Universidad Castro Carazo
Estudiante: Priscilla Corrales Fallas
Curso: Análisis de Datos
II cuatrimestre, 2023

1.1 ¿Qué es seaborn?

Según Rosidi (2023) Seaborn es una biblioteca creada en el año 2014 por Michael Waskom para la visualización de datos en Pyton y basada en la reconocida biblioteca de Matplotlib. La biblioteca Seaborn está diseñada específicamente para visualización estadística y se utiliza comúnmente en conjunto con Pandas para la exploración y análisis de datos.

Según Chandra (2023) el uso de Seaborn presenta ventajas significativas para la visualización de datos, tales como:

1. **Visualización atractiva y personalizable:** Seaborn proporciona diferentes estilos de visualización personalizables y además cuenta con paletas de colores incorporadas, lo anterior permite la creación de gráficos de calidad de publicación y que se ajustan a necesidades específicas.
2. **Soporte para relaciones estadísticas complejas:** con diversas funciones de trazado, Seaborn permite la visualización de relaciones estadísticas complejas, lo que permite la simplificación y comprensión de los datos.
3. **Soporte para variables categóricas:** Seaborn ofrece funciones de trazado diseñadas para trabajar con variables categóricas, facilitando la comprensión de las relaciones entre diferentes categorías de datos.
4. **Integración sencilla con Pandas:** Seaborn está diseñada para trabajar con Pandas, esta integración agiliza el trabajo y la visualización de datos en un DataFrame en Pandas

1.2 Tipos de gráficos en Seaborn

De acuerdo con la página web de Python Charts (s.f), el sitio web Geeksforgeeks (s.f) y Chandra (2023), Seaborn permite la creación de diferentes tipos de gráficos, entre los que se destacan:

1. **Gráficos de dispersión (Scatter Plots):** Estos gráficos representan puntos en un plano cartesiano, donde cada punto corresponde a un par de valores. Son útiles para identificar patrones y relaciones entre dos variables numéricas. Se pueden crear usando la función **scatterplot()**
2. **Gráficos de Líneas (Line Plots):** Muestran la relación entre dos variables numéricas mediante una línea continua. Son útiles para ilustrar tendencias a lo largo del tiempo o secuencias. Se pueden crear con la función **lineplot()**
3. **Relplot():** permite usar otras herramientas que muestran cómo dos cosas diferentes están relacionadas, pero lo hace de manera que tiene un significado especial para ciertos grupos dentro de los datos. Utilizamos una función llamada relplot() para hacer este tipo de gráfico.
4. **Gráficos de barras (Bar Plots):** Representan datos categóricos o numéricos en forma de barras verticales u horizontales. Son ideales para comparar categorías o mostrar recuentos. Se pueden crear con **barplot()** o **countplot()**.
5. **Gráficos de distribución (Distribution Plots)**: Muestran la distribución de una variable numérica y pueden incluir histogramas, estimaciones de densidad del núcleo (KDE) y gráficos de caja. Ayudan a entender la forma y dispersión de los datos. Se pueden crear con funciones como histplot(), kdeplot() y boxplot().
6. **Gráficos de correlación (Correlation Plots)**: Representan la relación entre múltiples variables numéricas, resaltando las correlaciones entre ellas. El heatmap es un ejemplo común de gráfico de correlación, y se puede crear usando **heatmap()**
7. **Gráficos de pares (Pair Plots)**: Son útiles para visualizar relaciones entre múltiples variables numéricas. Se muestran en una matriz de gráficos de dispersión y histogramas. La función **pairplot()** crea este tipo de gráficos.
8. **Gráficos de cajas (Box Plots)**: A veces también se conoce como gráfico de caja y bigotes. Muestran la distribución y resumen estadístico de una variable numérica, incluyendo mediana, cuartiles y posibles valores atípicos. Se crean con **boxplot()**.
9. **Gráficos de violín (Violin Plots)**: Combina elementos de gráficos de caja y gráficos de densidad para mostrar la distribución y resumen estadístico de una variable. La función **violinplot()** crea este tipo de gráficos.
10. **Gráficos de regresión (Regression Plots):** Son útiles para visualizar relaciones lineales entre dos variables numéricas. Los gráficos de regresión pueden incluir líneas de regresión y bandas de confianza. Se crean con **regplot()**.

