Seaborn

1 SEABORN

- Biblioteca para visualização de dados baseado em Matplotlib.
- Interface de alto nível para gráficos estatísticos.
- Fornece uma interface atraente e profissional para os gráficos.
- Simples e muito intuitiva de usar.

1.0.1 Quando utilizar?

- Útil para análise e exploração de dados.
- Apresentar análises visuais.

1.0.2 Instalação da Biblioteca

Caso esteja usando anaconda instale o seaborn com o gerenciador de pacotes pip ou usando o conda.

Exemplo:

> pip install seaborn
>
> ou
>
> conda install seaborn

---

2 Plots de distribuições

Vamos discutir alguns gráficos que nos permitem visualizar a distribuição de um conjunto de dados. Esses plots são:

- distplot
- jointplot
- pairplot
- rugplot
- kdeplot

2.1 Imports

[ ]:
```
import seaborn as sns
%matplotlib inline
```

2.2 Dados

Seaborn vem com conjuntos de dados embutidos.

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

[9]:
```
tips.head(500)
```

[9]:

	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	3.61	Female	No	Sun	Dinner	4
..	…	…	…	…	…	…	…
239	29.03	5.92	Male	No	Sat	Dinner	3
240	27.18	2.00	Female	Yes	Sat	Dinner	2
241	22.67	2.00	Male	Yes	Sat	Dinner	2
242	17.82	1.75	Male	No	Sat	Dinner	2
243	18.78	3.00	Female	No	Thur	Dinner	2

[244 rows x 7 columns]

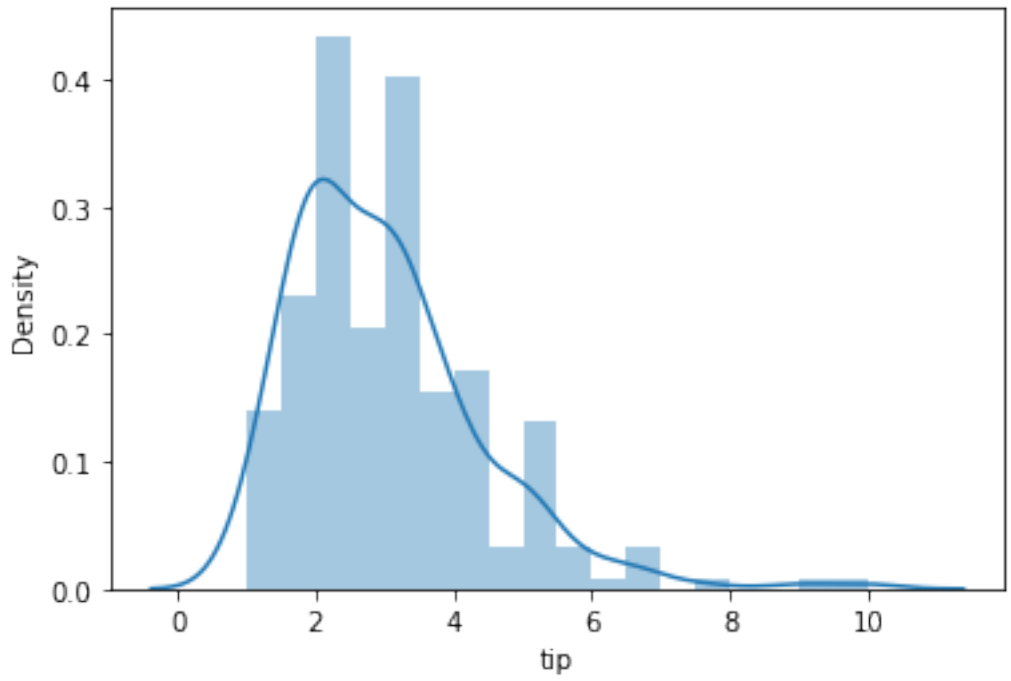
2.3 distplot

O distplot mostra a distribuição de um conjunto de observações de uma variável.

[ ]:
```
sns.distplot(tips['tip'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8ce7b7c4d0>
```

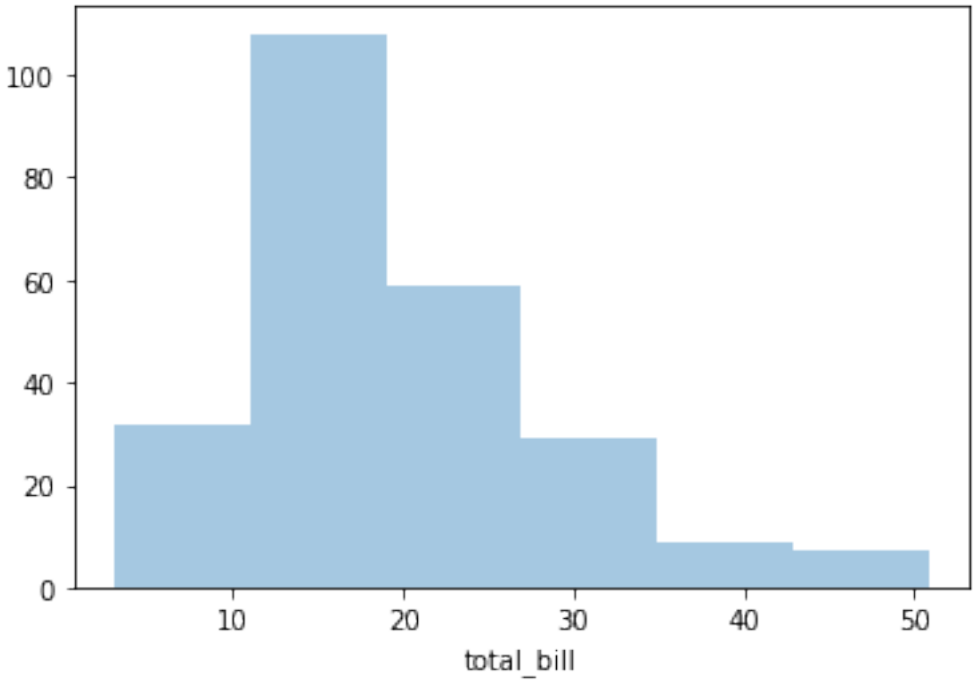


Para remover a camada kde e apenas usar o histograma:

[ ]:
```
sns.distplot(tips['total_bill'],kde=False,bins=6)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8ce76654d0>
```

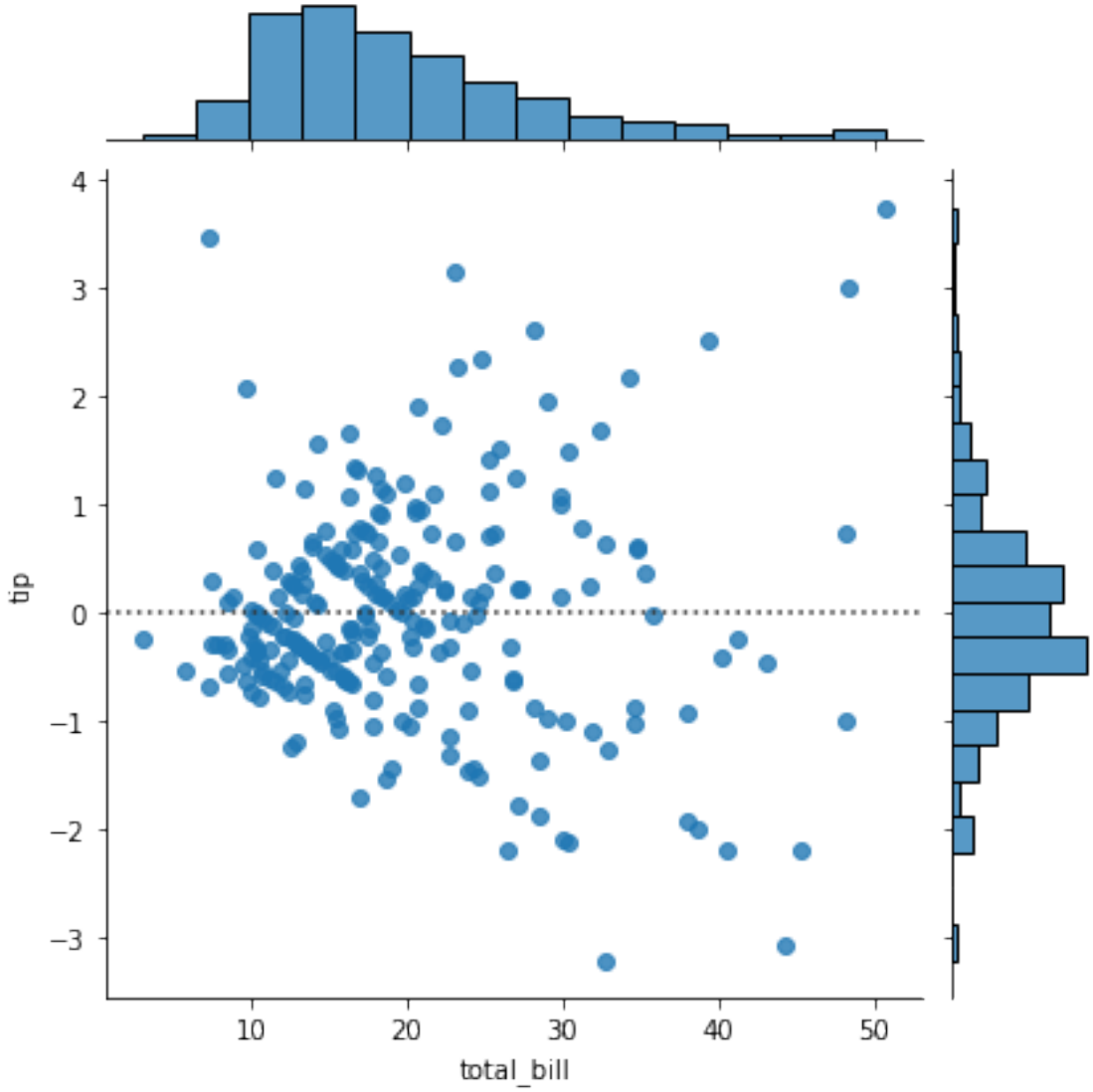


2.4 jointplot

jointplot() permite combinar basicamente dois distplots() para dados bivariados. Podemos visualizar os dados das seguintes formas (usando o **kind**): * “scatter” * “reg” * “resid” * “kde” * “hex”

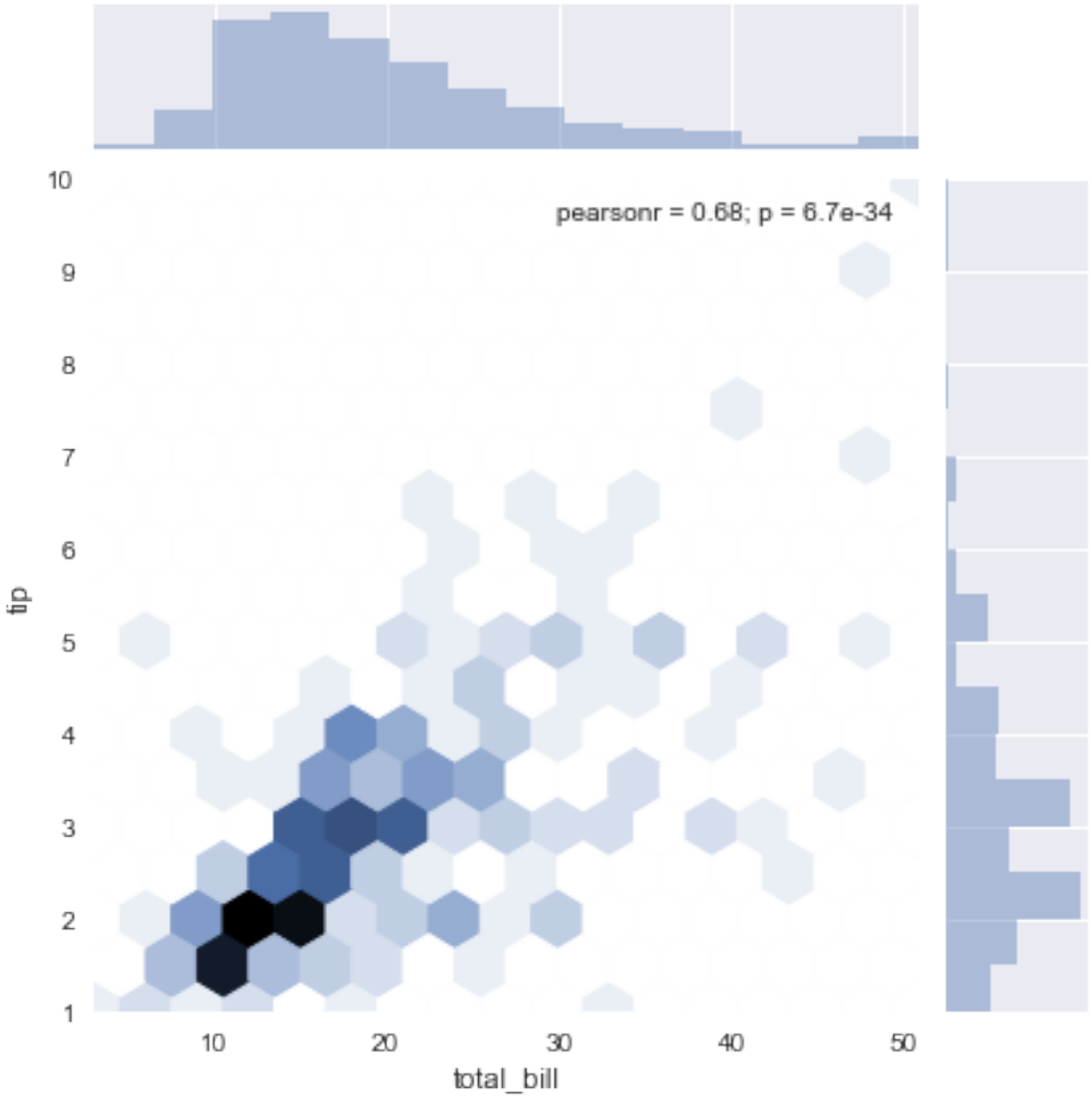
[10]:
```
sns.jointplot(x='total_bill',y='tip',data=tips,kind='resid')
```

[10]:
```
<seaborn.axisgrid.JointGrid at 0x7f8cdeaa4690>
```



[ ]:
```
sns.jointplot(x='total_bill',y='tip',data=tips,kind='hex')
```

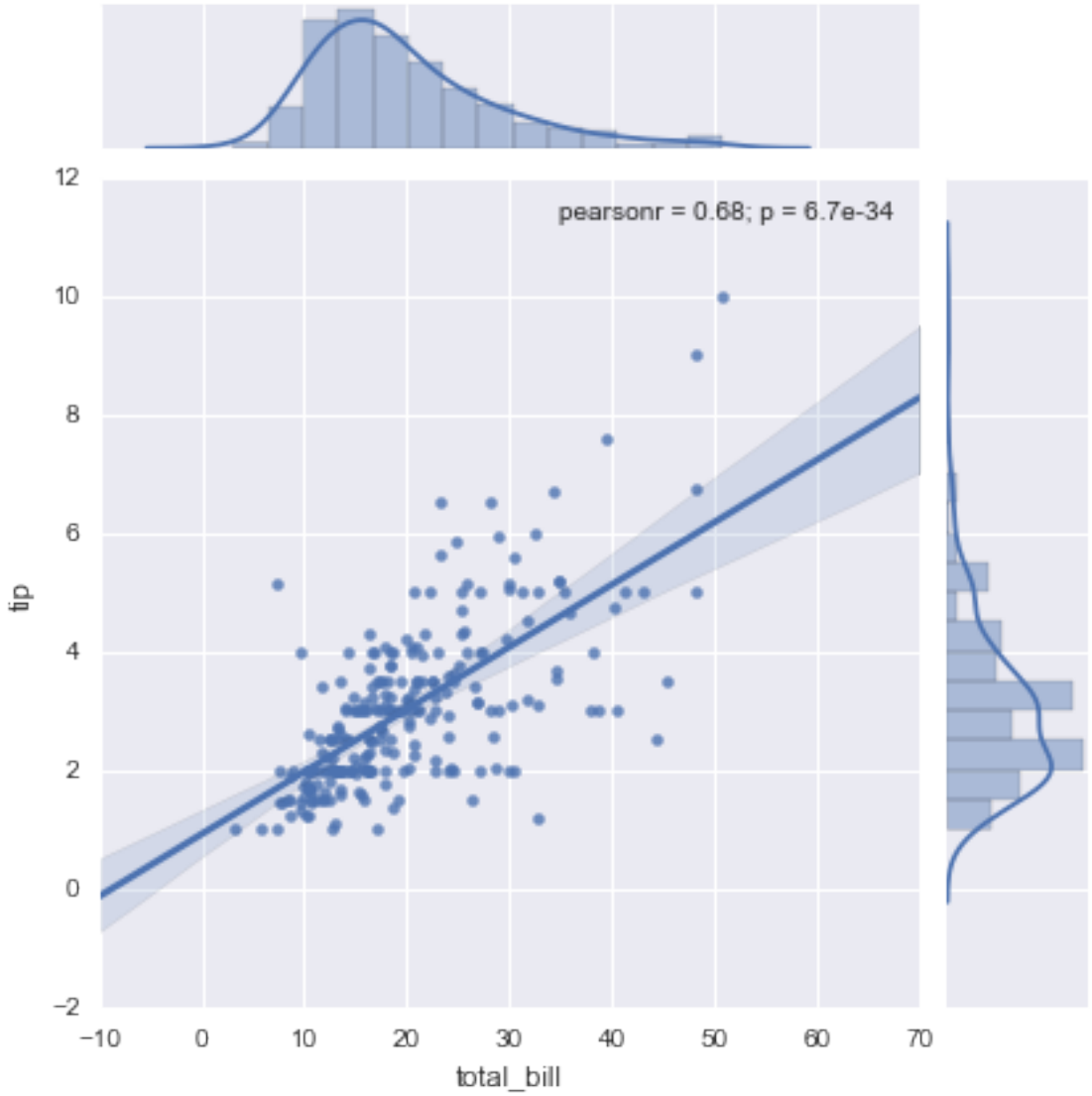
[ ]:
```
<seaborn.axisgrid.JointGrid at 0x158d2b0d1d0>
```



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

```
/Users/marci/anaconda/lib/python3.5/site-
packages/statsmodels/nonparametric/kdetools.py:20: VisibleDeprecationWarning:
using a non-integer number instead of an integer will result in an error in the
future
  y = X[:m/2+1] + np.r_[0,X[m/2+1:],0]*1j
```

