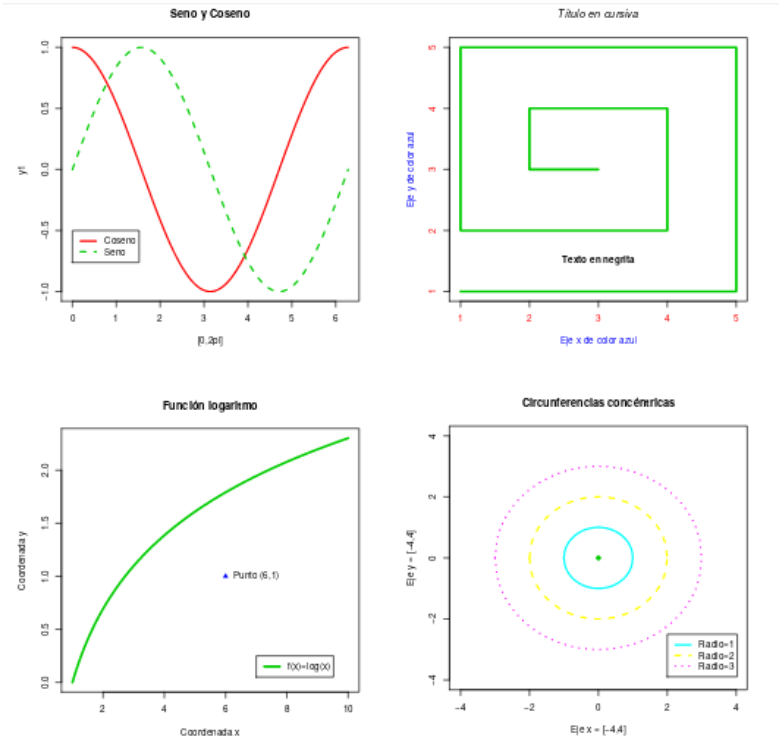
Lecturas Importantes

1. Artículo de Karlijn Willems, sobre Jupyter and R Markdown: Notebooks con R
   `https://www.datacamp.com/community/blog/jupyter-notebook-r`.

Preguntas

1. Supongamos que tenemos un conjunto de valores numéricos x de un vector X y un conjunto diferente de valores numéricos y en un vector Y.
   - Escribe un programa en R, de cómo calcular la distancia mínima entre un valor x y un valor y: $\min_{i,j} |x_i - y_j|$.
   - Determina el par de índices (i, j) para los que la distancia mínima definida anteriormente se alcanza.

2. Reproduce en lo posible las siguientes gráficas. Utiliza las funciones `plot` y `par`.



3. Pedro y Pablo juegan un juego que implica lanzamientos repetidos de una moneda. En un lanzamiento dado, si se observan caras, Pedro gana \$1 ; de lo contrario, Pedro le da \$1 a Pablo. Si Pedro comienza con cero dolares y se han realizado 50 lanzamientos.

```
> options(width=60)
> sample(c(-1, 1), size=50, replace=TRUE)
>
>
> win = sample(c(-1, 1), size=50, replace=TRUE)
```

```
> cum.win = cumsum(win)
> cum.win
>
> par(mfrow=c(2, 2))
> for(j in 1:4){
+   win = sample(c(-1, 1), size=50, replace=TRUE)
+   plot(cumsum(win), type="l" ,ylim=c(-15, 15))
+   abline(h=0)
+ }
```

Reescribe el siguiente código. Responde a las siguientes preguntas, adjuntando la parte del código que soporta tu respuesta.

- ¿ Cuál es la probabilidad de que Pedro se retire incluso después de 100 lanzamientos?
- ¿ Cuál es el número probable de lanzamientos en los que Pedro estará ganando?
- ¿ Cuál será el valor de la mejor ganancia de Pedro durante el juego?

4. (a) Escribe las funciones `func1` y `func2` tal que si `xVect` es el vector $(x_1, x_2, \ldots, x_n)$, entonces `func1(xVect)` retorna el vector $(x_1, x_2^2, \ldots, x^n)$ y `func2(xVect)` retorna el vector $\left(x_1, \frac{x_2^2}{2}, \ldots \frac{x_n^n}{n}\right)$.