Estos son solo algunos ejemplos de los tipos de gráficos disponibles en Seaborn. Cada tipo de gráfico es útil para diferentes propósitos y escenarios de análisis de datos, para visualizar con más detalle cada tipo de gráfico ver Anexo 1.

1.3 Creación de gráfico con Seaborn

En el siguiente apartado con la finalidad de profundizar en los pasos, métodos y funciones para la visualización de datos utilizando Seaborn, siguiendo a Rosidi, N. (2023), se desarrolla un ejemplo básico con un gráfico de dispersión para mostrar las relaciones entre la longitud de los pétalos y la longitud de los sépalos.

1. **Importar las bibliotecas**

Se importa la biblioteca seaborn

```
import seaborn as sns
```

2. **Cargar el conjunto de datos (DataFrame)**

La variable iris es un DataFrame que contiene el conjunto de datos de las flores, el cual se carga utilizando la función *'sns.load_dataset()'*

```
iris = sns.load_dataset('iris')
```

3. **Establecer el Estilo**

La función *'sns.set_style'* se utiliza para establecer el estilo del gráfico. Luego, el código crea un gráfico de dispersión utilizando la función *'sns.scatterplot'*.

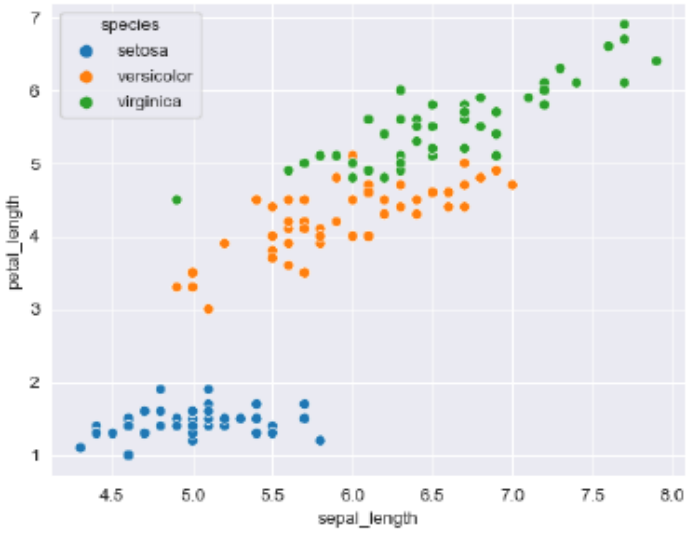
```
sns.set_style("darkgrid")
```

4. **Dibujar el gráfico**

El argumento *'data'* especifica el DataFrame que contiene los datos para el gráfico, y los argumentos '*x'* e '*y'* especifican las columnas a utilizar para los datos del eje x y el eje y. El argumento *'hue'* se utiliza para colorear los puntos según los valores en la columna "species", y el argumento *'legend'* se utiliza para mostrar la leyenda completa para el gráfico.

```
sns.scatterplot( data=iris, x='sepal_length', y='petal_length',
hue="species", legend="full")
```

A continuación, se presenta el gráfico:



Fuente: Rosidi, N. (2023). 4 Python Data Visualization Libraries you can't do without. (pág. 10)

El gráfico de dispersión resultante muestra la longitud del sépalo y la longitud del pétalo para cada flor en el conjunto de datos, donde cada especie de flor está representada por un color diferente. La leyenda muestra la correspondencia de colores con las especies.