[ ]:
```
<seaborn.axisgrid.JointGrid at 0x11e0cfba8>
```

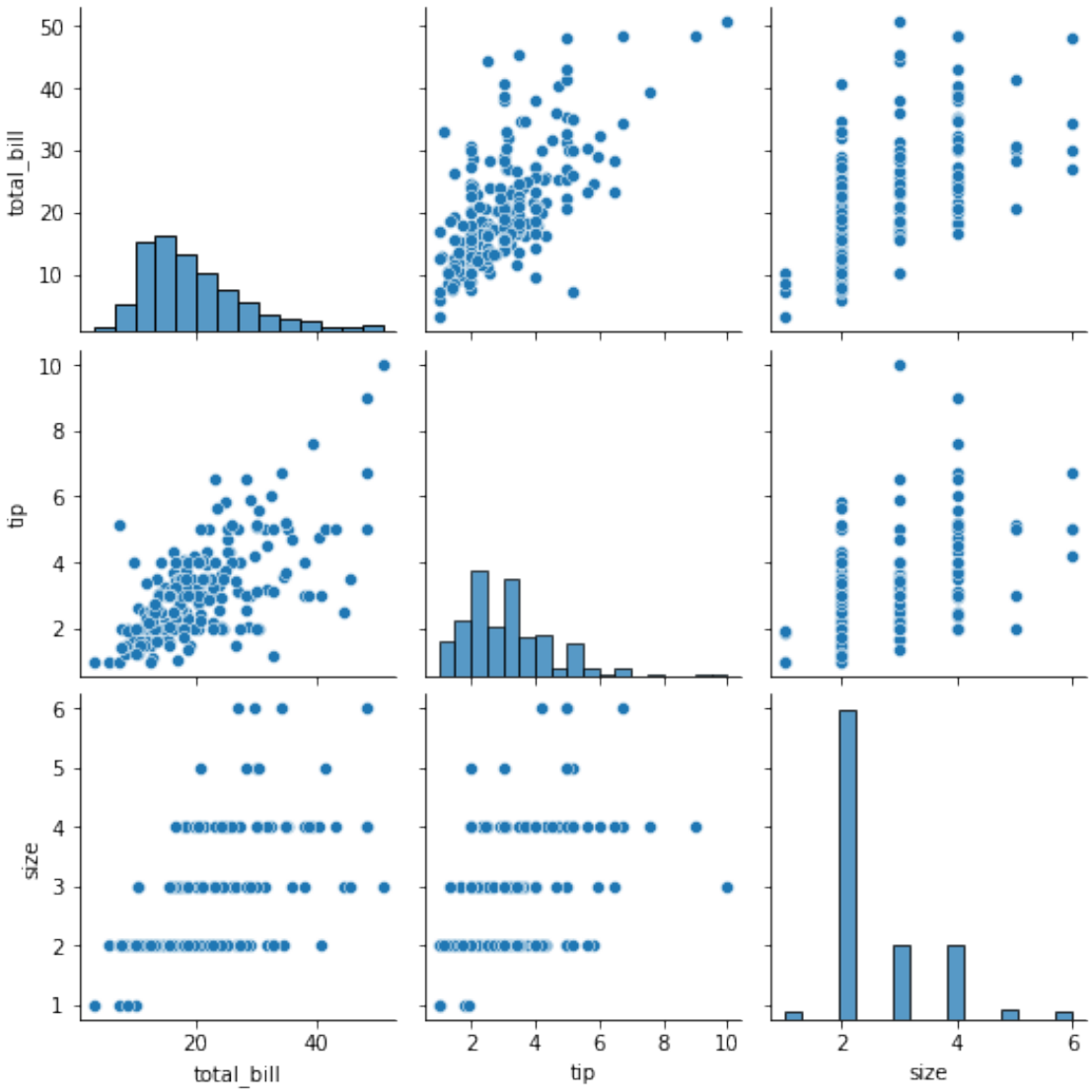


2.5 pairplot

pairplot irá traçar distribuições entre pares em todo o DataFrame (para as colunas numéricas) e suporta um argumento de matiz de cor (para colunas categóricas).

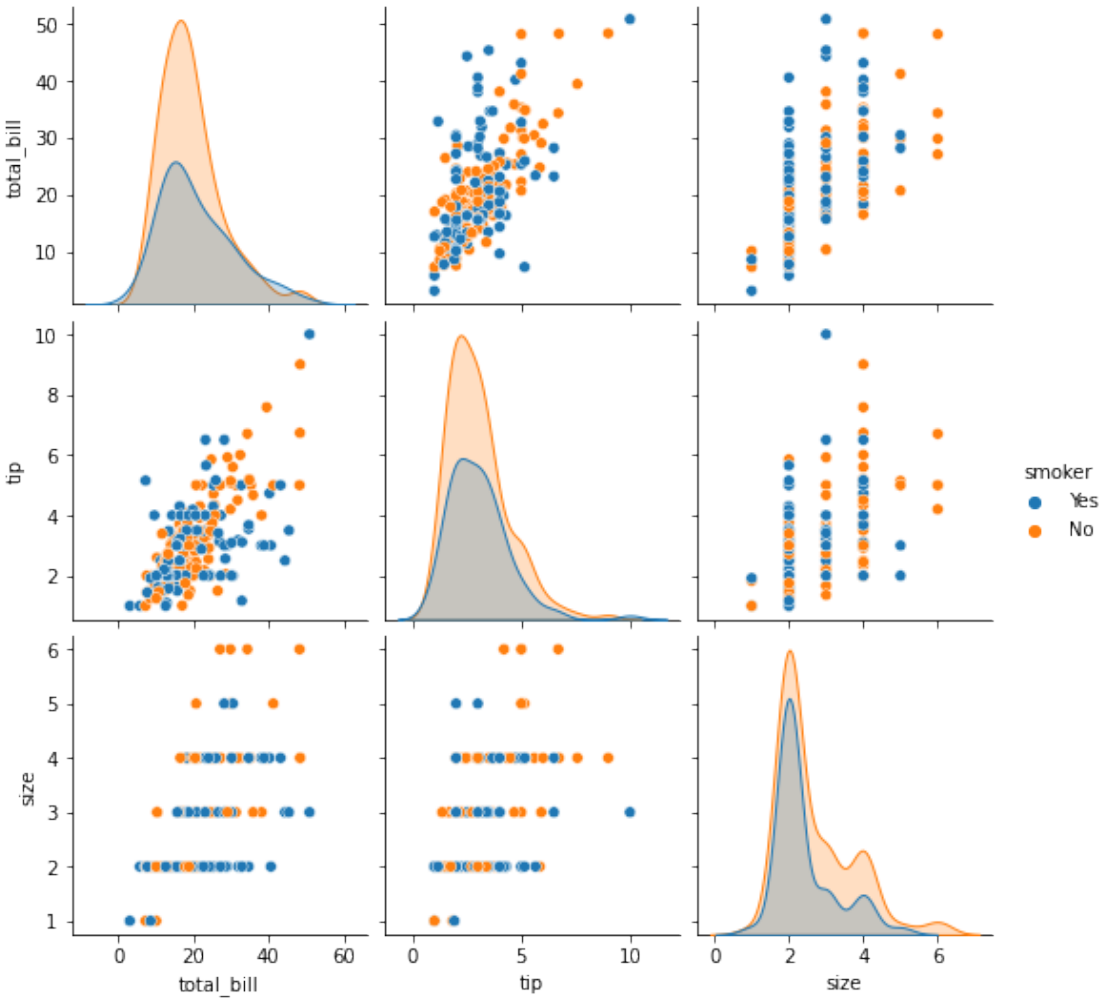
[ ]:
```
sns.pairplot(tips)
```

[ ]:
```
<seaborn.axisgrid.PairGrid at 0x7f72312c37d0>
```



[13]:
```
sns.pairplot(tips,hue='smoker')
```

[13]: <seaborn.axisgrid.PairGrid at 0x7f8cdc971e50>

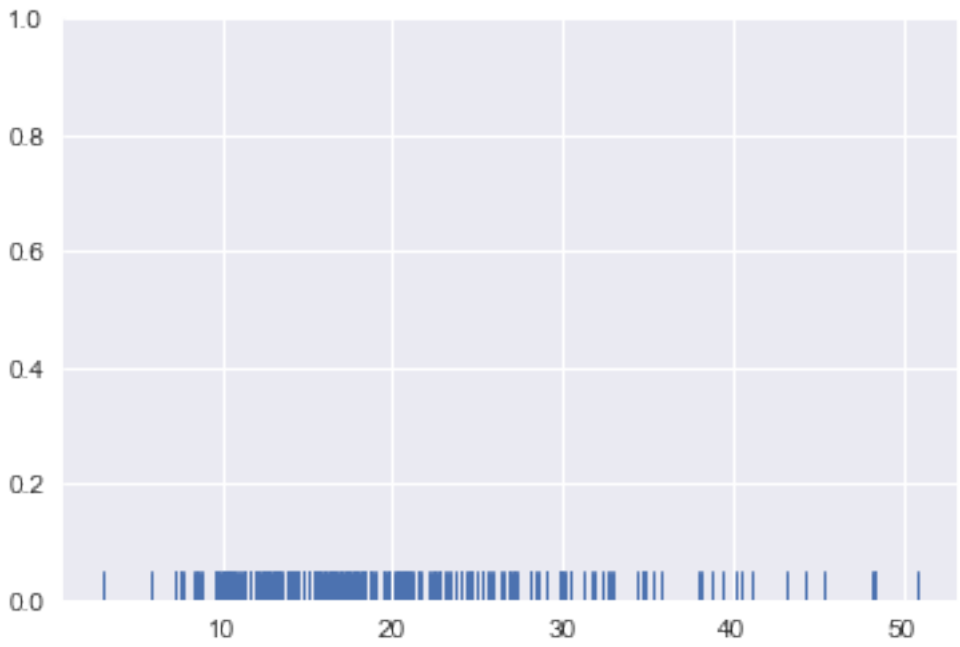


2.6 rugplot

rugplots possuem um conceito muito simples, eles apenas desenham uma marca de traço para cada ponto em uma distribuição univariada. Eles são o bloco de construção de um KDE:

[ ]:
```
sns.rugplot(tips['total_bill'])
```

[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x158d41a3128>
```



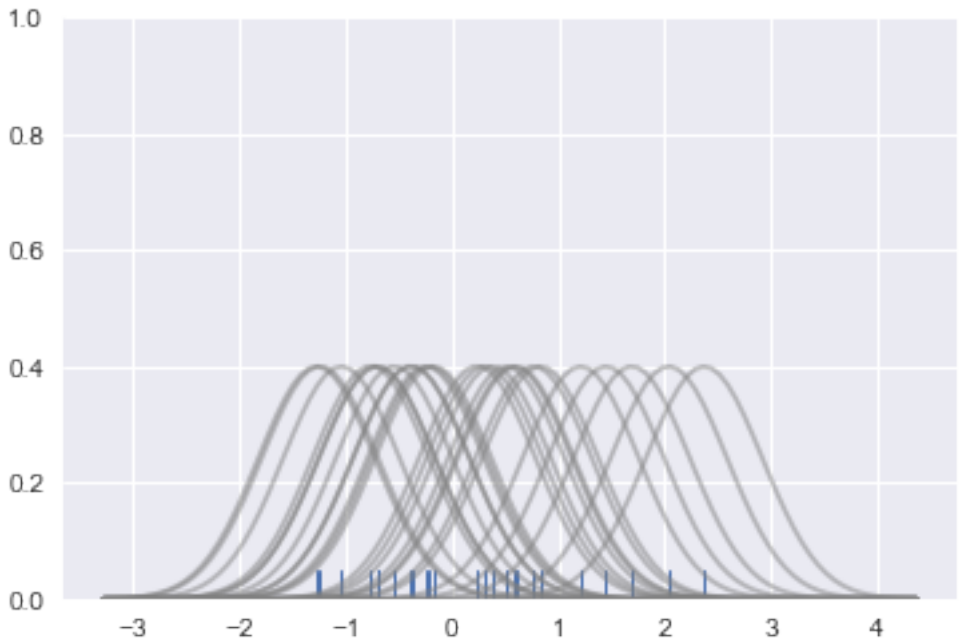
2.7 kdeplot

kdeplots são Plots de estimativa de densidade kernel. Esses plots KDE substituem cada observação com uma distribuição Gaussiana (Normal) centrada em torno desse valor. Por exemplo:

[ ]:
```
# Não se preocupe em entender este código!
# É apenas para o diagrama abaixo
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats

# Cria o dataset
dataset = np.random.randn(25)

# Cria outro rugplot
sns.rugplot(dataset);

# Configure o eixo dos x para o gráfico
x_min = dataset.min() - 2
x_max = dataset.max() + 2

# 100 pontos igualmente espaçados de x_min para x_max
x_axis = np.linspace(x_min,x_max,100)

# Configure a largura de banda. Para obter informações sobre isso:
```

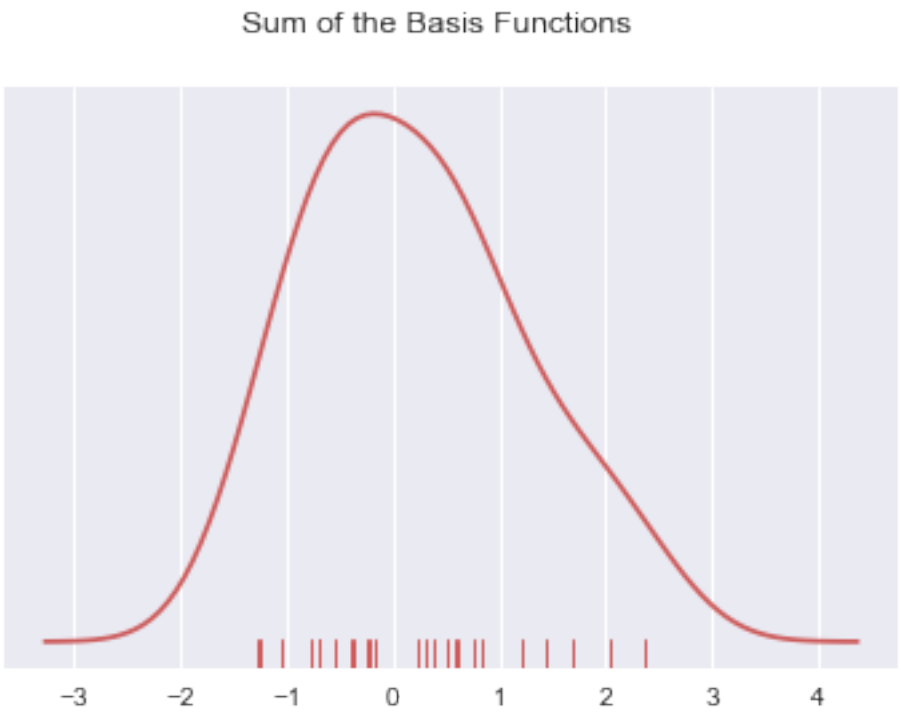
```
url = 'http://en.wikipedia.org/wiki/
 ↪Kernel_density_estimation#Practical_estimation_of_the_bandwidth'

bandwidth = ((4*dataset.std()**5)/(3*len(dataset)))**.2

# Crie uma lista de kernel vazia
kernel_list = []

# Traça cada função de base
for data_point in dataset:

    # Crie um kernel para cada ponto e acrescente à lista
    kernel = stats.norm(data_point,bandwidth).pdf(x_axis)
    kernel_list.append(kernel)

    # Ajusta a escala para plotar
    kernel = kernel / kernel.max()
    kernel = kernel * .4
    plt.plot(x_axis,kernel,color = 'grey',alpha=0.5)

plt.ylim(0,1)
```

[ ]: (0, 1)



[ ]:
```
# Para obter o gráfico do kde podemos somar essas funções de base.

# Traça a soma da função de base
sum_of_kde = np.sum(kernel_list,axis=0)

# Plota a figura
fig = plt.plot(x_axis,sum_of_kde,color='indianred')

# Adiciona o rugplot inicial
sns.rugplot(dataset,c = 'indianred')

# Livrar-se das marcas de "y-tick"
plt.yticks([])

# Define o título
plt.suptitle("Sum of the Basis Functions")
```

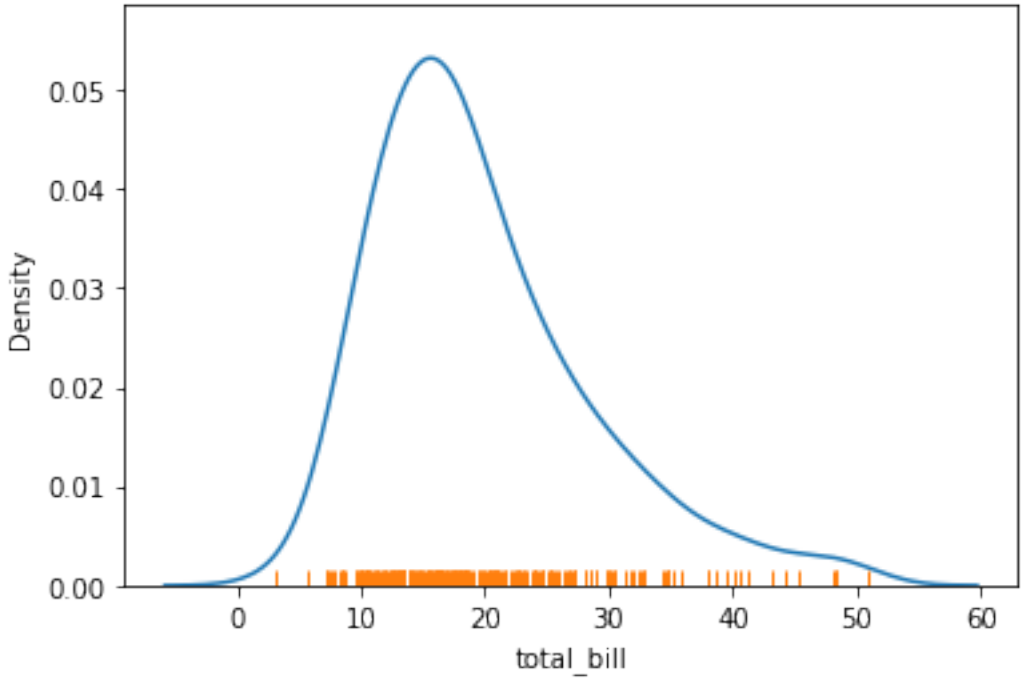
[ ]:
```
<matplotlib.text.Text at 0x158d467beb8>
```



Então, com nosso DataFrame tips:

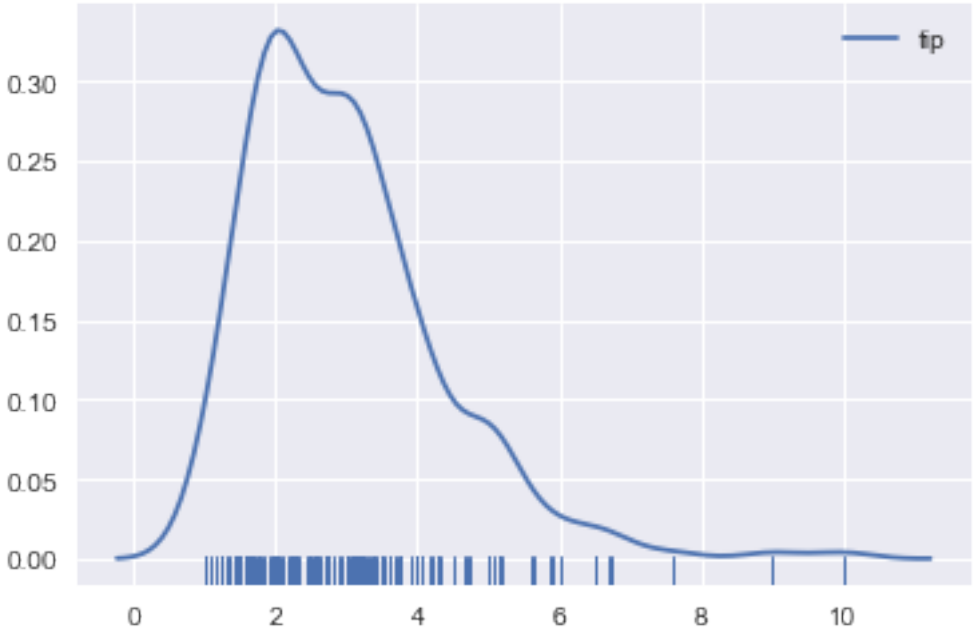
[14]:
```
sns.kdeplot(tips['total_bill'])
sns.rugplot(tips['total_bill'])
```

[14]: <matplotlib.axes._subplots.AxesSubplot at 0x7f8cdc5ec4d0>



[ ]:
```
sns.kdeplot(tips['tip'])
sns.rugplot(tips['tip'])
```

[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x158d4e1fc50>



3 Plots categóricos

Agora vamos discutir como usar seaborn para traçar dados categóricos. Existem alguns tipos de argumentos principais para isso:

- factorplot
- boxplot
- violinplot
- stripplot
- swarmplot
- barplot
- countplot

Vamos passar por exemplos de cada um.

[15]:
```
import seaborn as sns
%matplotlib inline
```

[16]:
```
tips = sns.load_dataset('tips')
tips.head()
```

[16]:

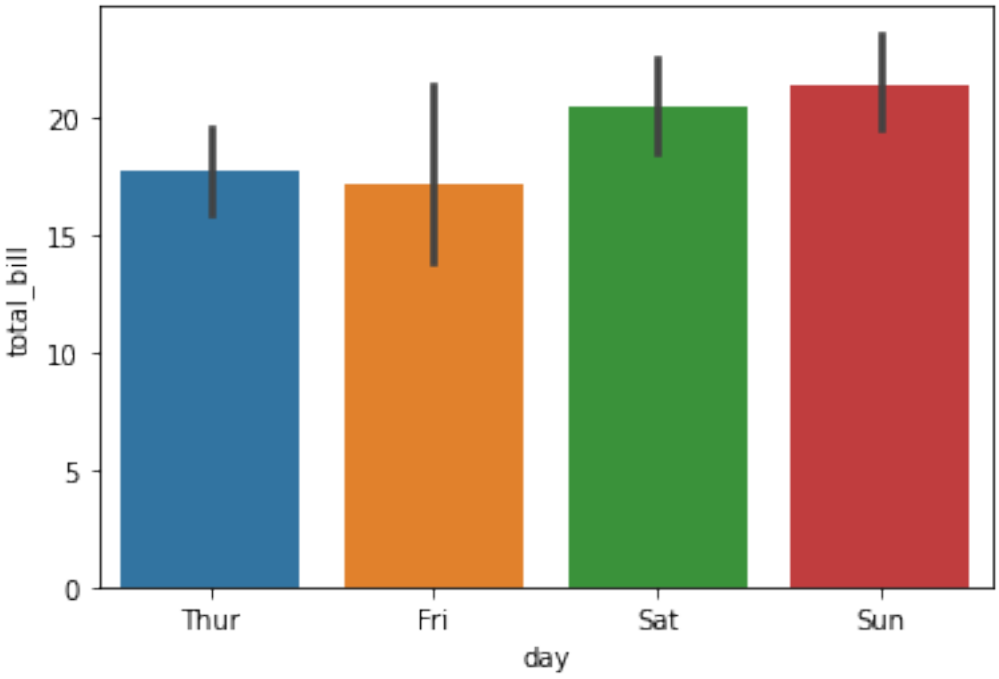
	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

3.1 barplot e countplot

Esses plots parecidos permitem que você obtenha dados agregados de um recurso categórico. ** barplot ** é um gráfico geral que permite que você agregue os dados categóricos baseados em alguma função, por padrão, a média:

[18]:
```
sns.barplot(x='day',y='total_bill',data=tips)
```

[18]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8cdc48e0d0>
```

