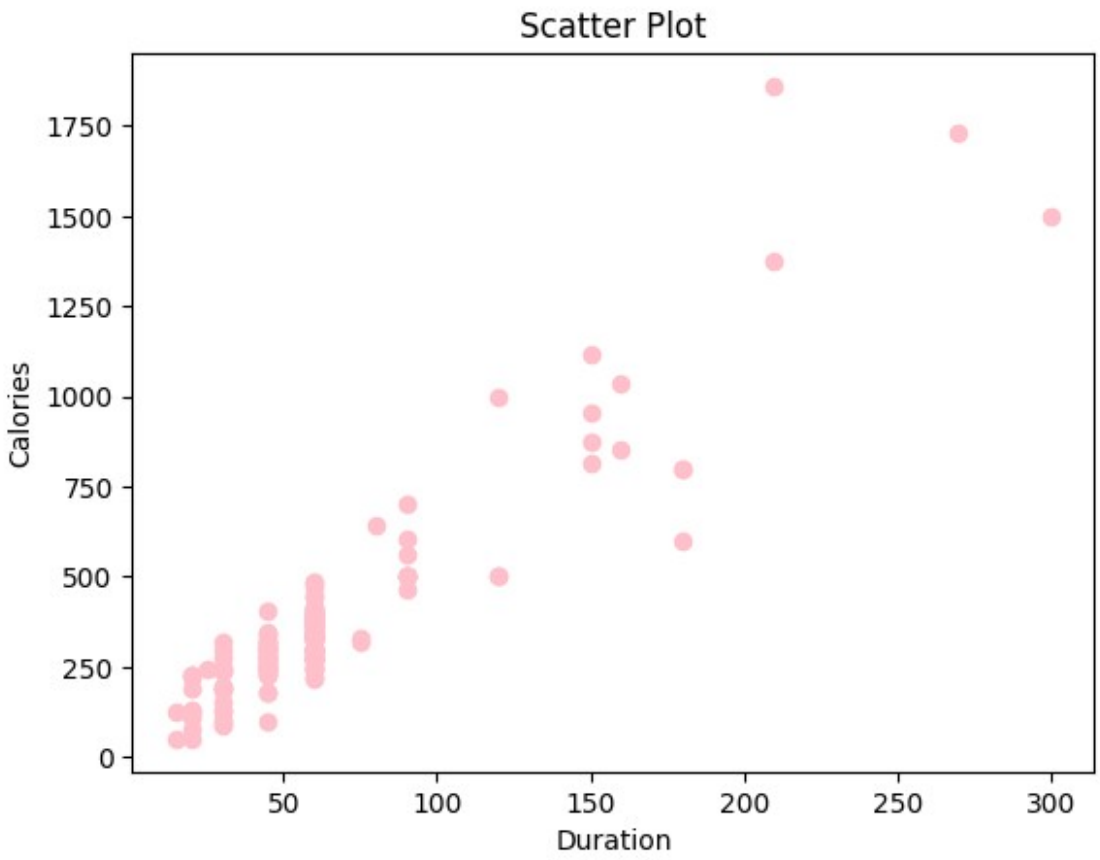
```
import pandas as pd
df =
pd.read_csv('https://raw.githubusercontent.com/yessss28/Estadistica/
refs/heads/main/data.csv')
df
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 169,\n  \"fields\": [\
n    {\n      \"column\": \"Duration\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 42,\n        \"min\": 15,\n
\"max\": 300,\n        \"num_unique_values\": 16,\n
\"samples\": [\n          60,\n          45,\n          210\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Pulse\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 14,\n        \"min\": 80,\n        \"max\": 159,\n
\"num_unique_values\": 47,\n        \"samples\": [\n          159,\n
112,\n          153\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Maxpulse\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 16,\n        \"min\": 100,\n
\"max\": 184,\n        \"num_unique_values\": 57,\n
\"samples\": [\n          130,\n          127,\n          146\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Calories\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
266.37991924435164,\n        \"min\": 50.3,\n        \"max\": 1860.4,\
n        \"num_unique_values\": 142,\n        \"samples\": [\n
328.0,\n          282.0,\n          229.4\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"df"}
```

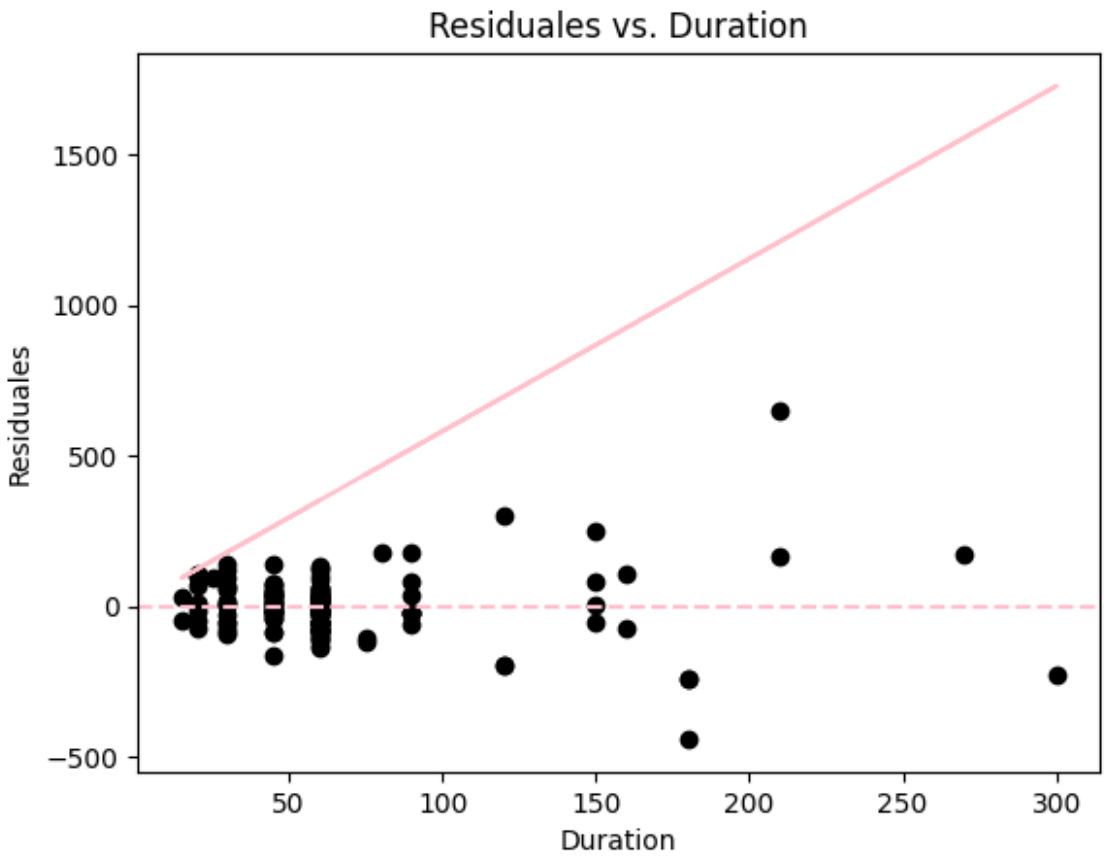

```
# a) Establezca una variable dependiente (Y) y una variable
dependiente (X).

import pandas as pd
df = pd.read_csv
('https://raw.githubusercontent.com/yessss28/Estadistica/refs/heads/
main/data.csv')
# eliminar registros cpn valores faltantes
df.dropna(inplace=True)

X = df['Duration'] # variable independiente
Y = df['Calories'] # variable dependiente

# b) Realiza un gráfico de dispersión y la recta de regresión
ajustada.
import matplotlib.pyplot as plt
plt.scatter(X, Y, color = 'pink')
plt.xlabel('Duration')
```

```
plt.ylabel('Calories')
plt.title('Scatter Plot')
plt.show()
# recta de regreción lineal.
import statsmodels.api as sm
X_constant = sm.add_constant(X)