1.4 Personalización de gráficos con Seaborn

Según el sitio web Geeksforgeeks (s.f) Seaborn ofrece una amplia gama de opciones de personalización que permiten crear gráficos visualmente atractivos y adaptados a las preferencias personales. Algunos métodos o funciones de personalización incluyen:

1. **Cambiar la estética de la figura:** El método **"set_style()"** se utiliza para establecer la estética del gráfico Esto significa que afecta cosas como el color de los ejes, si la cuadrícula está activa o no, u otros elementos estéticos. Hay cinco temas disponibles en Seaborn: darkgrid, whitegrid, dark, white, ticks.

    - **Sintaxis:** `sns.set_style("dark")`

2. **Eliminación de bordes (Ejes):** Los bordes son las líneas que indican los limites en los datos y conectan las marcas de graduación de los ejes. Se pueden eliminar utilizando el método **"despine()"**

    - **Sintaxis:** `sns.despine()`

3. **Cambiar el tamaño de la figura:** El tamaño de la figura se puede cambiar utilizando el método **"figure()"** de matplotlib. El método **"figure()"** crea una figura del tamaño especificado, que se pasa como parámetro **"figsize"**

    - **Sintaxis:** `plt.figure(figsize = (2, 4))`

4. **Escalado de los gráficos:** Esto se puede hacer utilizando el método **set_context()**. Permite sobrescribir los parámetros predeterminados. Esto afecta cosas como el tamaño de las etiquetas, las líneas y otros elementos del gráfico, pero no el estilo general. El contexto base es "notebook", y los otros contextos son "paper", "talk" y "poster". **font_scale** establece el tamaño de la fuente.

    - **Sintaxis:** `sns.set_context("paper")`

5. **Configuración Temporal del Estilo:** El método "**axes_style()"** se utiliza para establecer el estilo temporalmente. Se usa junto con la declaración "**with"**.

    - **Sintaxis:** `with sns.axes_style('darkgrid'):`

6. **Paleta de color:** Se puede usar diferentes tipos de mapas de colores para diferentes tipos de gráficos. El método "**color_palette()"** se utiliza para asignar colores al gráfico. Otra función**, "palplot()"**, se utiliza para manejar paletas de colores y representa la paleta de colores como un conjunto horizontal.

    - **Sintaxis:** `sns.palplot(palette)`

7. **Establecer paleta de colores predeterminada:** El método **"set_palette()"** se utiliza para establecer la paleta de colores predeterminada para todos los gráficos. Los argumentos tanto para **color_palette()** como para set_palette() son los mismos. **set_palette()** cambia los parámetros predeterminados de Matplotlib.

    - **Sintaxis:** `sns.set_palette('vlag')`

1.5 Referencias bibliográficas

Chandra, R. (2023). ¿Qué es Seaborn en Python? Visualización de datos usando Seaborn. Recuperado de https://docs.kanaries.net/es/tutorials/Seaborn/seaborn-python

Geeksforgeeks.(s.f). Python Seaborn Tutorial. Recuperado de https://www.geeksforgeeks.org/python-seaborn-tutorial/

Python Charts. (s.f.). La librería Seaborn. Recuperado de https://python-charts.com/es/seaborn/

Rosidi, N. (2023). 4 Python Data Visualization Libraries you can't do without. Recuperado de https://www.stratascratch.com/blog/4-python-data-visualization-libraries-you-can-t-do-without/#:~:text=Matplotlib%2C%20seaborn%2C%20Plotly%2C%20and,them%20with%20our%20code%20examples