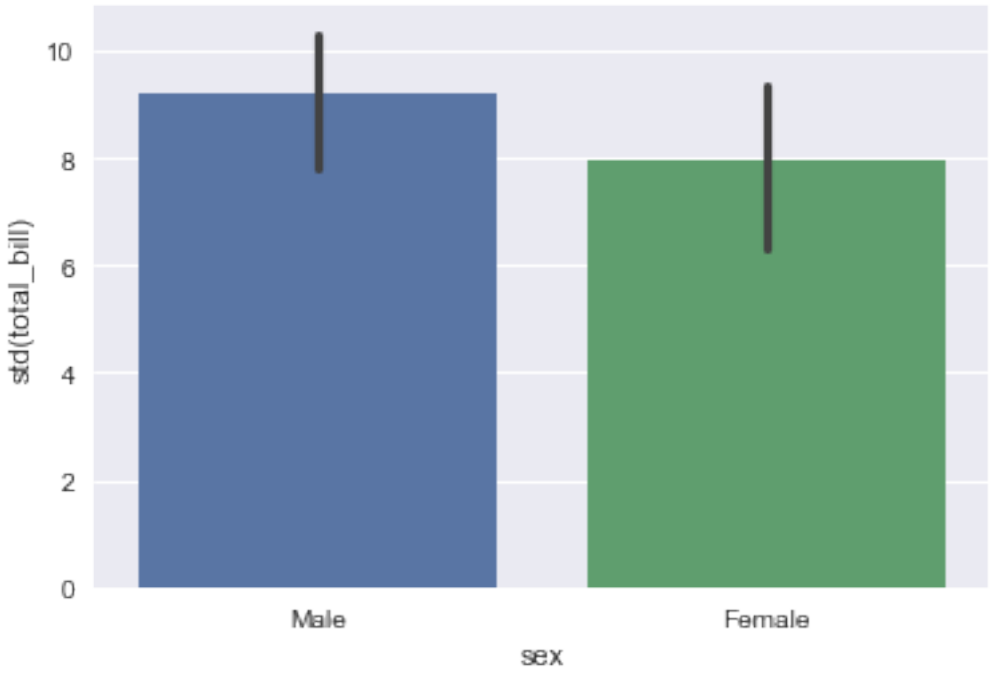


[19]:
```
import numpy as np
```

Você pode alterar o objeto estimador para sua própria função, que converte um vetor em um escalar:

[ ]:
```
sns.barplot(x='sex',y='total_bill',data=tips,estimator=np.std)
```

[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x16ec89fbcf8>
```

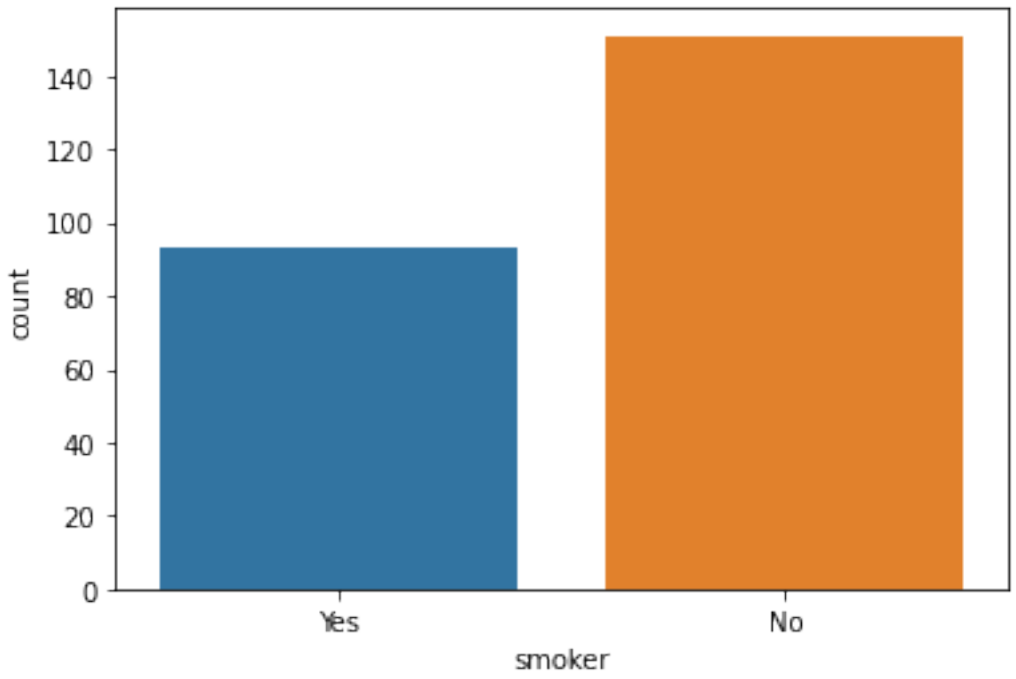


3.1.1 countplot

Isto é essencialmente o mesmo que o gráfico de barras, exceto que o estimador está explicitamente contando o número de ocorrências. É por isso que apenas passamos o valor x:

[20]:
```
sns.countplot(x='smoker',data=tips)
```

[20]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8cdc54d950>
```

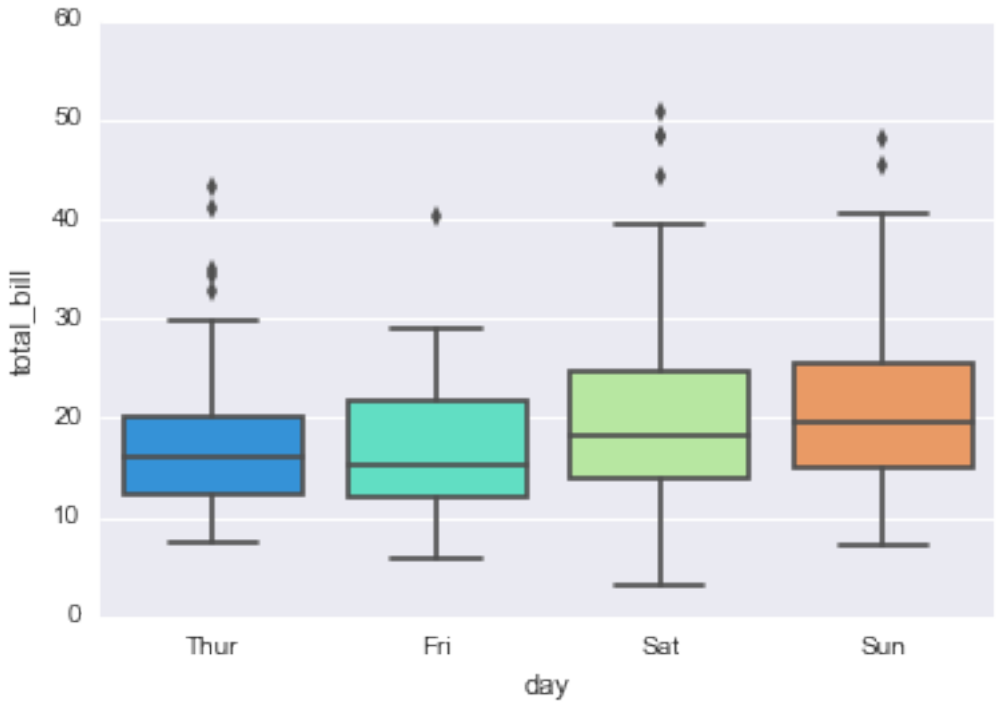


3.2 boxplot and violinplot

Boxplots e violinplots são usados para mostrar a distribuição de dados categóricos. Um boxplot (ou gráfico de caixa e espessura) mostra a distribuição de dados quantitativos de uma maneira que facilita comparações entre variáveis ou entre os níveis de uma variável categórica. A caixa mostra os quartis do conjunto de dados, enquanto as barras se estendem para mostrar o resto da distribuição, exceto pelos pontos que são determinados como "outliers".

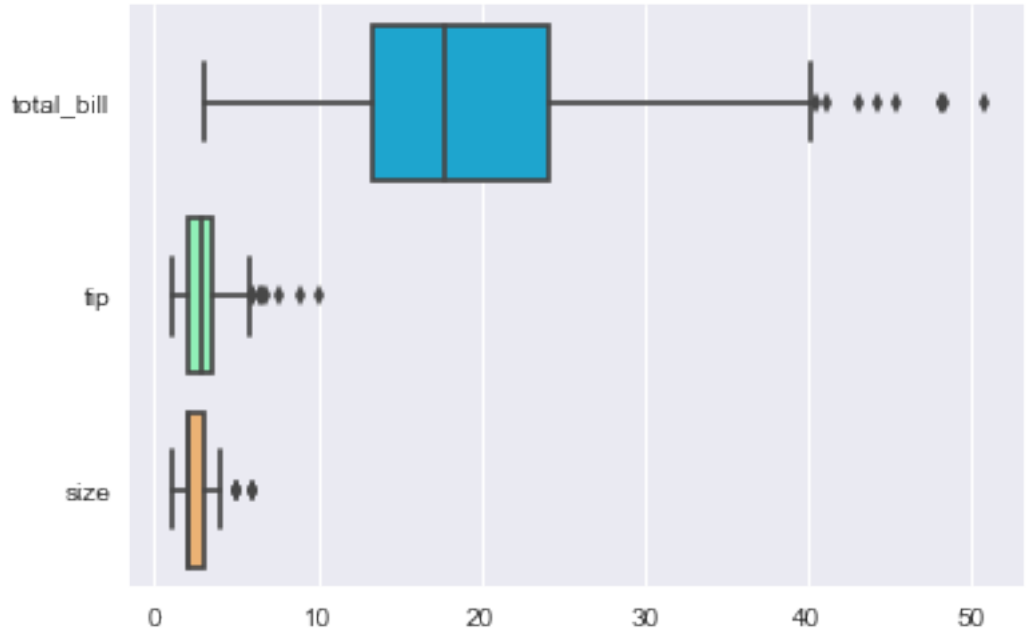
[ ]:
```
sns.boxplot(x="day", y="total_bill", data=tips,palette='rainbow')
```

[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x11db81630>
```



[ ]:
```
# Podemos orientar os dados para aparecerem na horizontal
sns.boxplot(data=tips,palette='rainbow',orient='h')
```

[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x16ec8bb1278>
```



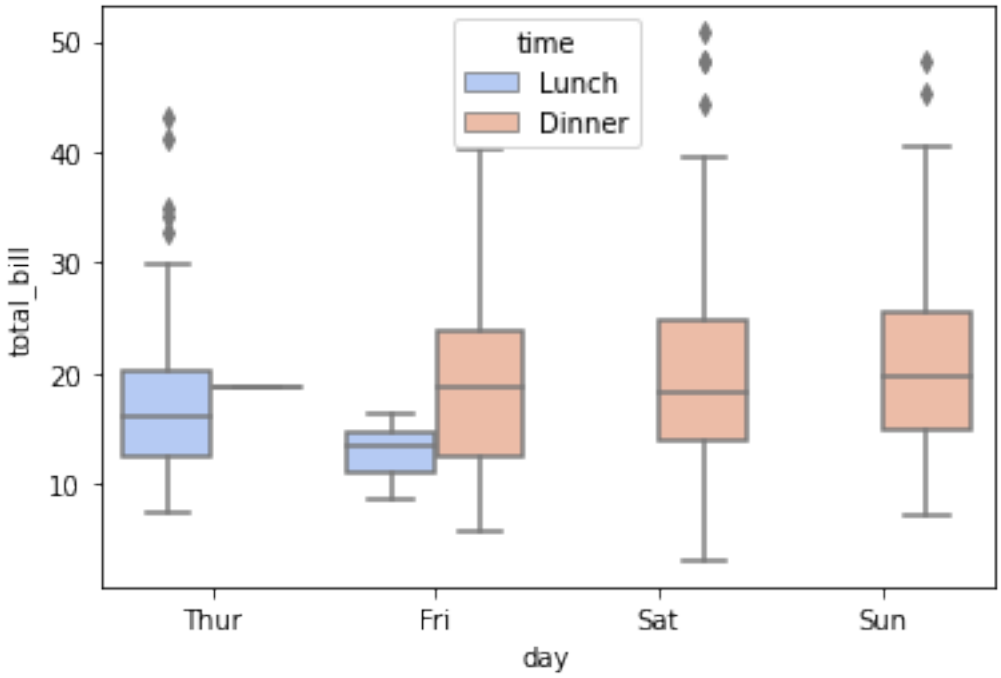
[ ]:
```
tips.head()
```

[ ]:

	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	3.61	Female	No	Sun	Dinner	4

[21]:
```
sns.boxplot(x="day", y="total_bill", hue="time",data=tips, palette="coolwarm")
```

