Categóricos

Variables categoricas o variables de texto

In [ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

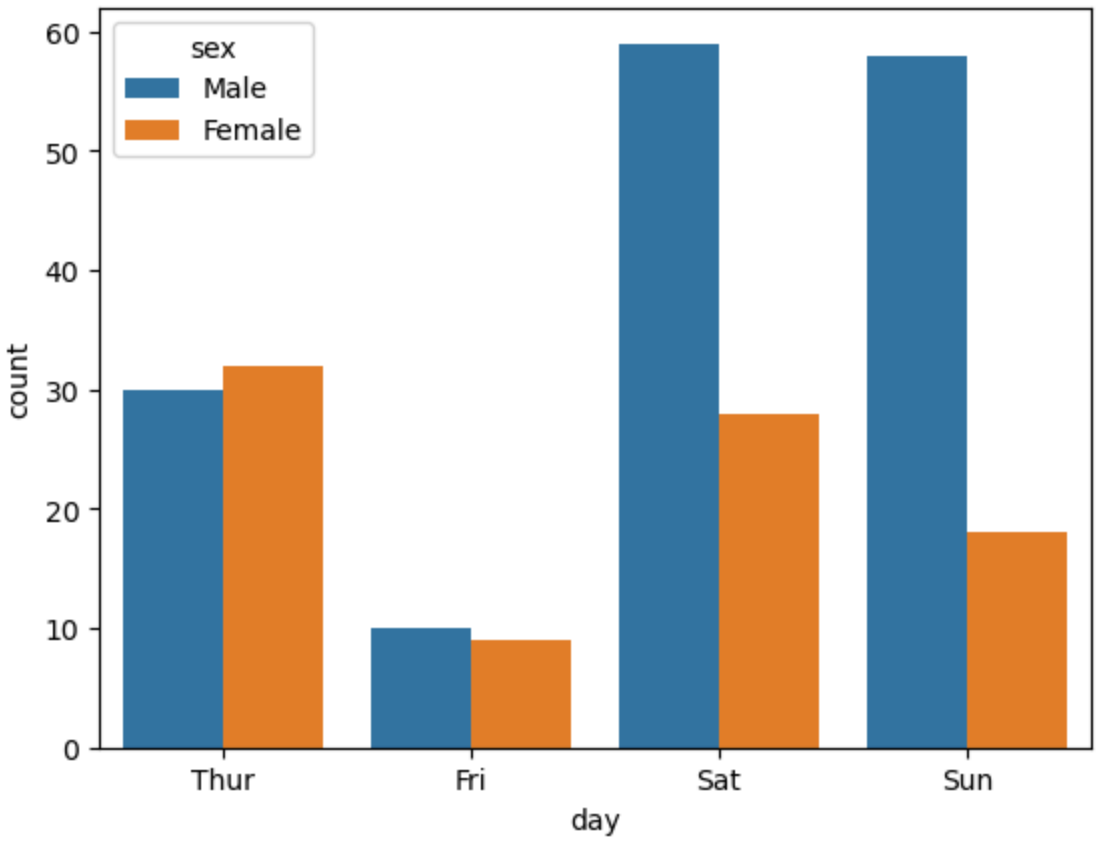
In [ ]:
```
tips = sns.load_dataset('tips')
tips.head(3)
```

Out[ ]:

	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

Countplot

In [ ]:
```
#Haciendo gráficas
sns.countplot(data=tips,x='day',hue='sex')
plt.show()
```



Explicación:

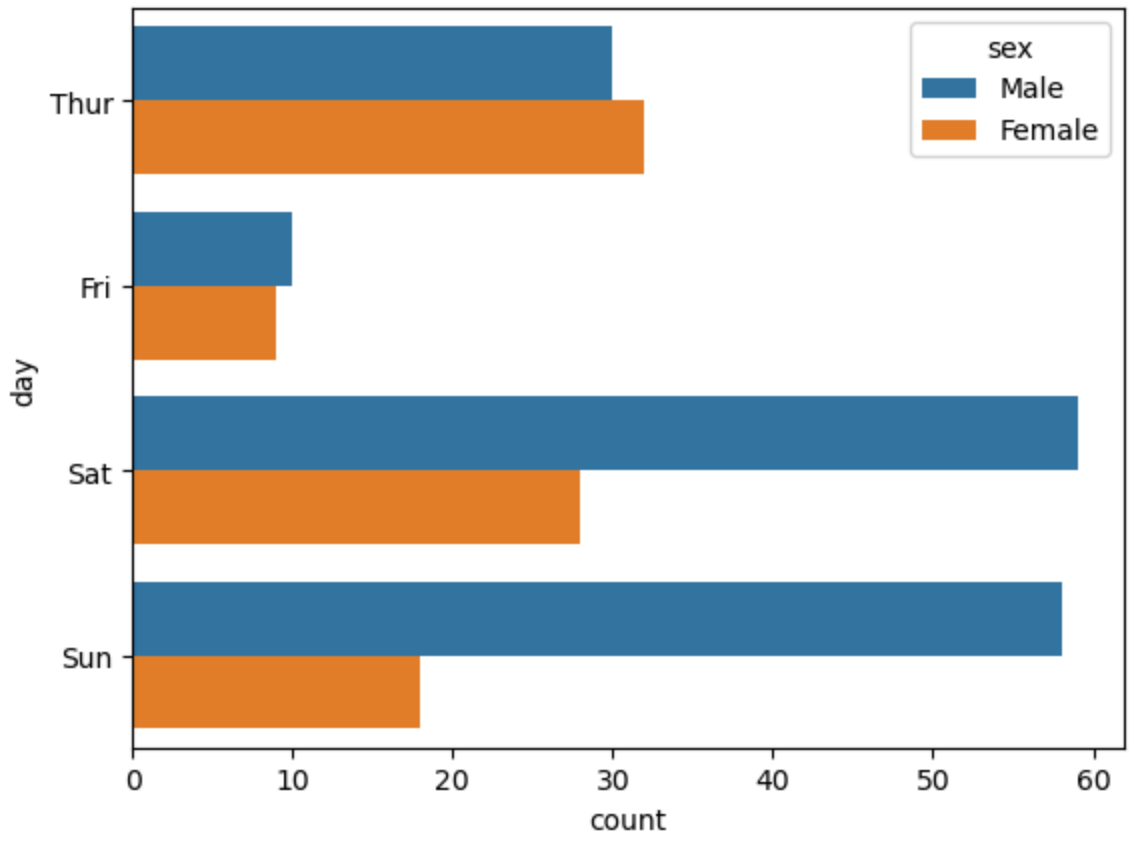
Haciendo una gráfica de nuestro dataframe `tips` de la variable `x = day` y haciendo una segmentación `(hue)` por variable `sex` que representa el Género.

De aquí podemos concluir que:

- Los días Viernes están dejando menor propina
- Los dias Sábados los hombres dejan más propina que las mujeres
- Los días Jueves las mujeres ligeramente dejan mayor propina que los hombres.

También lo podemos gráficar en el eje Y

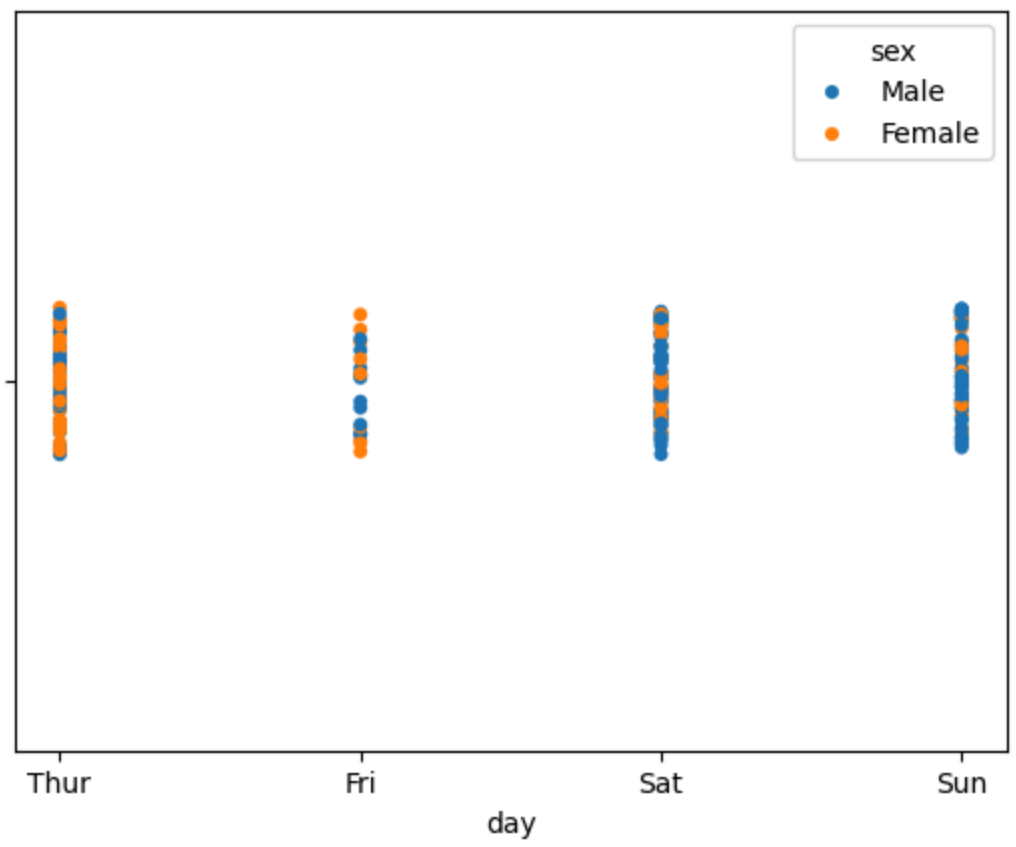
In [ ]:
```
#Countplot
sns.countplot(data=tips,y='day',hue='sex')
plt.show()
```



Como se puede observar el gráfico se acomoda automaticamente.