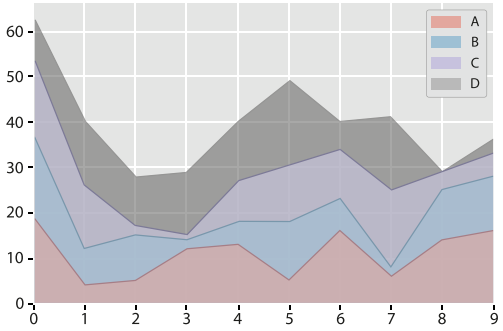
Pandas & Seaborn plots cheatsheet

V2 - feb. 2020

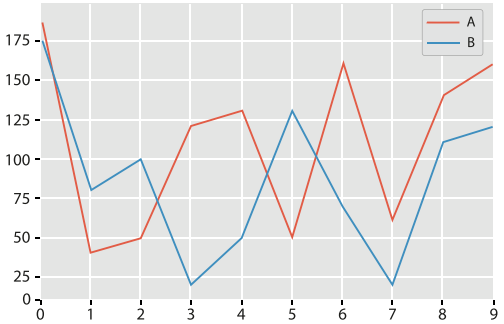
Paradigma

```
df1 = pd.DataFrame(np.random.randn(30,4),
      columns=list('ABCD'))

df2 = pd.DataFrame(np.random.randint(0,20,size=(10, 4)),
      columns=list('ABCD'))

df2["orden"] = ["primero", "primero", "primero",
                "primero", "primero", "segundo",
                "segundo", "segundo", "segundo",
                "segundo"]

df2["Mes"] = ["Enero", "Febrero", "Marzo", "Abril",
              "Mayo", "Junio", "Julio", "Agosto",
              "Septiembre", "Octubre"]

df3 = pd.DataFrame(np.random.randn(1000, 2),
      columns=['a', 'b'])
```