[21]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8cdc37d5d0>
```

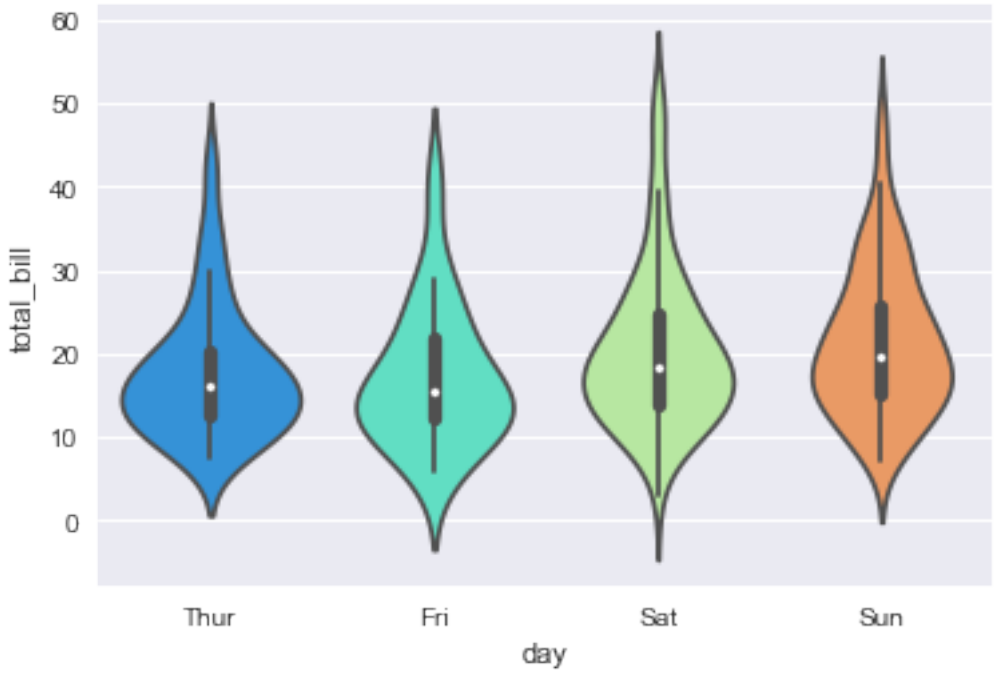


3.2.1 violinplot

Um violinplot desempenha um papel semelhante a um boxplot. Ele mostra a distribuição de dados quantitativos em vários níveis de uma (ou mais) variáveis categóricas, de modo que essas distribuições possam ser comparadas. Ao contrário de um boxplot, no qual todos os componentes do gráfico correspondem a pontos de dados reais, o gráfico de violino possui uma estimativa da densidade do núcleo da distribuição subjacente.

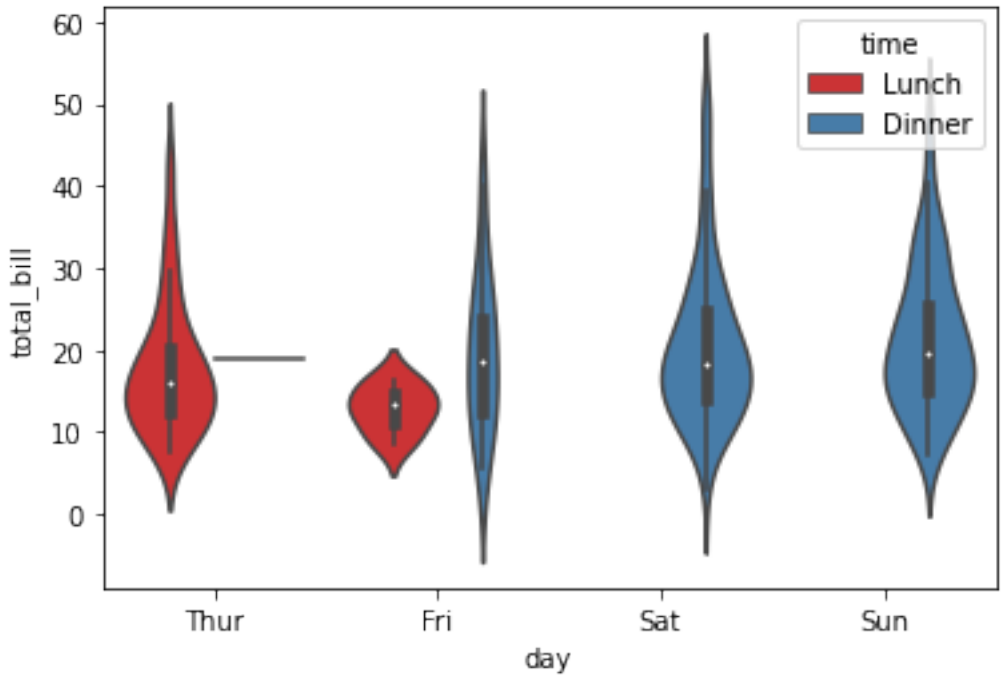
[ ]:
```
sns.violinplot(x="day", y="total_bill", data=tips,palette='rainbow')
```

[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x16ec9eb9f60>



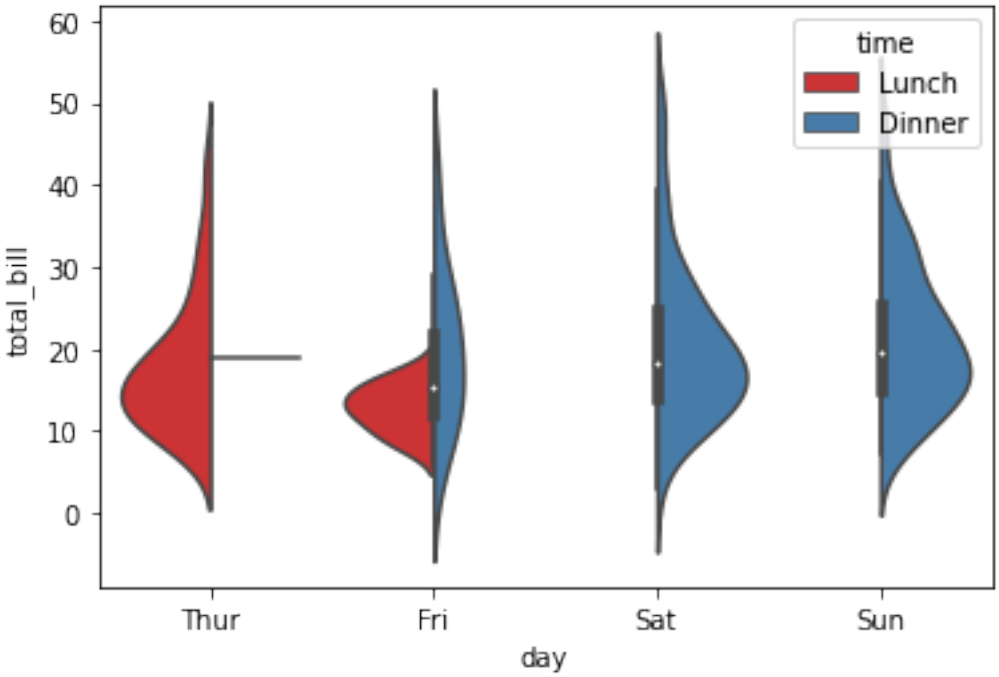
[22]:
```
sns.violinplot(x="day", y="total_bill", data=tips,hue='time',palette='Set1')
```

[22]: <matplotlib.axes._subplots.AxesSubplot at 0x7f8cdc2e77d0>



[23]:
```
sns.violinplot(x="day", y="total_bill",␣
↪data=tips,hue='time',split=True,palette='Set1')
```

[23]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8cdc1c6a50>
```



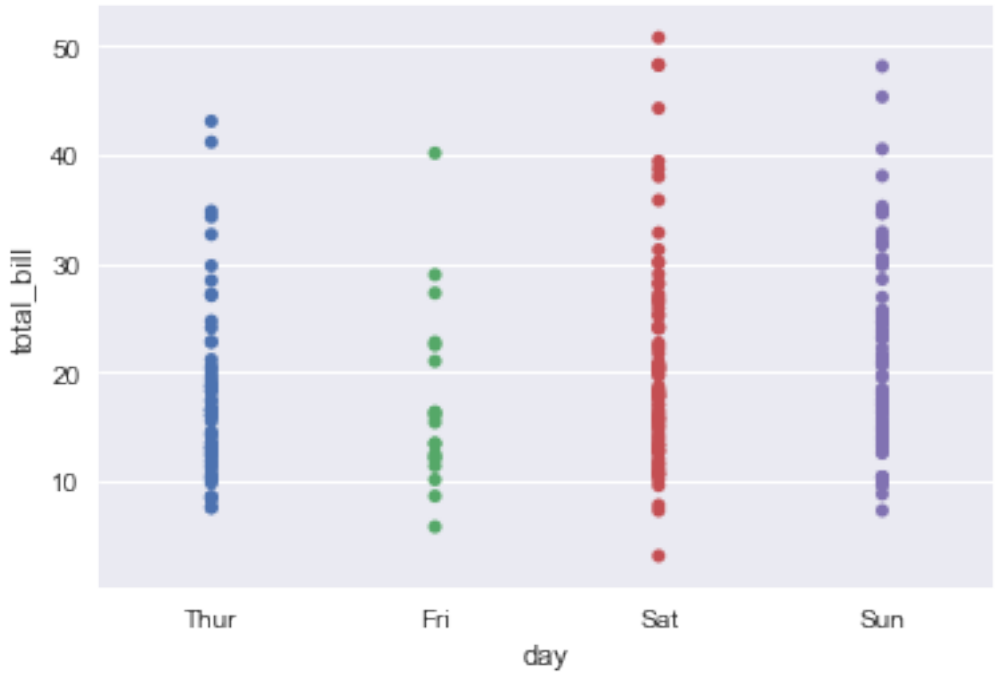
3.3 stripplot e swarmplot

O stripplot irá desenhar um scatterplot onde uma variável é categórica. Um stripplot pode ser desenhado por conta própria, mas também é um bom complemento para uma boxplot ou violinplot nos casos em que você deseja mostrar todas as observações juntamente com alguma representação da distribuição subjacente.

O swarmplot é semelhante ao stripplot (), mas os pontos são ajustados (somente ao longo do eixo categórico) para que eles não se sobreponham. Isso dá uma melhor representação da distribuição de valores, embora não se ajude também a um grande número de observações (tanto em termos de capacidade de mostrar todos os pontos quanto em termos da computação necessária para organizá-los).

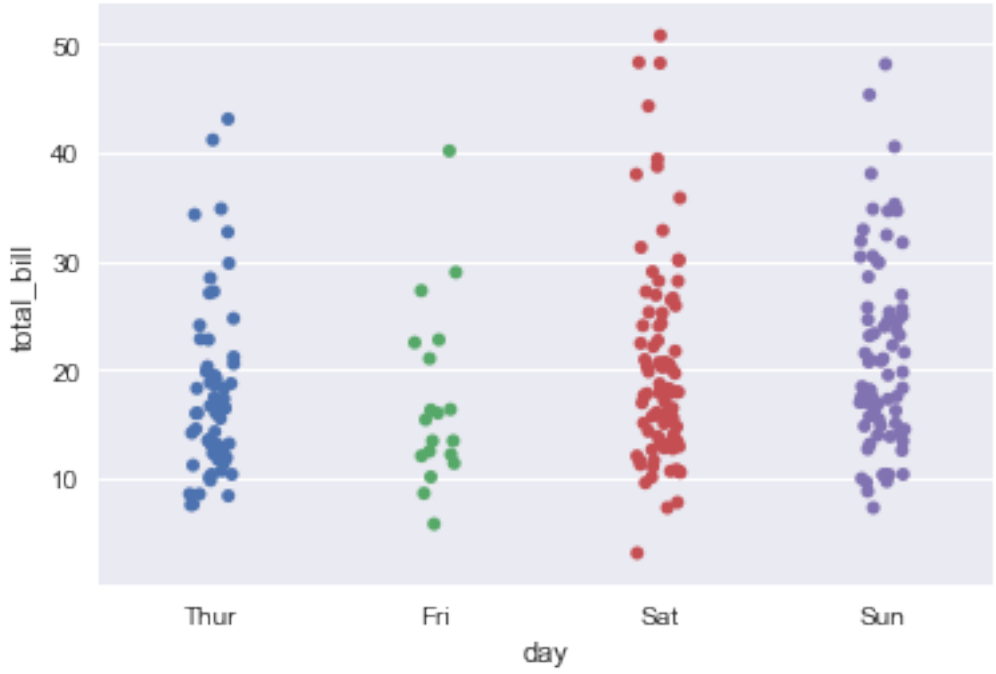
[ ]:
```
sns.stripplot(x="day", y="total_bill", data=tips)
```

[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x16eca0f50f0>
```



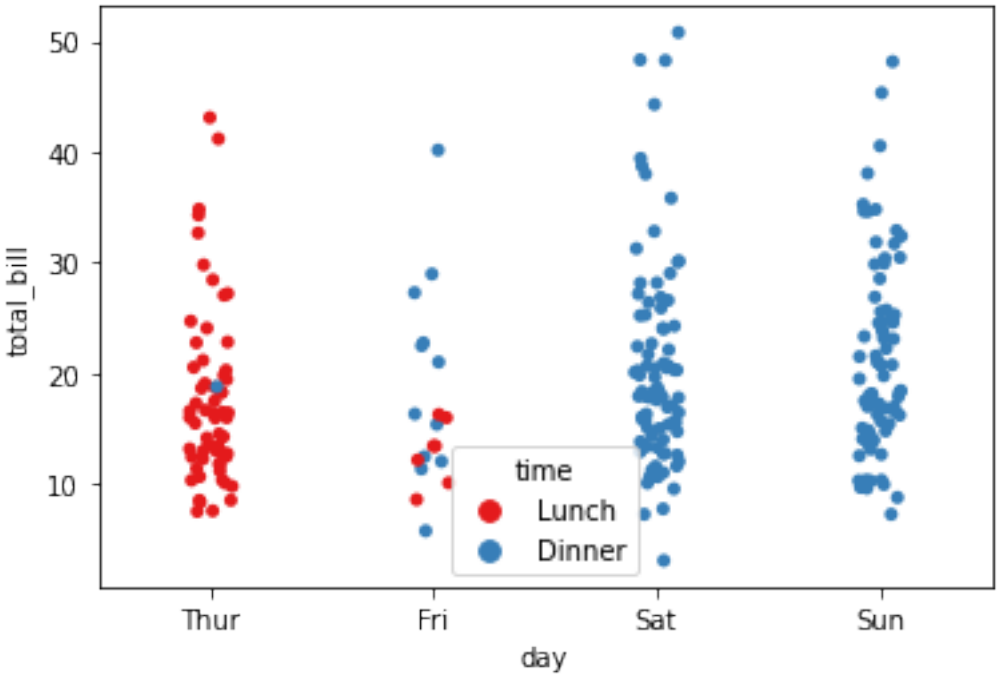
[ ]:
```
sns.stripplot(x="day", y="total_bill", data=tips,jitter=True)
```

[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x16eca1db668>
```



[ ]:
```
sns.stripplot(x="day", y="total_bill",␣
↪data=tips,jitter=True,hue='time',palette='Set1')
```

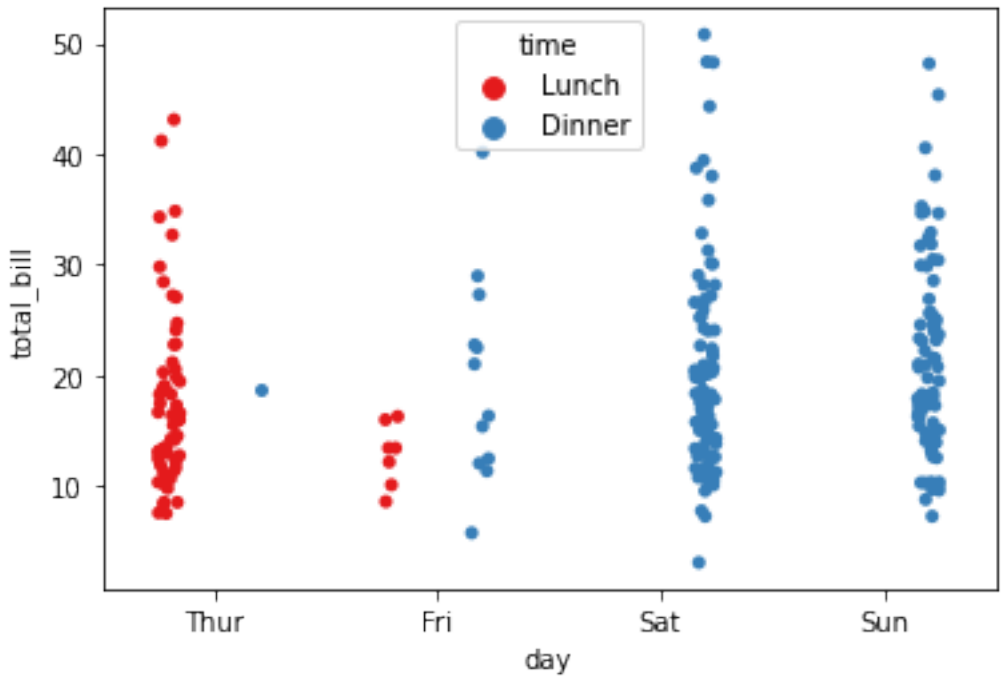
[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x7f72259de9d0>



[ ]:
```
sns.stripplot(x="day", y="total_bill",␣
↪data=tips,jitter=True,hue='time',palette='Set1',split=True)
```

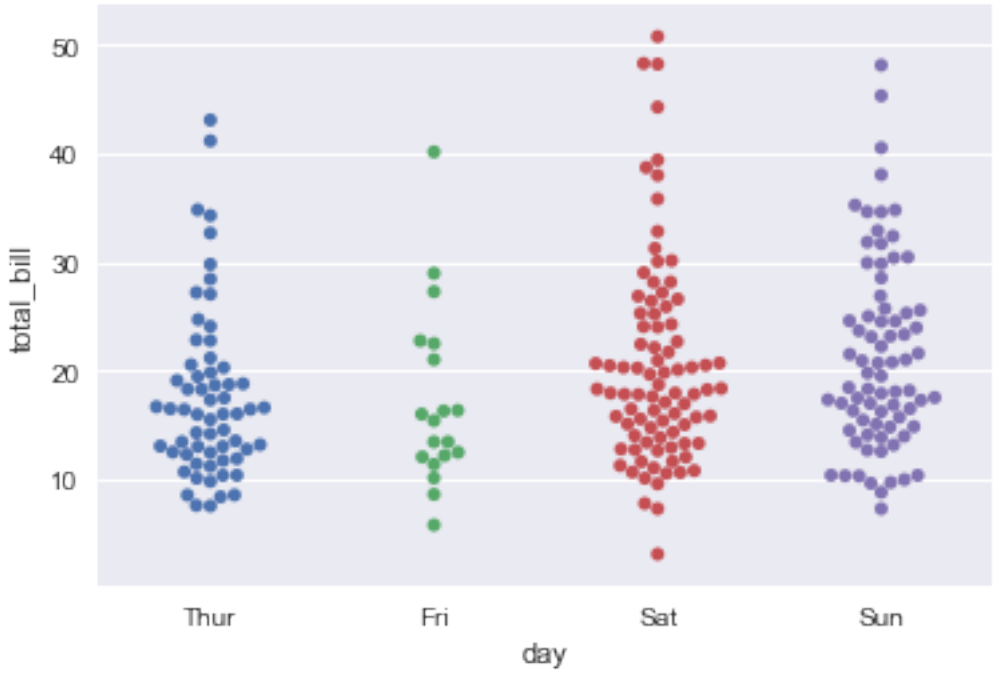
```
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:2802: UserWarning:
The `split` parameter has been renamed to `dodge`.
  warnings.warn(msg, UserWarning)
```

[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x7f72258ce790>



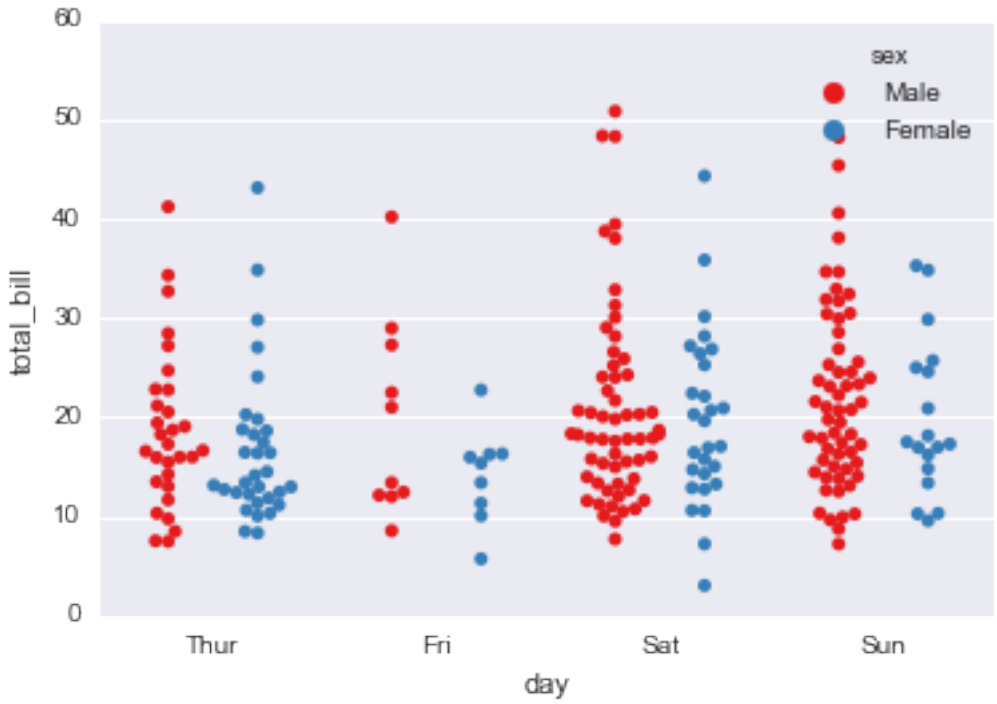
[ ]:
```
sns.swarmplot(x="day", y="total_bill", data=tips)
```

