

Clase 2: Trabajo con datos

Conceptos clave de programación para análisis de datos
o automatización de procesos

Fundación SOL

lunes, 12 may , 2025

Antecedentes de la programación

- Una parte relevante de la historia de la programación moderna ha sido su aplicación para el desarrollo tecnológico en la industria, la administración del Estado y la guerra, e incluso el arte.
- Un buen ejemplo es uno de los antecedentes directos de los ordenadores modernos, el *Telar de Jacquard* que utilizaba códigos en tarjetas perforadas para definir diseños o el *Motor Analítico* de Babbage.



Figura 1: Telar de Jacquard (1801) - Nota de prensa ([Science and Industry Museum - Manchester](#)).



Fabricante de tarjetas que utiliza una máquina para traducir un patrón en tarjetas perforadas, c. 1950.

Colección del Grupo del Museo de Ciencias,
fotografía con permiso de Garth Dawson,
Accrington.

Figura 2: Muestras de tejidos Telar de Jacquard (1801) - Nota de prensa ([Science and Industry Museum - Manchester](#)).



Figura 3: Muestras de tarjetas Telar de Jacquard (1801) - Nota de prensa ([Science and Industry Museum - Manchester](#)).



Figura 4: Muestras de tejidos Telar de Jacquard (1801) - Nota de prensa ([Science and Industry Museum - Manchester](#)).

