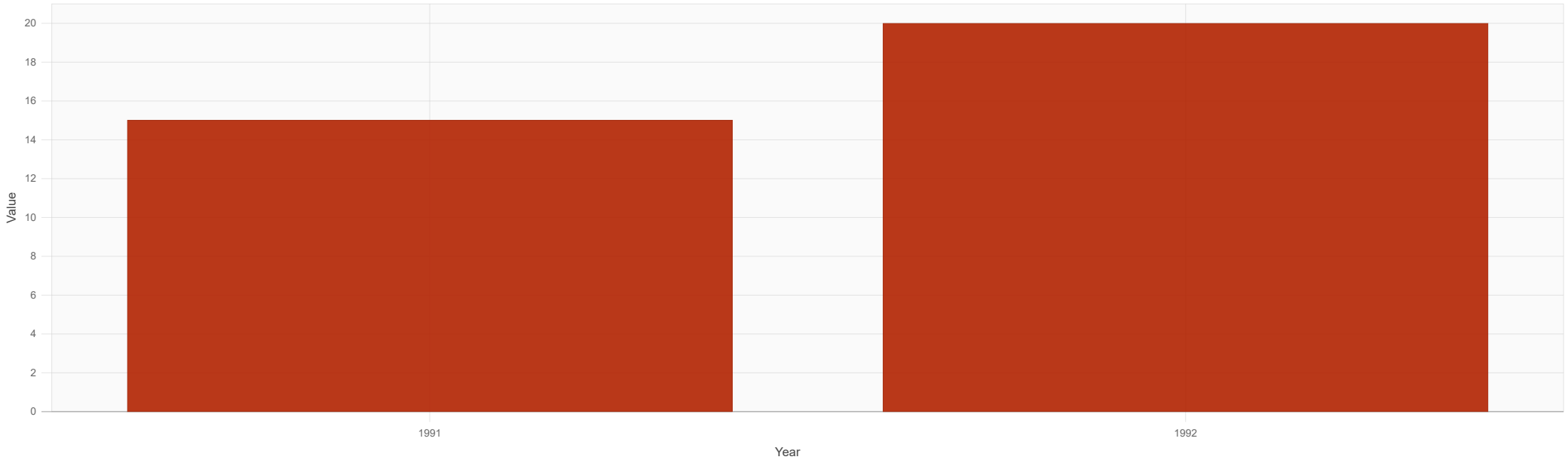
08MBID-VISUALIZACIÓN DE DATOS

Nombre: Jordy Carrión Lojan
Url: https://js-jevjq5.stackblitz.io
Google Colab: https://colab.research.google.com/drive/1SRhET9XVeer_-E306FecxUpjXoxCPtTL?usp=sharing
GitHub: https://github.com/Jordy98CL/08MBID.gitg