[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x16eca38beb8>
```



[ ]:
```
sns.swarmplot(x="day", y="total_bill",hue='sex',data=tips, palette="Set1",␣
↪split=True)
```

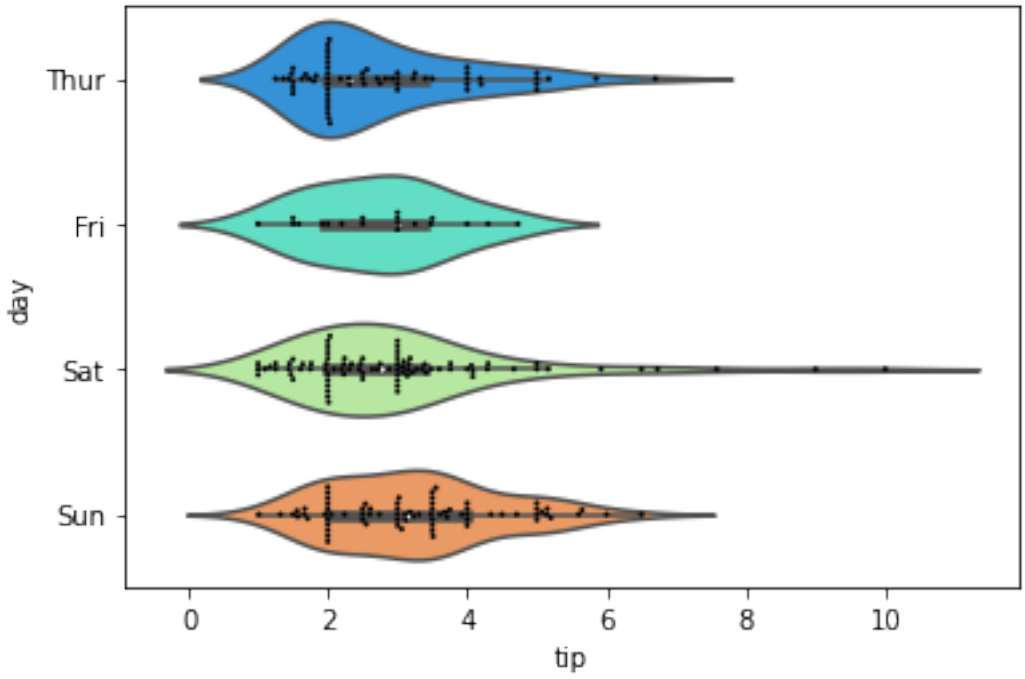
[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1211b6da0>
```



3.3.1 Combinando plots categóricos

[ ]:
```
sns.violinplot(x="tip", y="day", data=tips,palette='rainbow')
sns.swarmplot(x="tip", y="day", data=tips,color='black',size=2)
```

[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7225725310>
```

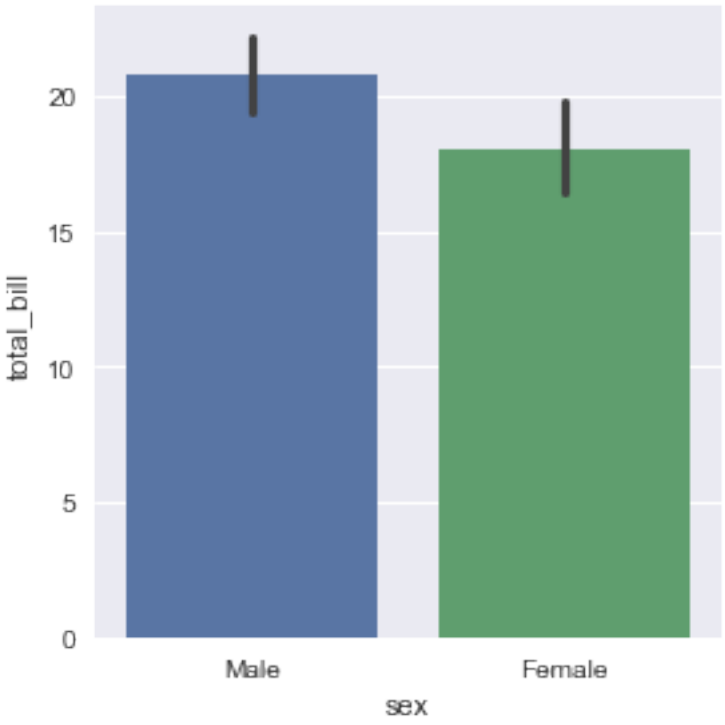


3.4 factorplot

O factorplot é a forma mais geral de um plot categórico. Pode aceitar um parâmetro ** kind ** para ajustar o tipo de plotagem:

[ ]:
```
sns.factorplot(x='sex',y='total_bill',data=tips,kind='bar')
sns.factorplot()
```

[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x16eca52e240>
```



4 Plots matriciais

Os gráficos matriciais permitem traçar dados como matrizes codificadas por cores e também podem ser usados para indicar clusters dentro dos dados (mais tarde, na seção de Machine Learning, aprenderemos a formatear dados de cluster).

Comecemos por explorar o mapa térmico e o clutermap de Seaborn:

[24]:
```
import seaborn as sns
%matplotlib inline
```

[25]:
```
flights = sns.load_dataset('flights')
```

[26]:
```
tips = sns.load_dataset('tips')
```

[27]:
```
tips.head()
```

[27]:

	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	3.61	Female	No	Sun	Dinner	4

[28]:
```
flights.head()
```

[28]:

	year	month	passengers
0	1949	Jan	112
1	1949	Feb	118
2	1949	Mar	132
3	1949	Apr	129
4	1949	May	121

4.1 Heatmap

Para que um mapa de calor funcione corretamente, seus dados já devem estar em uma forma de matriz e a função sns.heatmap basicamente apenas põe cor pra você. Por exemplo:

[ ]:
```
tips.head()
```

[ ]:

	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	3.61	Female	No	Sun	Dinner	4

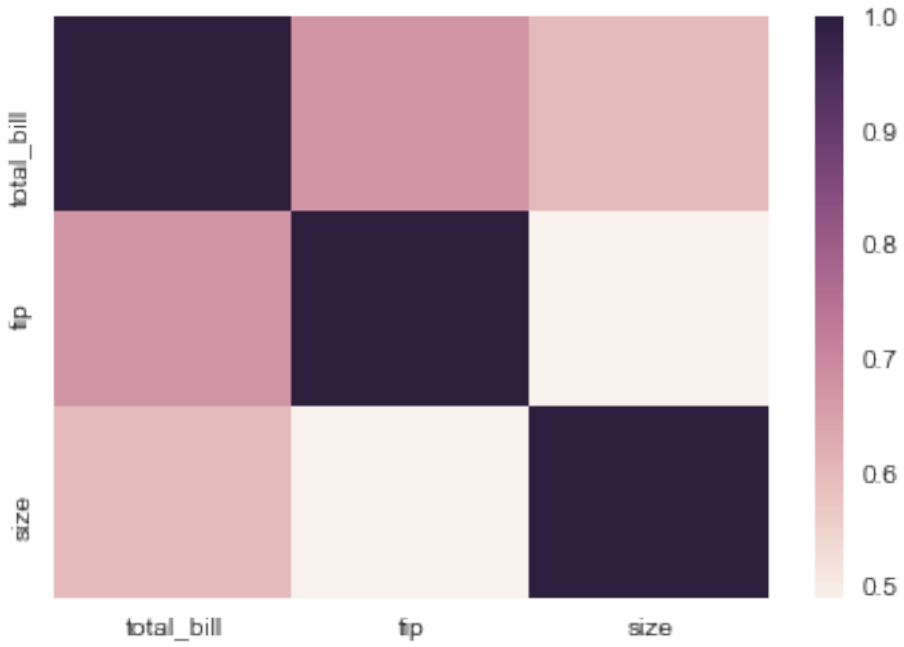
[29]:
```
# Correlograma
tips.corr()
```

[29]:

	total_bill	tip	size
total_bill	1.000000	0.675734	0.598315
tip	0.675734	1.000000	0.489299
size	0.598315	0.489299	1.000000

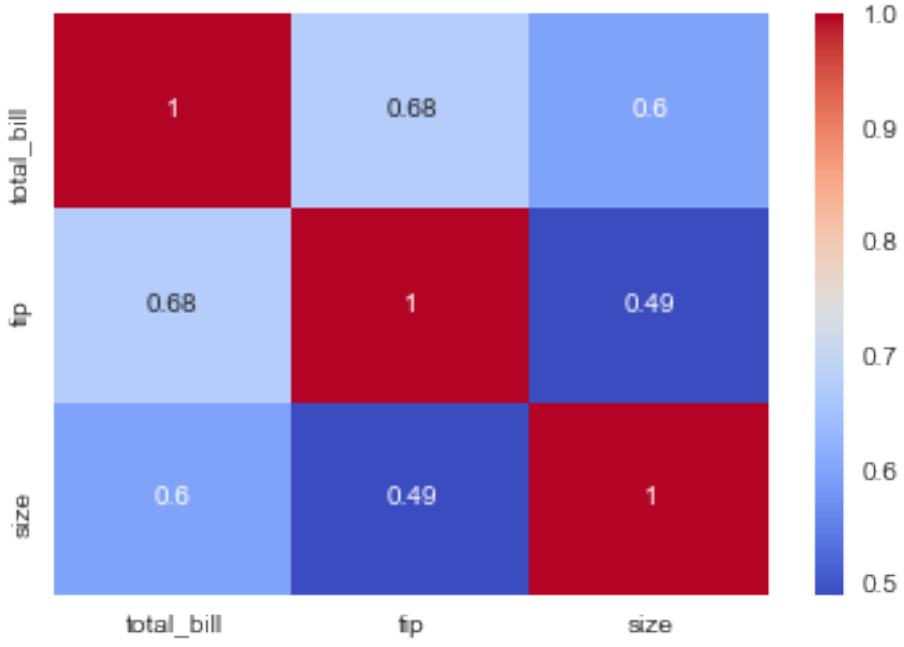
[ ]:
```
sns.heatmap(tips.corr())
```

[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1ed137966a0>
```



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

[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1ed135556d8>
```



Ou para os dados dos vôos:

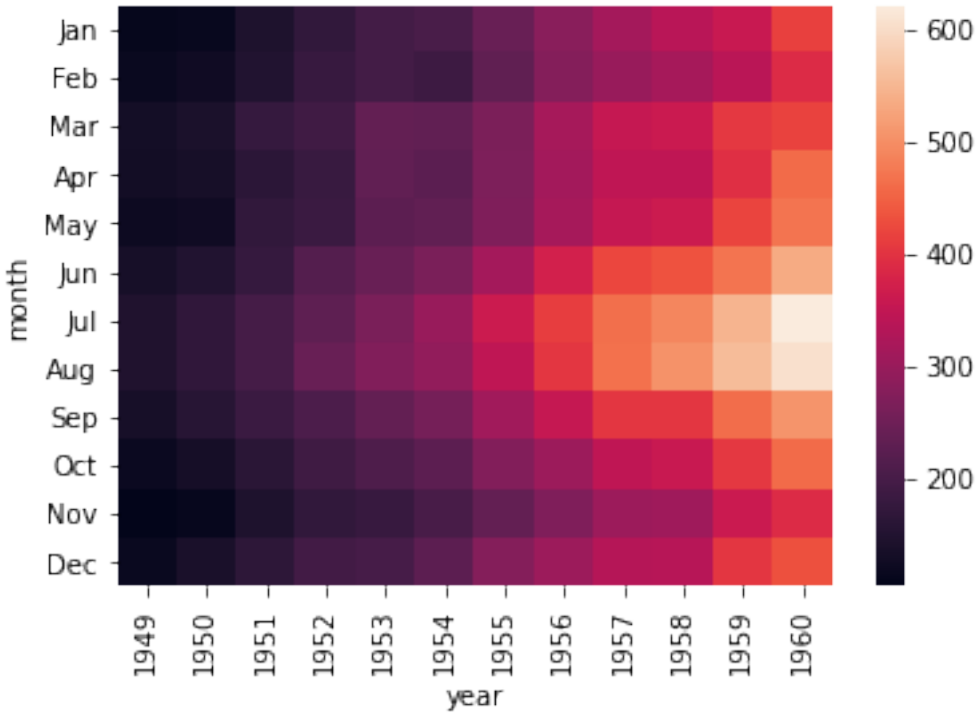
[30]:
```
flights.pivot_table(values='passengers',index='month',columns='year')
```

[30]:

month	1949	1950	1951	1952	1953	1954	1955	1956	1957	1958	1959	1960
Jan	112	115	145	171	196	204	242	284	315	340	360	417
Feb	118	126	150	180	196	188	233	277	301	318	342	391
Mar	132	141	178	193	236	235	267	317	356	362	406	419
Apr	129	135	163	181	235	227	269	313	348	348	396	461
May	121	125	172	183	229	234	270	318	355	363	420	472
Jun	135	149	178	218	243	264	315	374	422	435	472	535
Jul	148	170	199	230	264	302	364	413	465	491	548	622
Aug	148	170	199	242	272	293	347	405	467	505	559	606
Sep	136	158	184	209	237	259	312	355	404	404	463	508
Oct	119	133	162	191	211	229	274	306	347	359	407	461
Nov	104	114	146	172	180	203	237	271	305	310	362	390
Dec	118	140	166	194	201	229	278	306	336	337	405	432

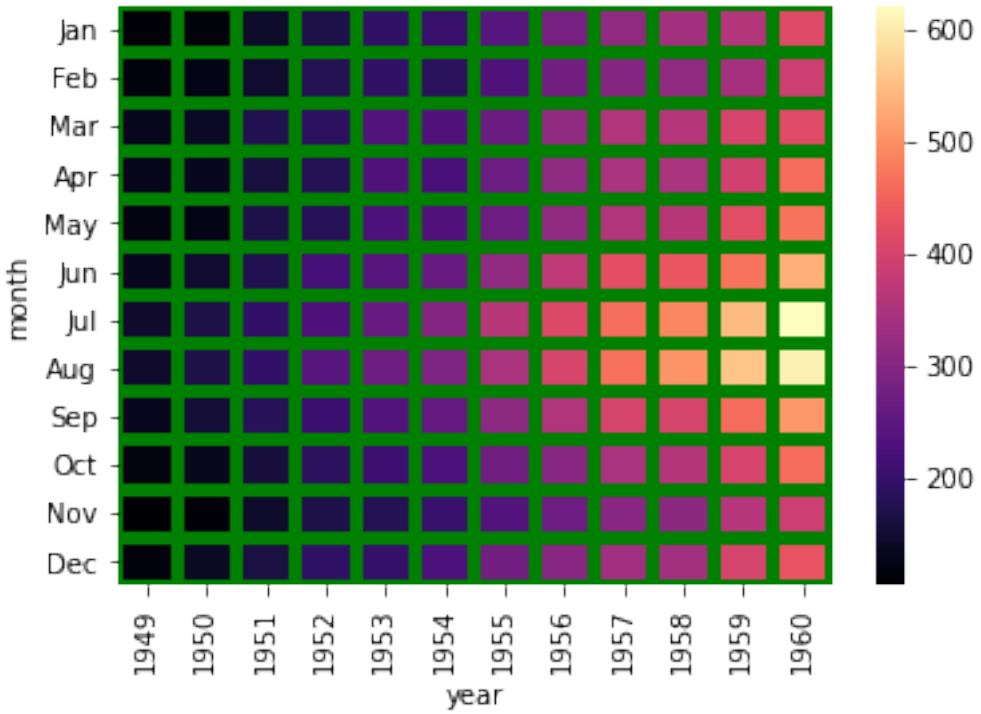
[ ]:
```
pvflights = flights.
 ↪pivot_table(values='passengers',index='month',columns='year')
sns.heatmap(pvflights)
```

[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7225632b90>
```



[ ]:
```
sns.heatmap(pvflights,cmap='magma',linecolor='green',linewidths=4)
```

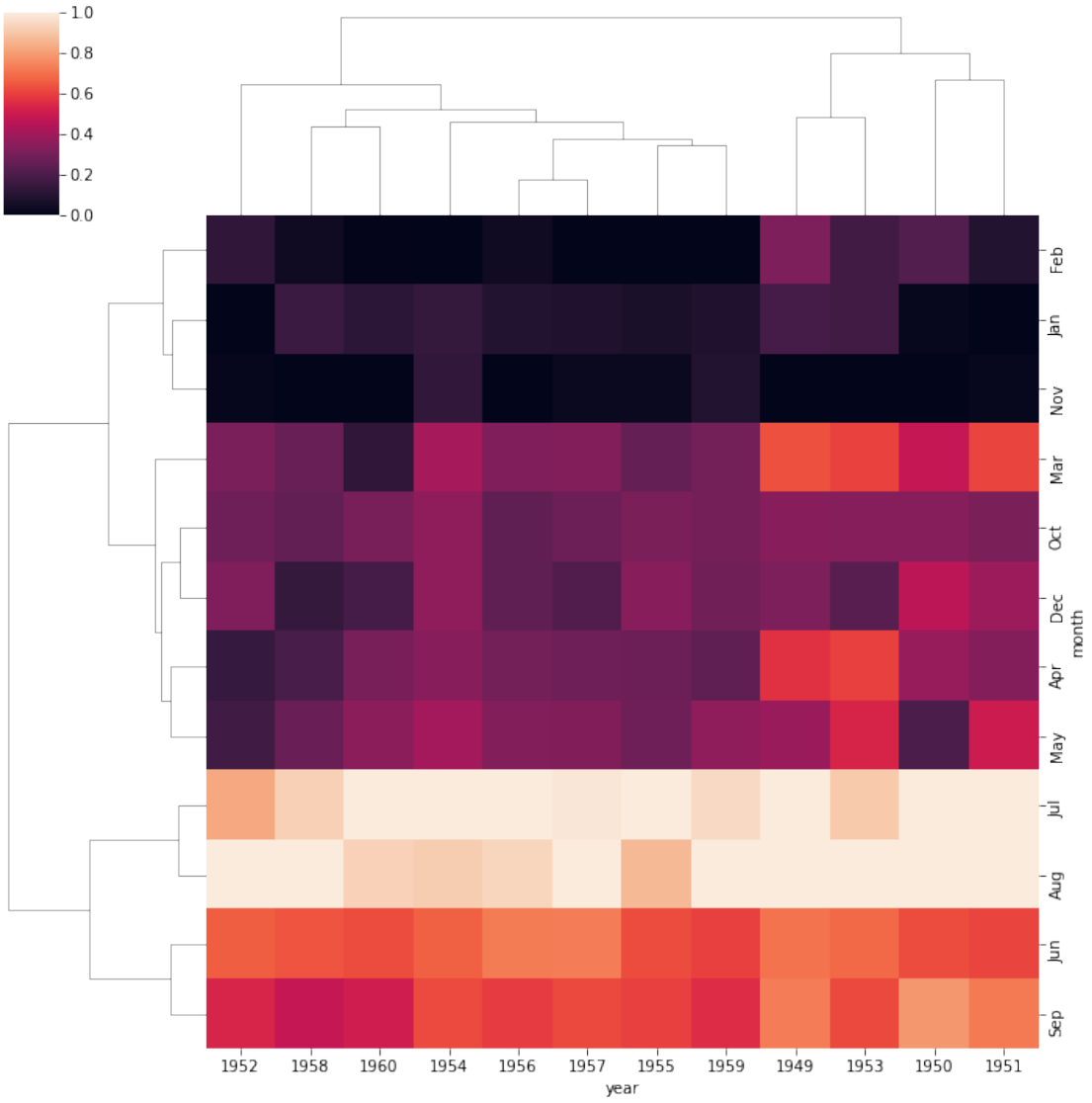
[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7225519e90>
```



