del público (tras llegar a acuerdos con la empresa, claro). Aquí tienes, por ejemplo, el enlace de la Comunidad de Madrid:

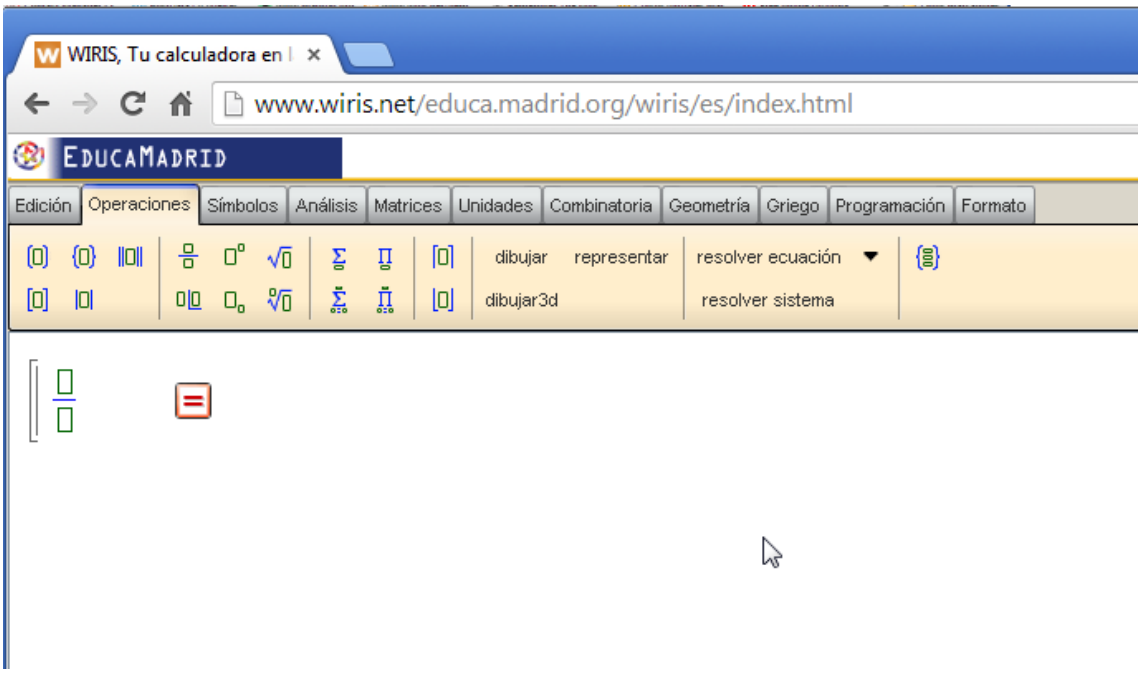
http://www.wiris.net/educa.madrid.org/wiris/es/index.html

Usemos esta última. Al abrirla (es necesario tener instalado Java en el ordenador; si no sabes si lo tienes, o si puedes o debes instalarlo, consulta a un ninja informático), el aspecto es este:

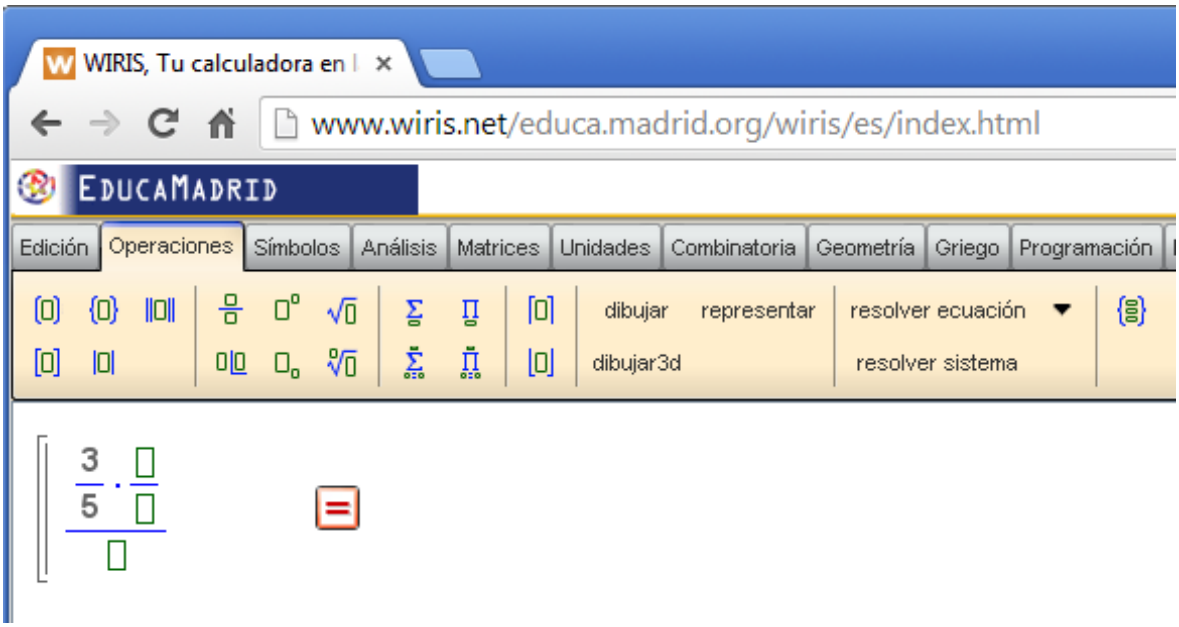


Wiris CAS nos permite escribir fórmulas matemáticas usando una notación muy parecida a la que usaríamos en el papel o la pizarra. Los símbolos y operaciones matemáticas están organizados por pestañas, pero para este primer ejemplo tan sencillo, todo lo que necesitamos está en la pestaña operaciones. Busca en ella el símbolo de fracción, que es el icono:

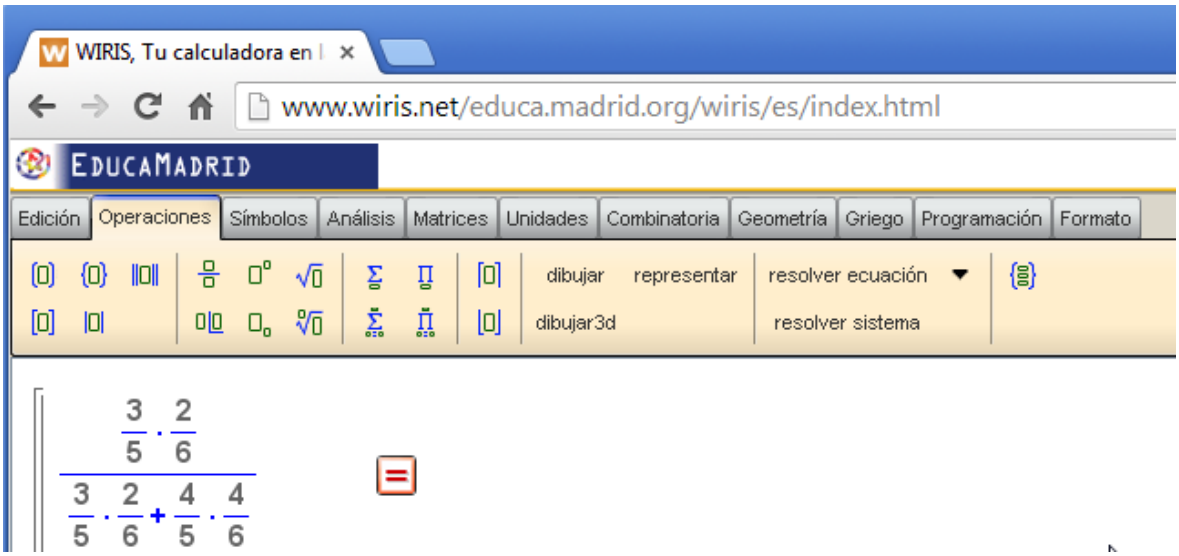
Púlsalo una vez con el ratón, y obtendrás



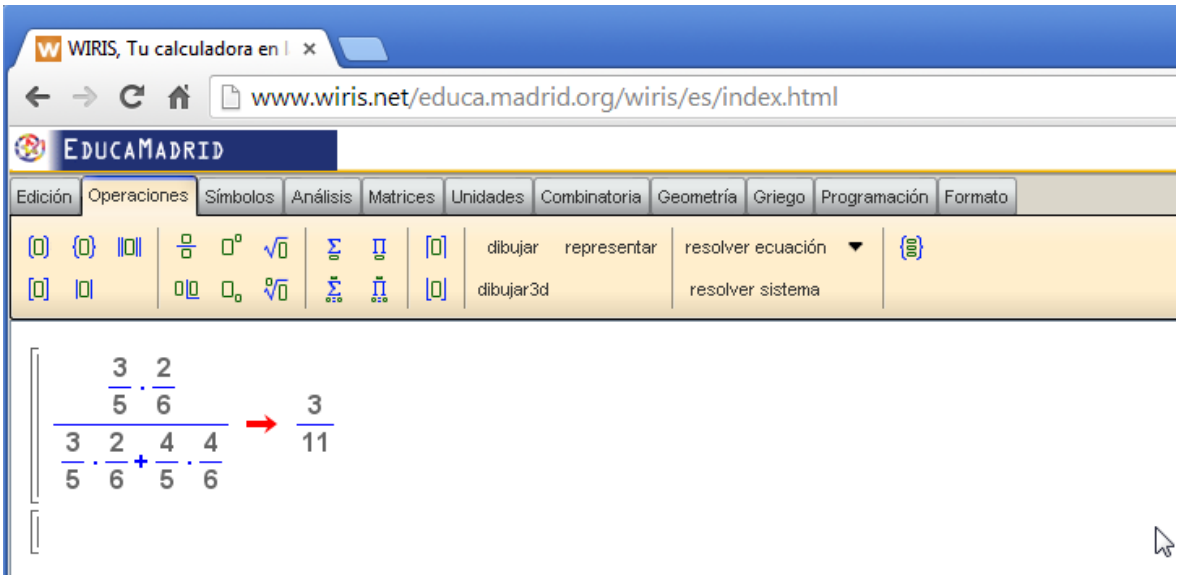
A partir de aquí las cosas son bastante intuitivas. Tienes que usar las teclas + para la suma, y * para la multiplicación, y usar el icono de fracción cada vez que quieras crear una nueva fracción. Tus primeros pasos te pueden llevar a algo como esto:



Ten en cuenta, para avanzar más rápido, que puedes seleccionar trozos de la fórmula con el ratón, y copiarlos y pegarlos. También, que en la pestaña `Edición` tienes dos iconos en forma de flechas curvas, para deshacer y rehacer operaciones. Debes llegar a:



Y ahora viene lo bueno. Una vez que estés ahí, pulsa sobre el icono *igual* que hay a la derecha de la fórmula (o, con el cursor situado en la fórmula, pulsa `Ctrl+Enter`). Al cabo de unos instantes (tu fórmula viaja por la red, se calcula, y la respuesta vuela de vuelta a tu ordenador), tendrás la respuesta:



Y Wiris CAS está listo para nuestra siguiente pregunta. Como ves, la respuesta es simbólica, no numérica, como queríamos. El inconveniente que encontramos es que la respuesta es realmente una imagen en la pantalla, no un número que podamos cortar y pegar para llevar a otro programa.