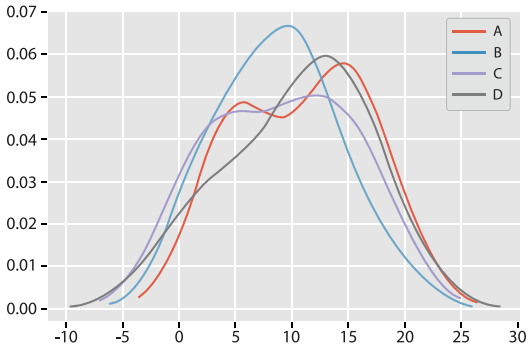
```
df2.plot.area(alpha=0.4)
```



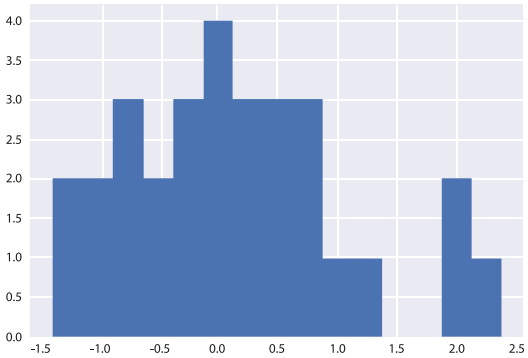

```
df2.plot.line(y = ['A','B'])
```



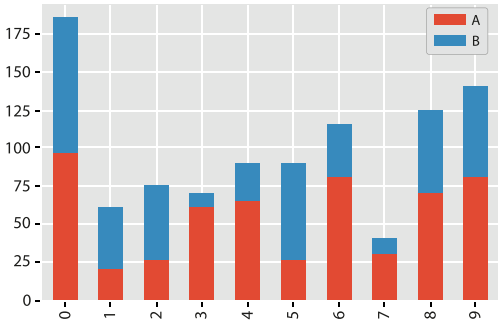

```
df2.plot.kde()
```



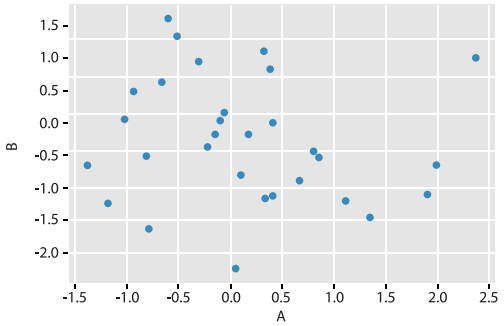

```
df1.A.hist(bins=15)
```



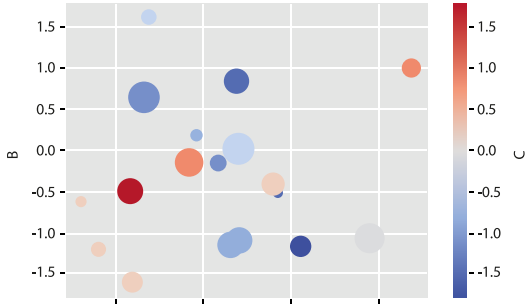
```
df2[["A","B"]].plot.bar(stacked=True)
```



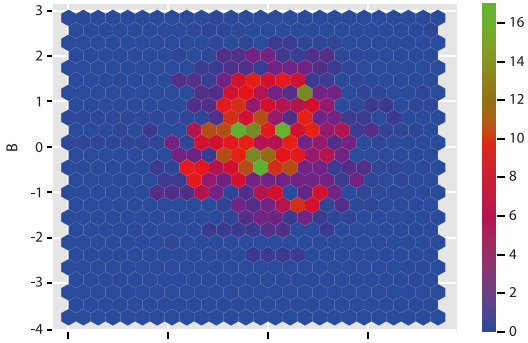

```
df1.plot.scatter(x="A", y="B")
```



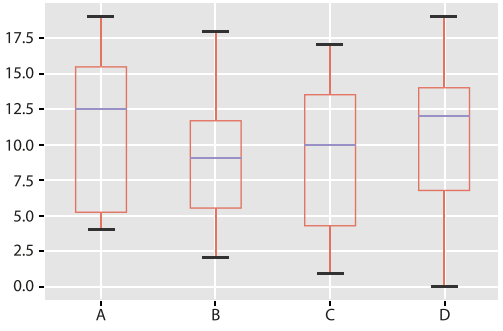

```
df1.plot.scatter(x="A", y="B", c = "C",cmap='coolwarm',
s=df1["D"]*300)
```



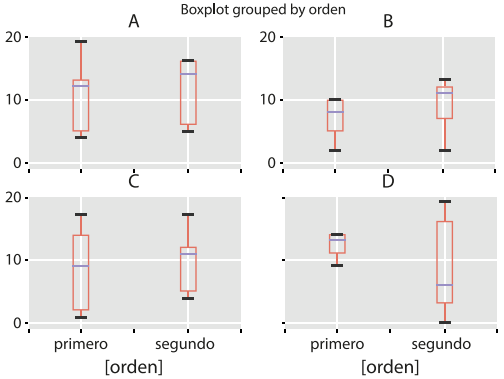

```
columns=['a', 'b'])
df3.plot.hexbin(x='a',y='b',gridsize=25,cmap='brg')
```



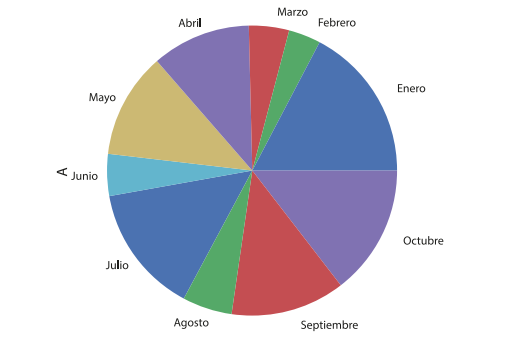

```
df2.plot.box()
```




```
df2.boxplot(by="orden")
```




```
df_2.set_index("Mes",inplace=True)
df_2.A.plot.pie()
```



Más información y comentarios: paradig.ma/pandas-seaborn

Referencias: https://matplotlib.org/3.1.0/contents.html | https://seaborn.pydata.org/

