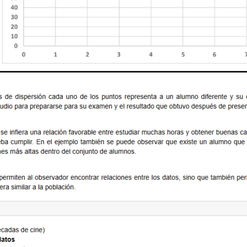
Gráficas de dispersión

Las **gráficas de dispersión** es la representación visual de una colección de puntos colocados en un plano utilizando coordenadas cartesianas que representan los valores de dos variables. Al mostrar una variable en cada eje, se puede detectar si existe una relación o correlación entre las dos variables.

Tiempo estudio	0.5	1.0	1.5	2.0	3.0	3.0	3.5	4.0	4.5	5.0	5.0	5.5	6.5
Nota obtenida	55	62	71	95	76	80	76	83	84	87	88	92	94	97



En este primer ejemplo de gráficas de dispersión cada uno de los puntos representa a un alumno diferente y su coordenada en la gráfica representa la intersección entre las horas que estudio para preparare para su examen y el resultado que obtuvo después de presentarlo.

A primera vista es muy posible que se infiera una relación favorable entre estudiar muchas horas y obtener buenas calificaciones, pero existen algunos casos donde lo anterior no siempre se deba cumplir. En el ejemplo también se puede observar que existe un alumno que estudio solamente dos horas y aún así logro obtener una de las calificaciones más altas dentro del conjunto de alumnos.

Las gráficas de dispersión no solo permiten al observador encontrar relaciones entre los datos, sino que también permiten que se identifiquen fácilmente los datos que no se comportan de manera similar a la población.

In [ ]:
```
import pandas as pd
```

Industria de las películas (Cuatro decadas de cine)
Visitar y conocer el conjunto de datos
https://www.kaggle.com/danielgrijalvas/movies

In [ ]:
```
df = pd.read_csv('data/movies/movies.csv') #dataset No actualizado, ejecutar la celda y observar el error
```

In [ ]:
```
df = pd.read_csv('data/movies/movies.csv', encoding = "cp1252") # Se usa codificación
```

In [ ]:
```
df.head()
```

In [ ]:
```
df.describe() # Se centra en las columnas donde hay números
```

In [ ]:
```
df = df.drop(df[df['budget'] == 0].index) # Borrar los datos que tienen presupuesto en cero
```

In [ ]:
```
df.describe()
```

In [ ]:
```
import matplotlib.pyplot as plt
#%matplotlib inline
```

In [ ]:
```
plt.scatter(df['budget'],df['gross']) # scatter es el equivalente a un gráfico de dispersión entre el presupuesto e ingresos bru
```

La gráfica representa en cada punto una película y más o menos cuanto invirtió y cuanto recaudo. No es muy precisa la información, pero visualmente vemos que la películas que invierten poco, poco recaudaron. Pero es mejor seguir investigando

Retorno de la Inversión

ROI = (Beneficio – Inversión) / Inversión

Usemos **La Ecuación del Retorno de Inversión por sus siglas ROI**

Nos dice: ROI = (beneficio - inversión) / inversión.

El beneficio que me trajo menos la inversión que yo gasté entre la inversión. Entonces el 'ROI' me va a decir el porcentaje (%) que gane, contra lo que inverti para el Data Frame en la columna 'ROI' que se va a agregar, va a ser igual al Data Frame de lo que me trajo de Beneficio menos (-) el Data Frame de lo que gasté entre (/) el Data Frame de lo que gasté y voy a centrar este estudio para filtrarlo de cierta manera entre los que tengan un ROI > 1.5 es decir, aquellos que tengan una recuperación de lo que invirtieron más el 50% como mínimo.

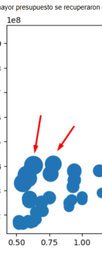
Aparte el Data Frame también va a estar filtrado por solamente las películas que tengan 7 de score o más y las películas que tengan un poquito más de 0.5e8 en Dólares.

In [ ]:
```
df['roi'] = (df['gross']-df['budget']) / df['budget'] # Se calcula el ROI

df = df[ (df['roi']>1.5) & (df['score']>7.0) & (df['budget']>0.5e8)] # Se aplica el filtro
```

In [ ]:
```
# Para graficar en X va el presupuesto, Y el beneficio y s= Size, tamaño del punto
plt.scatter(df['budget'],df['gross'], s=df['roi']*100) #s= roi * 100, un circulo grande las de mayor ROI
```

En general las películas de mayor presupuesto se recuperaron de manera proporcional, sin embargo, hay unas que invirtieron poco y el ROI fue alto.



La representación un poco más entendible pero, aún así, todos los circulos son del mismo color y no hay forma que sepamos qué película corresponde a cada círculo. En esta gráfica en particular vamos a colorear por géneros.