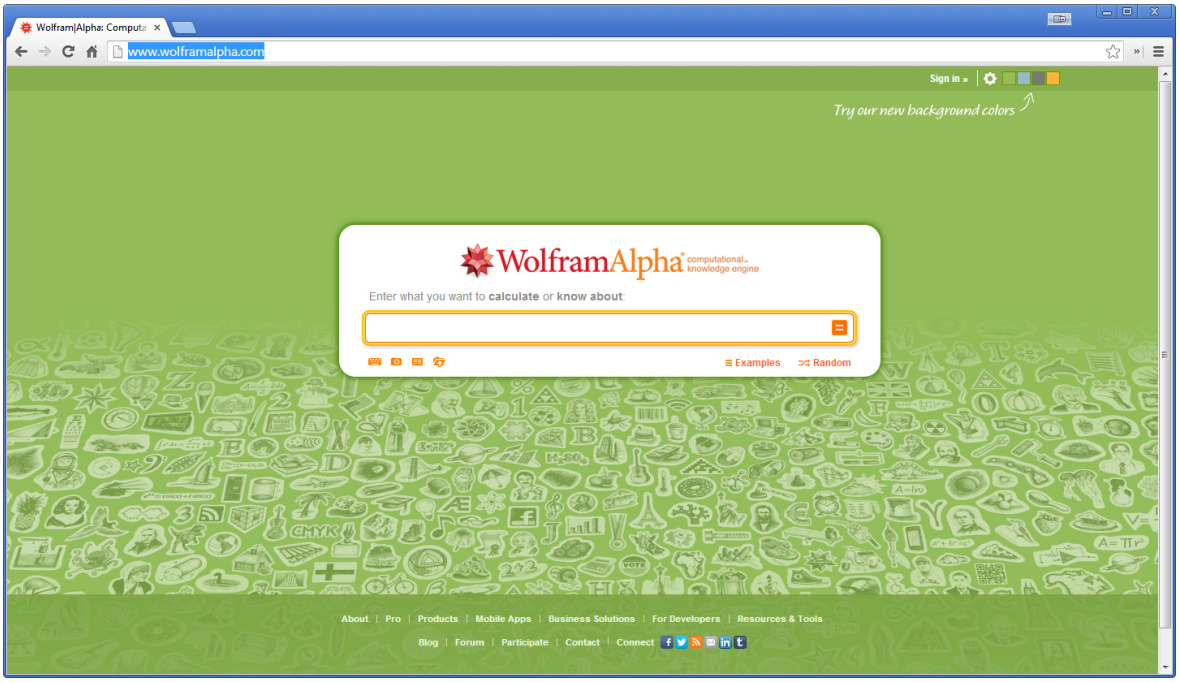
El cálculo de este ejemplo ha sido muy sencillo, pero a lo largo del curso iremos viendo más ejemplos en los que Wiris CAS nos puede ser de gran ayuda.

5.2. Wolfram Alpha.

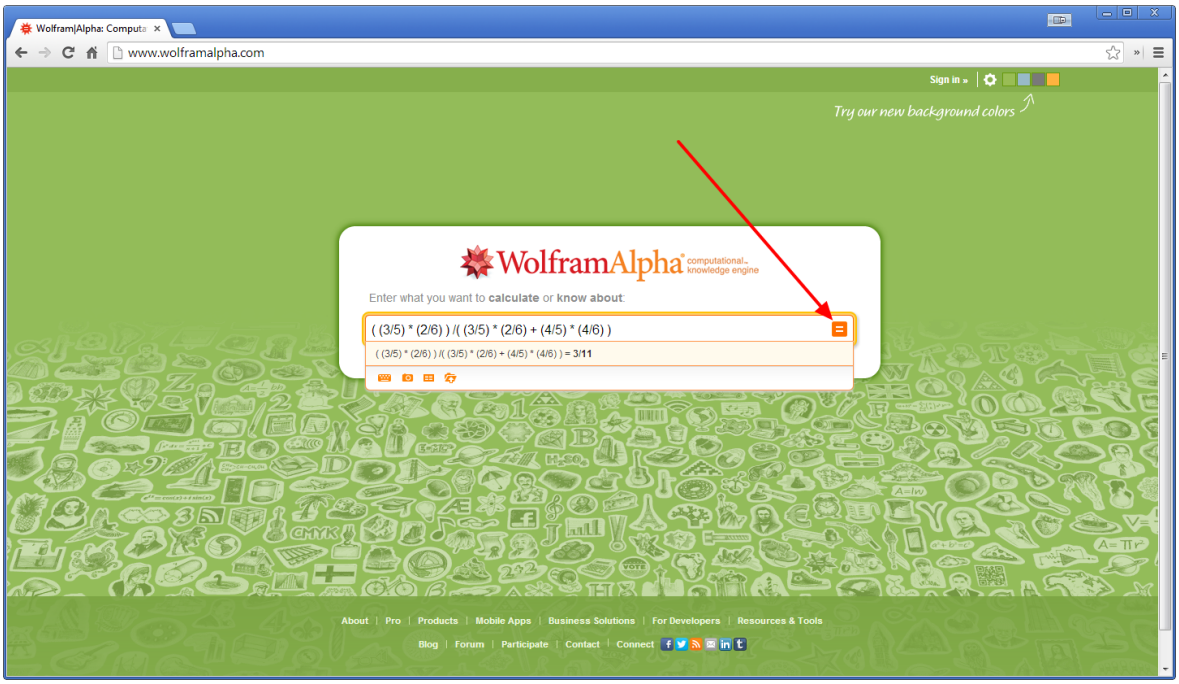
Se trata también de una herramienta sólo accesible a través de la Web, en este caso en inglés, pero que también es muy interesante (por ejemplo, es fácilmente accesible desde teléfonos móviles o tablets). Al abrir la dirección

http://www.wolframalpha.com/

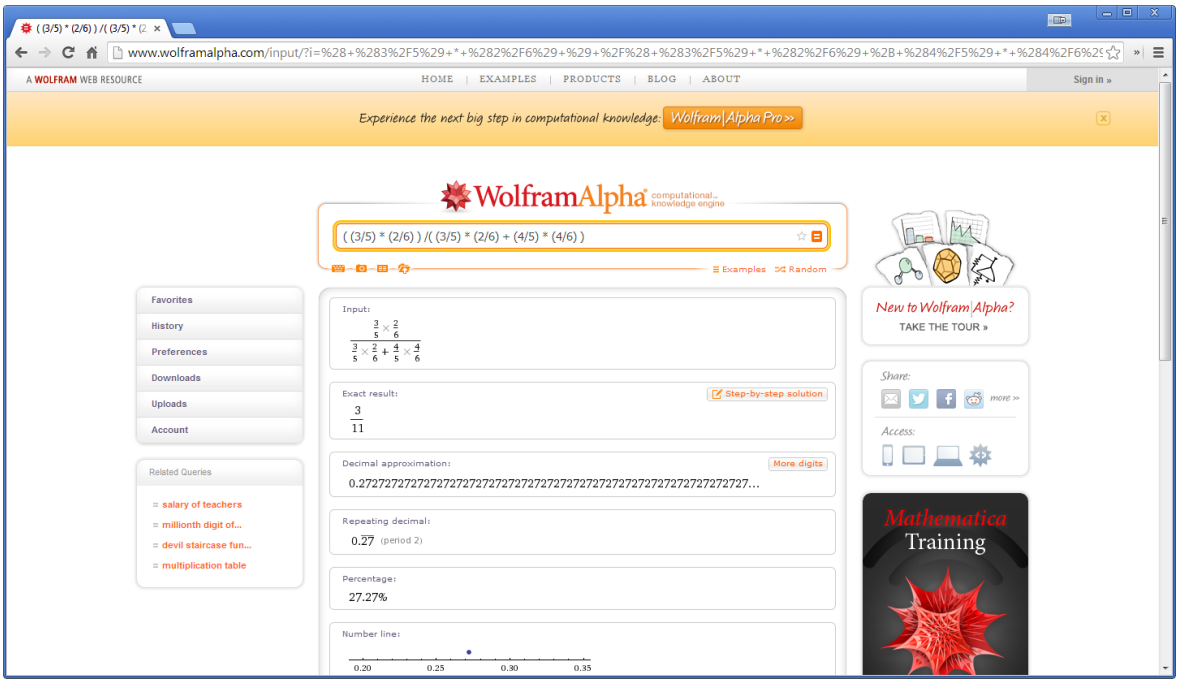
Wolfram Alpha te recibe con una pantalla como esta:



Debemos introducir lo que queremos calcular en el campo de entrada del centro de la ventana, usando en este caso la misma sintaxis que en R:



En este caso, como ves, en cuanto hemos tecleado la operación, Wolfram Alpha nos ha mostrado la respuesta. Pero para hacer el ejemplo más completo, pulsa sobre el símbolo igual que hay al final del campo de entrada. Verás una pantalla parecida a esta,



en la que, desde luego, está la respuesta a tu pregunta, pero que contiene además mucha más información matemática sobre esa pregunta (más de la que seguramente nunca pensaste que existiera...) Como en el caso de Wiris CAS, apenas hemos rozado la superficie de lo que Wolfram Alpha es capaz de hacer, y volveremos más adelante a seguir aprendiendo como usarlo. Si quieres, puedes pulsar en el enlace `Examples` que aparece bajo la barra de entrada para ver algunas de esas cosas.

5.3. Suma de series con Wiris y Wolfram Alpha.

Una serie es una suma con infinitos sumandos, como la suma

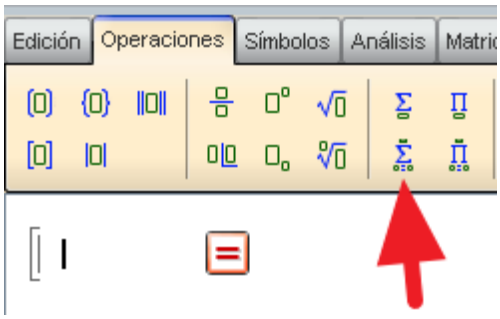
$$\frac{1}{2}+\frac{1}{2^3}+\frac{1}{2^5}+\frac{1}{2^7}+\cdots=\frac{2}{3}.$$

que vimos en el Ejemplo 3.3.5 (pág. 58) del libro. La teoría matemática de este tipo de sumas puede llegar a ser muy complicada. Pero nosotros nos vamos a quedar en la superficie, limitándonos a usar el ordenador para calcular los ejemplos que necesitemos.