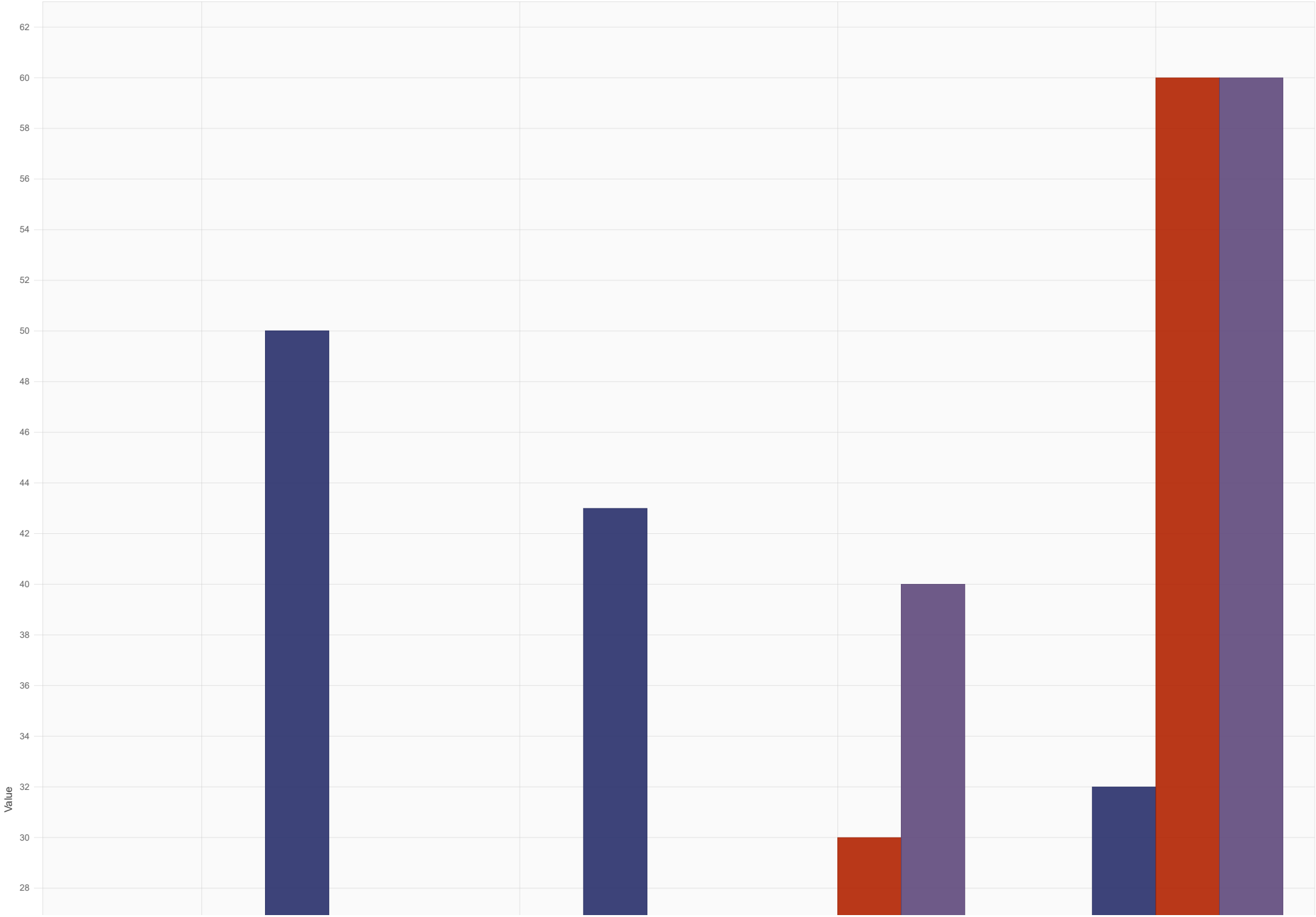
Gráfico de Columnas con D3Plus

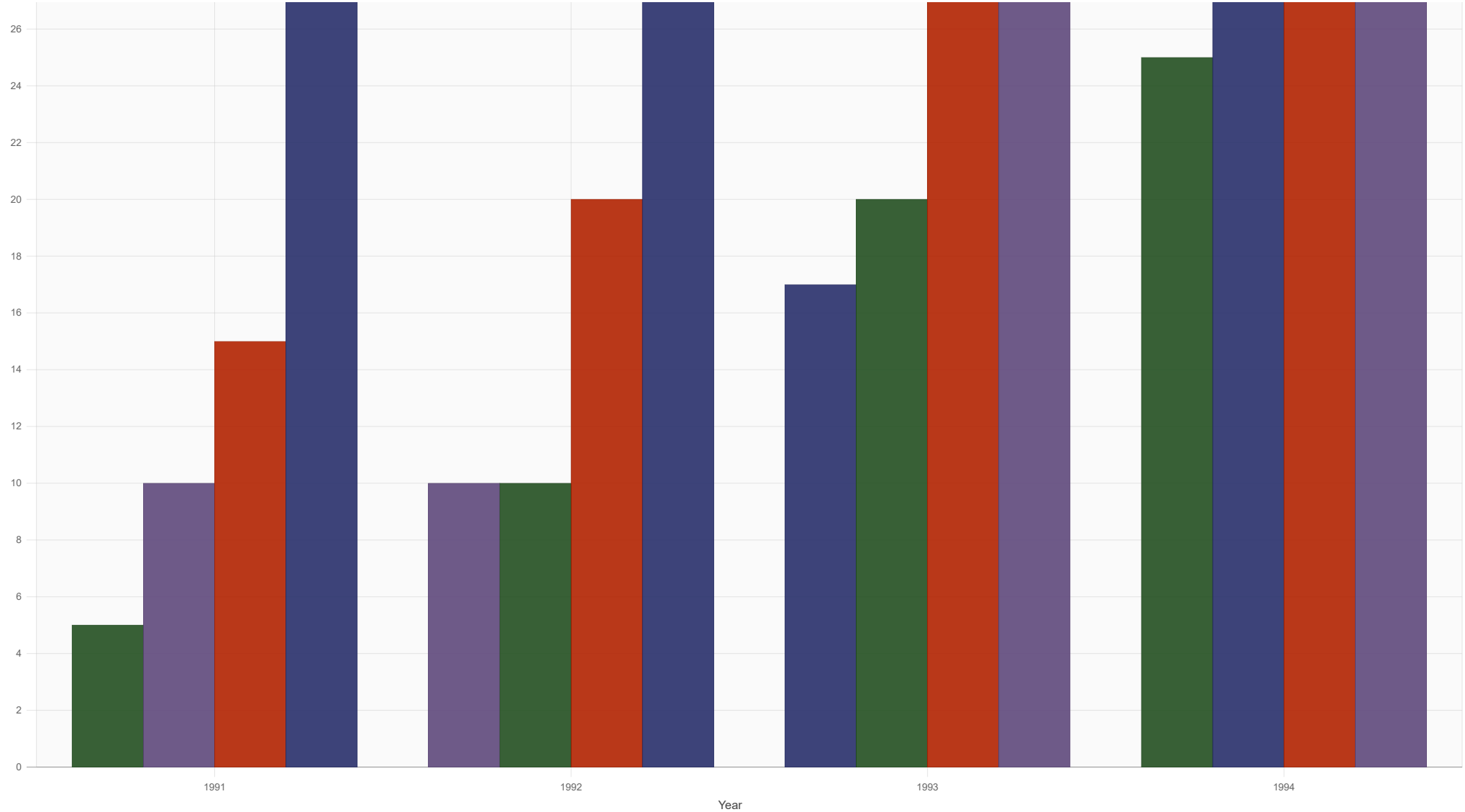
Gráfico de columnas simple



Reutilizar código. Barras

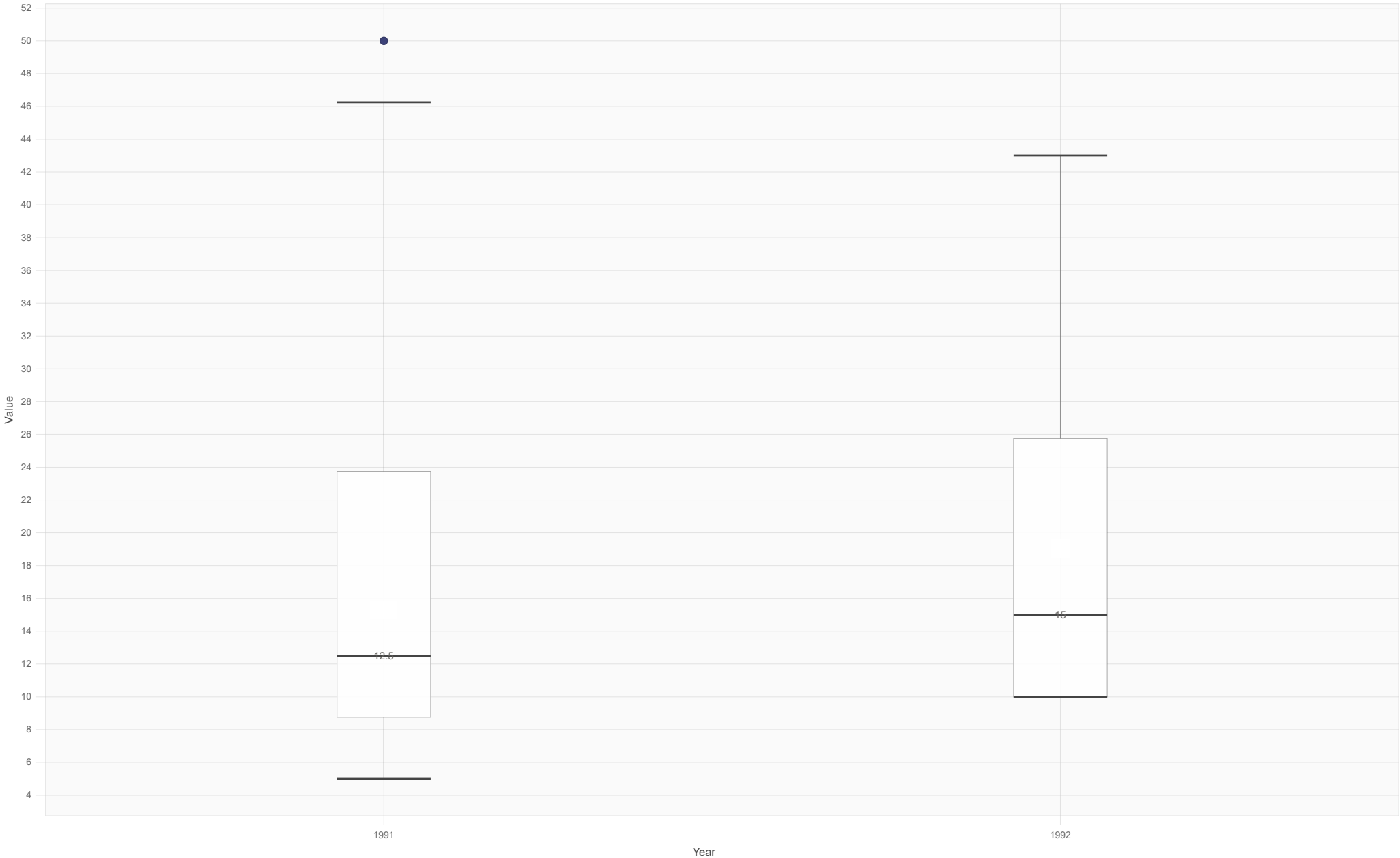
Gráfico de columnas múltiple


62
60
58
56
54
52
50
48
46
44
42
40
38
36
34
32
30
28
Value



Caja y Bigotes

Gráfico de Caja y Bigote (1991 y 1992)


52
50
48
46
44
42
40
38
36
34
32
30
28
26
24
22
20
18
16
14
12
10
8
6
4
Value
12.5
15
1991
1992
Year

```
#Importamos modulos
import pandas as pd
import io
import requests
import seaborn as sns
import timeit
import matplotlib.pyplot as plt
```

```
#Cargamos el dataset de los pasajeros del Titanic
url="https://raw.githubusercontent.com/mwaskom/seaborn-data/master/titanic.csv"
s=requests.get(url).content
titanic=pd.read_csv(io.StringIO(s.decode('utf-8')))
```

Explorar datos.

```
titanic.head()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True

```
#Informacion del dataset