Strip plot

In [ ]:
```
#stripplot
sns.stripplot(data=tips,x='day',hue='sex')
plt.show()
```



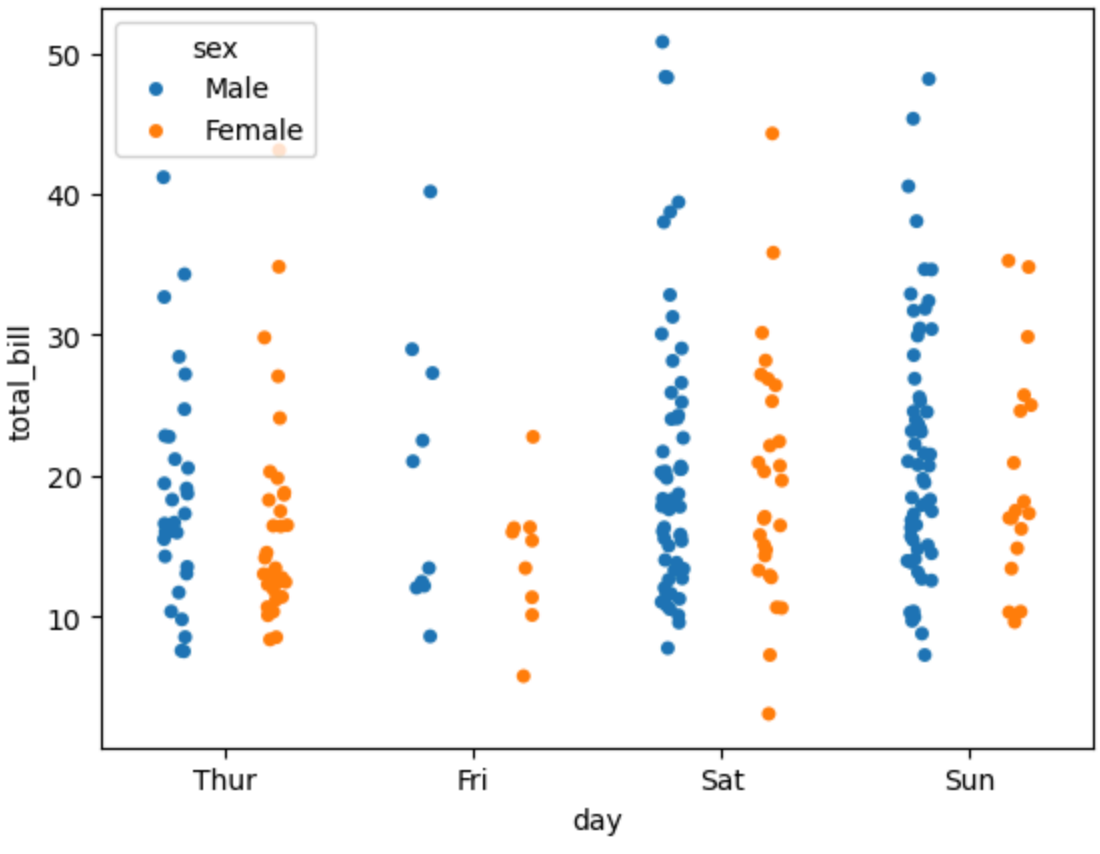
Este gráfico se ve algo complicado, pero le daremos contexto.

Explicación

Como ya no estoy contando las variables, debo especificarle que quiero ver de esas variables categóricas:

- Le especificamos que en `y ='total_bill'`, es decir le decimos que en `y` quiero el total de la cuenta.
- En x puedo ver el comportamiento a traves de la variable categorica por `day`.
- Le especificamos `dodge = True`, para poder ver el comportamiento de la segmentación que hicimos. En este caso ver el comportamiento de `total_bill` entre `hombres` y `mujeres`.

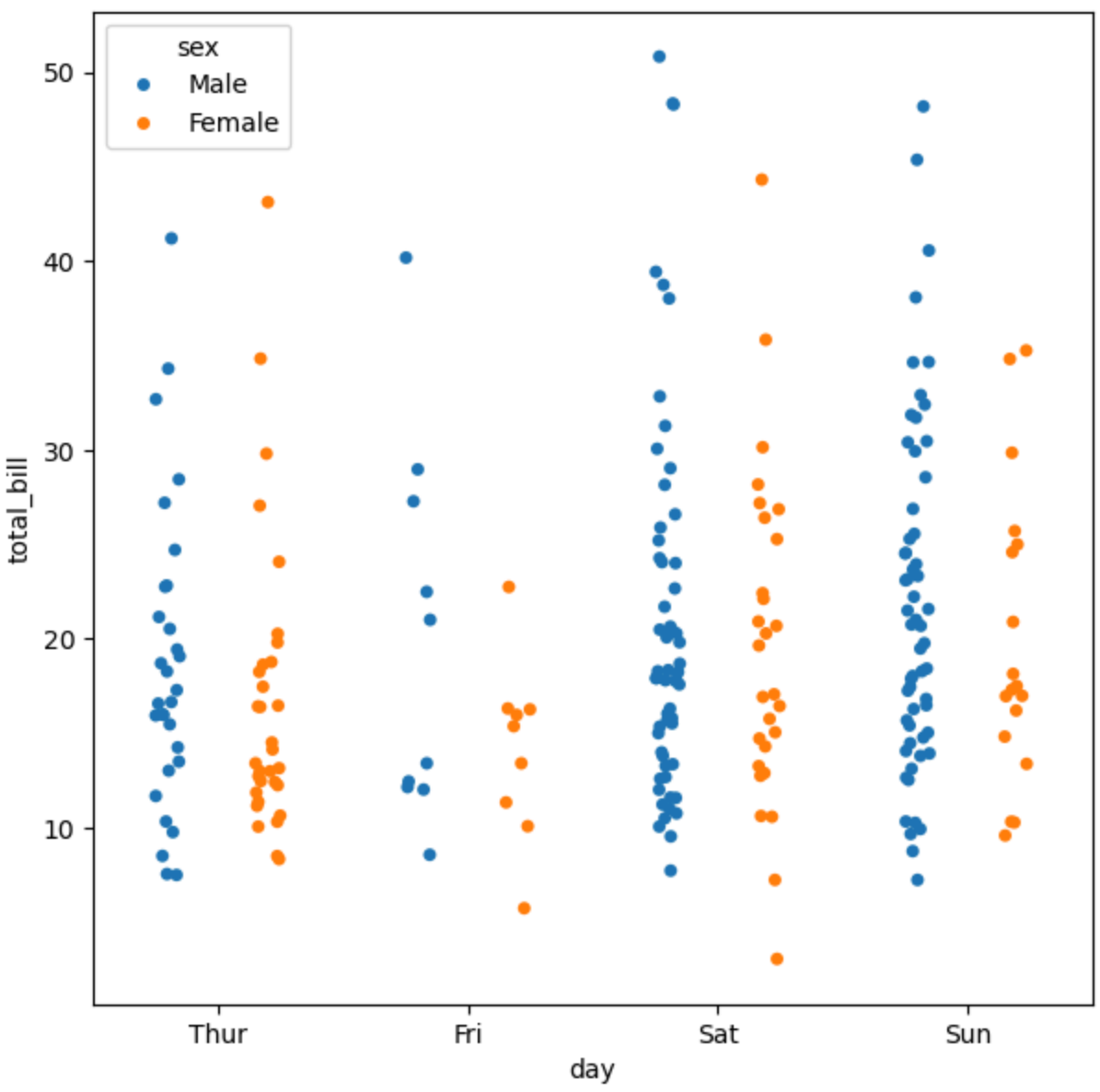
In [ ]:
```
```



Análisis:

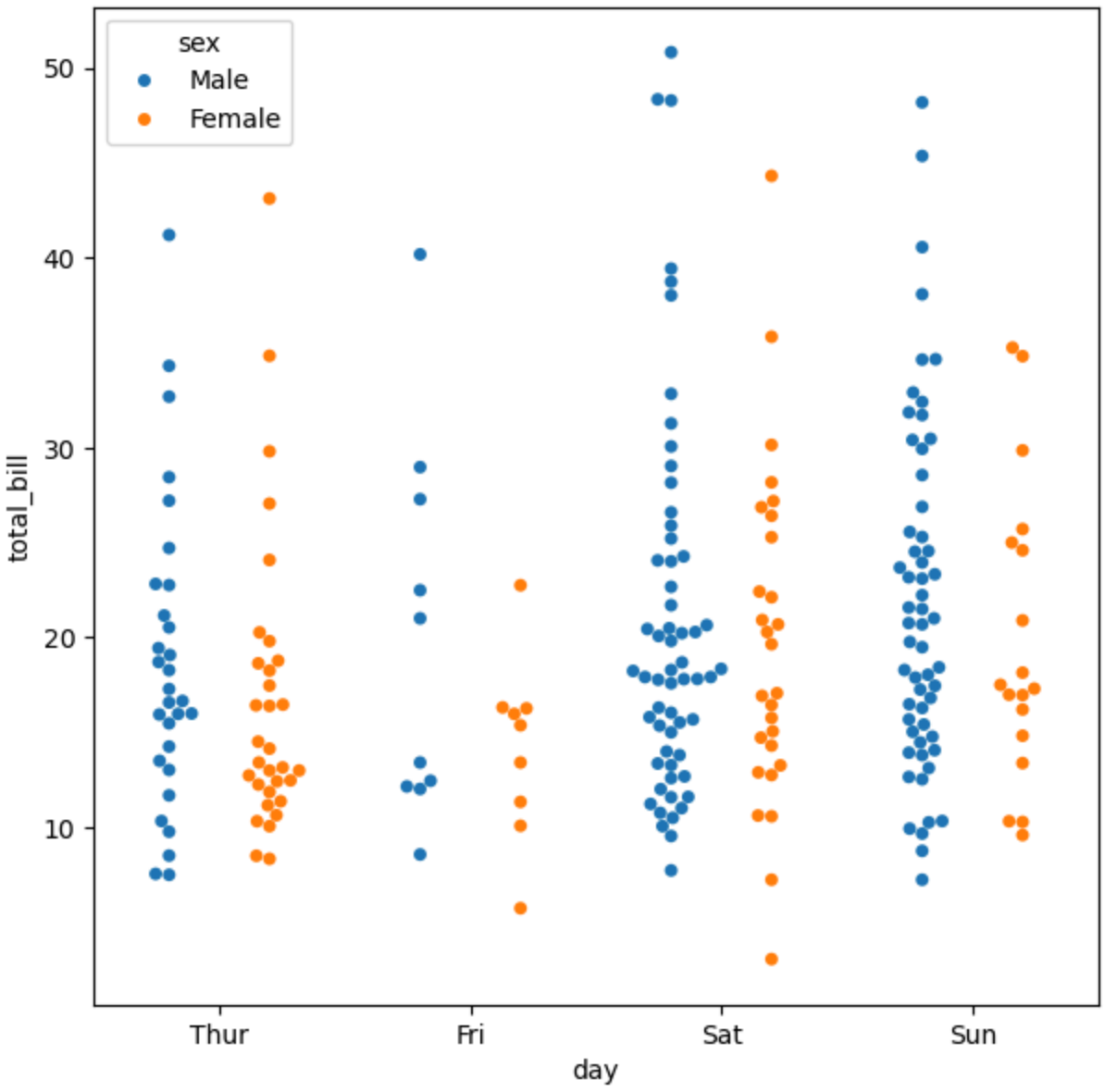
Se puede observar que en el día `Sábado` hay pocos registros por parte de los `hombres` que `pagan una cuenta excesiva` ,también se puede ver que hay `pocos puntos o registros` en el día `Viernes`