El primer ejemplo, el más sencillo de todos, nos permite comprobar que la probabilidad total asignada en ese Ejemplo 3.3.5 es igual a 1, como exigen las propiedades básicas de la Probabilidad. Es decir, que queremos ver que:

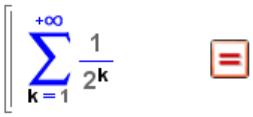
$$\frac{1}{2}+\frac{1}{2^2}+\frac{1}{2^3}+\frac{1}{2^4}+\cdots=1.$$

En la primera suma de esta sección sumábamos sólo los términos con exponentes impares, aquí los sumamos todos. Para calcular esta suma usaremos Wiris. Una vez abierto, en la pestaña `Operaciones`, busca el icono del sumatorio



Y usa la paleta de símbolos de esa misma pestaña para escribir la serie, como en la siguiente figura. Un par de advertencias:

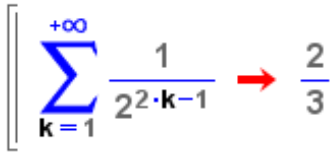
- Encontrarás el símbolo de infinito positivo ($+\infty$) en la pestaña `Símbolos`.
- Para escribir un exponente en Wiris puedes, desde luego, usar el botón de la pestaña `Operaciones`. Pero es más rápido usar el atajo de teclado `Ctrl+↑`.



Pulsa sobre el símbolo igual, y verás como Wiris te confirma que el resultado de sumar esa serie es un 1.

Ahora vamos a modificar ligeramente este cálculo, para obtener la suma de la serie de los términos con exponentes impares con la que empezamos esta sección. Para conseguir esto, vamos a modificar el exponente para que sólo tome valores impares. La forma de conseguirlo es sustituir, en el exponente, la variable k por la fórmula $2\cdot k-1$. Porque, cuando k recorre los valores $1,2,3,\ldots$, la fórmula $2\cdot k-1$ recorre a su vez los valores impares $1,3,5,\ldots$

Por lo tanto, hacemos ese cambio en Wiris, volvemos a pulsar el símbolo igual y obtenemos el resultado que se muestra en esta figura:



Como hemos dicho, las matemáticas de las series pueden ser muy complicadas. Queremos, para cerrar esta brevísima visita, añadir un ejemplo final, que ilustra un punto importante de esa teoría. La suma de esta serie

$$1+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{n}+\cdots$$

es infinito. Si le preguntas a Wiris, en este caso protestará, y dirá que no ha sido capaz de calcularlo (mira en la parte inferior de la pantalla). Lo malo es que a veces obtendremos esa misma respuesta con series que son "demasiado complicadas para Wiris", pero cuya suma es una cantidad finita. Si alguna vez necesitas necesitas más detalles, puedes consultar el manual de Wiris al respecto.

También puedes probar con Wolfram Alpha. Ya sabes, la dirección es

http://www.wolframalpha.com/

Prueba a escribir en el campo de entrada

sum (1/n)

y obtendrás esta respuesta

La frase `sum does not converge` (la suma –o serie– no converge) se debe, en este caso, a que el resultado no es un número finito[3]

¿Por qué esta suma es infinito, mientras que las anteriores daban resultados finitos? La respuesta es que una serie es una suma de infinitos números, cada vez más pequeños, y que el factor clave para que la suma sea finita es la *velocidad* a la que los números se hacen pequeños. Si se hacen pequeños muy rápido, la serie tendrá una suma finita (el resultado de la suma dependerá de cómo sea esa velocidad, en detalle). Pero si los números, por el contrario, aunque se hagan pequeños, lo hacen despacio, entonces la suma se irá haciendo cada vez más grande, hasta valer infinito, *en el límite*.

6. Combinatoria en R. Cómo instalar librerías adicionales.

R proporciona algunas herramientas básicas para hacer cálculos en Combinatoria. Para empezar, la función `factorial`, que se usa como muestra este ejemplo de código:

```
factorial(6)

## [1] 720
```

El factorial de n, `fact(n)` en el lenguaje de R, da como resultado el producto de todos los elementos que forman el vector `1:n`. Y no es de extrañar que en R exista una función `prod`, para multiplicar todos los elementos de un vector:

```
prod(1:6)

## [1] 720
```

La ventaja de `prod` frente a `fact` es, desde luego, que los elementos no tienen porque ser naturales, ni consecutivos, etc. Puesto que casi todos los cálculos de Combinatoria que vamos a usar en el

[3] Aparte de que el resultado sea infinito, pueden ocurrir otras cosas. La serie converge cuando el resultado es un número finito, y no converge en cualquier otro caso.

curso se basan en el factorial, podríamos parar aquí. Pero el factorial es una forma extremadamente poco eficiente de hacer cálculos. Para trabajar de una forma más sensata, por ejemplo, al calcular números combinatorios, R nos ofrece la función `choose`. La forma de utilizar esta función para calcular el número combinatorio

$$\binom{22}{7}$$

es esta:

```
choose(n=22,k=7)
```

```
## [1] 170544
```

6.1. Librerías adicionales en R.

Con estas dos herramientas podemos resolver la mayoría de las preguntas que surgen en los problemas elementales de Probabilidad, al aplicar la Regla de Laplace, cuando queremos calcular el *número* de casos posibles o de casos favorables a un suceso. Pero no nos sirven si lo que queremos es, de hecho, enumerar esos casos y construir realmente la lista completa de casos.

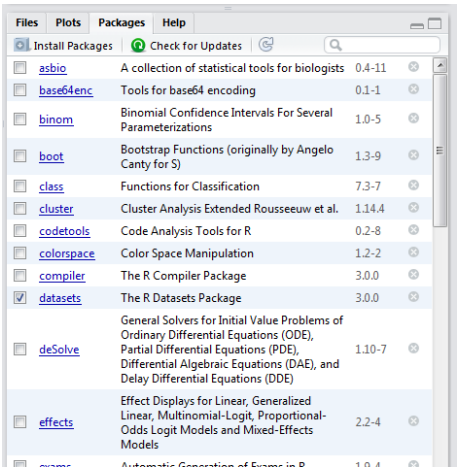
Por ejemplo, en el caso que usamos muchas veces de las tiradas de un par de dados. Sabemos por un lado que hay 36 resultados posibles distintos. Pero si queremos obtener la lista, entonces queremos obtener los resultados en una forma similar a

$$(1,1),(1,2),\ldots,(4,3),\ldots,(6,5),(6,6).$$

Hasta ahora nos hemos limitado a representar los resultados de las dos tiradas mediante números del 1 al 36. Pero esa representación no es siempre conveniente y, en problemas más complicados, las representaciones análogas, que nos alejan de la estructura real de los datos, se convierten en un inconveniente cada vez mayor. Así que nos conviene buscar maneras de representar los datos en R tal y como son, de la manera más fidedigna posible.

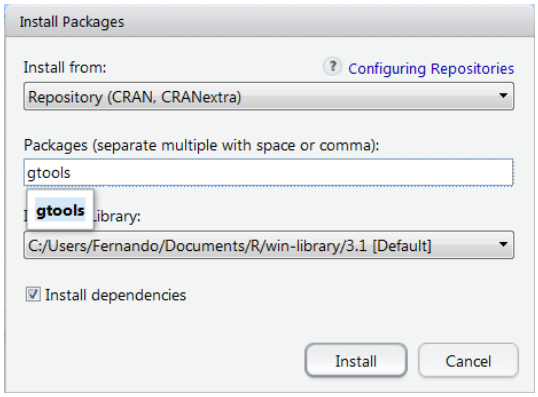
Vamos a utilizar este problema para aprender algunas cosas más sobre R. La instalación de R, tal como hemos aprendido a hacerla en el Tutorial02, incluye sólo una pequeña parte de todo lo que es posible hacer con R. Porque una de las ventajas que se desprenden del hecho de que R sea un programa de código abierto es la gran cantidad de paquetes de código R que se han escrito, para resolver los problemas más variados. Ya hemos visto cómo guardar nuestros propios fragmentos de código en R. Otros usuarios, a menudo expertos en R y en alguna de sus aplicaciones, escriben sus propios fragmentos de código, a veces muy sofisticados, para resolver problemas. Y después ponen esos fragmentos de código, llamados a veces paquetes o librerías, a disposición de la comunidad de usuarios de R, de forma libre y gratuita. A fecha de hoy, R-cran, que es el mayor almacén de código R, contiene más de 5600 de esos paquetes (y la cifra aumenta constantemente).

Cuando instalas R en tu ordenador, normalmente sólo instalas una pequeña parte de esos paquetes, para no hacer la instalación innecesariamente larga y complicada. En RStudio, el panel inferior izquierdo contiene una pestaña llamada `Packages`, en la que puedes ver la lista de las librerías que actualmente tienes instaladas. Tu lista será, muy probablemente, diferente de la mía (y, si tienes una versión reciente de RStudio, incluirá una pestaña adicional, llamada `Viewer`).



Si en algún momento necesitas uno de los paquetes que no has instalado, puedes conectar con un almacén de paquetes en Internet, descargarlo e instalarlo de forma casi automática. Para ello, asegúrate de que tu ordenador está conectado a la red. Si es así, en RStudio basta con que pulses sobre el botón `Install Packages` de esa ventana, y que teclees el nombre del paquete en el campo `Packages` de la ventana que se abre (ver la siguiente figura).

Para practicar esto y, de paso, mejorar las habilidades Combinatorias de R vamos a instalar un paquete llamado `gtools`. Así que escribe ese nombre en el campo *packages* (a medida que escribes el nombre, verás que RStudio lo reconoce como uno de los paquetes disponibles).