   (b) Escribe una función `func3` que toma 2 argumentos x y n donde x es un número y n es un entero positivo estricto. La función debe retornar el valor de:

$$1 + \frac{x}{1} + \frac{x^2}{2} + \frac{x^3}{3} + \cdots + \frac{x^n}{n}.$$

5. Un Sudoku es una cuadrícula 9×9 que se completa de números del 1 al 9 de manera que cualquier número entre 1 y 9 sólo aparece una vez en una fila, una columna o un bloque de 3×3 de la cuadrícula. Este ejercicio resuelve una cuadrícula de Sudoku simple donde las entradas vacías se llenan una a la vez mediante la exclusión de todas las posibilidades una vez.

   El Sudoku que resolvemos está dado por:

```
> s=matrix(0,ncol=9,nrow=9)
> s[1,c(6,8)]=c(6,4)
> s[2,c(1:3,8)]=c(2,7,9,5)
> s[3,c(2,4,9)]=c(5,8,2)
> s[4,3:4]=c(2,6)
> s[6,c(3,5,7:9)]=c(1,9,6,7,3)
> s[7,c(1,3:4,7)]=c(8,5,2,4)
> s[8,c(1,8:9)]=c(3,8,5)
> s[9,c(1,7,9)]=c(6,9,1)
```

   (a) Imprime la cuadrícula en la pantalla.

   (b) Definimos el array `pool= array (TRUE, dim = c(9,9,9))` de posibles valores para cada entrada (i, j) de la cuadrícula, `pool[i, j, k]` es FALSE si el valor de k puede ser excluido. Escribe el código R que actualiza `pool` para las entradas ya llenas.

   (c) Si i es un entero entre 1 y 81, explica el significado de $s[i]$.

   (d) Demuestra que, para una entrada dada (a, b), los índices de los números enteros en el mismo cuadro de 3×3 como (a, b) se definen por

```
> boxa=3*trunc((a-1)/3)+1
> boxa=boxa:(boxa+2)
> boxb=3*trunc((b-1)/3)+1
> boxb=boxb:(boxb+2)
```

(e) Deduce que los valores de una entrada (a, b) que todavia no están determinados se puede excluir por

```
> for (u in (1:9)[pool[a,b,]])
+   pool[a,b,u]=(sum(u==s[a,])+sum(u==s[,b])+
+       sum(u==s[boxa,boxb]))==0
```

y que ciertas entradas corresponden a

```
> if (sum(pool[a,b,])==1) s[i]=(1:9)[pool[a,b,]]
```

(f) Resolver la cuadrícula con una exploración aleatoria de entradas (a, b) que sigue, siempre y cuando la `sum(s == 0)> 0`.

6. Considere la siguiente curiosidad:

$$
\begin{aligned}
8 \times 8 + 13 &= 77 \\
8 \times 88 + 13 &= 717 \\
8 \times 888 + 13 &= 7117 \\
8 \times 8888 + 13 &= 71117 \\
8 \times 8888 + 13 &= 711117 \\
&\text{etc.}
\end{aligned}
$$

Escribe código R, que verifica las 10 primeras ecuaciones, imprimiendo los resultados, usando la fórmula dada

(a) usando el bucle `for`.

(b) usando una expresión vectorizada.

7. Supongamos que $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ denota el vector `xVect` y $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ denota el vector `yVect`.

(a) Crea el vector $(y_2 - x_1, \ldots, y_n - x_{n-1})$.

(b) Crea el vector $\left(\frac{\sin(y_1)}{\cos(x_2)}, \frac{\sin(y_2)}{\cos(x_3)}, \ldots, \frac{\sin(y_{n-1})}{\cos(x_n)}\right)$.

(c) Crea el vector $(x_1 + 2x_2 - x_3, x_2 + 2x_3 - x_4, \ldots, x_{n-2} + 2x_{n-1} - x_n)$.

(d) Calcula $\sum_{i=1}^{n-1} \frac{e^{-x_{i+1}}}{x_i + 10}$.

8. Escribe los programas correspondientes a los siguientes problemas:

(a) Un palillo se rompe al azar en 3 piezas. Escribe código en R que calcule y devuelva la probabilidad de que las piezas puedan formar un triángulo.

(b) Escribe una función que toma como un único argumento una matriz y que devuelve una matriz, que es la misma que el argumento, pero cada número impar es duplicado.