Pandas & Seaborn plots cheatsheet

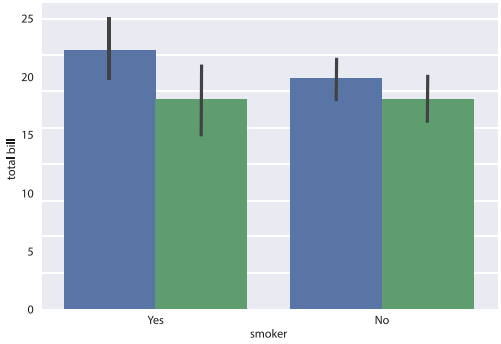
V2 - feb. 2020

Paradigma

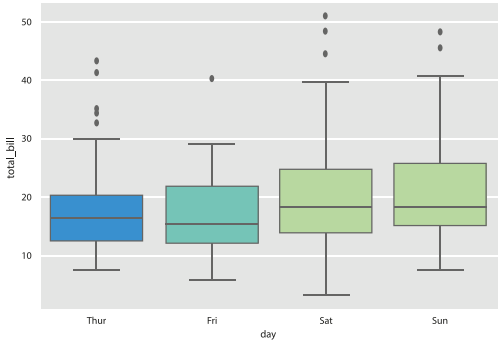
```
import seaborn as sns
tips = sns.load_dataset('tips')
```

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.63	Female	No	Sun	Dinner	4

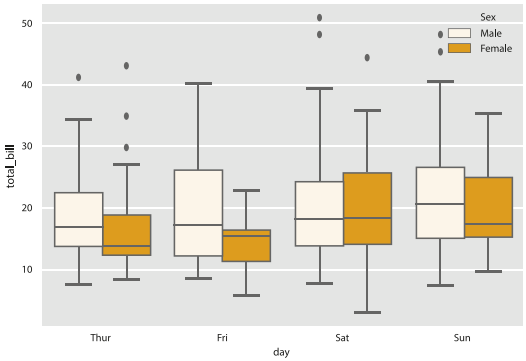
```
sns.barplot(x='smoker',y='total_bill',data=tips
hue="sex")
```



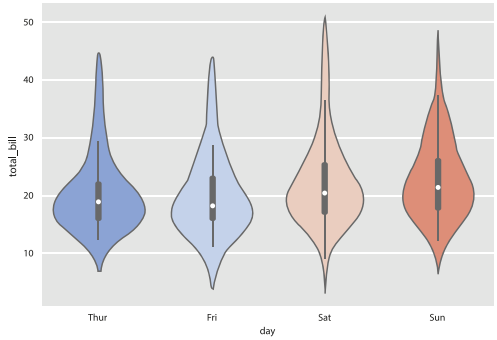

```
sns.boxplot(x="day", y="total_bill", data=tips,
palette='rainbow')
```



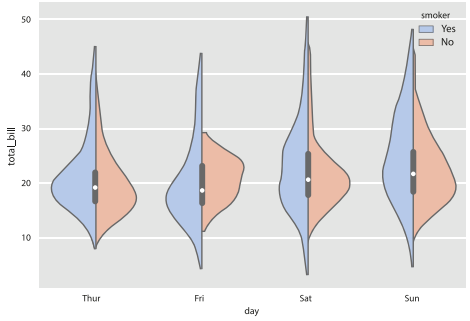

```
sns.boxplot(x="day", y="total_bill", data=tips,hue="sex",
color="orange")
```



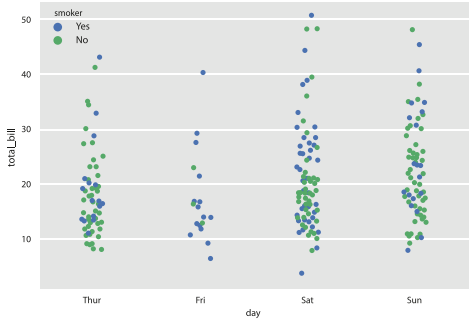

```
sns.violinplot(x="day", y="total_bill", data=tips,
palette="coolwarm")
```