In [ ]:
```
#stripplot
#Ampliando la gráfica para ver mejor los resultados
plt.figure(figsize=(7,7))
sns.stripplot(data=tips,x='day',y='total_bill',hue='sex',dodge=True)
plt.show()
```



Swarmplot

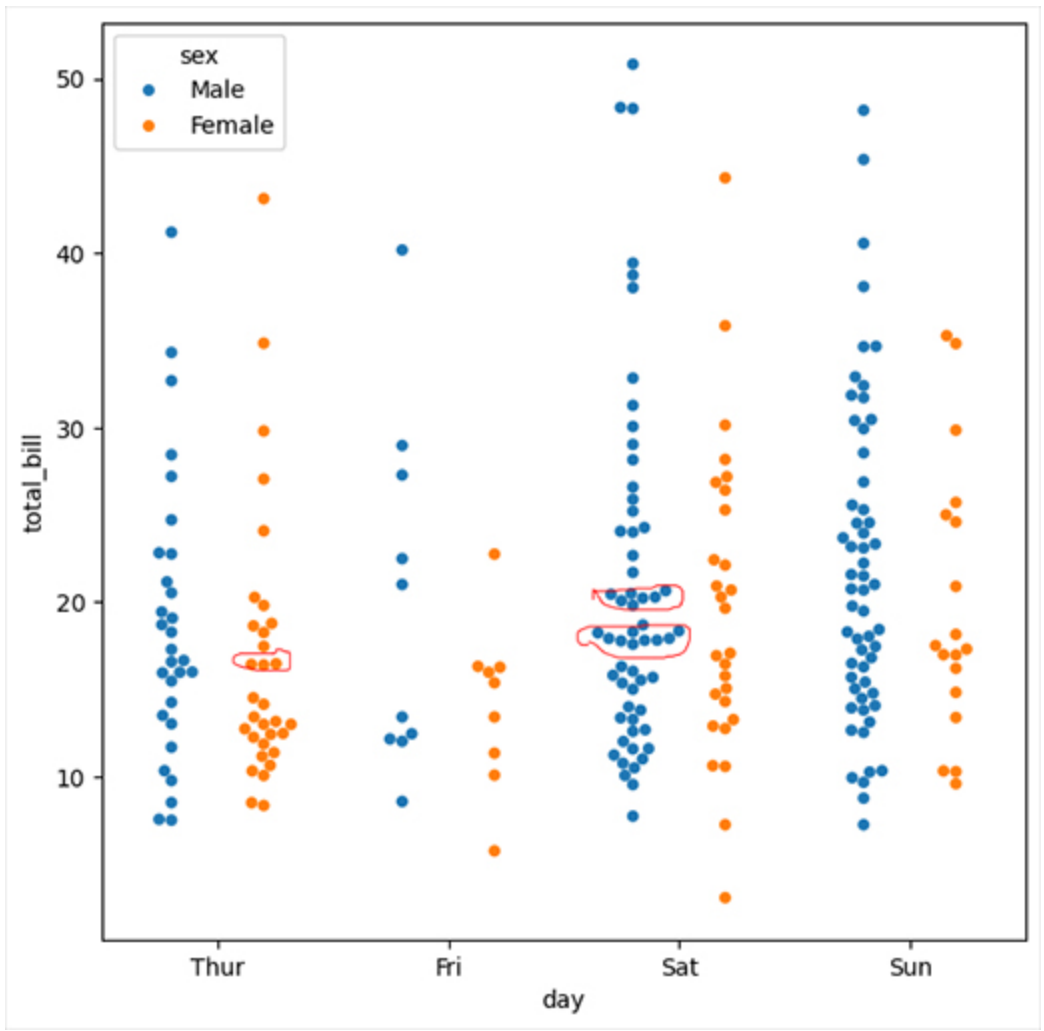
In [ ]:
```
#swarmplot
#Ampliando la gráfica para ver mejor los resultados
plt.figure(figsize=(7,7))
sns.swarmplot(data=tips,x='day',y='total_bill',hue='sex',dodge=True)
plt.show()
```



Explicación:

Este tipo de gráfico es similar al anterior, pero con una excepción.

> En la **concentración de datos** voy a tener un *eje Horizontal*

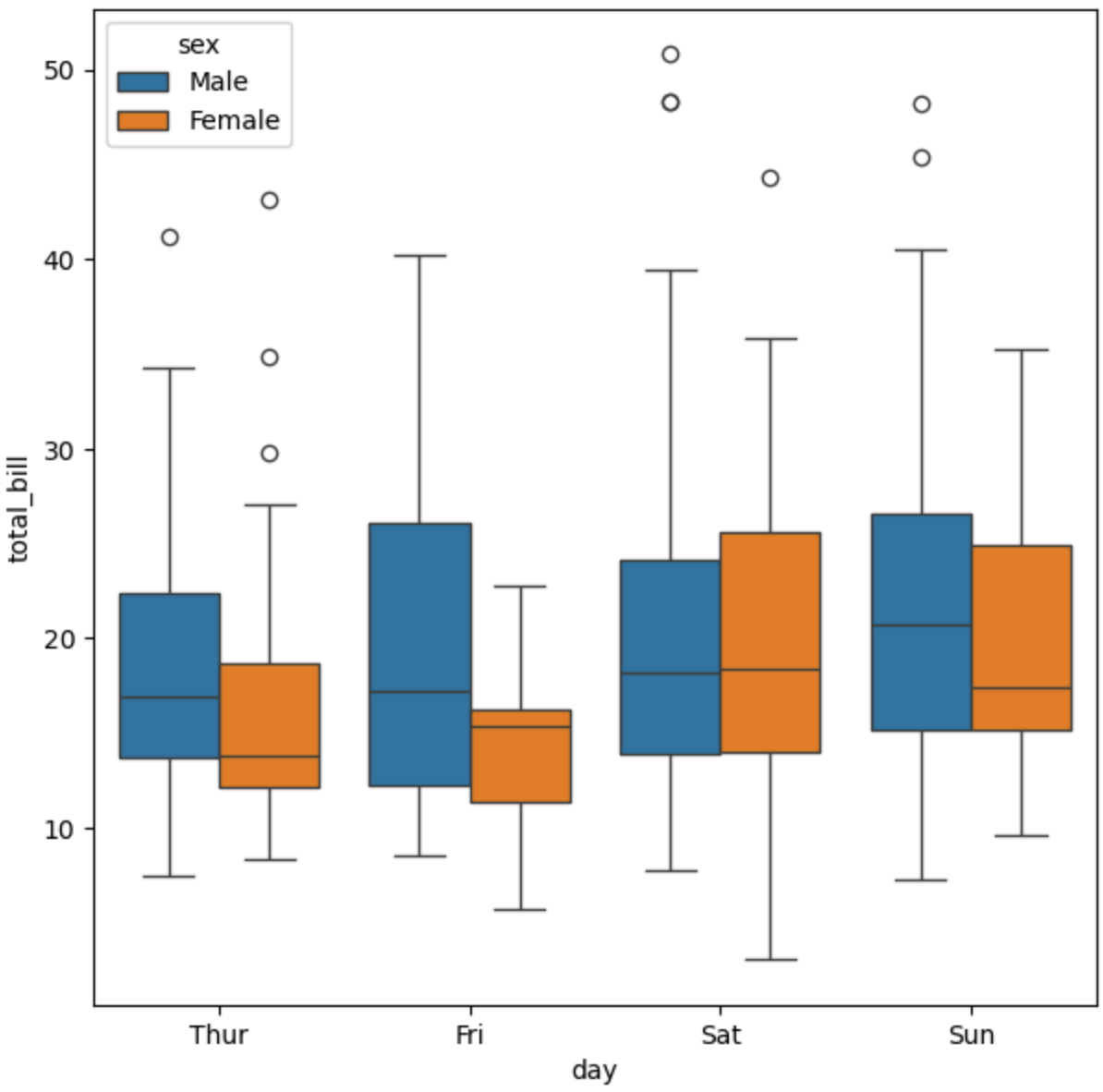


Análisis:

De aqui podemos decir que para el día `Jueves` las `mujeres` la gran mayoría pagaban entre 10-13 dls en la cuenta

Boxplot

In [ ]:
```
#Boxplot
#Ampliando la gráfica para ver mejor los resultados
plt.figure(figsize=(7,7))
sns.boxplot(data=tips,x='day',y='total_bill',hue='sex',dodge=True)
plt.show()
```