4.2 clustermap

O clustermap usa agrupamento hierárquico para produzir uma versão em cluster do heatmap. Por exemplo:

[ ]:
```
figura1 = sns.clustermap(pvflights, standard_scale=1)
```



[ ]:
```
figura1.savefig("figura.png")
```

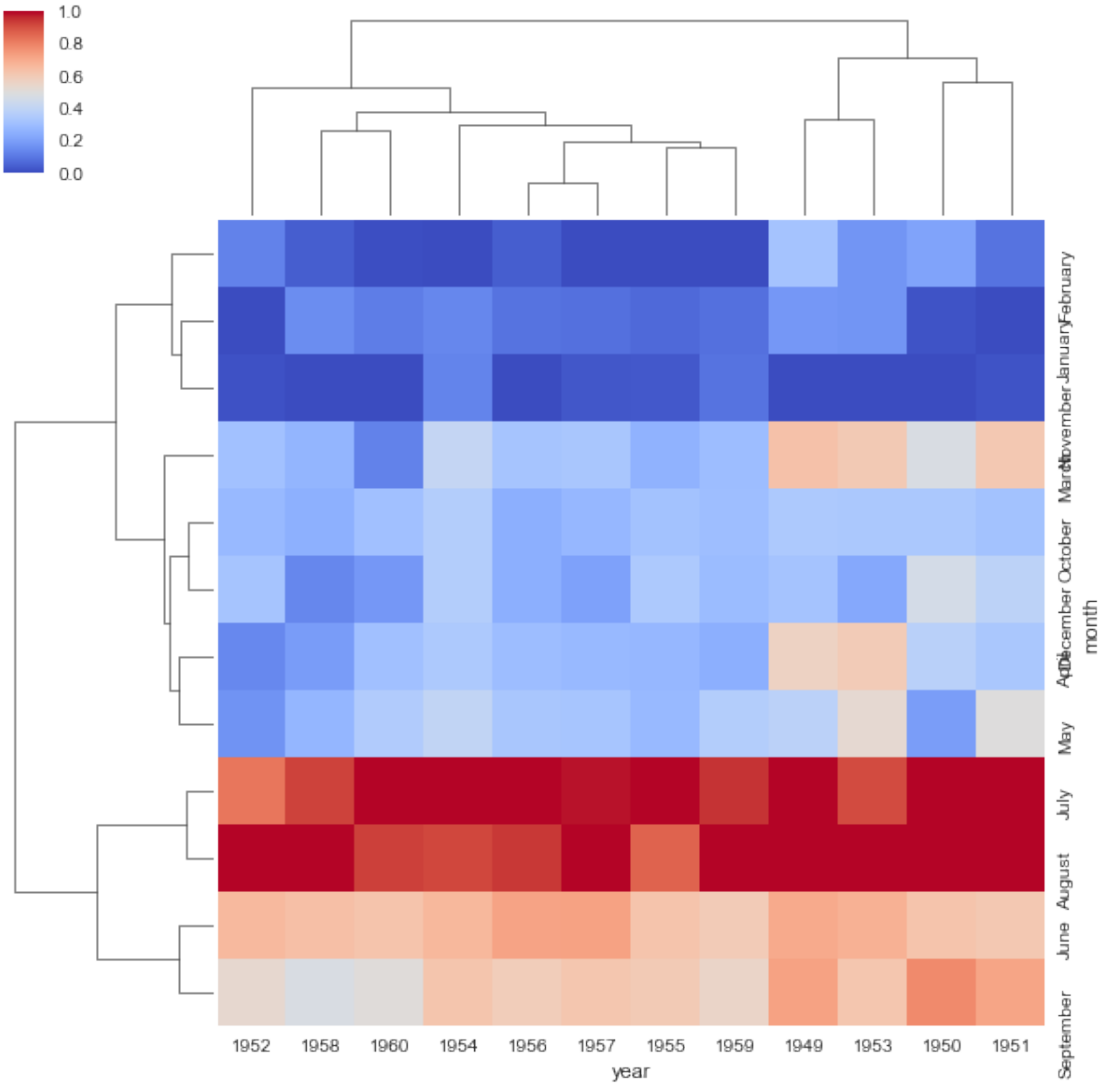
Observe agora como os anos e os meses não estão mais em ordem, em vez disso, eles são agrupados por similaridade em valor (contagem de passageiros). Isso significa que podemos começar a inferir coisas desse plot, como agosto e julho sendo semelhantes (faz sentido, uma vez que são ambos os meses de viagem de verão no hemisfério norte)

[ ]:
```
# Mais opções para obter a informação um pouco mais clara, como a normalização
sns.clustermap(pvflights,cmap='coolwarm',standard_scale=1)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook.py:136:
MatplotlibDeprecationWarning: The axisbg attribute was deprecated in version
2.0. Use facecolor instead.
```

```
    warnings.warn(message, mplDeprecation, stacklevel=1)
```

```
[ ]: <seaborn.matrix.ClusterGrid at 0x1ed151db7f0>
```



5 Plots de regressões

O Seaborn possui muitas ferramentas integradas para plots de regressão, no entanto, não discutiremos a regressão até a seção de Machine Learning do curso, de modo que apenas cobriremos a função ** lmplot () ** por enquanto.

** lmplot ** permite que você exiba modelos lineares, mas também permite que você divida esses gráficos com base em recursos, além de colorir a matiz de cores com base nos recursos.

Vamos explorar como isso funciona:

[ ]:
```
import seaborn as sns
%matplotlib inline
```

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

[ ]:
```
tips.head()
```

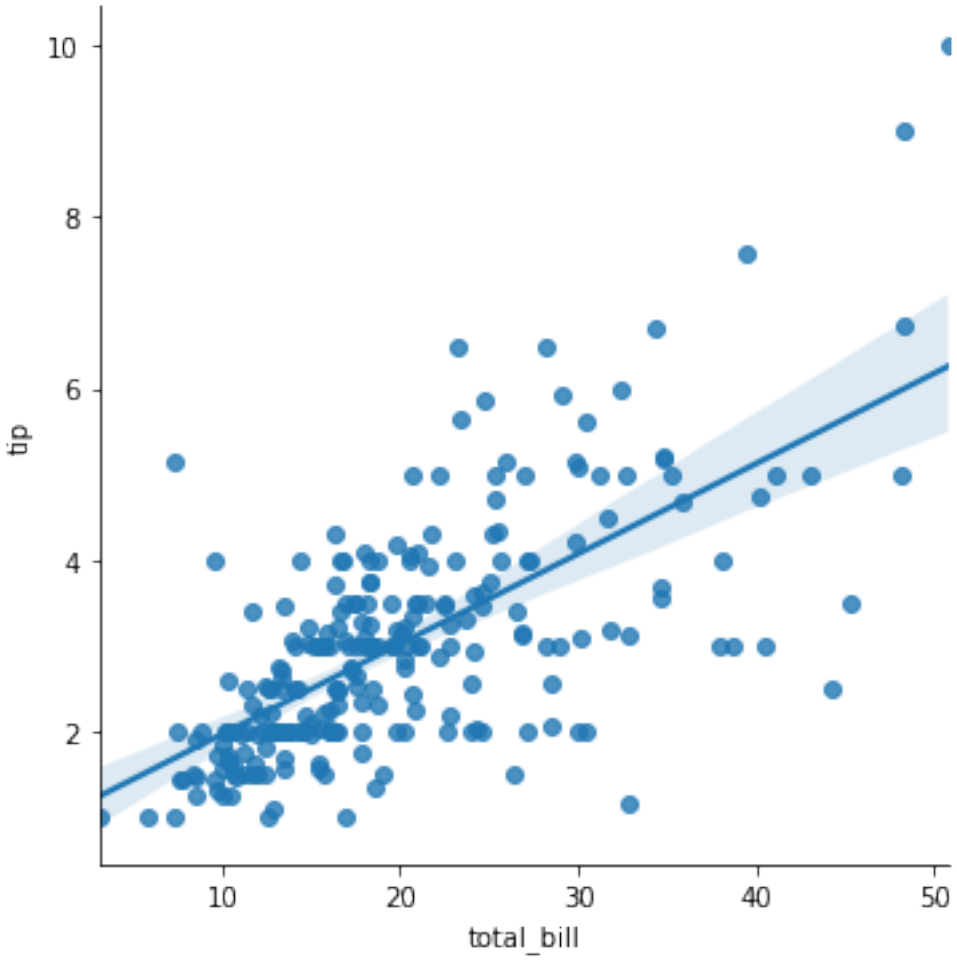
[ ]:

	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	3.61	Female	No	Sun	Dinner	4

5.1 lmplot()

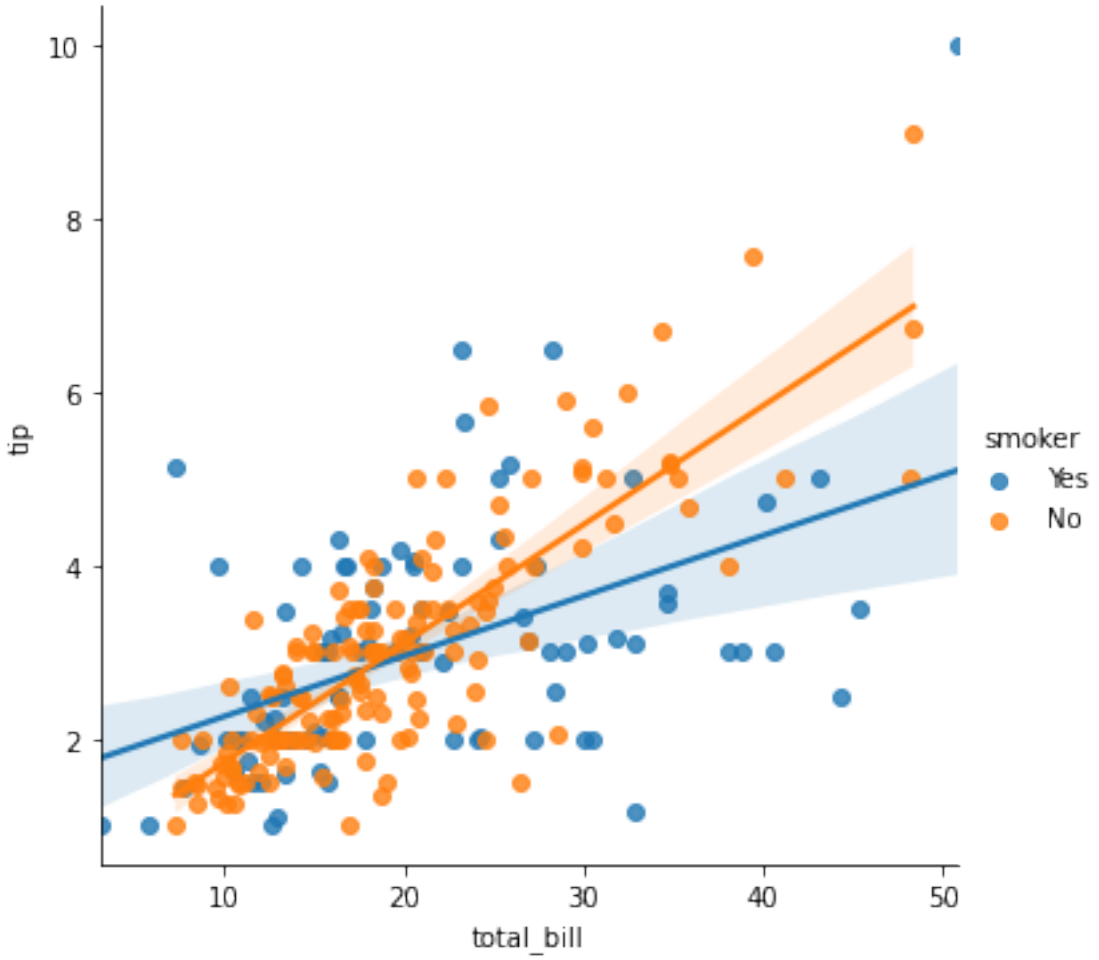
[ ]:
```
sns.lmplot(x='total_bill',y='tip',data=tips)
```

[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x7f7224fd8510>
```



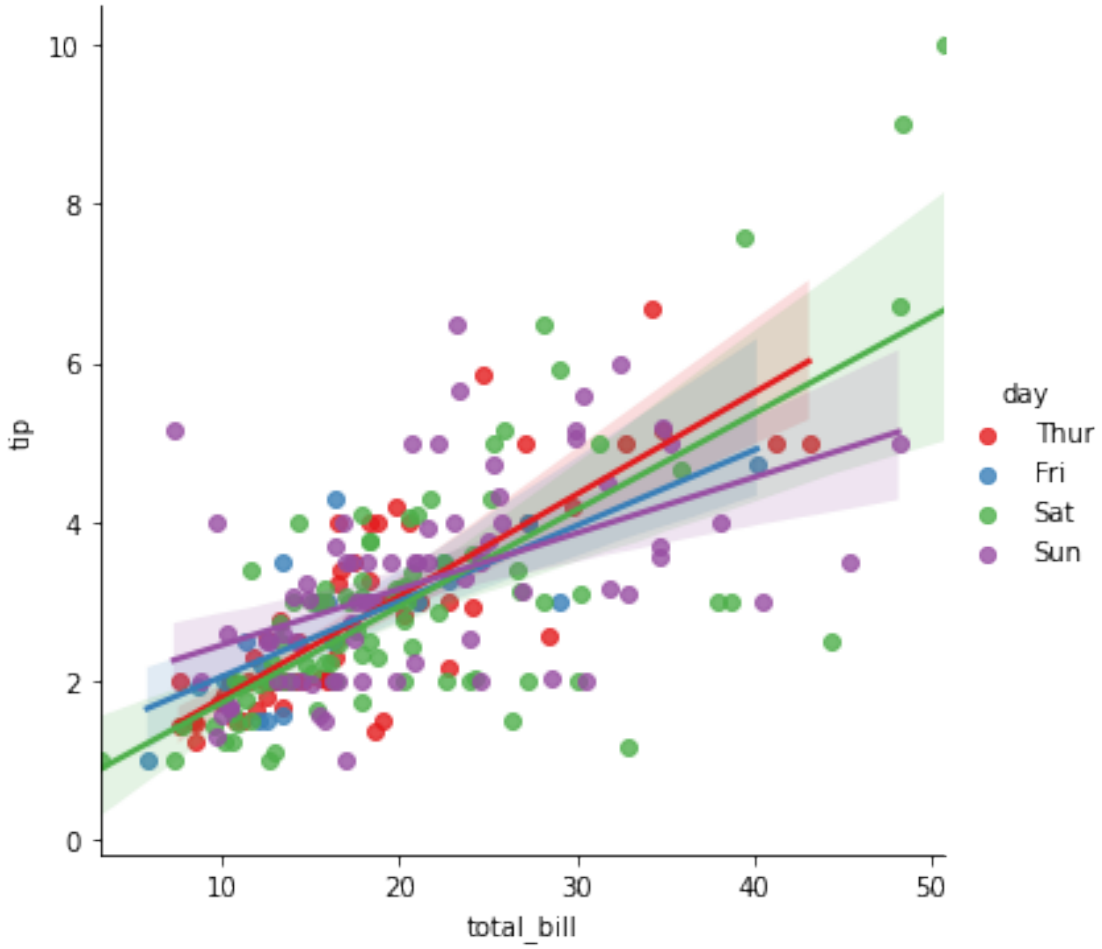
[ ]:
```
sns.lmplot(x='total_bill',y='tip',data=tips,hue='smoker')
```

[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x7f7224f951d0>
```



[33]:
```
sns.lmplot(x='total_bill',y='tip',data=tips,hue='day',palette='Set1')
```

[33]:
```
<seaborn.axisgrid.FacetGrid at 0x7f8cdbe770d0>
```

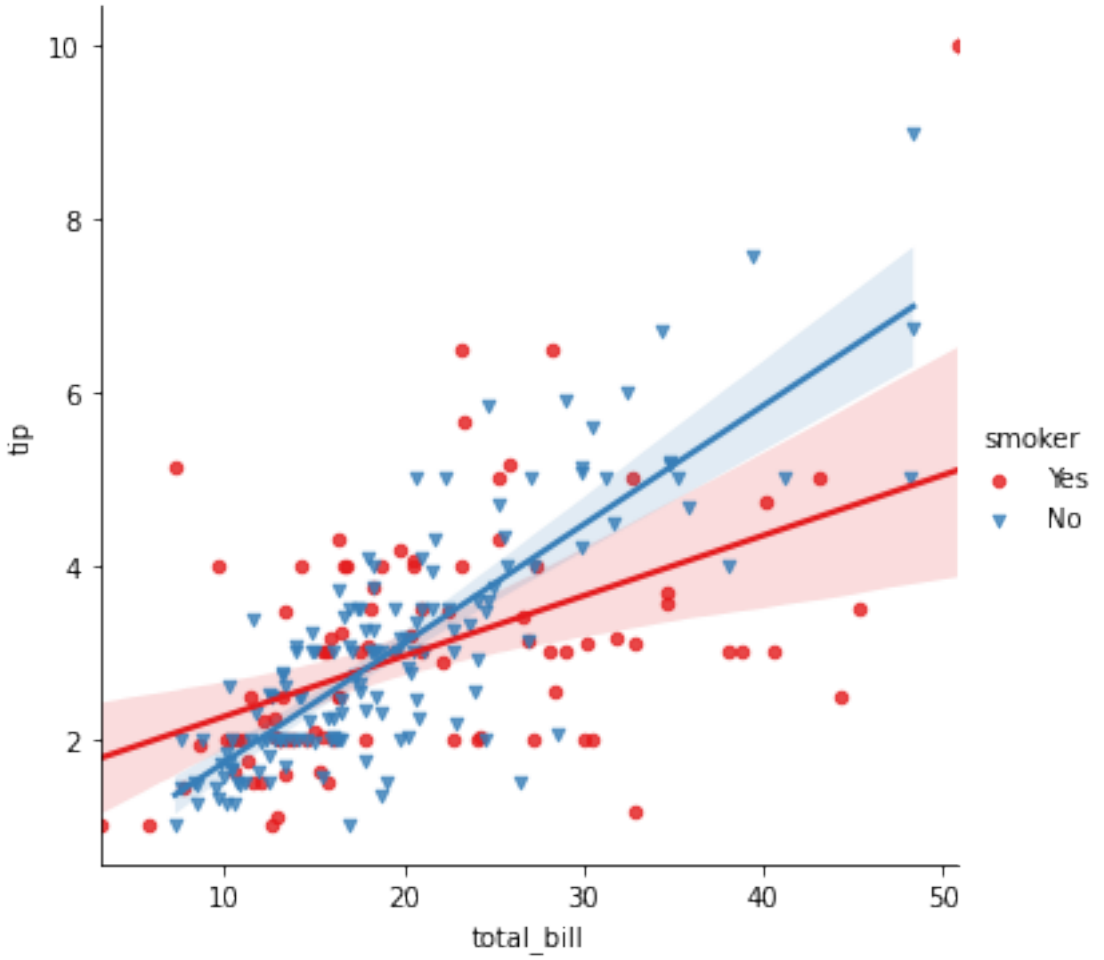


5.1.1 Trabalhando com marcadores

lmplot kwargs são passados através do ** regplot **, que é uma forma mais geral de lmplot (). O regplot possui um parâmetro scatter_kws é passado para plt.scatter. Então, você pode querer definir o parâmetro "s" nesse dicionário, o que corresponde ao tamanho dos marcadores. Em outras palavras, você acaba passando um dicionário com os argumentos base do matplotlib, neste caso, s para o tamanho do gráfico de dispersão. Em geral, você provavelmente não vai se lembrar disso sempre, porém, consulte sempre que achar necessário.