titanic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
 2   sex          891 non-null    object
 3   age          714 non-null    float64
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   embarked     889 non-null    object
 8   class        891 non-null    object
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         203 non-null    object
 12  embark_town  889 non-null    object
 13  alive        891 non-null    object
 14  alone        891 non-null    bool
dtypes: bool(2), float64(2), int64(4), object(7)
memory usage: 92.4+ KB
```

```
#Renombra la columna class=clase y fare=tarifa
titanic.rename(columns={'class': 'clase'},
inplace=True)
titanic.rename(columns={'fare': 'tarifa'},
inplace=True)
#Muestra los valores distintos para class(clase)
titanic.clase.unique()
```

```
array(['Third', 'First', 'Second'], dtype=object)
```

```
#Primeras 5 filas
titanic.head(5)
```

	survived	pclass	sex	age	sibsp	parch	tarifa	embarked	clase	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True

Definiendo nuevas columnas en el dataset

```
#Añadimos nuevas columnas: is_old, is_baby
##############################################
def is_old_func(row):
    return row['age'] > 60
titanic['is_old'] = titanic.apply(is_old_func, axis='columns')
#Otra forma de definir una  nueva columna
titanic.eval ( ' is_baby = age< 15 ' , inplace = True)
```

```
titanic.head()
```

	survived	pclass	sex	age	sibsp	parch	tarifa	embarked	clase	who	adult_male	deck	embark_town	alive	alone	is_
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False	Fa
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False	Fa
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True	Fa
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False	Fa
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True	Fa

```
# Encuentra las primeras filas con datos nulos en 'nombre_columna'
filas_con_nulos = titanic[titanic['age'].isnull()]