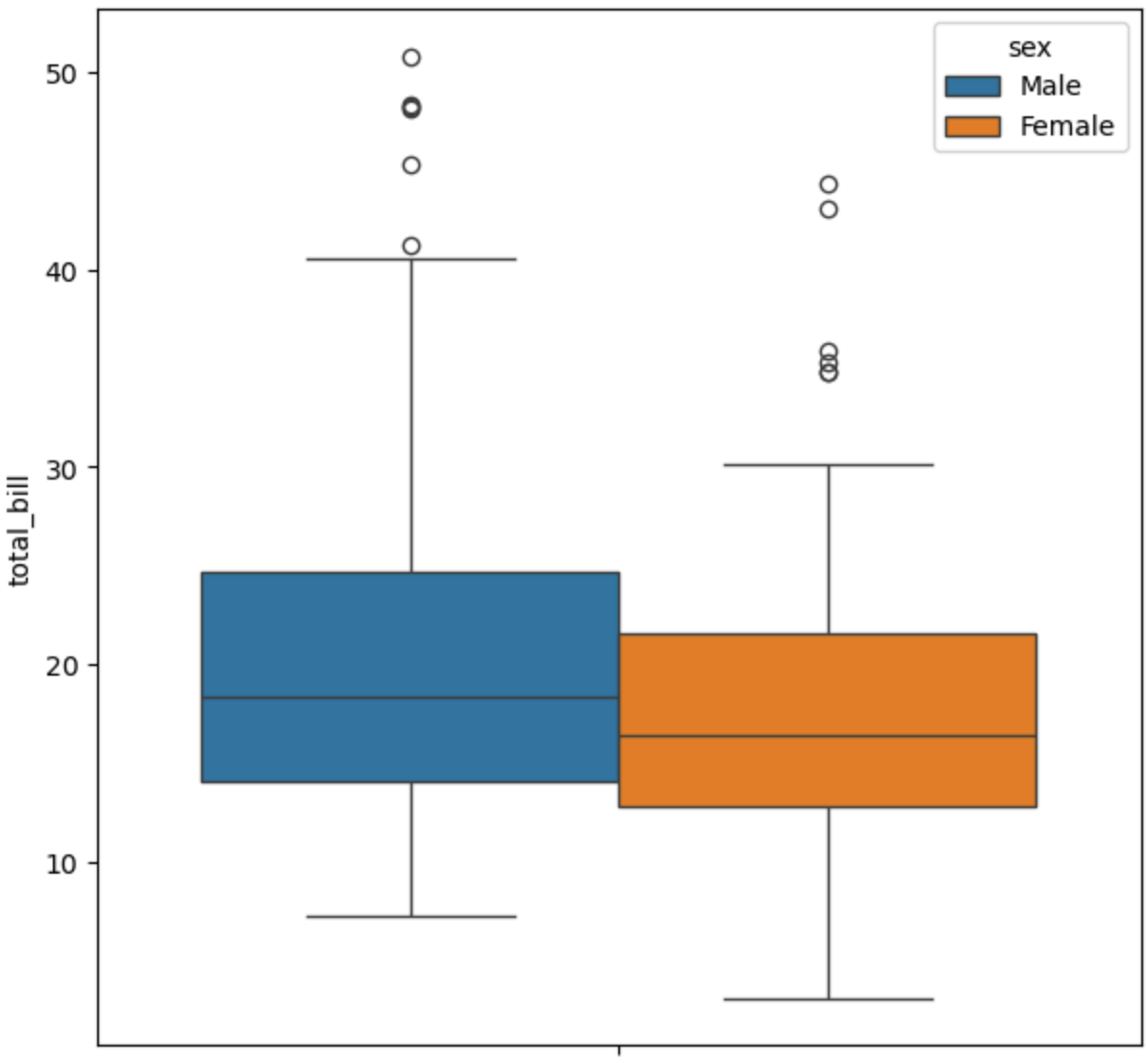


Explicación

Este tipo de gráfico nos ayuda a ver percentiles y otros parametros ademas, podemos ver la media de hombres casi en todos los días es más alta que de mujeres.
También me sirve para gráficar variables:

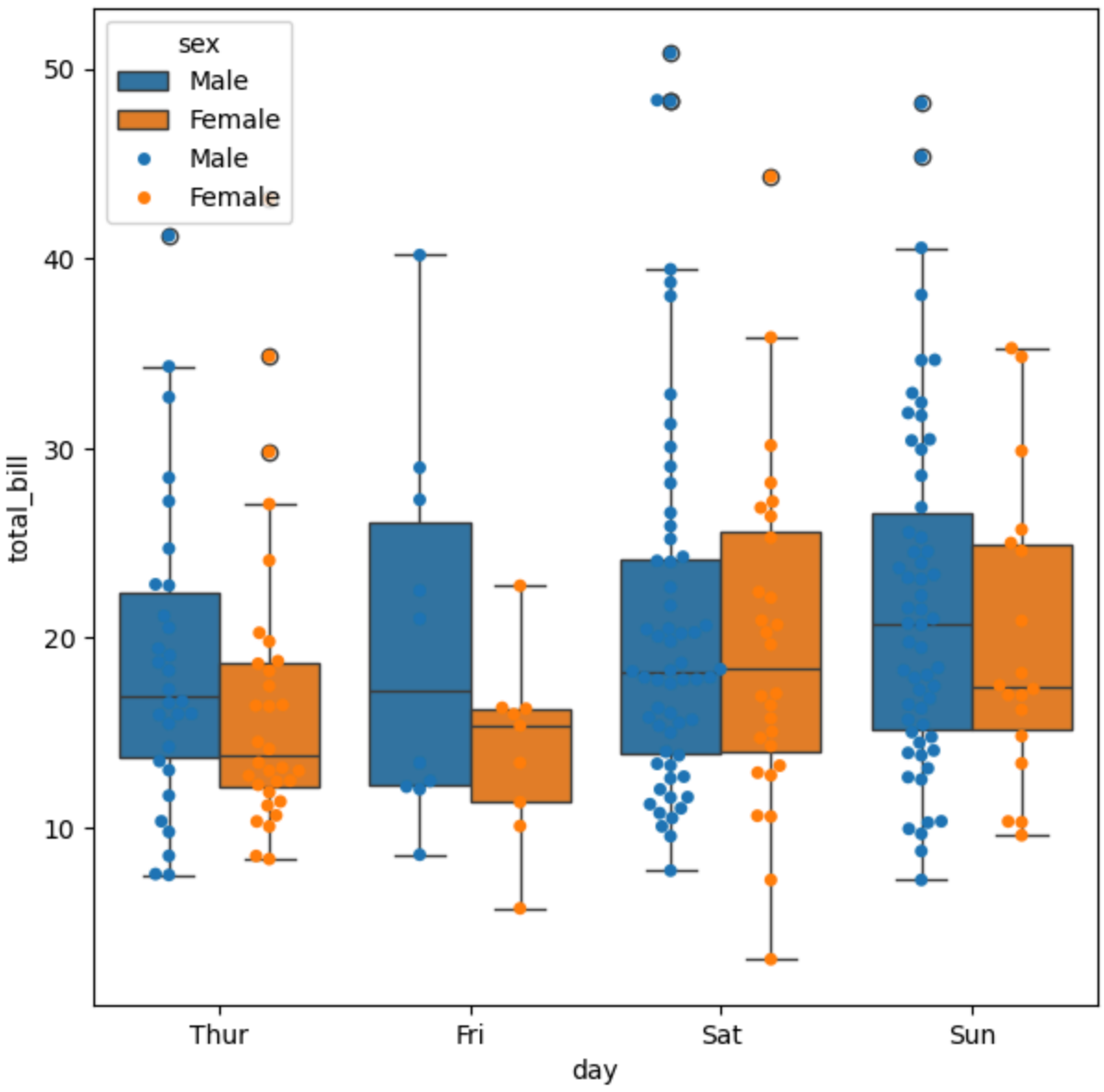
- Númericas
- Categoricas

In [ ]:
```
#Boxplot con total bill
#Ampliando la gráfica para ver mejor los resultados
plt.figure(figsize=(7,7))
sns.boxplot(data=tips,y='total_bill',hue='sex')
plt.show()
```



Puedo combinar Swarm plot y boxplot

In [ ]:
```
#Boxplot + Swarmplot
#Ampliando la gráfica para ver mejor los resultados
plt.figure(figsize=(7,7))
sns.boxplot(data=tips,x='day',y='total_bill',hue='sex',dodge=True)
sns.swarmplot(data=tips,x='day',y='total_bill',hue='sex',dodge=True)
plt.show()
```



Cambios:

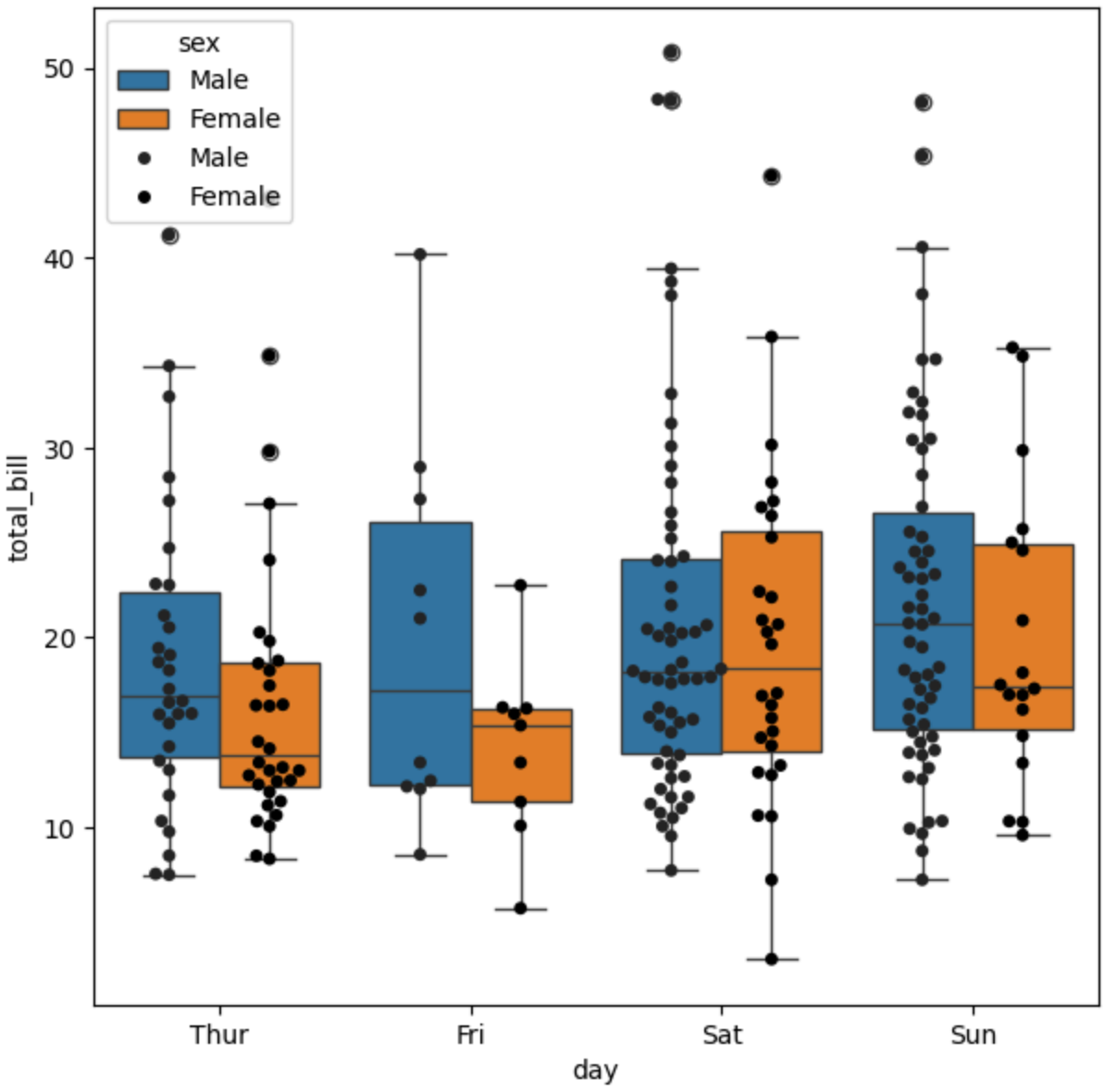
Le cambiare el color al Swarmplot `color=0`