[39]:
```
# http://matplotlib.org/api/markers_api.html
sns.lmplot(x='total_bill',y='tip',data=tips,hue='smoker',palette='Set1',
           markers=['o','v'],scatter_kws={'s':20})
```

[39]:
```
<seaborn.axisgrid.FacetGrid at 0x7f8cdbbd38d0>
```

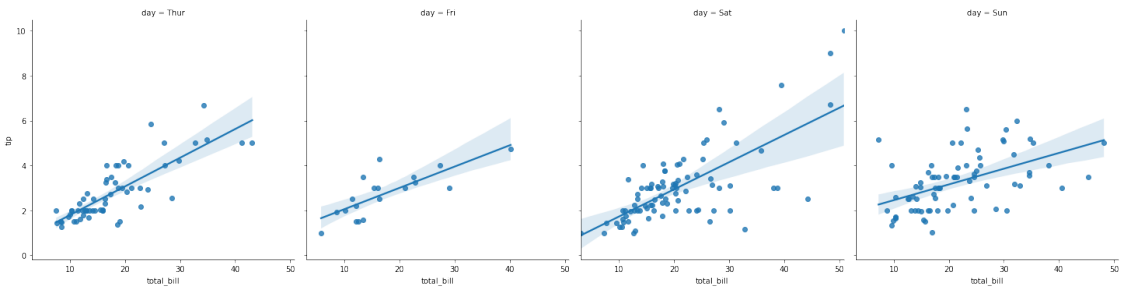


5.2 Usando grades

Podemos adicionar mais separação variável através de colunas e linhas com o uso de uma grade. Basta indicar isso com os argumentos col ou row:

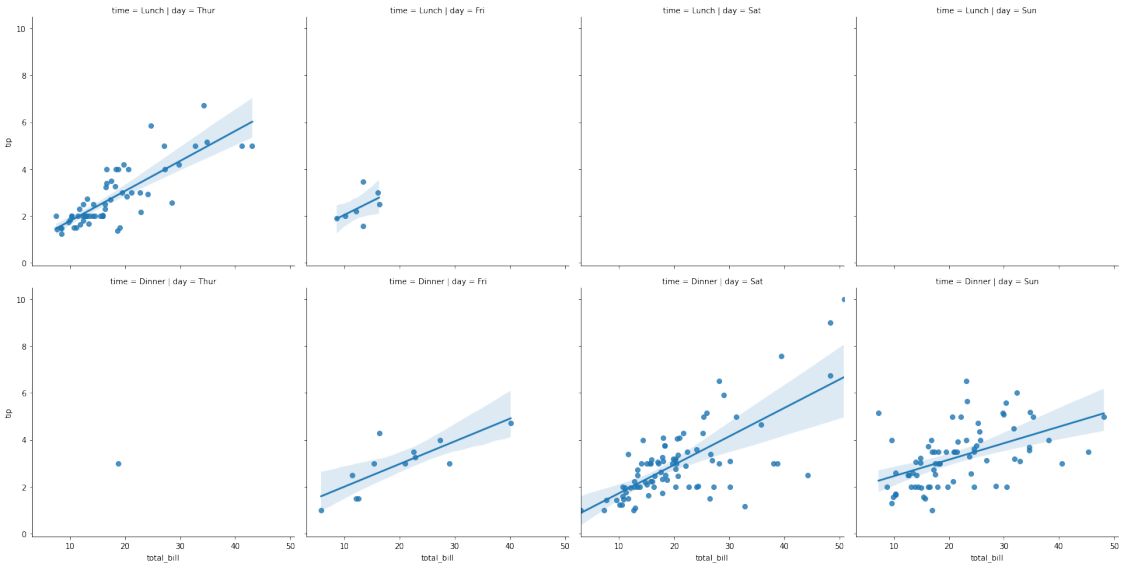
[41]:
```
sns.lmplot(x='total_bill',y='tip',data=tips,col='day')
```

[41]:
```
<seaborn.axisgrid.FacetGrid at 0x7f8cdba0f110>
```



[44]:
```
sns.lmplot(x="total_bill", y="tip", row="time", col="day",data=tips)
```

[44]:
```
<seaborn.axisgrid.FacetGrid at 0x7f8cdb298b90>
```



[ ]:
```
sns.
↪lmplot(x='total_bill',y='tip',data=tips,col='day',hue='sex',palette='coolwarm')
```

[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x1ea6af4a0b8>
```

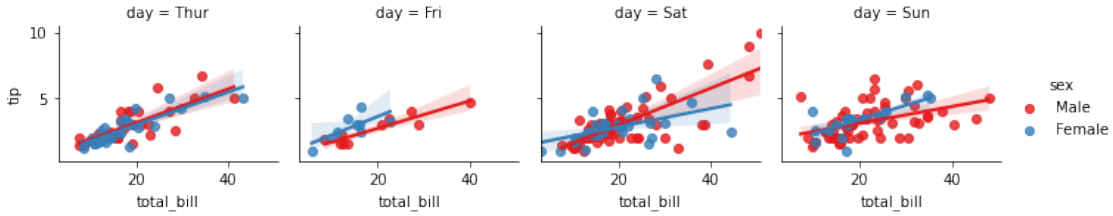
5.3 Aspecto e tamanho

As figuras de Seaborn podem ter seu tamanho e aspect ajustados com os parâmetros ** ** size ** e ** aspecto **:

[51]:
```
sns.lmplot(x='total_bill',y='tip',data=tips,col='day',hue='sex',palette='Set1',
           aspect=1.2,size=2)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/regression.py:580: UserWarning:
The `size` parameter has been renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
```

[51]:
```
<seaborn.axisgrid.FacetGrid at 0x7f8cda57f690>
```



Se desejar obter mais informações sobre como alterar outros aspectos visuais dos seus plots no seaborn, confira o Notebook sobre o assunto!

6 PairGrids

Os PairGrids são tipos gerais de gráficos que permitem mapear tipos de plotagem diferentes para linhas e colunas de um grid, isso ajuda você a criar plots similares separadas por categoias.

[53]:
```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

[55]:
```
iris = sns.load_dataset('iris')
```

[56]:
```
iris.head()
```

[56]:

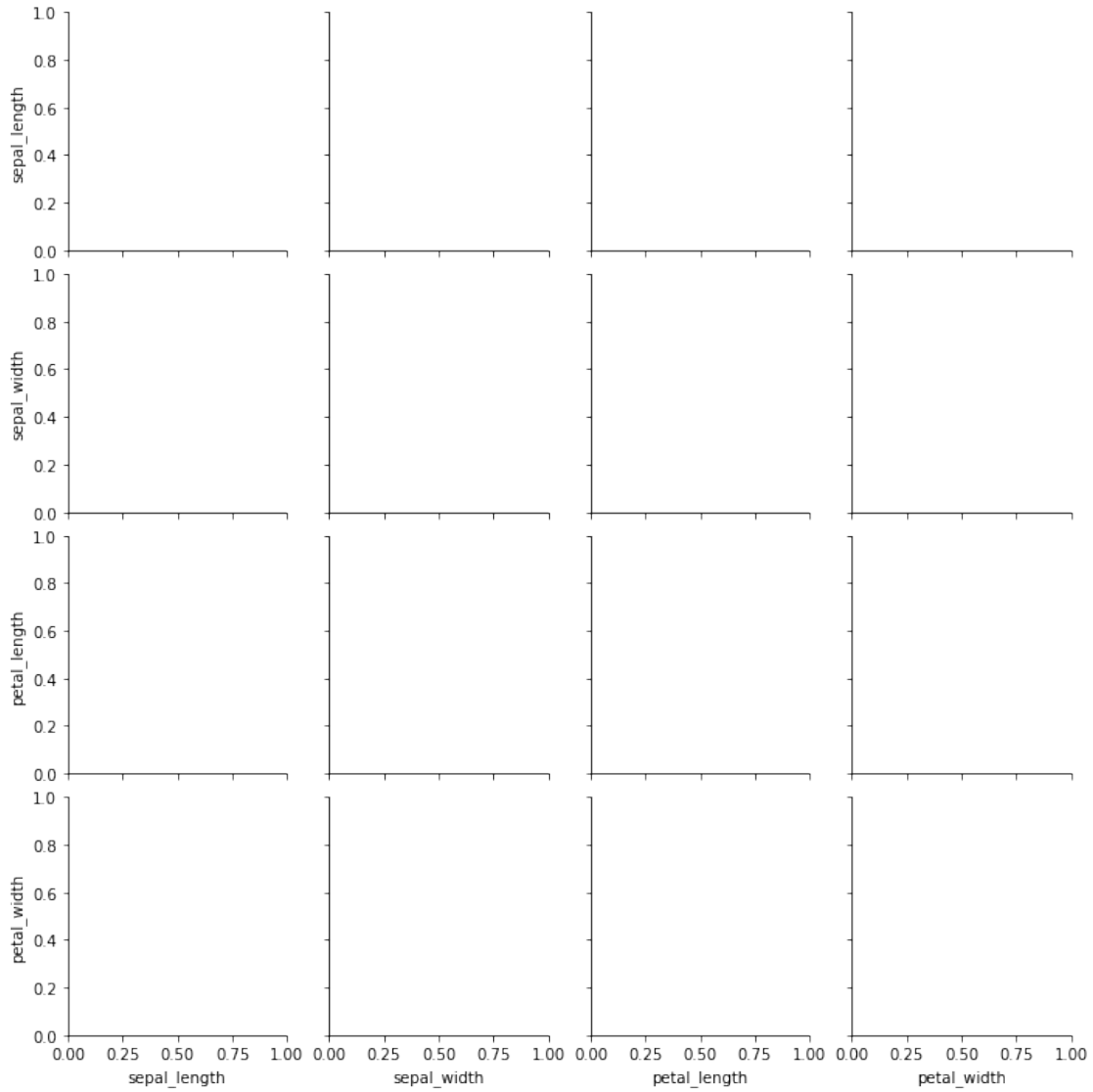
	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

6.1 PairGrid

Pairgrid é um plot de grade para traçar relacionamentos entre pares de um conjunto de dados.

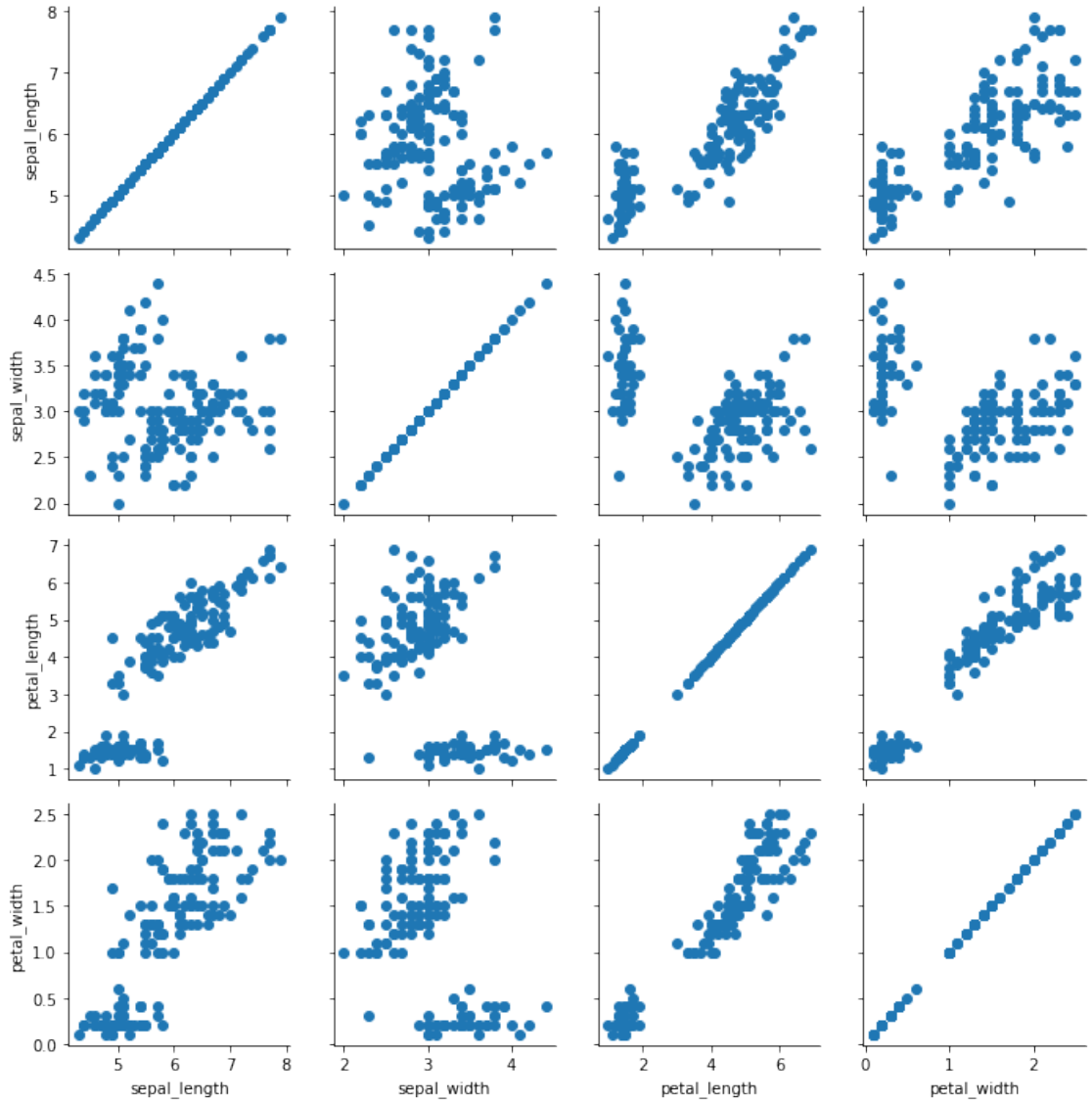
[57]:
```
# Just the Grid
sns.PairGrid(iris)
```

[57]:
```
<seaborn.axisgrid.PairGrid at 0x7f8cda323d90>
```



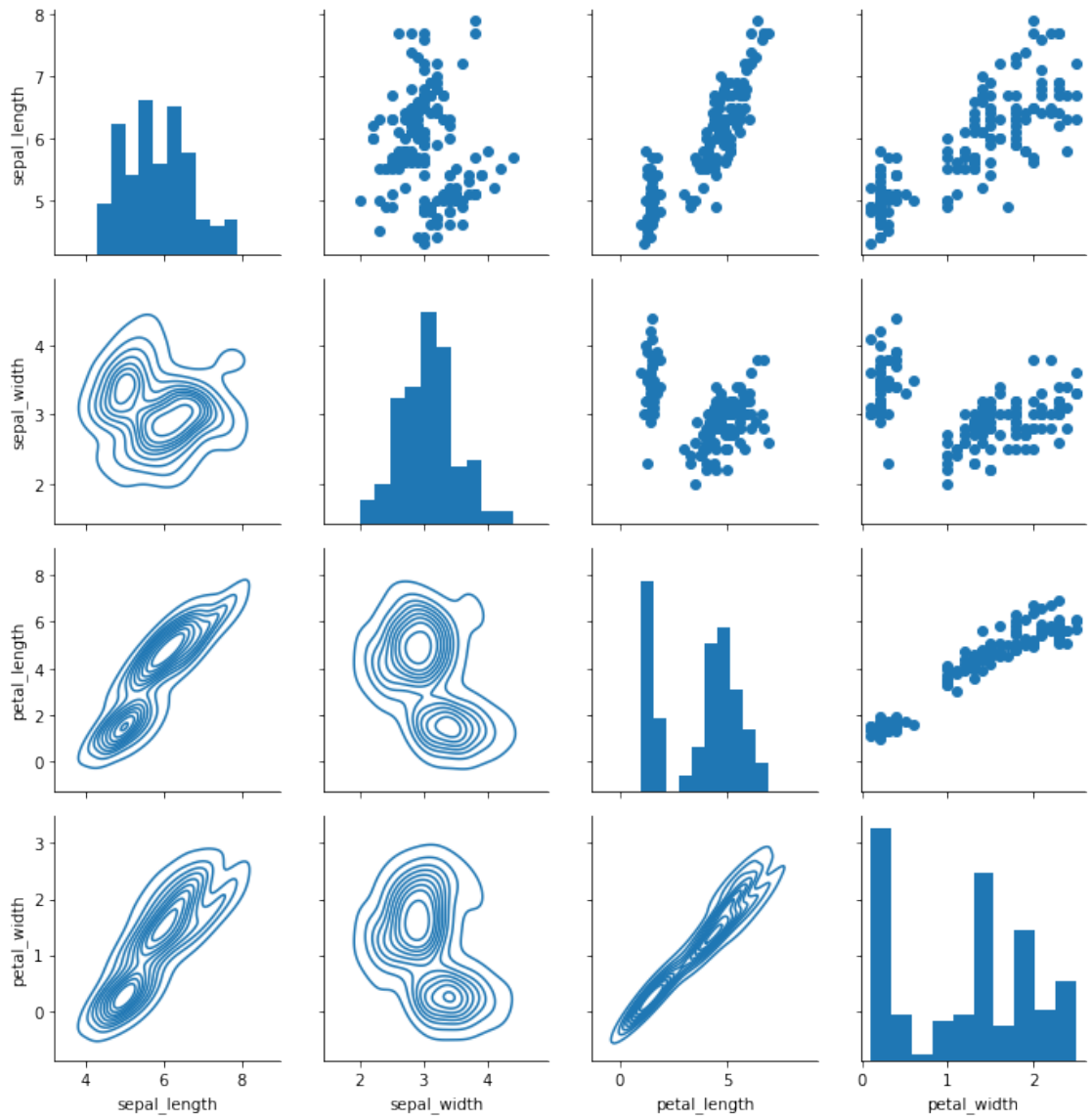
[58]:
```
# Then you map to the grid
g = sns.PairGrid(iris)
g.map(plt.scatter)
```

[58]: <seaborn.axisgrid.PairGrid at 0x7f8cd9dd0050>



[61]:
```
# Altera os tipos de plots na diagonal, parte superior e inferior.
g = sns.PairGrid(iris)
g.map_diag(plt.hist)
g.map_upper(plt.scatter)
g.map_lower(sns.kdeplot)
```