# Muestra las primeras filas con datos nulos en 'nombre_columna'
```

```
filas_con_nulos.head()
```

	survived	pclass	sex	age	sibsp	parch	tarifa	embarked	clase	who	adult_male	deck	embark_town	alive	alone	i
5	0	3	male	NaN	0	0	8.4583	Q	Third	man	True	NaN	Queenstown	no	True	
17	1	2	male	NaN	0	0	13.0000	S	Second	man	True	NaN	Southampton	yes	True	
19	1	3	female	NaN	0	0	7.2250	C	Third	woman	False	NaN	Cherbourg	yes	True	
26	0	3	male	NaN	0	0	7.2250	C	Third	man	True	NaN	Cherbourg	no	True	
28	1	3	female	NaN	0	0	7.8792	Q	Third	woman	False	NaN	Queenstown	yes	True	

```
#Define una variable numérica: class_num
def class_num_func(row):
    Clase={'Third':3,'First':1,'Second':2}
    return Clase[row.clase]
titanic['class_num'] = titanic.apply(class_num_func, axis='columns')
```

```
titanic.head()
```

	survived	pclass	sex	age	sibsp	parch	tarifa	embarked	clase	who	adult_male	deck	embark_town	alive	alone	is_
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False	Fa
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False	Fa
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True	Fa
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False	Fa
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True	Fa

Consulta de datos con condiciones

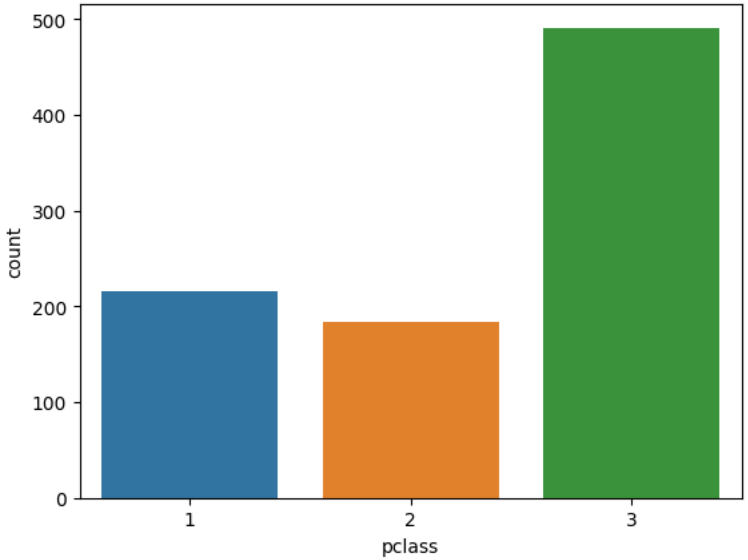
```
#Consulta con condiciones
titanic[
    (titanic.sex == 'female')
    & (titanic['clase'].isin(['First', 'Third']))
    & (titanic.age > 45 )
    & (titanic.survived == 0)
]
```

	survived	pclass	sex	age	sibsp	parch	tarifa	embarked	clase	who	adult_male	deck	embark_town	alive	alone	i
132	0	3	female	47.0	1	0	14.5000	S	Third	woman	False	NaN	Southampton	no	False	
177	0	1	female	50.0	0	0	28.7125	C	First	woman	False	C	Cherbourg	no	True	
736	0	3	female	48.0	1	3	34.3750	S	Third	woman	False	NaN	Southampton	no	False	

Distribución de las clases

```
#Distribución de las clases
sns.countplot(x="pclass", data=titanic)
```

```
<Axes: xlabel='pclass', ylabel='count'>
```

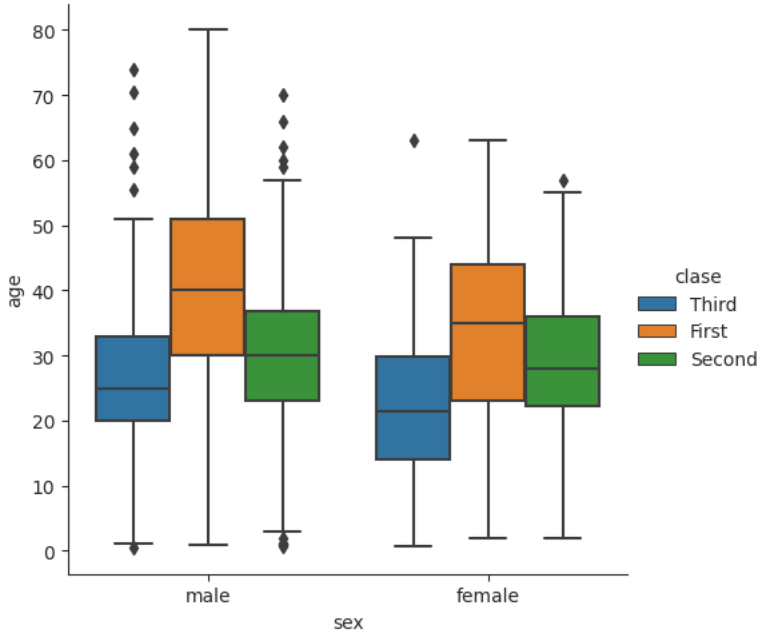


Distribución por edad