model = sm.OLS(Y, X_constant). fit()

b0, b1 = model.params
Fun = lambda x: b0 + b1 * x

Yc = Fun(X)

plt.plot(X, Yc, color = 'pink')

# C) Calcula el coeficiente de correlación y el coeficiente de
determinación e interpreta los resultados.

from scipy.stats import pearsonr

r,_ = pearsonr (X, Y)
print(f'Coeficiente de correlación: {r:0.4f}/n')
print(f'Coeficiente de determinación: {r ** 2: 0.4f}/n')
# d) Obtén un intervalo de confianza de 98% para la pendiente e
interpreta el resultado. Respalda tu conclusión usando ANOVA.

nivel_de_confianza = 0.98
intervalo_de_confianza = model.conf_int(alpha = 1 -
nivel_de_confianza)
intervalo_de_confianza_b1 = intervalo_de_confianza.iloc[1]
print(f'Intervalo de confianza de {nivel_de_confianza * 100}% para la
pendiente:')
print(f'{intervalo_de_confianza_b1 [0] - intervalo_de_confianza_b1[1]:
0.4f}')

# Tabla ANOVA
from statsmodels.formula.api import ols
# Y - X
model = ols('Y ~ X', data = df).fit()
tabla_anova = sm.stats.anova_lm(model)
print(tabla_anova)

# e) Verifica los supuestos.
resudiales = model.resid
plt.scatter(X, resudiales, color = 'black')
plt.xlabel('Duration')
plt.ylabel('Residuales')
plt.title('Residuales vs. Duration')
ax = plt.gca()
ax.axhline(y = 0, color = 'pink', linestyle = '--')
```

```
plt.show()

from scipy.stats import shapiro
_, valor_p_shapiro = shapiro(resudiales)
print(f'Valor p de Shapiro-Wilk: {valor_p_shapiro: 0.4f}')

from statsmodels.stats.api import het_breuschpagan
_, valor_p_breuschpagan, _, _ = het_breuschpagan(resudiales,
X_constant)
print(f'Valor p de Breusch-Pagan: {valor_p_breuschpagan: 0.4f}')
```




```
Coeficiente de correlación: 0.9227/n
Coeficiente de determinación:  0.8514/n
Intervalo de confianza de 98.0% para la pendiente:
-0.8839
             df        sum_sq       mean_sq           F         PR(>F)
X           1.0  9.847530e+06  9.847530e+06  928.219489  5.795220e-69
Residual  162.0  1.718667e+06  1.060905e+04         NaN           NaN
```




```
Valor p de Shapiro-Wilk:  0.0000
Valor p de Breusch-Pagan:  0.0000
```