9. (a) Supongamos que x es un vector numérico. **Explica en detalle**, como las siguientes expresiones son evaluadas y que valores toman:

```
> sum(!is.na(x))
> c(x,x[-(1:length(x))])
> x[length(x) + 1]/length(x)
> sum(x > mean(x))
```

(b) Considera el siguiente problema: Dada una matriz numérica X, determinar el índice de la primera fila cuyos elementos son todos números positivos (y que no contiene NA valores). Resuelve este problema usando un bucle `for`.

(c) La matriz de Hilbert $n \times n$ tiene a los elementos (i, j) dados por $1/(i + j - 1)$.

- Escribe una función que muestra una matriz de Hilbert $n \times n$ como salida para entero positivo n.
- ¿ Son todas las matrices de Hilbert invertibles?.
- Usa `solve()` y `qr.solve()` para calcular la inversa de las matrices Hilbert, por ejemplo, cuando $n = 10$.

10. Suponga que ha sido seleccionado para ayudar a Rafael Nadal para calcular la probabilidad de que gane un juego, un set o un partido. Como es sabido que usted es un científico, conocedor de matemáticas y programación, se le darán dos trabajos:

(a) Halle una expresión (puede ser recursiva o iterativa) para calcular lo deseado.

(b) Escribe un algoritmo en base a su expresión para dar una respuesta para algún valor.

Si es que no se acuerda de las reglas del tenis, las explicaremos en detalle:

- Punto: Cada punto es iniciado por el servicio de alguno de los jugadores y cuando uno de los jugadores falle en realizar una jugada legal, pierde la anotación.
- Juego: Cada juego tiene un sistema de puntuación de $15 - 30 - 40$ con cada anotación y un sistema de deduce en el caso de un empate $40 - 40$. El sistema en este caso indica que los puntos deben tener una diferencia de 2 entre sí para poder ganar, por lo que si un jugador hace una anotación más se va al $D - 40$ y con otra del mismo, ganaría el juego. En el mismo caso, si es que el jugador rival obtiene un punto, entonces los puntajes regresan a $40 - 40$. Algo como lo siguiente se daría: $J1$ anota. Puntuación: $D - 40$. $J2$ anota. Puntuación: $40 - 40$. $J1$ anota. Puntuación: $D - 40$. $J1$ anota. Fin del juego, gana $J1$.
- Set: Un jugador gana un set si es que gana al menos 6 juegos y lleva una ventaja a su rival de al menos 2 juegos. Al igual que en el caso del juego, se usa un sistema de tiebreak para determinar al ganador en el caso de que se llegue a dar una puntuación de $5 - 5$. En este caso, gana el que llegue a $7 - 5$. En el caso de que de $5 - 5$ se vayan a $6 - 6$, se usa el sistema tiebreak de 12 puntos.
- Tiebreak (12 puntos): Con este juego se determina cuál de los dos jugadores será el ganador del set, en este juego especial se dan las mismas condiciones que en el juego simple, pero para poder ganar se deben realizar como mínimo 7 puntos y tener una ventaja de 2 como mínimo, usando el sistema de deduce del juego simple.
- Partido: El ganador de un partido es el primero en ganar 2 sets.

Rafael sabe que su probabilidad de obtener un punto es de p sin importar si es él el que sirve o no.

11. Este script simula la probabilidad de obtener 3 caras en lanzamientos de monedas, está divido en 3 partes: Escribe el código para la prueba, determina el éxito de la prueba y implementa la replicación. El número 1 representa las caras y 0 los sellos.

```
> #Prueba Experimental
>
> prueba <- sample(0:1, 3, replace=TRUE)
> # Exito
>
> if (sum(prueba )==3) 1 else 0
> # Repeticion
>
> n <- 10000   # Numero de iteraciones
> simlista <- replicate(n, 0) ## Inicializa la lista con 0's
```

```
> for (i in 1:n)
+ {
+         prueba <- sample(0:1, 3, replace=TRUE)
+         exito <- if (sum(prueba )==3) 1 else 0
+         simlista[i] <- exito
+ }
> # Resultado simulado
> mean(simlista)
```

Modifica el código anterior para simular la probabilidad de obtener exactamente una cara en cuatro lanzamientos de moneda.