Después pulsa sobre el botón `Install`, y verás como R realiza una serie de operaciones (descarga e instalación, básicamente). Si, por alguna razón, no usas RStudio, siempre puedes ejecutar este comando en la consola de R:

```
install.packages(pkgs="gtools",dependencies=TRUE)
```

Ya hemos instalado el paquete `gtools`, pero todavía falta un paso más. De la misma forma que la instalación de R no incluye todos los paquetes que existen (más de 5600, sólo en CRAN), es posible que al cabo de un tiempo tengas instalados en tu ordenador decenas o cientos de paquetes. Y no queremos sobrecargar la memoria del ordenador, haciéndole cargar todos esos paquetes en cada sesión de trabajo (además, algunos de ellos podrían interferir entre sí). Por esa razón, cada vez que vayamos a usarlo tenemos que decirle a R que, para nuestro trabajo en esa sesión, queremos usar ese paquete en concreto. La forma de decirle a R que quiero usar ese paquete es mediante el comando:

```
library("gtools")
```

Al ejecutarlo puede que aparezca alguna advertencia (`warning`) de R sobre el número de versión. No te preocupes, mientras no aparezcan mensajes de error. Tras la instalación, ya podemos empezar el trabajo de Combinatoria con R.

6.2. Lista de Permutaciones y combinaciones con R.

El paquete `gtools` añade (entre otras muchas cosas) dos funciones nuevas a R, `permutations` y `combinations`, que nos van a servir precisamente para obtener esos resultados. En España, como hemos visto en el curso, distinguimos entre variaciones y permutaciones, pero la tradición anglosajona engloba dentro del término *permutations* tanto las variaciones como las permutaciones.

Insistimos en que no se trata de saber cuántos hay, sino, de hecho, de construirlos y enumerarlos. Para ver en acción a estas funciones de R, vamos a utilizarlas para fabricar todas las listas posibles de tres elementos, elegidos de entre cuatro posibles. Para empezar, no admitimos repeticiones de los elementos. Entonces, si el orden no importa, estamos formando las combinaciones de cuatro elementos tomados de tres en tres. Y si el orden importa, entonces se trata de las variaciones (un inglés diría permutaciones) de cuatro elementos tomados de tres en tres.

Los correspondientes comandos son estos (recuerda que has debido ejecutar primero `library("gtools")`). Para las combinaciones:

```
combinations(4,3)
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    1    2    4
## [3,]    1    3    4
## [4,]    2    3    4
```

Cada fila de la salida contiene una de las combinaciones que buscábamos. Y para las permutaciones:

```
permutations(4,3)
```

```
##       [,1] [,2] [,3]
##  [1,]    1    2    3
##  [2,]    1    2    4
##  [3,]    1    3    2
##  [4,]    1    3    4
##  [5,]    1    4    2
##  [6,]    1    4    3
##  [7,]    2    1    3
##  [8,]    2    1    4
##  [9,]    2    3    1
## [10,]    2    3    4
## [11,]    2    4    1
## [12,]    2    4    3
## [13,]    3    1    2
## [14,]    3    1    4
## [15,]    3    2    1
## [16,]    3    2    4
## [17,]    3    4    1
## [18,]    3    4    2
## [19,]    4    1    2
## [20,]    4    1    3
## [21,]    4    2    1
## [22,]    4    2    3
## [23,]    4    3    1
## [24,]    4    3    2
```

Ejercicio 14. *Hemos visto ya que en R hay muchos tipos de objetos distintos (vectores, matrices, tablas, etc.) ¿Qué tipo de objetos son los que estamos obteniendo? Solución en la página 42.* □

¿Cómo se usa esto para las tiradas de dos dados? Bueno, si queremos obtener resultados equiprobables, debemos distinguir el orden (como si cada dado fuera de un color) y, además, a diferencia de los dos ejemplos anteriores, debemos permitir resultados repetidos. Con esas premisas, hacemos

```
     (dosDados=permutations(6,2,repeats.allowed=TRUE))
```

```
##       [,1] [,2]
##  [1,]    1    1
##  [2,]    1    2
##  [3,]    1    3
##  [4,]    1    4
##  [5,]    1    5
##  [6,]    1    6
##  [7,]    2    1
##  [8,]    2    2
##  [9,]    2    3
## [10,]    2    4
## [11,]    2    5
```

```
## [12,]    2    6
## [13,]    3    1
## [14,]    3    2
## [15,]    3    3
## [16,]    3    4
## [17,]    3    5
## [18,]    3    6
## [19,]    4    1
## [20,]    4    2
## [21,]    4    3
## [22,]    4    4
## [23,]    4    5
## [24,]    4    6
## [25,]    5    1
## [26,]    5    2
## [27,]    5    3
## [28,]    5    4
## [29,]    5    5
## [30,]    5    6
## [31,]    6    1
## [32,]    6    2
## [33,]    6    3
## [34,]    6    4
## [35,]    6    5
## [36,]    6    6
```

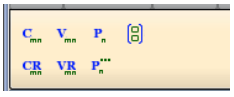
Fíjate que cada tirada de los dos dados ocupa una fila de la matriz que se obtiene como resultado.

7. Combinatoria con Wiris y Wolfram Alpha

En la Sección 3.6 (pág. 72) del libro hemos discutido la forma de calcular el número de permutaciones, variaciones, y combinaciones, con o sin repetición. Aquí vamos a aprender a utilizar algunas de las herramientas software que conocemos para facilitar el trabajo en problemas de Combinatoria.

7.1. Wiris

De nuevo en Wiris, en la pestaña de Combinatoria, encontrarás los iconos de esta figura:



Los significados son evidentes, así que nos limitamos a invitarte a que uses Wiris para calcular el resultado de estos ejercicios:

Ejercicio 15.

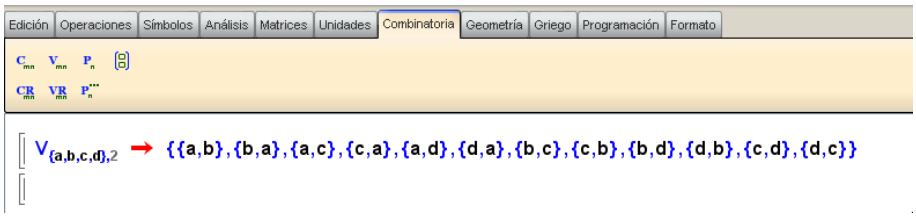
1. *Permutaciones de 6 elementos.*
2. *Variaciones de 100 elementos, tomados de 30 en 30.*
3. *Combinaciones de 22 elementos, tomados de 7 en 7.*
4. *Permutaciones con repetición de 10 elementos, divididos en grupos de elementos idénticos de* $(2, 2, 3, 3)$ *(de modo que hay* 2 *del primer tipo,* 2 *del segundo,* 3 *del tercero y* 3 *del cuarto tipo).*
5. *Variaciones con repetición de 8 elementos, tomados de 13 en 13.*
6. *Combinaciones con repetición de 4 elementos tomados de 6 en 6.*

Soluciones en la pág. 43. □

Aparte de calcular estos números, puedes usar Wiris para enumerar (es decir, hacer la lista, explícitamente) las variaciones o combinaciones. La forma de hacer esto es darle a Wiris como primer argumento una lista de los valores entre los que tiene que hacer la selección. Por ejemplo, para obtener la lista de variaciones de las 5 letras

$$\{a, b, c, d\}$$

tomadas de 2 en 2 (hay 12 posibles), hacemos esto (se muestra el resultado de ejecutar el comando)



Puedes aprender más sobre las capacidades combinatorias de Wiris en el manual:

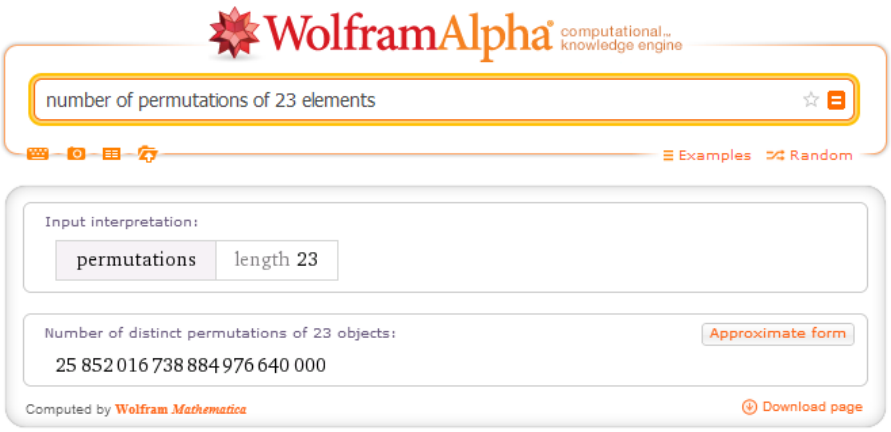
http://www.wiris.net/educa.madrid.org/wiris/manual/es/html/tour/combinatoria.html

7.2. Wolfram Alpha

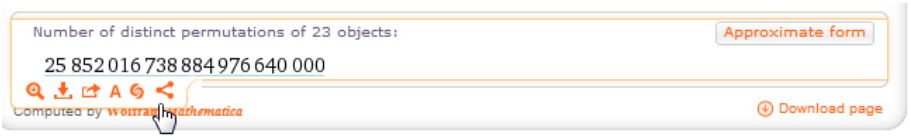
Para usar Wolfram Alpha, debemos expresar (en inglés, claro) lo que deseamos calcular en esa mezcla de lenguaje natural y símbolos matemáticos, característica de este sistema. Por ejemplo, si escribes en el campo de entrada:

`number of permutations of 23 elements`

obtendrás como respuesta



Si quieres copiar el resultado para poder pegarlo en otro programa, sitúa el ratón sobre ese resultado, y verás aparecer una barra de herramientas, como en la siguiente figura. Haz clic en la letra `A` para obtener una versión copiable como texto del resultado.



Para ver más ejemplos de como usar Wolfram Alpha en Combinatoria, teclea `Combinatorics` en la línea de entrada. Ten en cuenta, en cualquier caso, que muchos de los ejemplos que verás no son relevantes para nuestro curso. Así que asegúrate de, al menos, hacer el siguiente

Ejercicio 16. *Teclea en el campo de entrada de Wolfram Alpha los siguientes comandos y, para practicar, copia el resultado como texto en un editor de texto, como el Bloc de Notas.*

1. `permutations of 6 elements`
2. `permutations(100,30)`
3. `combinations(22,7)`
4. `number of permutations of a,a,b,b,c,c,c,d,d,d`
5. *Aunque no hay una sintaxis directa (al menos, yo no la conozco), usa Wolpram Alpha para calcular el número de variaciones con repetición de 8 elementos, tomados de 13 en 13, y el de combinaciones con repetición de 4 elementos tomados de 6 en 6.*

Soluciones en la pág. 43. □

8. Ejercicios adicionales y soluciones

Ejercicios adicionales

17. Se lanzan dos dados. Hallar la probabilidad de estos sucesos:

    a) la suma de los resultados es ocho y (simultáneamente) su diferencia es cuatro.

    b) la suma de los resultados es cinco y (simultáneamente) su producto es cuatro.

    c) la suma de los resultados sea mayor que 12.

    d) la suma de los resultados sea divisible entre 3.