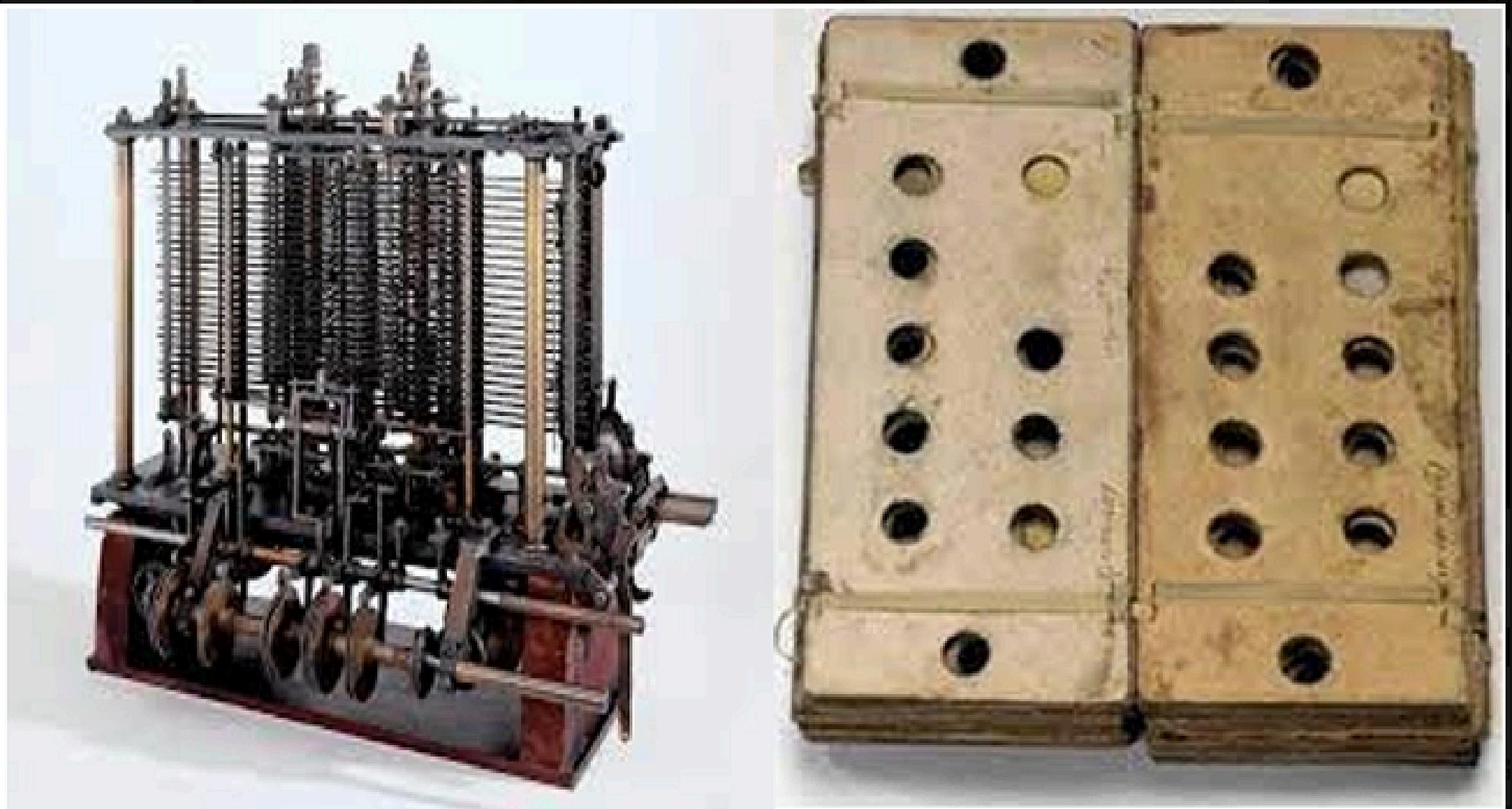


Figura 5: Charles Babbage - Analytical Engine (1837).