In [ ]:
```
#Boxplot + Swarmplot
#Ampliando la gráfica para ver mejor los resultados
plt.figure(figsize=(7,7))
sns.boxplot(data=tips,x='day',y='total_bill',hue='sex',dodge=True)
sns.swarmplot(data=tips,x='day',y='total_bill',hue='sex',dodge=True,color='0')
plt.show()
```

```
/tmp/ipykernel_63830/2098932137.py:5: FutureWarning:

Setting a gradient palette using color= is deprecated and will be removed in v0.14.0. Set `palette='dark:0'` for the same effect.

  sns.swarmplot(data=tips,x='day',y='total_bill',hue='sex',dodge=True,color='0')
```

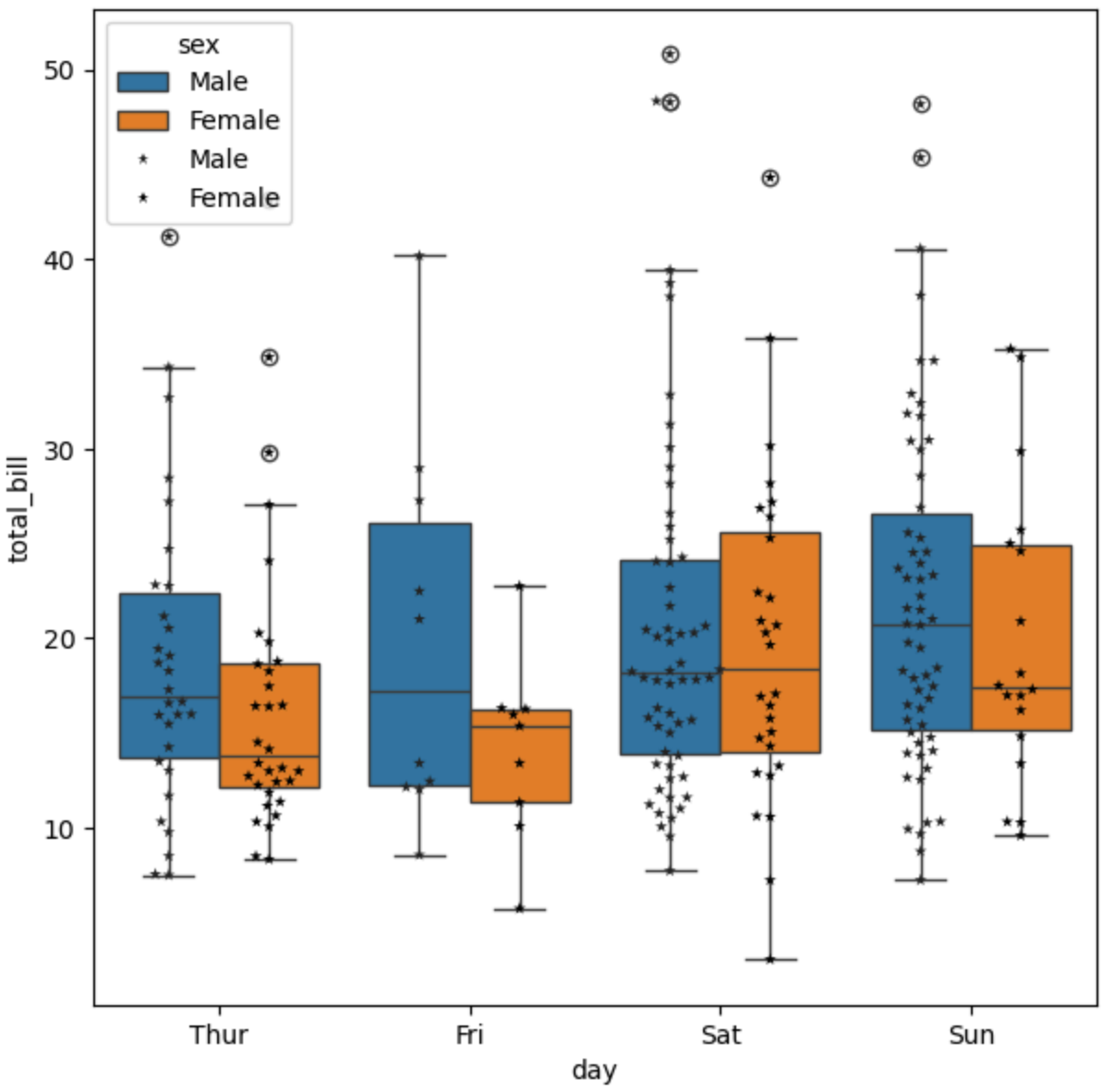


Hay cambios en la versión de Seaborn

palette='dark:0'

Además `marker='*'`

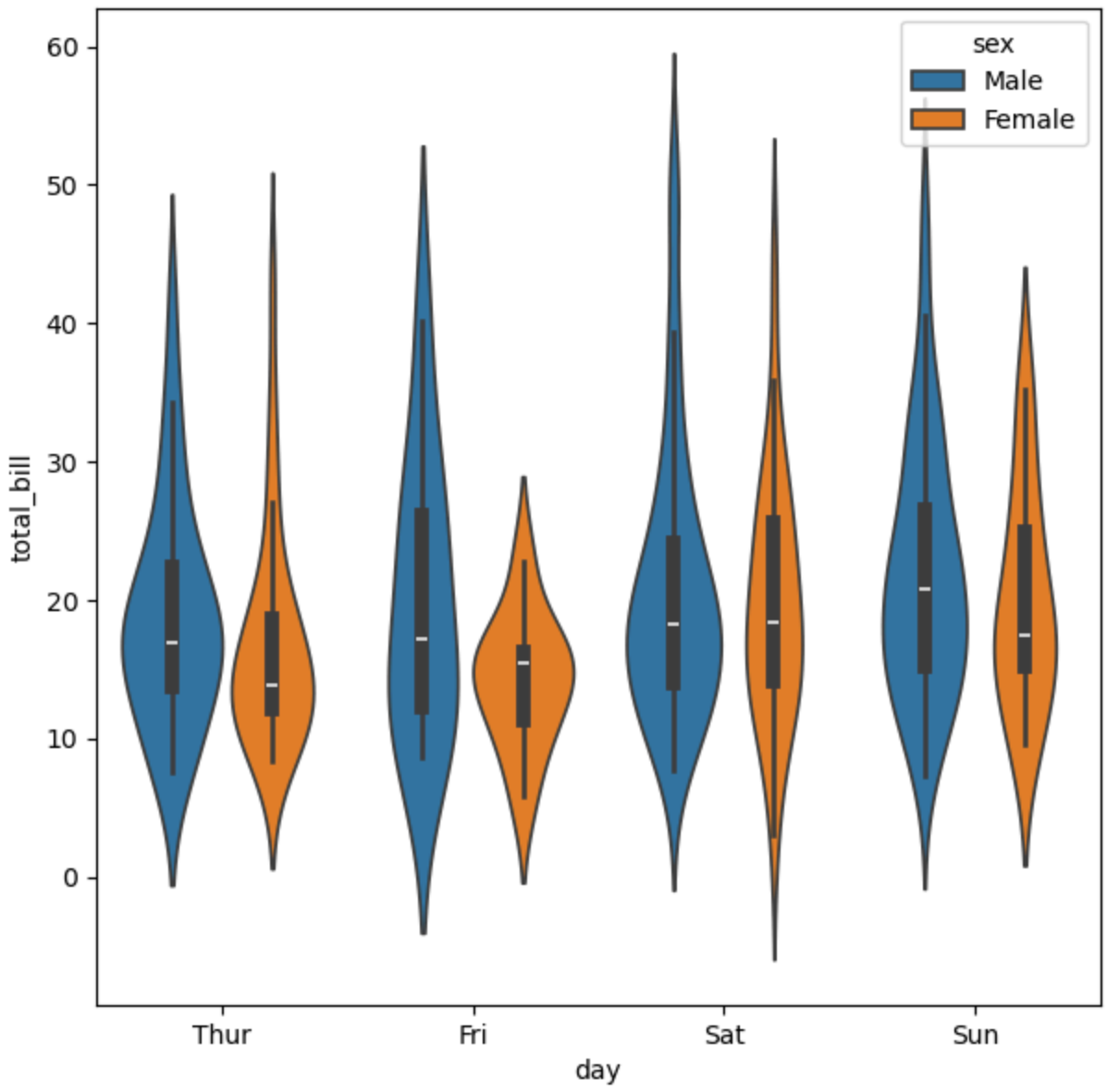
In [ ]:
```
#Boxplot + Swarmplot
#Ampliando la gráfica para ver mejor los resultados
plt.figure(figsize=(7,7))
sns.boxplot(data=tips,x='day',y='total_bill',hue='sex',dodge=True)
sns.swarmplot(data=tips,x='day',y='total_bill',hue='sex',dodge=True,palette='dark:0',marker='*')
plt.show()
```