    Solución en la página 43.

18. Hallar la probabilidad de que al lanzar una moneda dos veces se obtenga al menos una vez una cruz. Solución en la página 44.

19. En una caja hay seis fichas iguales numeradas del uno al 6. Se extraen una por una (sin reemplazarlas) todas las fichas de la caja. ¿Cuál es la probabilidad de que salgan en el orden natural? (Es decir, primero la ficha número uno, luego la dos, etc.) Solución en la página 44.

20. En un paquete hay 20 tarjetas numeradas del 1 al 20. Se escogen al azar dos tarjetas. ¿Cuál es la probabilidad de que las dos que se han elegido sean la número 1 y la número 20? ¿Hay alguna diferencia entre sacar las dos tarjetas a la vez, o sacarlas consecutivamente sin reemplazamiento? ¿Y si es con reemplazamiento (sacamos una, la devolvemos al paquete y sacamos otra al azar)? Solución en la página 44.

21. Una clase consta de 10 hombres y 20 mujeres. La mitad de los hombres y la mitad de las mujeres tienen los ojos castaños. Hallar la probabilidad de que una persona elegida al azar sea un hombre o tenga los ojos castaños. Solución en la página 45.

22. Las siguientes afirmaciones son necesariamente falsas. Explica por qué.

    a) En un hospital, la probabilidad de que un paciente permanezca ingresado durante más de dos días es de 0.5. La probabilidad de que un paciente permanezca hospitalizado durante más de un día es de 0.3.

    b) La probabilidad de que llueva el sábado es del 50 % y de que llueva el domingo es del 50 %. Por tanto, durante el fin de semana es seguro que lloverá.

    Solución en la página 45.

23. Se escogen al azar tres lámparas de entre 15, y sabemos que de esas 15, cinco son defectuosas. ¿Cuál es la probabilidad de que al menos una de las tres elegidas sea defectuosa? Solución en la página 46.

24. Hallar la probabilidad de que al tirar tres dados aparezca el seis en uno de los dados (no importa cual), pero sólo en uno de ellos. Solución en la página 46.

25. En una baraja de cartas española (40 cartas repartidas entre 4 palos) se desechan un número de cartas indeterminado. De las cartas que quedan se tiene una serie de probabilidades a la hora de sacar una carta:

$$P(\{\text{sacar rey}\}) = 0.15, \quad P(\{\text{sacar bastos}\}) = 0.3,$$

y además

$$P(\{\text{sacar carta que no sea rey ni bastos}\}) = 0.6.$$

a) ¿Está entre las cartas no desechadas el rey de bastos? En caso afirmativo, calcula la probabilidad de sacar esta carta.

b) ¿Cuántas cartas hay?

Solución en la página 47.

26. En cierta facultad, se sabe que (1) un 25 % de los estudiantes suspendió Matemáticas, (2) un 15 % suspendió Química y (c) un 10 % suspendió ambas. Se selecciona un estudiante al azar:

a) si suspendió Química, ¿cuál es la probabilidad de que también suspendiera Matemáticas?

b) si suspendió Matemáticas, ¿cuál es la probabilidad de que también suspendiera Química?

c) ¿Cuál es la probabilidad de que suspendiera al menos una de las dos?

Solución en la página 47.

27. Un hospital tiene dos quirófanos en funcionamiento. En el primero se han producido incidentes en el 20 % de sus operaciones y el segundo sólo en el 4 %. El número de operaciones es el mismo en ambos quirófanos. La inspección hospitalaria analiza el expediente de una operación, elegido al azar y observa que en esa operación se produjo un incidente. ¿Cuál es la probabilidad de que la operación se realizara en el primer quirófano? Solución en la página 48.

28. Un equipo de investigación está preparando un nuevo test para el diagnóstico de la enfermedad de Alzheimer. El test se ha probado en una muestra aleatoria con 450 pacientes diagnosticados con Alzheimer y una muestra aleatoria independiente de 500 pacientes que no presentan síntomas de la enfermedad. La siguiente tabla resume los resultados del ensayo:

		Padecen Alzheimer		
		Sí	No	Total
Resultado del Test	Positivo	436	5	441
	Negativo	14	495	509
	Total	450	500	950

Con estos datos, responder a las siguientes preguntas:

a) ¿Cuál es la probabilidad de que un sujeto sano haya dado positivo en el test?

b) ¿Cuál es la probabilidad de que un sujeto enfermo haya dado negativo en el test?

c) Sabiendo que un sujeto ha dado positivo en el test, ¿cuál es la probabilidad de que esté enfermo?

d) Sabiendo que un sujeto ha dado negativo en el test, ¿cuál es la probabilidad de que esté sano?

Solución en la página 48.

29. Una empresa produce anillas para identificación de tortugas marinas en tres fábricas. El volumen de producción diario es de 500, 1000 y 2000 unidades respectivamente. Se sabe que la fracción de producción defectuosa de las tres fábricas es de 0.005, 0.008, 0.010 respectivamente. Si se selecciona una anilla de forma aleatoria del total de producción de un día y se descubre que es defectuosa, ¿de qué fábrica es más probable que provenga esa anilla? Solución en la página 48.

30. Usa R para simular el experimento que se describe en el Ejemplo 3.4.1 del libro (pág. 62). Por si te sirve de guía visual, en la hoja de cálculo (del programa Calc):

Cap03-Lanzamientos2Dados-ProbabilidadCondicionada.ods

se ha realizado una simulación para comprobar estos resultados.
Una extensión natural de este ejemplo es tratar de calcular $P(S|D)$. ¿Puedes modificar la hoja de cálculo para simular este otro caso?

31. Usa R para simular el experimento que se describe en el Ejemplo 3.5.1 del libro (pág. 67), y que ilustramos con el fichero Calc

    Cap03-ProbabilidadesTotales-Urnas.ods

32. Usa R para simular los experimentos de los Ejemplos 3.6.3 y 3.6.4 del libro (pág. 78)

33. Elegimos al azar cinco números del 1 al 10, con reemplazamiento. Puedes pensarlo así: en una caja hay 10 bolas numeradas del 1 al 10. Sacamos una bola, anotamos el número, devolvemos la bola a la caja, y la agitamos bien. ¿Cuál es la probabilidad de que no haya repeticiones y, por tanto, obtengamos cinco números distintos? Solución en la página 49.

34. De entre los números naturales $1, 2, \ldots, 20$ se seleccionan cinco al azar sin reemplazamiento. Calcular la probabilidad de que: (a) los cinco sean pares. (b) exactamente dos de ellos sean múltiplos de 3. (c) dos sean impares y tres pares. Solución en la página 49.

35. En la lotería primitiva gana quien acierta 6 números de entre 64 sin importar el orden en el que salgan. ¿Cuál es la probabilidad de ganar con una única apuesta? Solución en la página 50.

36. De las 28 fichas del dominó, se extraen dos al azar (sin remplazamiento). ¿Cuál es la probabilidad de que con ellas se pueda formar una cadena, conforme a las reglas del juego (debe haber un número que aparezca en ambas fichas)? Solución en la página 50.

37. Calcular la probabilidad de que un número de cuatro cifras (una matrícula, o un pin)

    a) tenga cuatro cifras diferentes.

    b) tenga alguna cifra repetida.

    c) tenga exactamente dos cifras iguales.

    d) tenga dos parejas de cifras iguales (pero distintas entre sí).

    e) tenga exactamente tres cifras iguales.

    f) tenga todas las cifras iguales.

    Solución en la página 51.

38. **"La paradoja del cumpleaños".** Si en una sala hay 367 personas, entonces, con total seguridad, habrá dos personas con la misma fecha de cumpleaños (hemos usado 367 para cubrir incluso el caso de los años bisiestos, por si alguien de la sala nació el 29 de Febrero). Así que, si llamamos

    $$A_n = \{\text{al menos dos de las } n \text{ personas cumplen años el mismo día}\}$$

    entonces $P(A_{367}) = 1$. ¿Cuántas personas tiene que haber en la sala para que la probabilidad $P(A_n)$ sea mayor que $1/2$? Muchas menos de las que imaginas. Empieza por calcular ¿Cuánto vale $P(A_5)$ usando el ejercicio anterior.
    Este resultado se conoce como "la paradoja del cumpleaños", aunque no tiene nada de paradójico. Lo único que realmente demuestra este resultado es que, como hemos señalado en el curso, la intuición en materia de probabilidades es, en general, para la mayoría de nosotros, muy pobre.Solución en la página 52.

Soluciones de algunos ejercicios

- **Ejercicio 1, pág. 4**

```
set.seed(2014)
(caja=rep( 1:5, c(35, 15, 10, 10, 30) ))
```

```
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [36] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4
##  [71] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
```

```
(sorteo1 = sample(caja, 20, replace = TRUE))
```

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

```
(sorteo2 = sample(caja, 20, replace = FALSE))
```

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

```
(tabFrecAbs1  = table(sorteo1))
```

```
## sorteo1
## 1 2 3 4 5
## 9 1 2 6 2
```

```
(tabFrecRel1  = table(sorteo1) / length(sorteo1) )
```

```
## sorteo1
##    1    2    3    4    5
## 0.45 0.05 0.10 0.30 0.10
```

```
(tabFrecAbs2  = table(sorteo2))
```

```
## sorteo2
## 1 2 3 4 5
## 3 3 3 3 8
```

```
(tabFrecRel2  = table(sorteo2) / length(sorteo2) )
```

```
## sorteo2
##    1    2    3    4    5
## 0.15 0.15 0.15 0.15 0.40
```

- **Ejercicio 2, pág. 6**

Cuando `byrow=FALSE` (que es la opción por defecto), los elementos se colocan por columnas.

```
(dosPartidas = matrix(head(dado4000, 8), ncol = 4, byrow = FALSE))
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    2    4    4    6
## [2,]    2    2    1    4
```

```
(dosPartidas = matrix(head(dado4000, 8), ncol = 4))
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    2    4    4    6
## [2,]    2    2    1    4
```

- **Ejercicio 3, pág. 6**

```
(M = matrix(1:36, nrow=4, byrow=TRUE) )
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
## [1,]    1    2    3    4    5    6    7    8    9
## [2,]   10   11   12   13   14   15   16   17   18
## [3,]   19   20   21   22   23   24   25   26   27
## [4,]   28   29   30   31   32   33   34   35   36
```

- **Ejercicio 4, pág. 7**

```
dim(M)= c(4, 9)
M
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
## [1,]    1    2    3    4    5    6    7    8    9
## [2,]   10   11   12   13   14   15   16   17   18
## [3,]   19   20   21   22   23   24   25   26   27
## [4,]   28   29   30   31   32   33   34   35   36
```