In [ ]:
```
for genre in df['genre'].unique(): # Filtra los géneros únicos
    data = df[df['genre'] == genre] # Por cada genero se calcula una nueva data y es contar para calcular el ROI
    plt.scatter(data['budget'],data['gross'],
                s=50*data['roi']**2,
                alpha = 0.5,
                label=genre)
plt.legend()
```

Las películas de acción tienen un presupuesto muy alto tienen una recuperación alta, pero el ROI es pequeño, muchos dolares invertidos pero pocas ganancias.

Observar el circulo de aventura que con poco dinero genero mucha taquilla y los dólares que se invirtieron fueron mejor aprovechados.

In [ ]:
```
for genre in df['genre'].unique():
    data = df[df['genre'] == genre]
    plt.scatter(data['budget'],
                data['gross'],
                s=50*data['roi']**2,
                alpha = 0.5,
                label=genre)

# Para cada etiqueta o rótulo se le va a asignar un tamaño de 100
lgnd = plt.legend()
for handle in lgnd.legendHandles:
    handle.set_sizes([100])
```

La gráfica anterior se dibuja correctamenet pero puede aparecer una advertencia, se invita a consultar la documentación y mejorar el código

In [ ]:
```
# ticker: permete dar formato  a Los ejes

from matplotlib.ticker import FuncFormatter ###---

fig, ax = plt.subplots() ###---

for genre in df['genre'].unique():
    data = df[df['genre'] == genre]
    plt.scatter(data['budget'],
                data['gross'],
                s=50*data['roi']**2,
                alpha = 0.5,
                label=genre)

lgnd = plt.legend()
for handle in lgnd.legendHandles:
    handle.set_sizes([100])

# Se define una función propia llamada millones
# Va a tomar un valor numérico y la va a devolver en un formato de un número antes del punto decimal acompañado
# de la palabra millones con el formato de un exponente '- 6' y lo va a cambiar de la forma tradicional de escritura,
# a la notación científica de millones.
def millones(x, pos):
    return '${:1.0f} M'.format(x*1e-6)

formatter = FuncFormatter(millones)
ax.xaxis.set_major_formatter(formatter)
ax.yaxis.set_major_formatter(formatter)
```

In [ ]:
```
from matplotlib.ticker import FuncFormatter

fig, ax = plt.subplots()

for genre in df['genre'].unique():
    data = df[df['genre'] == genre]
    plt.scatter(data['budget'],
                data['gross'],
                s=50*data['roi']**2,
                alpha = 0.5,
                label=genre)


lgnd = plt.legend()
for handle in lgnd.legendHandles:
    handle.set_sizes([100])

def millones(x, pos):
    return '${:1.0f} M'.format(x*1e-6)

formatter = FuncFormatter(millones)
# fig, ax = plt.subplots()
ax.xaxis.set_major_formatter(formatter)
ax.yaxis.set_major_formatter(formatter)


best_movies = df.sort_values('budget',ascending=False).head(5) # Se ordena de manera descendente y traer 5 valores
for index, row in best_movies.iterrows():
    plt.annotate(row['name'],
                 (row['budget'], row['gross']),
                 textcoords="offset points",
                 xytext=(0,5),
                 ha='right')
```

Se puede observar que las películas de acción tienen mayor inversión y mas taquilla, asi el ROI sea pequeño.

In [ ]:
```
from matplotlib.ticker import FuncFormatter

## Tamaño de la imagén o grafico
fig, ax = plt.subplots(figsize=(20,10))

for genre in df['genre'].unique():
    data = df[df['genre'] == genre]
    plt.scatter(data['budget'],
                data['gross'],
                s=100*data['roi']**3,
                alpha = 0.5,
                label=genre)


lgnd = plt.legend()

## Titulo de la gráfica
plt.xlabel('Presupuesto') # Eje X
plt.ylabel('Valor Taquilla') # Eje Y
plt.title('Películas más taquilleras (2016)') # Ver el archivo hasta que año tiene Valores

# O usar programación:
anio=(max(df.year))

plt.title('Películas más taquilleras ('+str(anio)+')')
for handle in lgnd.legendHandles:
    handle.set_sizes([100])

def millones(x, pos):
    return '${:1.0f} M'.format(x*1e-6)

formatter = FuncFormatter(millones)
# fig, ax = plt.subplots()
ax.xaxis.set_major_formatter(formatter)
ax.yaxis.set_major_formatter(formatter)


# Las películas se van a clasificar según taquilla
best_movies = df.sort_values('budget',ascending=False).head(10) # Se ordena de manera descendente y traer 10 valores
for index, row in best_movies.iterrows():
    plt.annotate(row['name'],
                 (row['budget'], row['gross']),
                 textcoords="offset points",
                 xytext=(0,5),
                 ha='right')
plt.savefig("dispersion.png",dpi=300)
```

Taller:

Descargar el archivo actualizado de peliculas y comparar los cambios de los últimos años

Realizar los mismos pasos pero con los datos de peliculas de Wikipedia:
https://es.wikipedia.org/wiki/Anexo:Pel%C3%ADculas_con_las_mayores_recaudaciones

In [ ]:
```
```