```
#Distribución de la edad(ege)
sns.histplot(titanic.age.dropna( ), kde=True)
plt.show( )
```

BoxPlot de la edad por sexo y clase

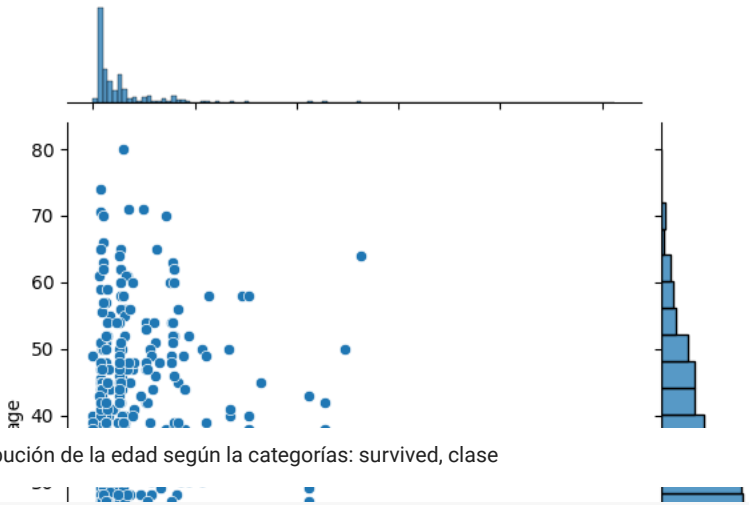
```
#BoxPlot de la edad por sexo y clase
with sns.axes_style(style='ticks'): ax = sns.catplot(data=titanic, x="sex", y="age", hue="clase", kind="box")
```



Distribución cruzada de Edad y Tarifa

```
#Distribución cruzada de Edad y Tarifa
sns.jointplot(x='tarifa',y='age',data=titanic)
```

```
<seaborn.axisgrid.JointGrid at 0x7dfc691412a0>
```



Distribución de la edad según la categorías: survived, clase