Puedo personalizar como se trabaja la gráfica

Violin plot

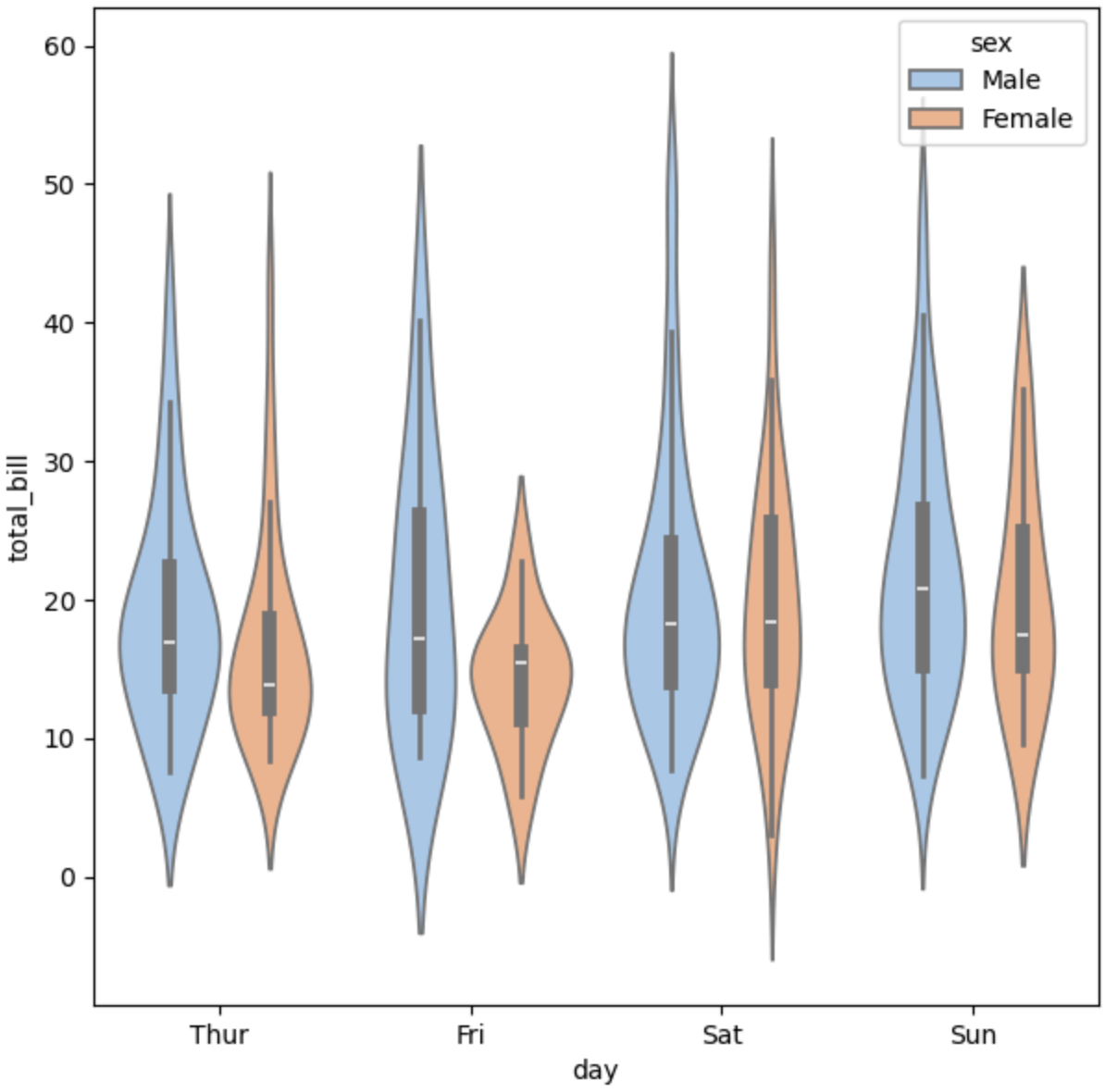
In [ ]:
```
#Violin plot
plt.figure(figsize=(7,7))
sns.violinplot(data=tips,x='day',y='total_bill',hue='sex',dodge=True)
plt.show()
```



Explicación:

Es muy parecido al `boxplot` solo que no me marca el `25%`, `75%`. En cuanto a percentiles.
Este gráfico me dirá como se están concentrando los datos con respecto a la figura, **es decir cuando se ensancha la curvatura, es ahí donde se concentran los datos**

In [ ]:
```
#Violin plot
plt.figure(figsize=(7,7))
#Cambiando la paleta de colores
sns.violinplot(data=tips,x='day',y='total_bill',hue='sex',dodge=True,palette='pastel')
plt.show()
```



Combinando los graficos

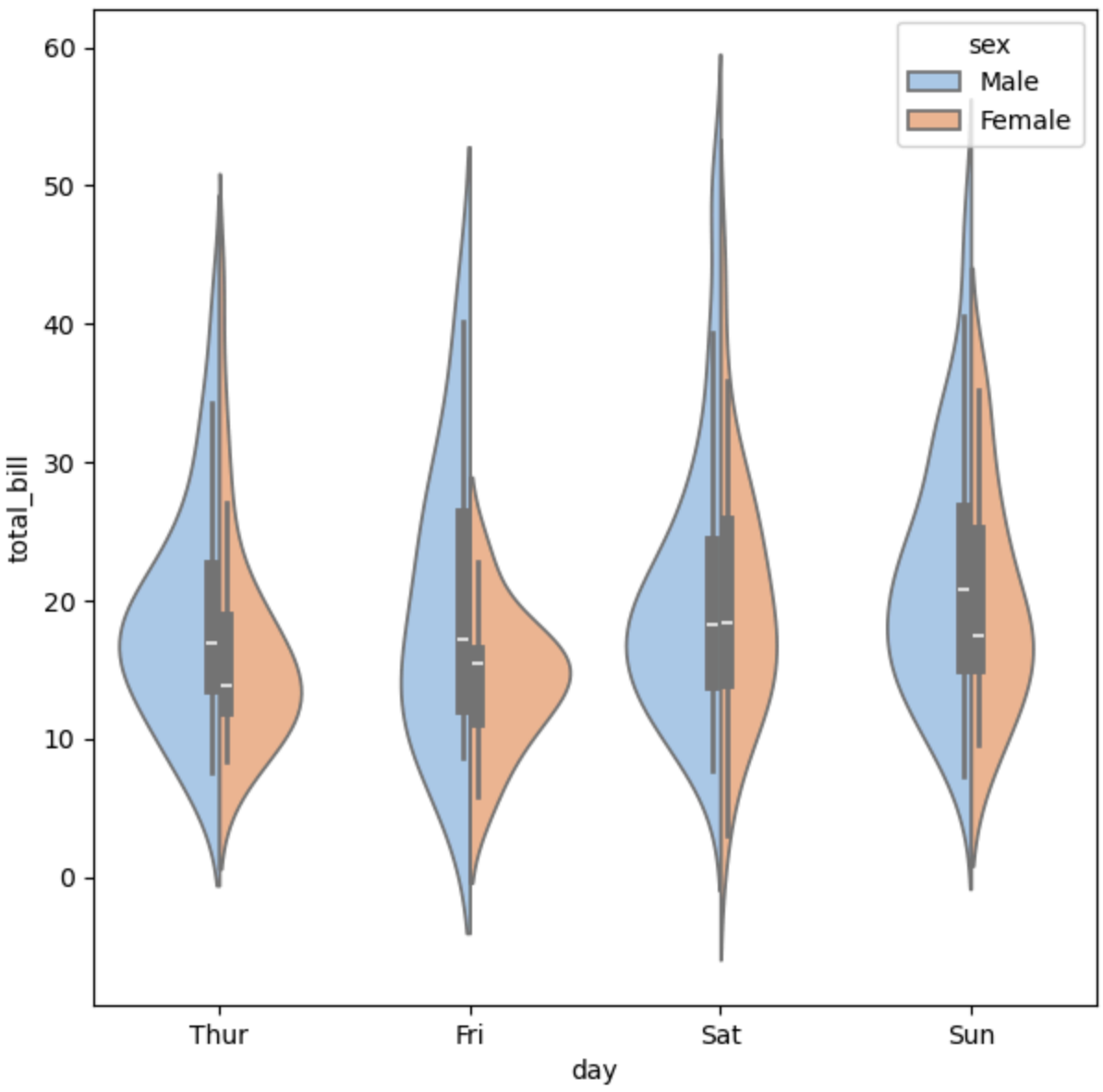
Recordemos:

- `dodge=True` : Me permite ver 2 gráficos separados por la `Segmentación` que hicimos.

Resaltemos:

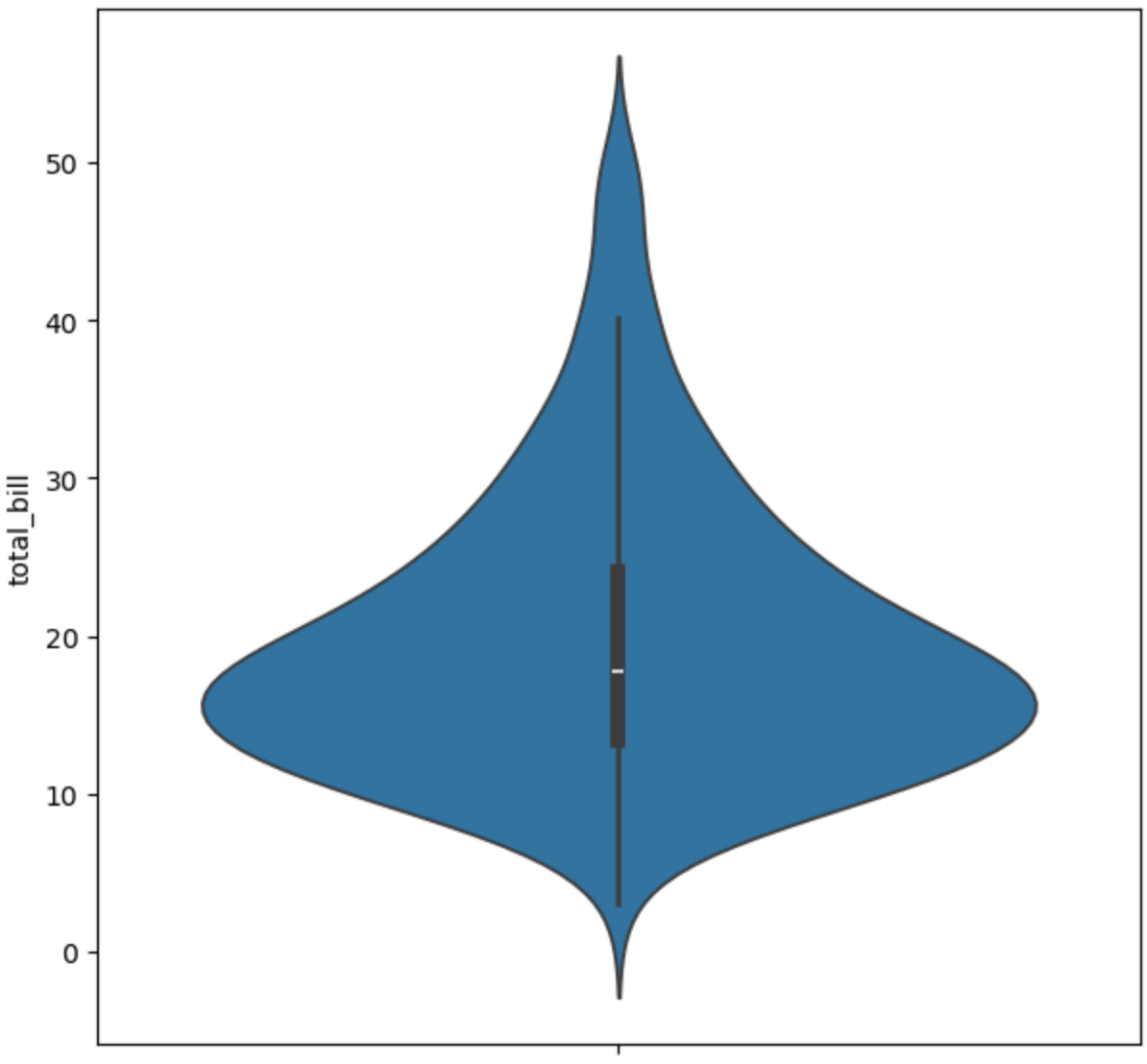
- `split=True` : Me permite combinar los 2 gráficos que me entrega la `Segmentación` de `dodge` , en un solo gráfico. Es decir los junta o divide la visualización '*mitad y mitad*'.