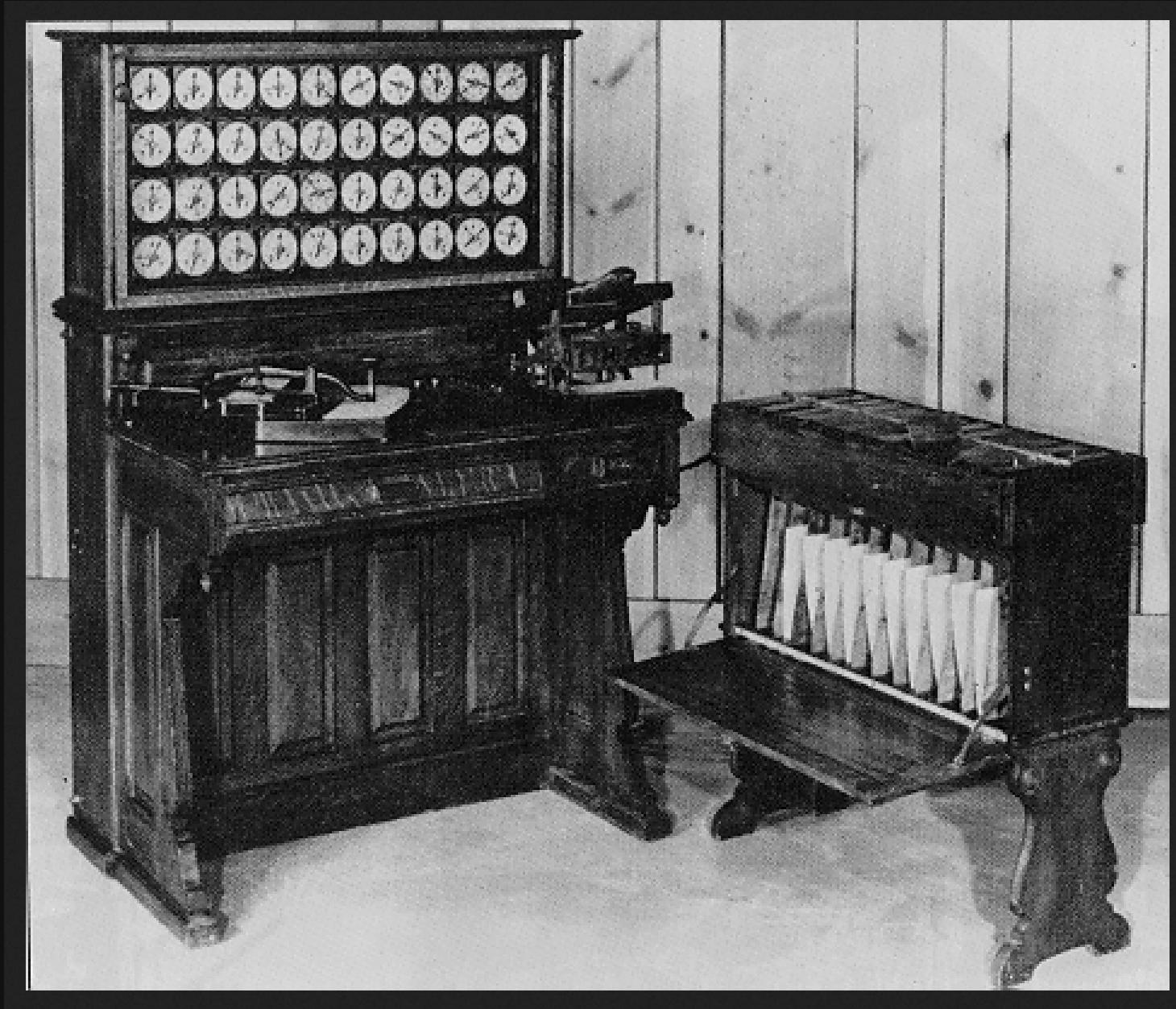


Figura 6: Máquina tabuladora Censo (1890) - Hollerith (Columbia Edu).

Hollerith y el Censo de 1890

- La máquina de Hollerith permitió a la oficina de estadísticas de Estados Unidos finalizar el censo en 6 meses y 2 años de análisis de datos. En referencia al censo de 1880 fue un éxito considerando que este demoró 8 años.
- El uso de tarjetas perforadas se mantuvo en el desarrollo de la computación, programación y análisis de datos, con aplicaciones concretas. Una de las empresas asociadas al uso de esta máquina fue la multinacional IBM.