- **Ejercicio 5, pág. 7**

Los vectores *puros* de R no tienen dimensión cuando son creados. La adquieren al convertirlos en matrices.

```
unVector = 1:10
dim(unVector)
```

```
## NULL
```

```
class(unVector)
```

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

```
M = matrix(unVector, nrow=1)
dim(M)
```

```
## [1]  1 10
```

```
class(M)
```

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

```
a=letters[1:12]
(Ma = matrix(a, nrow=3))
```

```
##      [,1] [,2] [,3] [,4]
## [1,] "a"  "d"  "g"  "j"
## [2,] "b"  "e"  "h"  "k"
## [3,] "c"  "f"  "i"  "l"
```

- **Ejercicio 6, pág. 7**

```
(letras = matrix(letters[1:12], nrow=4, byrow=TRUE))

##      [,1] [,2] [,3]
## [1,] "a"  "b"  "c"
## [2,] "d"  "e"  "f"
## [3,] "g"  "h"  "i"
## [4,] "j"  "k"  "l"

dim(letras)

## [1] 4 3

dim(letras)=c(3,4)
letras

##      [,1] [,2] [,3] [,4]
## [1,] "a"  "j"  "h"  "f"
## [2,] "d"  "b"  "k"  "i"
## [3,] "g"  "e"  "c"  "l"
```

- **Ejercicio 7, pág. 8**

La comprobación es:

```
t(W)

##      [,1] [,2] [,3]
## [1,]    1    6   11
## [2,]    2    7   12
## [3,]    3    8   13
## [4,]    4    9   14
## [5,]    5   10   15

dim(W) = c(5,3)
W

##      [,1] [,2] [,3]
## [1,]    1   12    9
## [2,]    6    3   14
## [3,]   11    8    5
## [4,]    2   13   10
## [5,]    7    4   15
```

- **Ejercicio 8, pág. 9**

```
apply(M, 1, sd)

## [1] 2.74 2.74 2.74 2.74

apply(M, 2, median)

## [1] 14.5 15.5 16.5 17.5 18.5 19.5 20.5 21.5 22.5
```

```
apply(M, 2, summary)

##          [,1]  [,2]  [,3] [,4] [,5] [,6] [,7] [,8] [,9]
## Min.     1.00  2.00  3.00  4.0  5.0  6.0  7.0  8.0  9.0
## 1st Qu.  7.75  8.75  9.75 10.8 11.8 12.8 13.8 14.8 15.8
## Median  14.50 15.50 16.50 17.5 18.5 19.5 20.5 21.5 22.5
## Mean    14.50 15.50 16.50 17.5 18.5 19.5 20.5 21.5 22.5
## 3rd Qu. 21.20 22.20 23.20 24.2 25.2 26.2 27.2 28.2 29.2
## Max.    28.00 29.00 30.00 31.0 32.0 33.0 34.0 35.0 36.0

class(apply(M, 2, summary))

## [1] "matrix"
```

- **Ejercicio 9, pág. 11**

1. La matriz que buscamos es:

```
B == 2

##       [,1]  [,2]  [,3]  [,4]
## [1,]  TRUE FALSE FALSE FALSE
## [2,]  TRUE FALSE FALSE FALSE
## [3,] FALSE FALSE FALSE FALSE
```

2. Se produce un error, porque al tratarse de matrices de dimensiones distintas, R no sabe como emparejar los elementos de `A` con los de `B`.

```
dim(A) = c(2, 6)
A * B

## Error in A * B: arreglos de dimensón no compatibles
```

- **Ejercicio 10, pág. 13**

```
(B = cbind(A, A[, 1]^2))

##      [,1] [,2] [,3] [,4] [,5] [,6] [,7]
## [1,]   -4    3    1    9   -8    3   16
## [2,]   -7   -4   -9    2   -7   -9   49
```

- **Ejercicio 11, pág. 17**

```
which(rowSums(esSeis) == max(rowSums(esSeis)))

## [1] 602
```

- **Ejercicio 12, pág. 18**

Puedes ejecutar varias veces este código (que puedes abrir en el fichero adjunto Tut03-DeMere2.R) para experimentar.

```
# Limpieza inicial
rm(list=ls())

# Generamos las 24000 tiradas del dado
dado24000 = sample(1:36, 24000, replace = TRUE)

# Los agrupamos en partidas cada 24 tiradas.
deMere2 = matrix(dado24000, ncol = 24, byrow = TRUE)

# Localizamos las apariciones del 36 (representa 6 doble).
es36 = (deMere2 == 36)

# Contamos cuantas apariciones del 36 hay en cada partida.
cuantos36 = rowSums(es36)

# Localizamos aquellas partidas con al menos un 36.
partidasGanadoras = (cuantos36 > 0)

# Y medimos la proporcion de partidas ganadoras.
(proporcion = table(partidasGanadoras)/length(partidasGanadoras))

## partidasGanadoras
## FALSE  TRUE
## 0.518 0.482
```

- **Ejercicio 13, pág. 19**

El código (que puedes abrir en el fichero adjunto Tut03-DeMere1a.R) es una adaptación sencilla del que corresponde al primer juego del Caballero de Méré. Puedes ejecutarlo varias veces para comprobar que el resultado se parece más a $\frac{11}{36} \approx 0.306$ que a $\frac{12}{36} \approx 0.333$.

```
# Limpieza inicial
rm(list=ls())
# Generamos las 6000 tiradas del dado
dado6000 = sample(1:6, 6000, replace = TRUE)
# Los agrupamos en partidas cada dos tiradas.
deMere1a = matrix(dado6000 , ncol = 2, byrow = TRUE)
# Localizamos las apariciones del 6.
esSeis = (deMere1a == 6)
# Contamos cuantas apariciones del seis hay en cada partida.
cuantosSeis = rowSums(esSeis)
# Localizamos aquellas partidas con al menos un 6.
partidasGanadoras = (cuantosSeis > 0)
# Y medimos la proporcion de partidas ganadoras.
(proporcion = table(partidasGanadoras)/length(partidasGanadoras))

## partidasGanadoras
## FALSE  TRUE
## 0.694 0.306
```

- **Ejercicio 14, pág. 32**

Puedes averiguarlo preguntando a R de esta forma:

```
class(permutations(4,3) )

## [1] "matrix"

class(combinations(4,3) )

## [1] "matrix"
```

Como puedes ver, lo que hemos obtenido es un objeto de tipo `matrix`.

- **Ejercicio 15, pág. 33**

1. 720.
2. 77910971370578048745872324992773214403583277006848000000000. Este número podría representar el número de subgrupos distintos de 30 alumnos, que podemos formar a partir de una clase de 100 alumnos. Su orden de magnitud es de 10^{57}. Para que te hagas una idea, el número estimado de estrellas en el universo es del orden de 10^{24}.
3. 170544.
4. 25200.
5. 549755813888.
6. 84.

- **Ejercicio 16, pág. 35**

Todas las preguntas aparecen en el Ejercicio 15; consulta las respuesta a ese ejercicio.

- **Ejercicio 17, pág. 35**

Se trata de contar casos favorables y casos posibles.

1. Una posibilidad consiste en escribir todo el espacio muestral y contar a mano.

$$\begin{array}{cccccc} (1,1) & (1,2) & (1,3) & (1,4) & (1,5) & (1,6) \\ (2,1) & (2,2) & (2,3) & (2,4) & (2,5) & (2,6) \\ (3,1) & (3,2) & (3,3) & (3,4) & (3,5) & (3,6) \\ (4,1) & (4,2) & (4,3) & (4,4) & (4,5) & (4,6) \\ (5,1) & (5,2) & (5,3) & (5,4) & (5,5) & (5,6) \\ (6,1) & (6,2) & (6,3) & (6,4) & (6,5) & (6,6) \end{array}$$

   Otra posibilidad consiste en llamar x_i al resultado del lanzamiento i. Lo que nos dice el enunciado es que

$$\begin{cases} x_1 + x_2 = 8 \\ |x_1 - x_2| = 4 \end{cases}$$

   donde las barras de valor absoluto estan para tener en cuenta la posibilidad de que, al restar el resultado de las dos caras del dado, la segunda sea mayor que la primera y el resultdo sea -4. Al eliminar el símbolo de valor absoluto, ese sistema de ecuaciones es equivalente a

$$\begin{cases} x_1 + x_2 = 8 \\ x_1 - x_2 = 4 \end{cases}, \qquad \begin{cases} x_1 + x_2 = 8 \\ x_1 - x_2 = -4 \end{cases}$$

   Al resolver cada uno de los sistemas tenemos $x_1 = 6, x_2 = 2$ y $x_1 = 2, x_2 = 6$. Es decir, dos casos favorables frente a 36 posibles.

2. Análogo al anterior
3. No es posible
4. Debe ser $x_1 + x_2 = 3n$, donde n sólo puede valer $n = 1, 2, 3, 4$. La probabilidad es $\frac{12}{36}$

• Ejercicio 18, pág. 35
Si escribes el espacio muestral, $(C, C), (C, X), (X, C), (XX)$ verás que hay 4 casos posibles y 3 favorables.

• Ejercicio 19, pág. 35
Hay 1 caso favorable y 6! casos posibles, por lo que la probabilidad es $\frac{1}{6!}$ donde puedes calcular 6! con la orden

```
    factorial(6)

## [1] 720
```

• Ejercicio 20, pág. 35

- Empezamos suponiendo que sacamos dos fichas a la vez. Usaremos la regla de Laplace para calcular la probabilidad. Hay un caso favorable, que es sacar $1, 20$. Observa que el orden NO es importante, nos da lo mismo sacar $1, 20$ que $20, 1$. El número de casos posibles esel número de subconjuntos de dos tarjetas que podemos hacer a partir de 20 tarjetas, sin que importe el orden. Es decir, combinaciones de 20 elementos tomados de 2 en 2. Podemos usar R para hacer el cálculo con la función 'choose()'. Si no recuerdas su significado, usa el tabulador

```
1/choose(20,2)

## [1] 0.00526
```

- Supongamos ahora que sacamos las dos fichas una tras otra SIN reemplazamiento. En este caso debemos tener en cuenta el orden, ya que el experimento tendría resultado positivo tanto si primero sale un 1 y despues el 20 como si sucediera al revés.

$$P(A = \text{sacar primero un 1 y después un 20}) = P(\text{sacar un 1})P(\text{sacar un 20}|\text{ha salido un 1})$$
$$= \frac{1}{20}\frac{1}{19}$$

De la misma manera,

$$P(B = \text{sacar primero un 20 y después un 1}) = \frac{1}{20}\frac{1}{19}$$

Y la probabilidad total es

```
(1/20)*(1/19)+(1/20)*(1/19)

## [1] 0.00526
```

- Vamos a usar un razonamiento diferente para este último caso (pero que se puede aplicar a los anteriores, y recíprocamente, el argumento con elresolvimoslos anteriores apartados se puede aplicar a este)

$$P(\text{sacar un 1 y un 20}) = P(\text{sacar un 1 o un 20})P(\text{sacar laficha que falta} \mid \text{ya salió un 1 o un 20})$$
$$= \frac{\mathbf{2}}{20}\frac{1}{\mathbf{20}}$$

porque se ha devuelto la primera ficha, de modo que las probabilidades con y sin reemplazamiento son diferentes.