In [ ]:
```
#Violin plot
plt.figure(figsize=(7,7))
#Aplicando split
sns.violinplot(data=tips,x='day',y='total_bill',hue='sex',dodge=True,palette='pastel',split=True)
plt.show()
```



También se puede usar en variables númericas

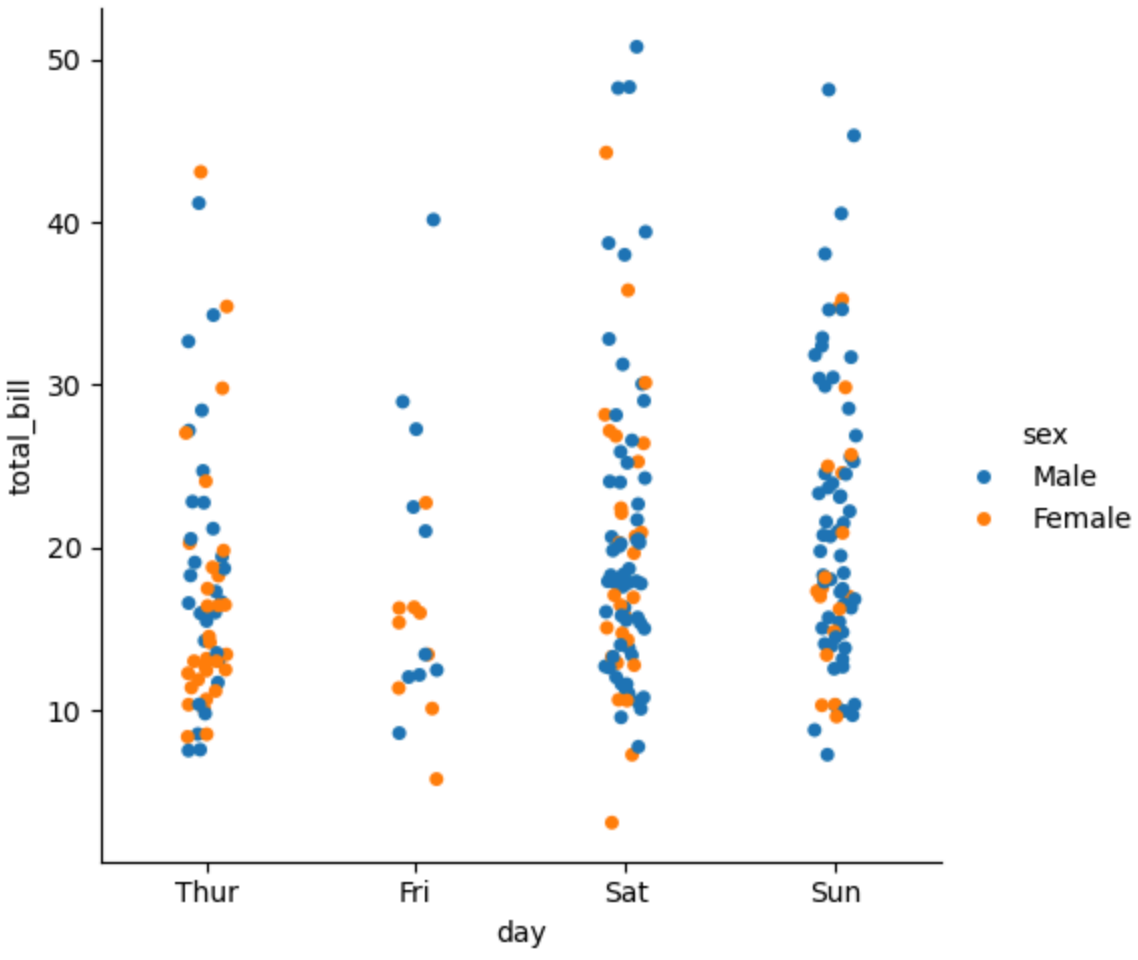
In [ ]:
```
#Violin plot para variables númericas
plt.figure(figsize=(7,7))
#Aplicando split
sns.violinplot(data=tips,y='total_bill')
plt.show()
```



Cat plot

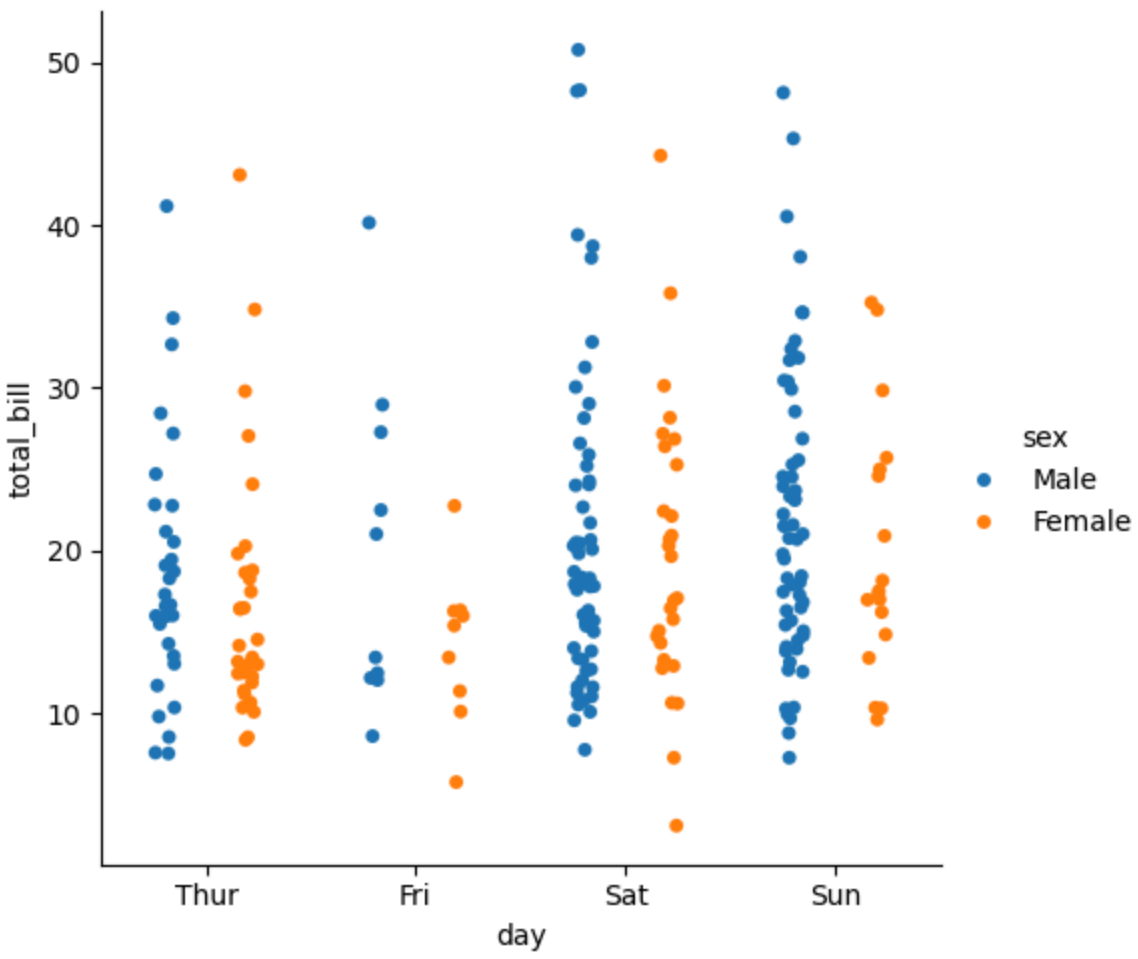
Va a graficar variables categoricas

In [ ]:
```
#Catplot
sns.catplot(data=tips,x='day',y='total_bill',hue='sex')
plt.show()
```