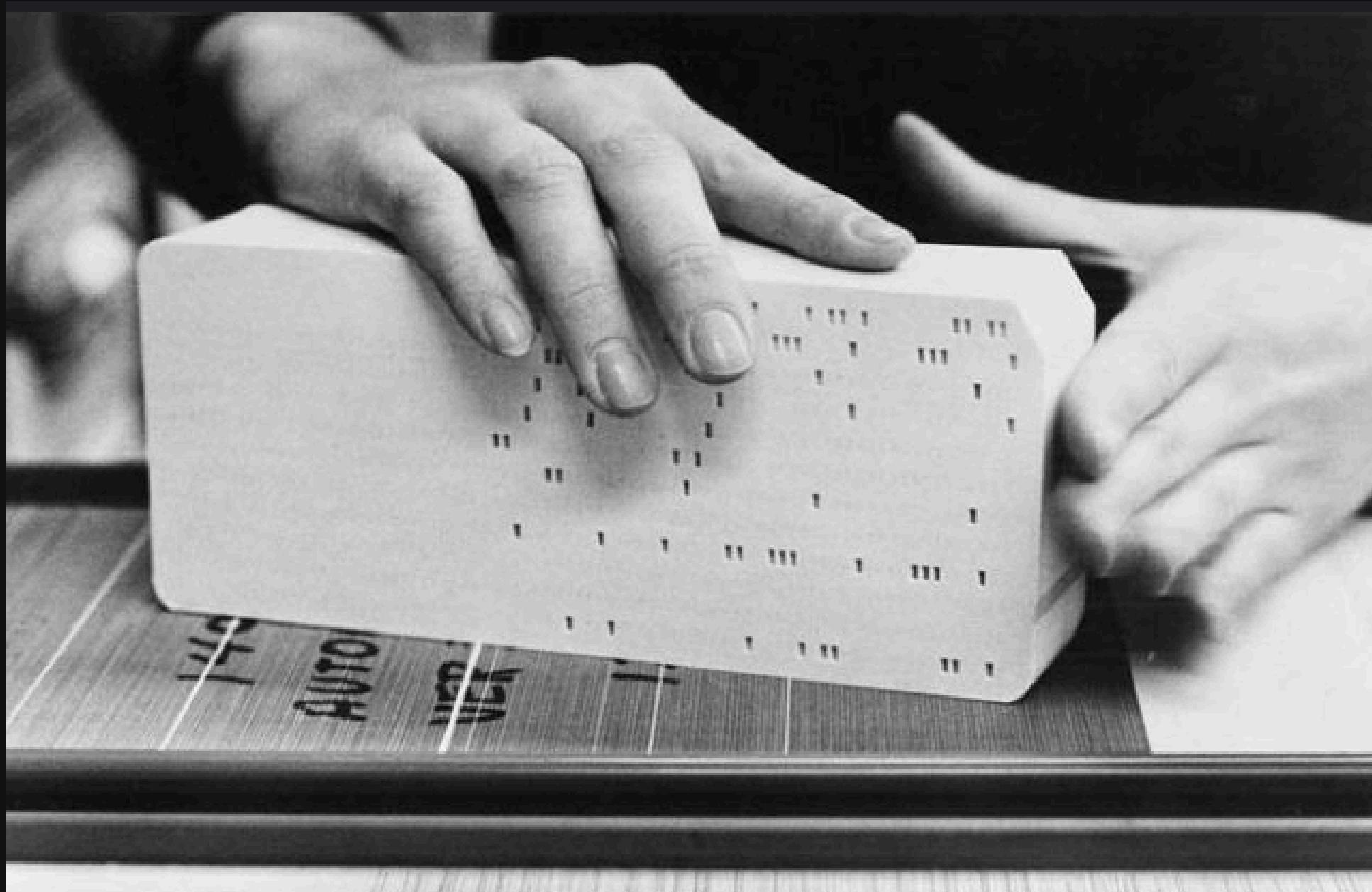


Figura 7: Tarjetas perforadas - IBM (siglo XX).

La programación y sus aplicaciones

Programación y algoritmos

- El primer algoritmo escrito directamente para una computación automatizada es el algoritmo publicado por **Ada Lovelace** en 1843 para el cálculo de números de Bernoulli.
- Charles Babbage fue invitado a explicar el funcionamiento de la *Analytical Engine* y un matemático, Luigi Menabrea, escribió una nota al respecto en francés. Se pidió a Ada la traducción y que añadiera notas. Estas eran 3 veces más largas que el artículo original.

Programación y algoritmos

- En la Nota G, Lovelace describe como puede hacerse uso de esta máquina, mediante su operación binaria, para realizar cómputos avanzados.
- La ecuación propuesta era recursiva, pues utilizaba la información de una operación para realizar la siguiente.
- Según la autora, al hacer un uso binario de las tarjetas perforadas, se abrían posibilidades como el componer y elaborar *piezas científicas de música* de cualquier grado de complejidad.



Figura 8: Ada Lovelace - 1834

¿Cómo utilizar funciones y algoritmos?

Funciones en R

- En este caso entenderemos una función como un código para ser usado con una variedad de inputs y evita que debamos escribir el mismo código una y otra vez.
- Recordemos que R es un lenguaje de programación entre otros lenguajes existentes como Python, C, etc.
- Cada lenguaje de programación tiene reglas y formas de programación que definen estilos y formas particulares.

Estructura de una función básica

```
1 nombre<-function(variables){  
2   cuerpo  
3 }
```

Ejemplo con dataset df_personas

```
1 df_personas <- data.frame(  
2   ID = 1:10,  
3   Nombre = c("Ana", "Luis", "Eva", "Diego", "Juan", "Marcela", "Diego", "Jocelyn", "Raquel",  
4   Edad = c(28, 34, 39, 45, 14, 65, 52, 27, 62, 28),  
5   Ocupación = c(TRUE, TRUE, FALSE, TRUE, FALSE, FALSE, TRUE, FALSE, FALSE, TRUE),  
6   Ingreso = c(510636, 600000, 80000, 420000, 224004, 70000, 530000, 50000, 224004, 380000),  
7   Gasto = c(700000, 900000, 120000, 920000, 200000, 300000, 1130000, 600000, 700000, 880000)  
8 )
```