```
#Cambiamos el font
sns.set(font_scale=1)
#FacetGrid - Construir una matriz de gráficos
g = sns.FacetGrid(titanic, row='survived',col='clase'
)
g.map(sns.histplot, "age", kde=True)
plt.show()
```

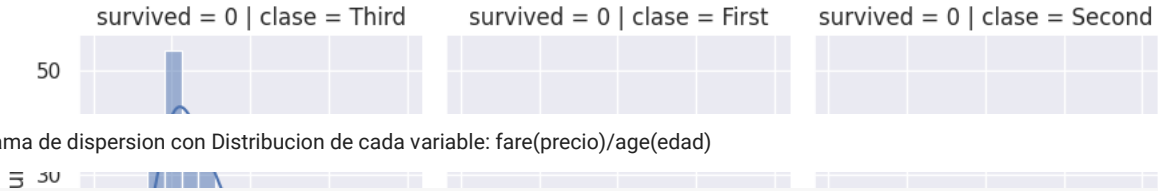
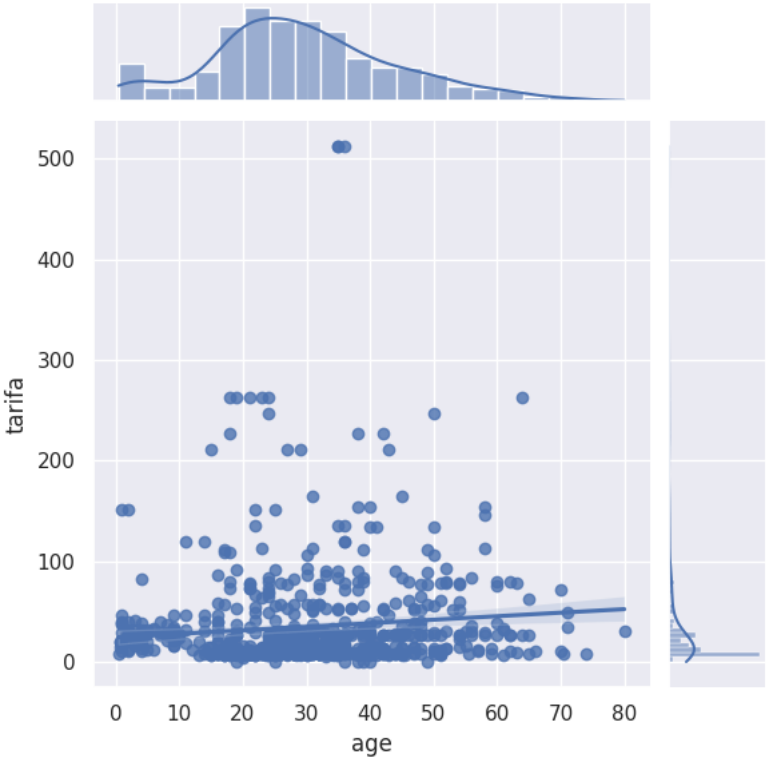


Diagrama de dispersion con Distribucion de cada variable: fare(precio)/age(edad)

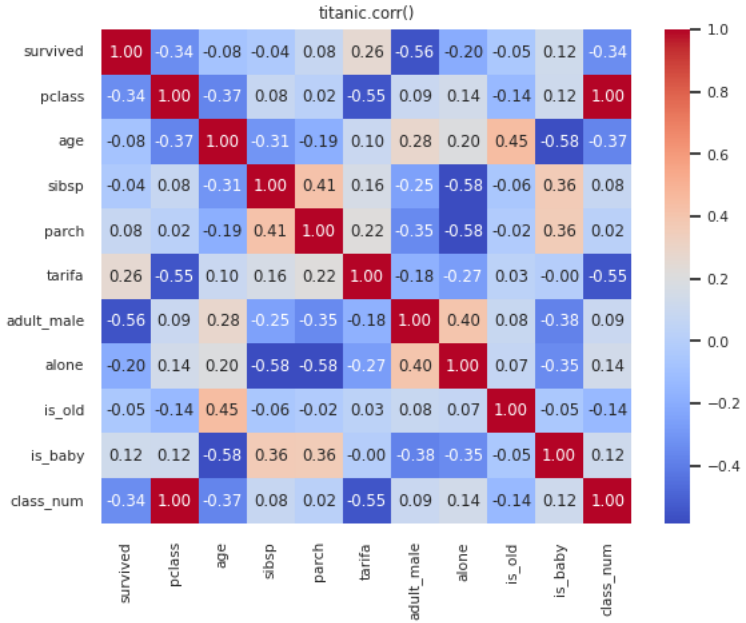
```
#Diagrama de dispersion con Distribucion de cada variable: fare(precio)/age(edad)
sns.jointplot(data=titanic, x='age', y='tarifa',
kind='reg', color='b')
plt.show()
```



Mapa de calor de correlaciones

```
# Mapa de calor de correlaciones
tc = titanic.corr()
sns.set(font_scale=0.7)
sns.heatmap(tc,annot=True, cmap='coolwarm', fmt=".2f")
plt.title('titanic.corr()')
```