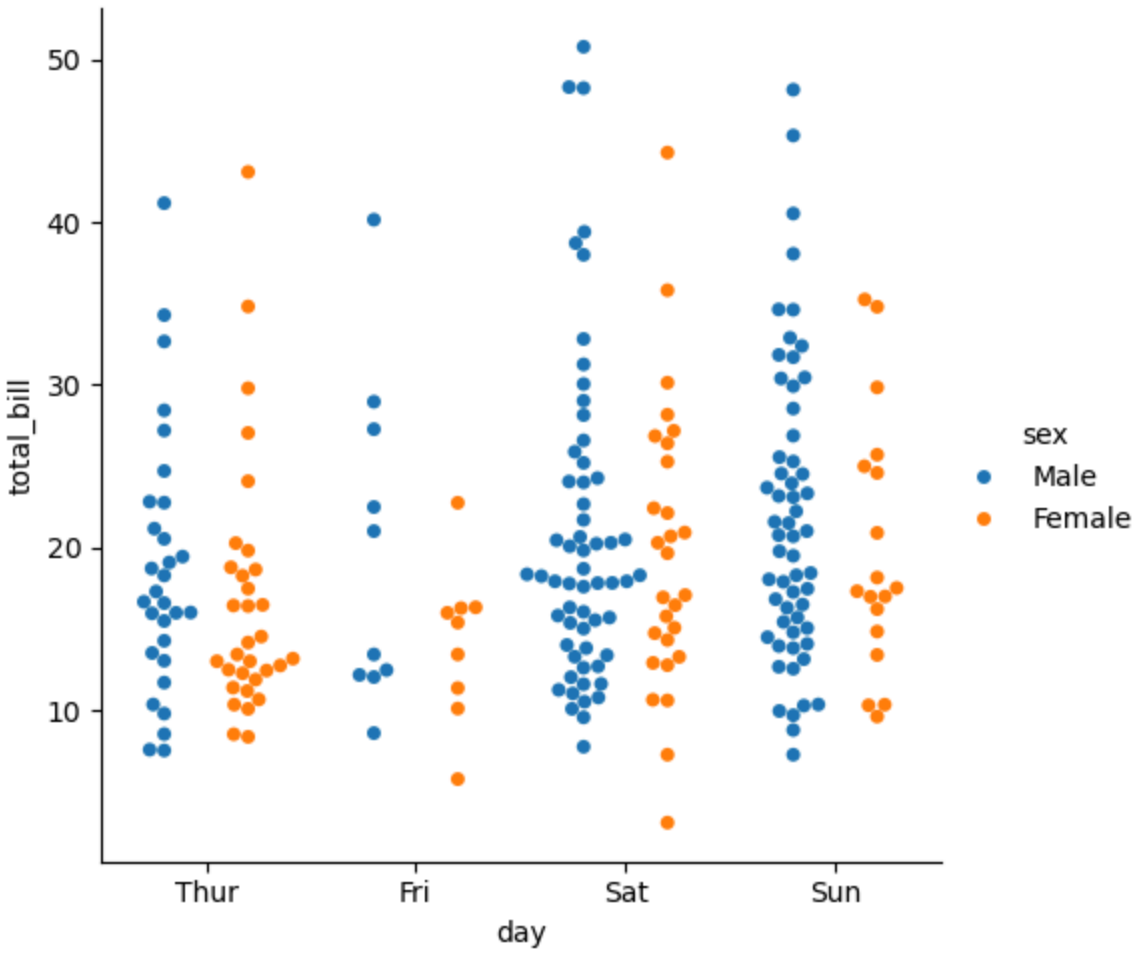
En este caso sin definir el tipo de gráfico, este método decide que trabajará con `stripplot`. Simplemente lo deja así, pero apliquemosle un `dodge=True`

In [ ]:
```
#Cat plot
sns.catplot(data=tips,x='day',y='total_bill',hue='sex',dodge=True)
plt.show()
```

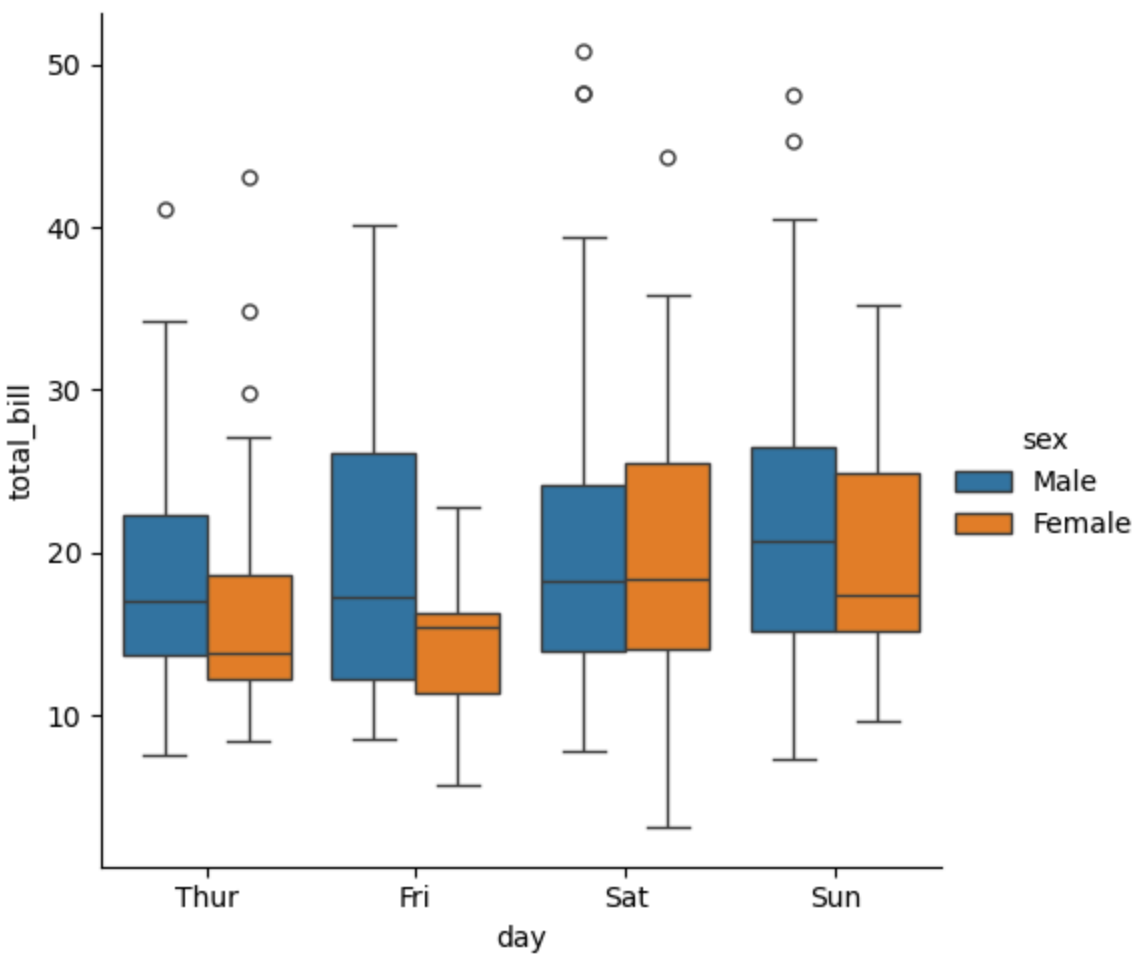


Parametro `kind=`

In [ ]:
```
#Cat plot
#kind='swarm'
sns.catplot(data=tips,x='day',y='total_bill',hue='sex',dodge=True,kind='swarm')
plt.show()
```

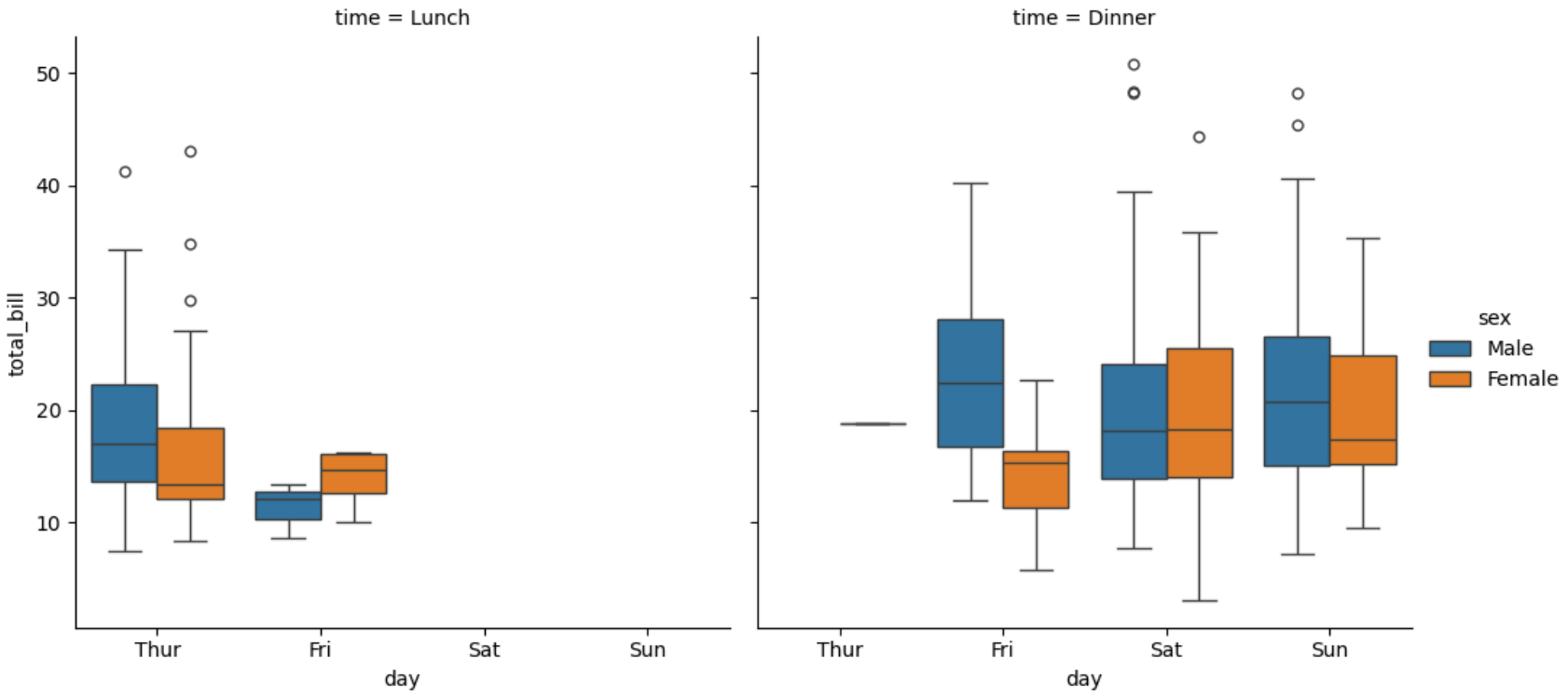


In [ ]:
```
#Cat plot
#kind='box'
sns.catplot(data=tips,x='day',y='total_bill',hue='sex',dodge=True,kind='box')
plt.show()
```



Puedo usar cualquier gráfica que use anteriormente, solo cambiando el parámetro de `kind`. Ademas puedo usar un parámetro extra `col`

In [ ]:
```
#Cat plot
#kind='box'
#col='time'
sns.catplot(data=tips,x='day',y='total_bill',hue='sex',dodge=True,kind='box',col='time')
plt.show()
```



Ahora tengo 2 gráficas `Almuerzo` y `Cena`, para saber como se comportaron entre los distintos dias `day` en cuanto a cuenta total `total_bill`.

Así puedo trabajar con 3 distintas variables solo con agregar un parametro como `col`

Referencia:

- Visualizing categorical data