OBJETIVOS

- **Revisar los principales conceptos aprendidos**
  - **Leer / escribir archivos**
  - **Extraer datos de matrices y vectores**
  - **Trazar datos (series de líneas y nubes de puntos)**
  - **Imprimir mensajes formateados**
  - **Funciones**
  - **Estructuras condicionales**

EJERCICIO 7.1 VOLUMEN DE CILINDROS

Queremos desarrollar una versión mejorada de la función que calcula los volúmenes de varios cilindros, que implementamos en el laboratorio 4. Esta vez, la función solo considerará las dimensiones válidas (radio y altura) de los cilindros para calcular los volúmenes ($v = \pi r^2 h$) . Además, devolverá los índices (posiciones) de las medidas incorrectas (radio negativo o alturas negativas).

```
function [vol,avg,stddev,wrongMeasures] = volumenCilindro(r, h)
```

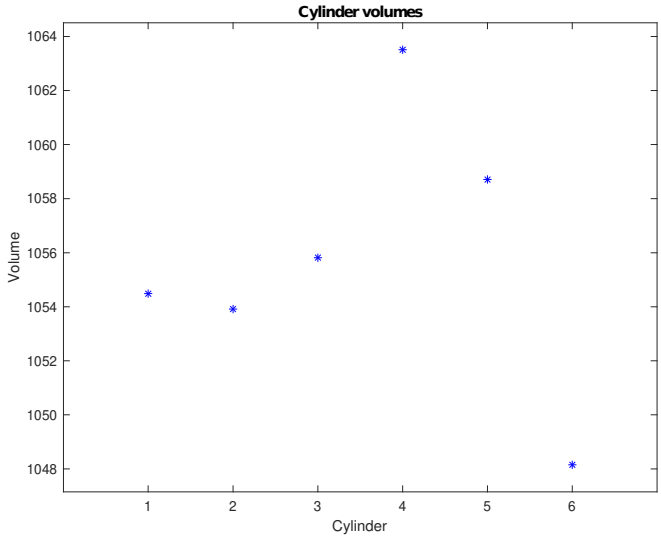
Una vez que haya implementado la función, escriba un script llamado testCilindro.m que defina los vectores que representan el radio y las alturas de los volúmenes, muestre los volúmenes de los cilindros en una gráfica (vea la figura a continuación) e imprima el volumen promedio junto con la desviación estándar y las medidas incorrectas.

Por ejemplo, los siguientes radios y alturas:

```
r = [5.499 -5.498 5.5 5.5 5.52 5.51 5.5 5.48];
h = [11.1 11.12 11.09 11.11 11.11 11.1 -11.08 11.11];
```

Deberíamos obtener el siguiente resultado:

```
Media: 1055.77  Desviación estandar: 5.13
Medidas incorrectas:      2     7
```



EJERCICIO 7.2 GRAFICO DE LOS DATOS DE ARCHIVO

Escriba un script que pregunte a los usuarios qué archivos quieren mostrar en un gráfico. Si el archivo no existe, el script mostrará un mensaje de error. De lo contrario, le pedirá al usuario las r columnas que deben mostrarse en el gráfico (cada una de las cuales se mostrará en un color diferente). Si todas las columnas que especificaron los usuarios son válidas, se mostrará la figura. De lo contrario, si al menos una de las columnas no existe, se debe mostrar un mensaje de error.