• **Ejercicio 21, pág. 35**
Hay varios enfoques posibles. Uno de ellos consiste en contar los casos favorables, que son 20 (10 hombre y las 10 mujeres que tienen los ojos castaños) y dividir entre los casos posibles, que son 30, para obtener

$$P(\text{ser hombre o tenerlos ojos castaños}) = \frac{2}{3}$$

• **Ejercicio 22, pág. 35**

1. El evento "permanecer ingresado durante más de dos días" está contenido (estrictamente) en el evento "permanecer hospitalizado durante más de un día", en el sentido de que si alguien está hospitalizado 2 díaso más ha tenidoque estar (forzosamente) hospitalizado un día o más. Por tanto, el primero no puede tener una probabilidad (estríctamente) mayor que el segundo.
2. Los eventos S = "que llueva el sábado" y D = "que llueva el domingo" son independientes y ambos tiene probabilidad 1/2. Por las reglas de la probabilidad

$$P(\text{llueva el fin de semana}) = P(S \cup D) = P(S) + P(D) - P(S \cap D) =$$

$$P(S) + P(D) - P(S)P(D) = \frac{1}{2} + \frac{1}{2} - \frac{1}{2}\frac{1}{2} = \frac{3}{4} \neq 1$$

• **Ejercicio 23.**

Código R para comprobar experimentalmente el resultado:

```
#set.seed(2014)
n = 10000

# Creo una matriz vacía de n filas y tres columnas
ensayos = matrix(nrow = n, ncol = 3)

# Ahora para cada número i desde 1 hasta n,
for(i in 1:n){
# hago una elección al azar de tres lamparas ¡¡sin reemplazamiento!!:
tresLamparas = sample(1:15, 3)
# y la guardo en la fila número i de la matriz
ensayos[i, ] = tresLamparas
}
# La matriz ensayos contiene n repeticiones del experimento
head(ensayos, 10)

##       [,1] [,2] [,3]
##  [1,]   11    8   15
##  [2,]    4    7    8
##  [3,]   12    3    4
##  [4,]   15    8   14
##  [5,]    2    4   10
##  [6,]    3    5   15
##  [7,]   12    8   14
##  [8,]   10   14    9
##  [9,]    5   10    3
## [10,]   15    6    7

# supongamos que las defectuosas corresponden a los numeros del 1 al 5
# miramos si cada elemento de la matriz es una lampara defectuosa

esDefectuosa = (ensayos<= 5)

# El aspecto de esDefectuosa es este:
head(esDefectuosa, 10)
```

```
##        [,1]  [,2]  [,3]
##  [1,] FALSE FALSE FALSE
##  [2,]  TRUE FALSE FALSE
##  [3,] FALSE  TRUE  TRUE
##  [4,] FALSE FALSE FALSE
##  [5,]  TRUE  TRUE FALSE
##  [6,]  TRUE  TRUE FALSE
##  [7,] FALSE FALSE FALSE
##  [8,] FALSE FALSE FALSE
##  [9,]  TRUE FALSE  TRUE
## [10,] FALSE FALSE FALSE
```

```
# Para que en un experimento (fila de la matriz) no haya ninguna defectuosa
# la suma de esa fila tiene que ser cero

sinDefecto = (rowSums(esDefectuosa) == 0)

# Podemos ver cuantas filas hay con y sin defecto:
table(sinDefecto)
```

```
## sinDefecto
## FALSE  TRUE
##  7412  2588
```

```
# Y para estimar la probabilidad (mediante la frecuencia relativa) basta con hacer:

table(sinDefecto) / n
```

```
## sinDefecto
## FALSE  TRUE
## 0.741 0.259
```

• **Ejercicio 23, pág. 35**
Si llamamos $X =$ "número de lámpara defectuosas", lo que queremos es calcular, en términos de la función de probabilidad, es $P(X \geq 1)$. Este planteamiento abre tres subcasos: según sean una, dos o las tres lámparas defectuosas. Un enfoque alternativo consiste en usar el evento complemetario y calcular $P(X \geq 1) = 1 - P(X = 1)$. Llamamos A_i al evento "la lámpara extraida en i-ésimo lugar no es defectuosa". Y el cálculo que resuelve el ejercicio es

$$P(X \geq 1) = 1 - P(X = 0) = 1 - P(A_1 \cap A_2 \cap A_3)$$

Los tres eventos no son independientes, ya que el hecho de elegir (extraer) una lámpara, cambia tanto el número de casos favorables como el de casos posibles. Por tanto, hay que usar la regla de la multiplicación:

$$1 - P(A_1 \cap A_2 \cap A_3) = 1 - P(A_1)P(A_2|A_1)P(A_3|A_1 \cap A_2)$$

para obtener

```
1-(10/15)*(9/14)*(8/13)
```

```
## [1] 0.736
```

• **Ejercicio 24, pág. 35**

Claramente hay 6^3 casos posibles (piensa en el arbol que usamos en el libro, en la Sección 3.6, para calcular el número de permutaciones de orden n). Por otro lado, los casos favorables son 35^2, ya

que el 6 puede salir sólo uno de los tres lanzamientos y hay 5^2 posibles resultados (diferentes de 6) para los otros dos dados. La probabilidad pedida es $\frac{35^2}{6^3}$

También se puede resolver con la variable binomial X = "número de seises al lanzar 3 dados", con probabilidad de éxito en cada lanzamiento $1/6$ y de fracaso $5/6$.

$$P(X=1) = \left(\begin{array}{c} 3 \\ 1 \end{array} \right) \left(\frac{1}{6}\right) \left(\frac{5}{6}\right)^2$$

• **Ejercicio 25, pág. 36**

1. Si la carta está aún en la baraja la probabilidad, al elegir una carta al azar, de sacarla debe ser positiva. Vamos a llamar R = "sacar rey" y B = "sacar bastos", y lo que hay que calcular

   $$P(R \cap B) = P(\text{sacar el rey de bastos})$$

   El enunciado nos dice

   $$0.6 = P(\{\text{sacar carta que no sea rey ni bastos}\} = P(R^C \cap B^C)) =$$

   $$P((R \cup B)^C)) = 1 - P(R \cup B)$$

   de donde tenemos

   $$P(R \cup B) = 1 - 0.6 = 0.4$$

   Por otro lado, para hacer aparecer el término $P(R \cap B)$ usamos

   $$0.4 = P(R \cup B) = P(R) + P(B) - P(R \cap B) = 0.15 + 0.3 - P(R \cap B)$$

   es decir, $P(R \cap B) = 0.05 > 0$. Luego el rey de bastos está en la baraja.

2. Usando la regla de Laplace:

   $$P(R \cap B) = \frac{\text{casos fav}}{\text{casos pos}} = 0.05$$

   Como sólo hay un caso favorable, al despejar tenemos que hay 20 casos posibles, es decir, quedan 20 cartas.

• **Ejercicio 26, pág. 36**
Si definimos los eventos Q = "suspender química" y M = "suspenden matemáticas", el enunciado nos da la siguiente información

$$P(Q) = 0.15 \qquad P(M) = 0.25 \qquad P(Q \cap M) = 0.1$$

Lo primero es traducir cada pregunta al lenguaje de la probabilidad, y a continuación, hacer el cálculo correspondiente

1. $P(M|Q) = \frac{P(M \cap Q)}{P(Q)} = \frac{0.1}{0.15} = 0.667$
2. $P(Q|M) = \frac{P(M \cap Q)}{P(M)} = \frac{0.1}{0.25} = 0.5$
3. $P(M \cup Q) = P(M) + P(Q) - P(M \cap Q) = 0.25 + 0.15 - 0.1 = 0.3$

• **Ejercicio ??, pág. ??**
Si fueran incompatibles tendríamos $P(A \cap B) = 0$ porque sería $A \cap B = \emptyset$. En ese caso, tendríamos

$$P(A \cup B) = P(A) + P(B) - P(A \cap B) = P(A) + P(B) = 0.5 + 0.6 = 1.1$$

lo cual es imposible.

• **Ejercicio 27, pág. 36**
Llamaremos Q_k al evento "la operación se realizó en el quirófano $k = 1, 2$" e I al evento "ocurrió un incidente".

El enunciado proporciona los siguientes datos:

$$P(Q_1) = 0.5, \qquad P(Q_2) = 0.5$$

$$P(I|Q_1) = 0.2, \qquad P(I|Q_2) = 0.04$$

Lo que nos preguntan es $P(Q_1|I)$ y, para calcularlo, usaremos el teorema de Bayes, es decir

$$P(Q_1|I) = \frac{P(Q_1)P(I|Q_1)}{P(Q_1)P(I|Q_1) + P(Q_2)P(I|Q_2)}$$

```
Prob_quirof = c(1/2, 1/2)
Prob_incid_x_quirof = c(0.2, 0.04)
# P(Q_1|I) =
Prob_quirof[1]*Prob_incid_x_quirof[1]/(sum(Prob_quirof*Prob_incid_x_quirof))

## [1] 0.833
```

• **Ejercicio 28, pág. 36**
En todos los casos se trata de probabilidades condicionadas

1. $P(+|No) = \frac{P(+\cap No)}{P(No)} = \frac{5/950}{500/950}$. A la vista de este resultado, podemos usar directamente las frecuancias absolutas de cada evento (y no las relativas). Así procedemos en los siguientes apartados
2. $P(-|Si) = \frac{P(-\cap Si)}{P(Si)} = \frac{14}{450}$.
3. $P(Si|+) = \frac{P(+\cap Si)}{P(+)} = \frac{436}{441}$.
4. $P(No|-) = \frac{P(-\cap No)}{P(-)} = \frac{495}{509}$.

• **Ejercicio 29, pág. 36**
Llamaremos F_k al evento "la anilla viene de la fábrica k" y D al evento "la anilla es defectuosa".