Función ejemplo para estadísticas de resumen

```
1 #Escribimos la función
2 resumen <- function(vector){
3     maximo <- max(vector)
4     media <- mean(vector)
5     mediana <- median(vector)
6     minimo <- min(vector)
7     return(c(maximo,media,mediana,minimo))
8 }
```

Función aplicamos función a vectores edad, ingreso y gasto

```
1 #Aplicamos sobre el vector  
2 edad<-as.vector(df_personas$Edad)  
3 print(resumen(edad))
```

```
[1] 65.0 39.4 36.5 14.0
```

```
1 ingreso<-as.vector(df_personas$Ingreso)  
2 print(resumen(ingreso))
```

```
[1] 600000.0 308864.4 302002.0 50000.0
```

```
1 gasto<-as.vector(df_personas$Gasto)  
2 print(resumen(gasto))
```

```
[1] 1130000 645000 700000 120000
```

Función intervalo de confianza

```
1 mean_ci <- function(x, conf = 0.95) {  
2   se <- sd(x) / sqrt(length(x))  
3   alpha <- 1 - conf  
4   mean(x) + se * qnorm(c(alpha / 2, 1 - alpha / 2))  
5 }
```

```
1 mean_ci(edad)
```

```
[1] 29.1856 49.6144
```

```
1 mean_ci(ingreso)
```

```
[1] 181068.2 436660.6
```

```
1 mean_ci(gasto)
```

```
[1] 435137.4 854862.6
```

Funciones anidadas

```
1 resumen <- function(vector){  
2   mean_ci <- function(x, conf = 0.95) {  
3     se <- sd(x) / sqrt(length(x))  
4     alpha <- 1 - conf  
5     mean(x) + se * qnorm(c(alpha / 2, 1 - alpha / 2))  
6   }  
7   interval <-mean_ci(vector)  
8   maximo <-max(vector)  
9   media <-mean(vector)  
10  mediana <- median(vector)  
11  minimo <- min(vector)  
12  return(c(maximo,media,mediana,minimo,interval))  
13 }
```

```
1 print(resumen(edad))
```

```
[1] 65.0000 39.4000 36.5000 14.0000 29.1856 49.6144
```

```
1 print(resumen(ingreso))
```

```
[1] 600000.0 308864.4 302002.0 50000.0 181068.2 436660.6
```

```
1 print(resumen(gasto))
```

```
[1] 1130000.0 645000.0 700000.0 120000.0 435137.4 854862.6
```

Aplicar una función a una lista de vectores

```
1 lista<-list(a=edad,b=ingreso,c=gasto)
2 lapply(lista,resumen)
```

```
$a
[1] 65.0000 39.4000 36.5000 14.0000 29.1856 49.6144

$b
[1] 600000.0 308864.4 302002.0 50000.0 181068.2 436660.6

$c
[1] 1130000.0 645000.0 700000.0 120000.0 435137.4 854862.6
```

¿Qué es un loop?

```
1 for (i in 1:5) {  
2   print(i)  
3 }
```

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

Siguiendo el ejemplo anterior, en lugar de lapply con un loop

```
1 for (element in lista) {  
2   print(resumen(element))  
3 }
```

```
[1] 65.0000 39.4000 36.5000 14.0000 29.1856 49.6144  
[1] 600000.0 308864.4 302002.0 50000.0 181068.2 436660.6  
[1] 1130000.0 645000.0 700000.0 120000.0 435137.4 854862.6
```

Aplicar función según condición

```
1 ocupa <- ifelse(df_personas$Ocupación == TRUE, "Ocupado/a", "No ocupado/a")  
2 ocupa2 <- as.vector(df_personas$Ocupación)  
3 ocupa2
```

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