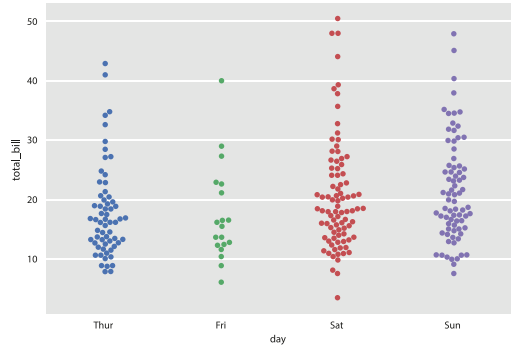

```
sns.violinplot(x="day", y="total_bill", data=tips,
palette="coolwarm",hue="smoker",split=True)
```



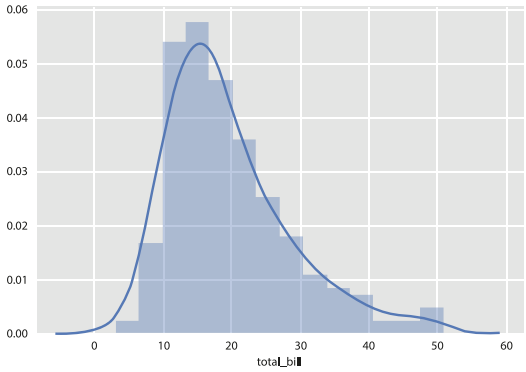

```
sns.stripplot(x="day", y="total_bill",
data=tips,hue="smoker")
```



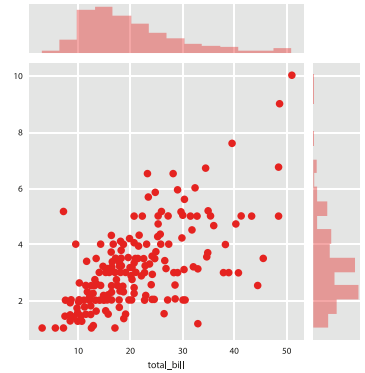

```
sns.swarmplot(x="day", y="total_bill", data=tips)
```



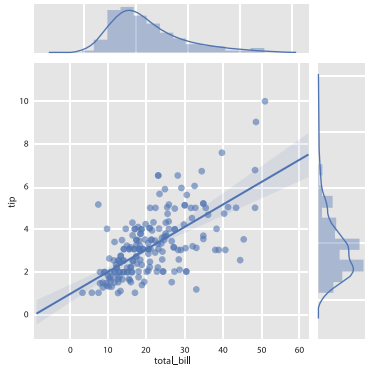

```
sns.distplot(tips['total_bill'])
```



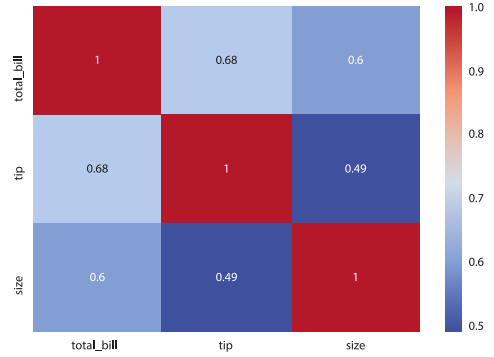

```
sns.jointplot(x='total_bill',y='tip',data=tips,
kind='scatter', color="red")
```




```
sns.jointplot(x='total_bill',y='tip',data=tips,
kind='reg')
```




```
sns.heatmap(tips.corr(),cmap='coolwarm',annot=True)
```



Más información y comentarios: paradig.ma/pandas-seaborn

Referencias: https://matplotlib.org/3.1.0/contents.html | https://seaborn.pydata.org/