EJERCICIO 7.3 TEMPERATURA MÁXIMA

Escribe una función llamada temperaturaMaxDia. Esta función tiene tres parámetros, un vector que representa los días del mes en que se midieron las temperaturas, un vector que contiene las temperaturas y un vector que especifica el día de la semana correspondiente a las fechas (1 se refiere al lunes y 7 se refiere a Domingo). La función devuelve tres valores, la temperatura máxima, el día del mes en que se midió la temperatura y el nombre correspondiente del día de la semana. En caso de que la temperatura máxima se mida varios días, se deberá devolver el primero de ellos.

```
function [maxTem,dayOfMonth,dayOfWeek] = temperaturaMaxDia(diasMes,
```

```
                                                    temperaturas, diaSemana)
```

Por ejemplo, la siguiente llamada a la función

```
[t, d, n] = temperaturaMaxDia([1 3 7 9 12], [18.2 15.3 16.2 15.0
                            14.1], [5 7 4 6 2])
```

devolverá

```
t = 18.2

d = 1

n = “Viernes”
```

EXERCISE 7.4 ¿SE PUEDEN MULTIPLICAR LAS DOS MATRICES?

Muchos problemas pueden requerir que se multipliquen dos matrices (o incluso más). Sin embargo, esta operación puede hacer que nuestro programa se bloquee si las matrices no son adecuadas para la multiplicación. Escriba una función llamada canMultiplyMatrices, que dadas dos matrices, devuelve si las matrices se pueden multiplicar o no y un mensaje de error.

```
function [posible,error] = canMultiplyMatrices(matrix1,matrix2)
```

Se pueden multiplicar dos matrices si no están vacías (es decir, tienen un número positivo de filas y columnas) y el número de columnas de la primera matriz es igual al número de filas de la segunda matriz. El mensaje de error será "matriz vacía" siempre que una de las matrices de entrada esté vacía, "Sin matrices numéricas" cuando alguna de las matrices contenga un elemento no numérico o "Matrices incompatibles" cuando el número de columnas de la primera matriz no lo haga corresponden al número de filas de la segunda matriz.

Ejemplos:

```
m1 = [[];[]; []];
m2 = ones(3,3);
[p, e] = canMultiplyMatrices(m1,m2)
p = logical 0
e = “matriz vacia”

m3 = randi([0 10], 3, 4);
m4 = randi([0 10], 3, 4);
[p, e] = canMultiplyMatrices(m3,m4)
p = logical 0
e = “Matrices incompatibles”

m5 = ["a" "b" "c"];
m6 = randi([0 100], 4, 2);
```

```
[p, e] = canMultiplyMatrices(m4,m5)

p = logical 0
e = “Sin matrices numéricas”

m5 = randi([0 10], 3, 4);

m6 = randi([0 10], 4, 2);

[p, e] = canMultiplyMatrices(m5,m6)

p = logical 1

e = “”
```

EJERCICIO 7.5 EVOLUCION COVID

En este ejercicio queremos analizar la evolución del COVID. Para ello se debe implementar la función calculateIncidenceRate14Days, que dado un vector que representa el número de casos positivos detectados en diferentes localizaciones del territorio en los últimos 14 días y la población que se encuentra sana, devuelve la tasa de incidencia acumulada y el color correspondiente. para el territorio.

La tasa de incidencia acumulada se calcula con la siguiente fórmula[1]

$$rate = \frac{totalCases14Days}{population} * 100000$$

donde totalCases14Days es la suma de los casos positivos detectados en los últimos 14 días y la población se refiere a los habitantes sanos. El color está determinado por los siguientes criterios:

- Si la tasa de incidencia acumulada es inferior a 60, el color será "Verde"
- Si la tasa de incidencia acumulada está en el rango [60, 300), el color será "Amarillo"
- Si la tasa de incidencia acumulada está en el rango [300, 500), el color será "Naranja"
- Si la tasa de incidencia acumulada es mayor o igual a 500, el color será "Rojo"

Ejemplos:

```
[r,c] = calculateIncidenceRate14Days([74 79 112 139 132 126 85 108 149 169 149
                                      133 107 112],2199711)
r = 76.1009
c = 'Amarillo
```

1 This formula has been simplified for the lab

```
[r,c] = calculateIncidenceRate14Days([198 236 213 282 296 334 341 288 265 293
                                      456 397 389 355],720459)
r = 602.8102
c = 'Rojo
```