El enunciado proporciona los siguientes datos:

$$P(F_1) = 500/3500, \qquad P(F_2) = 1000/3500, \qquad P(F_3) = 2000/3500$$

$$P(D|F_1) = 0.005, \qquad P(D|F_2) = 0.008, \qquad P(D|F_3) = 0.01$$

Para determinar de qué fábrica es más probable que provenga esa anilla hay que calcular

$$P(F_1|D), \qquad P(F_2|D), \qquad P(F_3|D)$$

Es decir, hay que usar tres veces el teorema de Bayes que, para la fábrica i, dice así

$$P(F_i|D) = \frac{P(F_i)P(D|F_i)}{P(F_1)P(D|F_1) + P(F_2)P(D|F_2) + P(F_3)P(D|F_3)}$$

```
# usar notación vectorial para reaprovechar
# las expresiones
Prob_fabr = c(500/3500,1000/3500, 2000/3500)
Prob_defecto_x_fabr = c(0.005,0.008, 0.01)
# P(F_1|D)
Prob_fabr[1]*Prob_defecto_x_fabr[1]/(sum(Prob_fabr*Prob_defecto_x_fabr))

## [1] 0.082
```

```r
# P(F_2|D)
Prob_fabr[2]*Prob_defecto_x_fabr[2]/(sum(Prob_fabr*Prob_defecto_x_fabr))

## [1] 0.262

# P(F_3|D)
Prob_fabr[3]*Prob_defecto_x_fabr[3]/(sum(Prob_fabr*Prob_defecto_x_fabr))

## [1] 0.656
```

Un enfoque alternativo es el siguiente: con lo aprendido en el tutorial 3, podemos crear un vector que contenga tantas veces "A", "B" y "C" como anillas defectuosas se producen en cada una de las fábricas, respectivamente. Luego, basta con contar cuántas "A", "B" y "C" hay, la que mayor frecuencias absoluta tenga, esa será lamás probable.

```r
anillas=rep(c("A","B", "C"),c(0.005*500,0.008*1000,0.010*2000))

esA= (anillas=="A")
(cuantasA=sum(esA))

## [1] 2

esB= (anillas=="B")
(cuantasB=sum(esB))

## [1] 8

esC= (anillas=="C")
(cuantasC=sum(esC))

## [1] 20
```

Por tanto, es más probable que la anilla defectuosa venga de la tercera empresa.

- **Ejercicio 33, pág. 37**

Como hay reemplazamiento (recuerda el ejercicio 20)

```r
(10/10)*(9/10)*(8/10)*(7/10)*(6/10)

## [1] 0.302
```

- **Ejercicio 34, pág. 37**

a) Puedes razonar como en el ejercicio 20, cuando se trabajaba SIN reemplazamiento, para obtener que la probabilidad vale

$$\frac{10}{20}\frac{9}{19}\frac{8}{18}\frac{7}{17}\frac{6}{16} = 0.016$$

b) Entre los 20 primeros números naturales existen seis múltiplos de 3. La probabilidad de que al extraer un número, éste sea múltiplo de 3 es (6/20), mientras que la probabilidad de que no lo sea es de (14/20). Además, si extraemos 5 números y queremos obtener 2 múltiplos de 3, tenemos que calcular el número de posibles combinaciones de obtener 2 múltiplos en 5 extracciones. Considerando todo esto, la probabilidad de obtener dos múltiplos de 3 en 5 extracciones es de:

$$P = \left(\begin{array}{c} 5 \\ 2 \end{array} \right) \left(\frac{6}{20}\right)^2 \left(\frac{14}{20}\right)^3 = 0.3$$

Esto se puede expresar en términos de una variable binomial . . . piensa en el problema del caballero de Méré (puedes ir al libro, a la Sección 5.1.2) e intercambia "sacar 2 seises" por "sacar 2 múltiplos de 3".

c) De la misma forma que en el apartado anterior, la probabilidad de obtener un número impar en una extracción es 1/2. Por tanto, la probabilidad de obtener 2 números impares y 3 pares será:

$$P = \left(\begin{array}{c} 5 \\ 2 \end{array} \right) \left(\frac{1}{2}\right)^2 \left(\frac{1}{2}\right)^3 = 0.31$$

• **Ejercicio 35, pág. 37**
En el bombo de la lotería hay 64 números de los que se extraen 6 (es decir, tenemos que trabajar sin reemplazo, dado que las bolas extraidas ya no se vuelven a introducir en el bombo). Podemos llamar A_i al evento "en la extracción i-ésima sale uno de mis números". Y queremos calcular $P(A_1 \cap A_2 \cap \cdots \cap A_6)$. Si usamos la regla de la multiplicación, tenemos

$$P(A_1 \cap A_2 \cap \cdots \cap A_6) = P(A_1)P(A_2|A_1)P(A_3|A_1 \cap A_2) \cdots P(A_6|A_1 \cap \cdots A_5)$$

En la primera extración hay 6 casos favorables frente a 64 posibles. En la segunda sólo hay 5 posibles (porque supongo que ya ha salido uno de mis números) frete a63 posibles, y así sucesivamente. Por tanto, la probabilidad de que nuestros seis números sean iguales a los extraídos del bombo:

$$\frac{6}{64}\frac{5}{63}\frac{4}{62}\frac{3}{61}\frac{2}{60}\frac{1}{59} = 1.33 \times 10^{-8}$$

También puedes razonarlo de esta forma: hay 1 caso favorable, y el número de casos posibles es el número de subconjuntos de 6 elementos que se puede formar con las 64 bolas, es decir, el número de combinaciones de 64 elementos tomados de 6 en 6. Por tanto, la probabilidad es

```
1/choose(64,6)
```

```
## [1] 1.33e-08
```

• **Ejercicio 36, pág. 37**
Tengamos en cuenta que existen dos tipos de fichas en el dominó: las dobles (hay 7: del blanco al 6) y las que no son dobles (hay 21).

- El número de casos posibles es el número de formas en que podemos extraer 2 fichas de las 28, es decir, en número de combinaciones de 28 elementos tomados de 2 en 2

$$\frac{28!}{2!(28-2)!} = \frac{28 \times 27}{2} = 378$$

- Para contar los casos favorables, hay que tener en cuenta que
  - Una ficha doble se puede emparejar con otras seis fichas. Como hay siete dobles tenemos un total de $6 \times 7 = 42$ emparejamientos.
  - Una ficha no doble se puede emparejar con 12, como hay 21 fichas no dobles, tenemos otros $12 \times 21 = 252$ emparejamientos.
  - Como hemos contando dos veces cada emparejamiento dobles, tendremos: $(42+252)/2 = 147$ emparejamientos favorables

Finalmente, la probabilidad pedida es

$$147/378 = 0.39$$

- **Ejercicio 37, pág. 37**

1. Los casos favorables son las variaciones de 10 elementos tomados de 4 en 4, y hay 10000 casos posibles (si contamos el 0000)

```
(10*9*8*7)/10000

## [1] 0.504
```

2. Es el evento complementario del anterior, es decir

```
1-(10*9*8*7)/10000

## [1] 0.496
```

3. Por un lado, hay 10000 casos posibles. Vamos a contar los favorables: para cada una de las 10 cifras (del 0 al 9) hay $\binom{4}{2}$ formas en las que cada una de ellas puede aparecer repetida 2 veces en un número de 4 dígitos (si no lo ves claro, puedes volver sobre el ejercicio 34 (pág 37). En cada caso, habremos usado un número del 0 al 9, de manera que disponemos de 9 números con los que rellenar las dos cifras restantes (sin repetirlos), lo que puede hacerse de 98 formas distintas (variaciones de 9 elementos de orden 2) porque el orden importa: 2298 es diferente de 2289.

   Resumiendo, la probabilidad pedida es

```
(10*choose(4,2)*9*8)/10000

## [1] 0.432
```

4. De nuevo, hay 10000 casos posibles. Razonando como en el apartado anterior, hay $\binom{4}{2}$ formas en las que cada una de ellas puede aparecer repetida 2 veces en un número de 4 dígitos. Y los dos dígitos restantes sólo los podemos completar de 9 formas diferentes (porque hay 9 cifras del 0 al 9 que son diferentes de la que ya aparece). Por tanto, la probabilidad pedida es

```
(10*choose(4,2)*9)/10000

## [1] 0.054
```

5. Hay $\binom{4}{3}$ posibilidades de obtener tres cifras iguales (puesto que el orden no nos preocupa). La probabilidad de obtener un número de los diez posibles es (1/10). Por tanto, la probabilidad de obtener tres cifras iguales es:

```
choose(4,3)*(1/10)^3*(9/10)

## [1] 0.0036
```

6. Hay 10 casos favorables (0000, 1111,..., 9999) y 10000 posibles, por lo que la probabilidad es

$$10/10000 = 0.001$$

• **Ejercicio 38, pág. 37**
La probabilidad de que al menos dos personas cumplan años el mismo día es $P(A_n) = 1 - P(\bar{A}_n)$ donde $P(\bar{A}_n)$ es la probabilidad de que ninguna de las n personas cumpla años el mismo día. Para que $P(A_2) \geq 1/2$, se tiene que cumplir que $P(\bar{A}_n) \leq 1/2$. Si n es el número de personas, entonces

$$P(\bar{A}_n) = \frac{366}{366}\frac{365}{366}\frac{364}{366}\cdots\frac{366-n+1}{366}$$

o, agrupando términos

$$\frac{366!}{366^n(366-n)!}$$

Hay que ir probando con con distintos valores de n hasta dar con el primero para el que se cumple la condición requerida:

```
# probamos con, aproximadamente 366/2 días
m =180
n = 1:m
# si no sabes qué pace la función prod(), usa el tabulador!
prod((366-n)/366)

## [1] 2.28e-24

# elnúmero debe ser menor.
# probamos con (aprók.) el punto medio del rango de días 2-180
m =90
n = 1:m
prod((366-n)/366)

## [1] 0.00000481

# y así sucesivamente, hasta aislar el valor deseado
m =35
n = 1:m
prod((366-n)/366)

## [1] 0.169

m =17
n = 1:m
prod((366-n)/366)

## [1] 0.654

m =26
n = 1:m
prod((366-n)/366)

## [1] 0.374

m =21
n = 1:m
prod((366-n)/366)

## [1] 0.525

m =22
n = 1:m
prod((366-n)/366)

## [1] 0.494
```

Para n=22, $P(A_n)$ vale

```
m =22
n = 1:m
1-prod((366-n)/366)

## [1] 0.506
```

Si hay 22 personas en la sala, tenemos una probabilidad de 1/2 de que haya dos que hayan nacido el mismo día.

Observa que un intento de cálculo "directo" es inviable

```
factorial(366)

## Warning in factorial(366): value out of range in 'gammafn'

## [1] Inf
```

Fin del Tutorial-03. ¡Gracias por la atención!