```
<ipython-input-19-3c53aca726db>:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future
  tc = titanic.corr()
Text(0.5, 1.0, 'titanic.corr()')
```



Diagramas de dispersión de parejas de variables cuantitativas

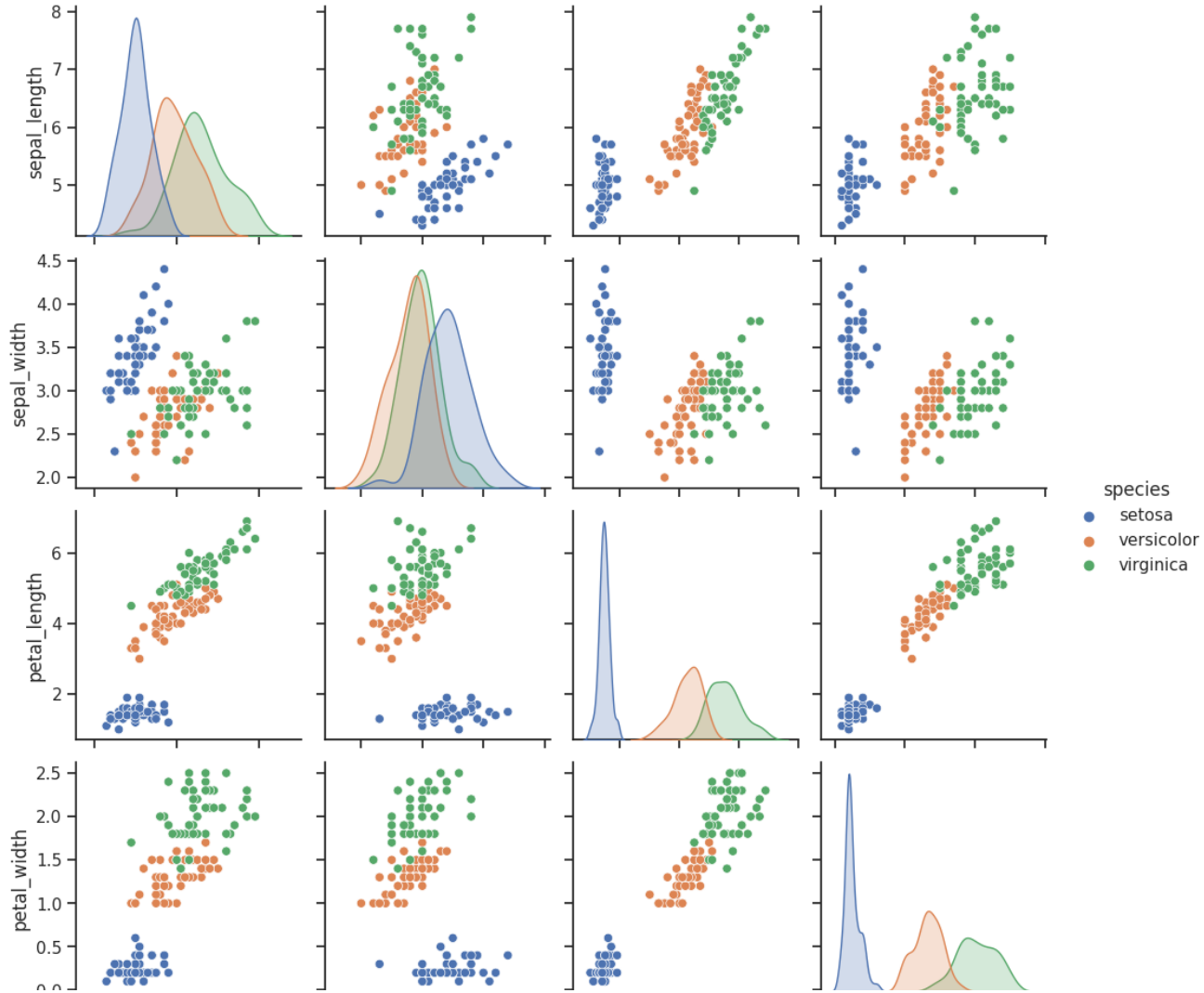
```
#Define un subconjuto de datos con las variables numéricas
titanic_num = titanic[['survived','pclass','sibsp','parch','tarifa']]
#Hace una matriz de diagramas de dispersión de parejas de variables.
sns.pairplot(titanic_num, hue="survived")
plt.show()
```

Dataframe Iris

```
#Establece el estilo estético de las tramas
sns.set(style="ticks")

#Carga el data set
df = sns.load_dataset("iris")
#matriz de diagramas de dispersion
sns.pairplot(df, hue="species")
```

```
<seaborn.axisgrid.PairGrid at 0x7dfc66ee6230>
```



Ampliación de la Práctica

```
import seaborn as sns
import matplotlib.pyplot as plt

# Suponiendo que ya tienes el DataFrame del Titanic cargado como 'titanic'

# Filtrar los datos para cada combinación de sexo y clase
g = sns.FacetGrid(titanic, col="pclass", row="sex")
g.map_dataframe(sns.histplot, x="age", hue="survived", multiple="stack")
```

```
# Configurar las etiquetas para la leyenda
legend_labels = {0: "No sobrevivió", 1: "Sobrevivió"}

# Añadir leyenda con etiquetas personalizadas y título
g.add_legend(title="Sobreviviente", labels=legend_labels)

# Mostrar el gráfico
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/seaborn/axisgrid.py:181: UserWarning: You have mixed positional and keyword arguments, s
  figlegend = self._figure.legend(handles, labels, **kwargs)
```