[61]: <seaborn.axisgrid.PairGrid at 0x7f8cd87dcb50>

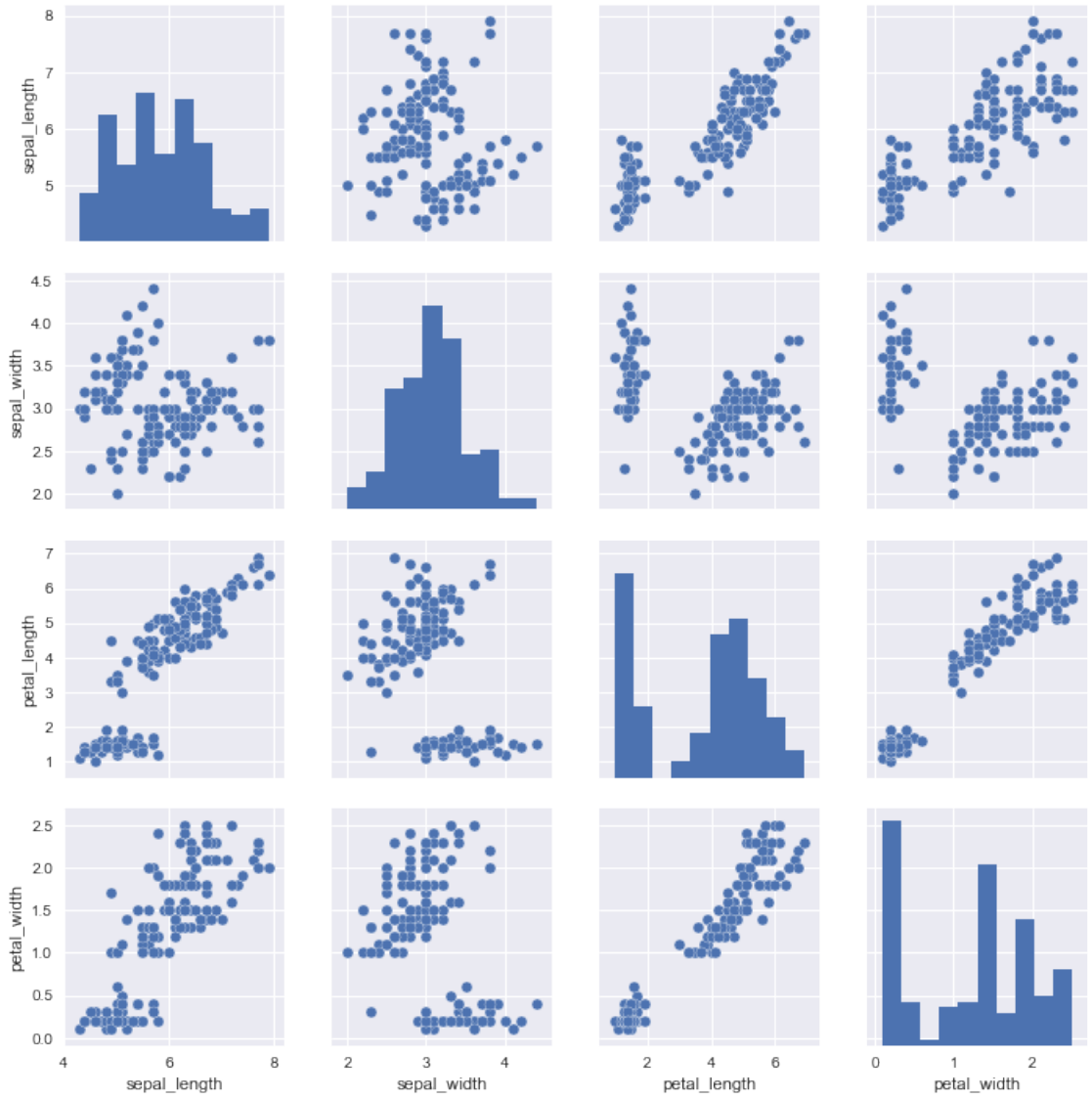


6.2 pairplot

Pairplot é uma versão mais simples do PairGrid (você usará com bastante frequência)

[ ]:
```
sns.pairplot(iris)
```

[ ]:
```
<seaborn.axisgrid.PairGrid at 0x25b9a4c6128>
```



[ ]:
```
sns.pairplot(iris,hue='species',palette='rainbow')
```

[ ]:
```
<seaborn.axisgrid.PairGrid at 0x25b9b9a59b0>
```

6.3 FacetGrid

FacetGrid é a maneira geral de criar plots de grades com base em um recurso:

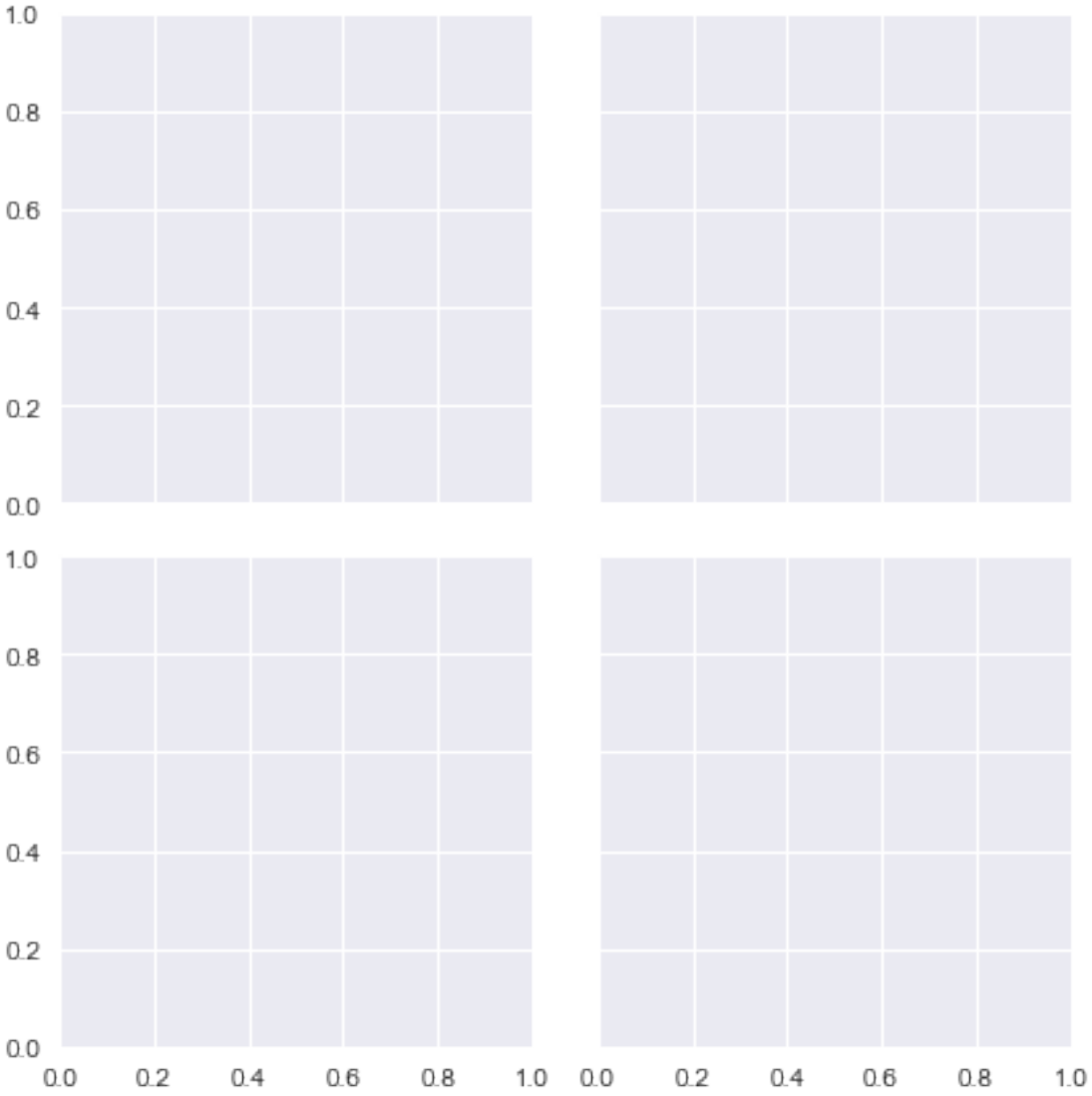
[ ]:
```
tips = sns.load_dataset('tips')
```

[ ]:
```
tips.head()
```

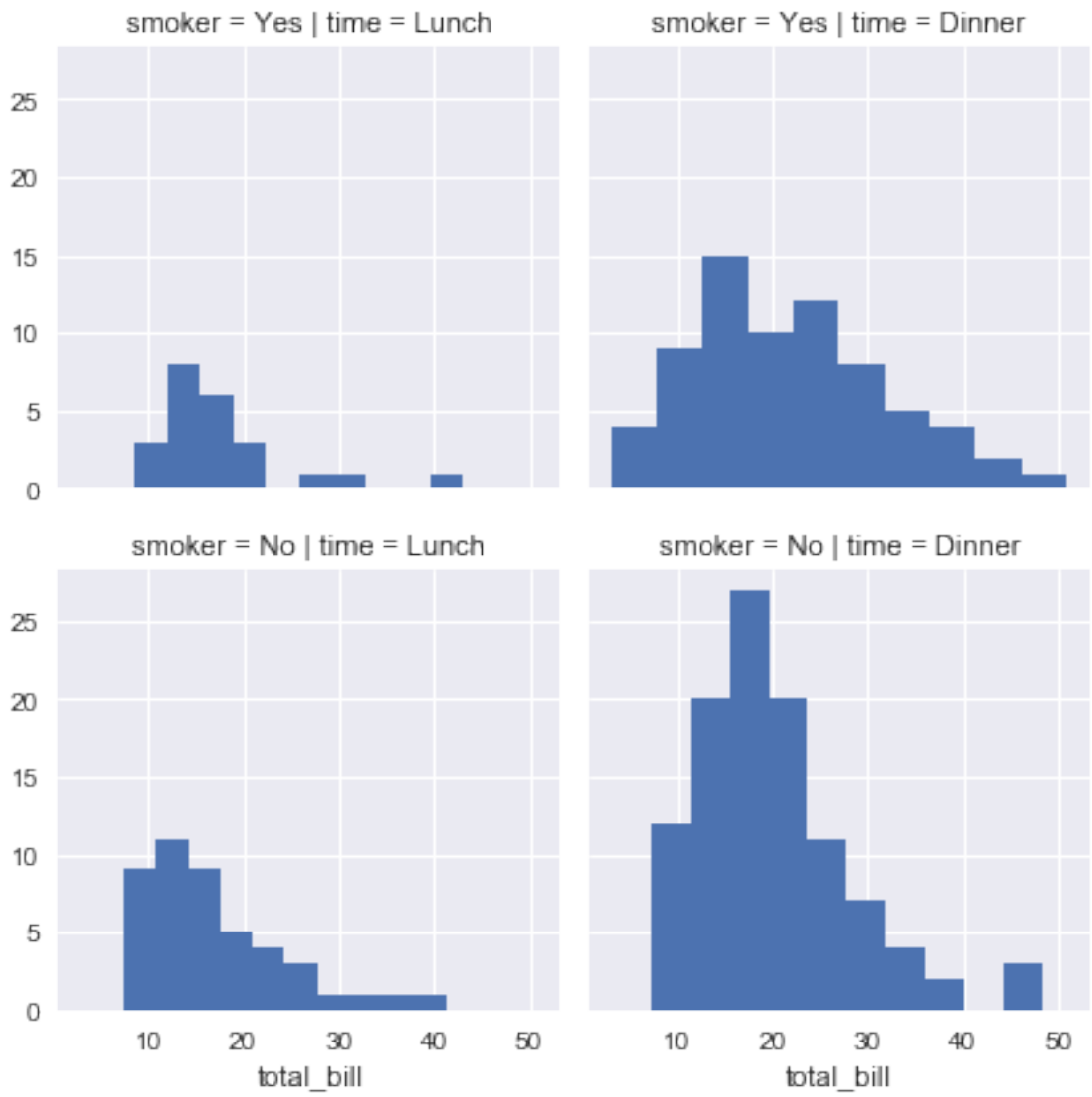
[ ]:

	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	3.61	Female	No	Sun	Dinner	4

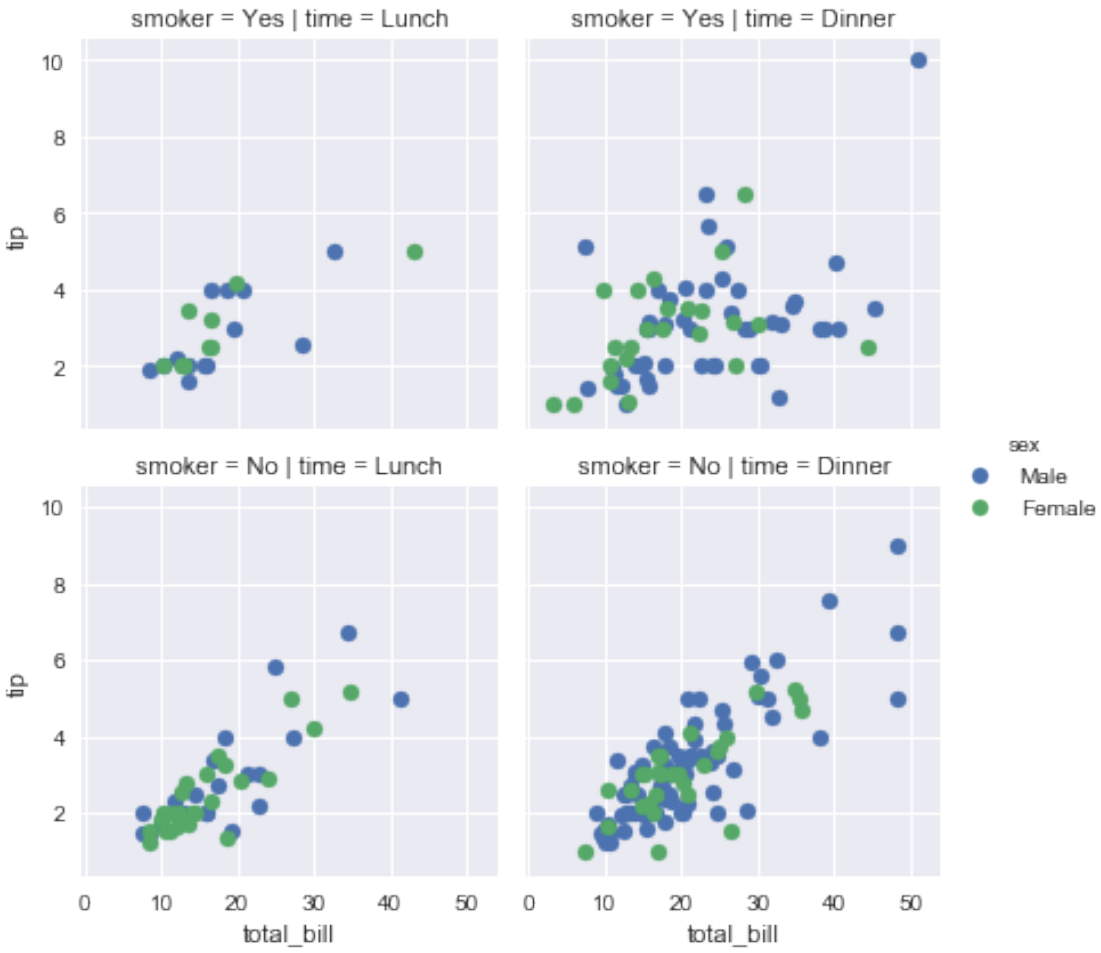
[ ]:
```
# Só a grade
g = sns.FacetGrid(tips, col="time", row="smoker")
```



[ ]:
```
g = sns.FacetGrid(tips, col="time",  row="smoker")
g = g.map(plt.hist, "total_bill")
```



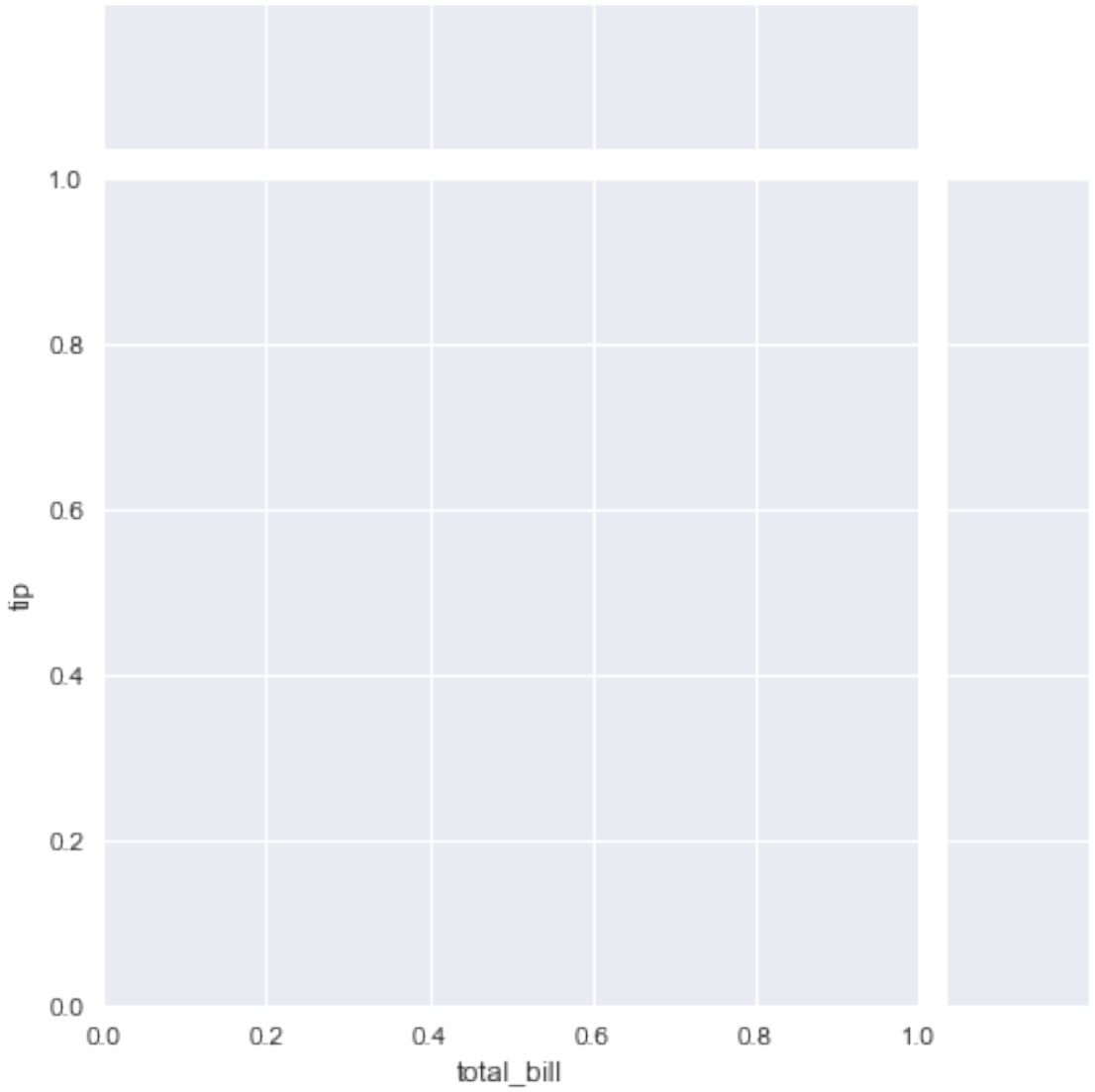
[ ]:
```
g = sns.FacetGrid(tips, col="time",  row="smoker",hue='sex')
# Observe como os argumentos vêm após a chamada do plt.scatter
g = g.map(plt.scatter, "total_bill", "tip").add_legend()
```



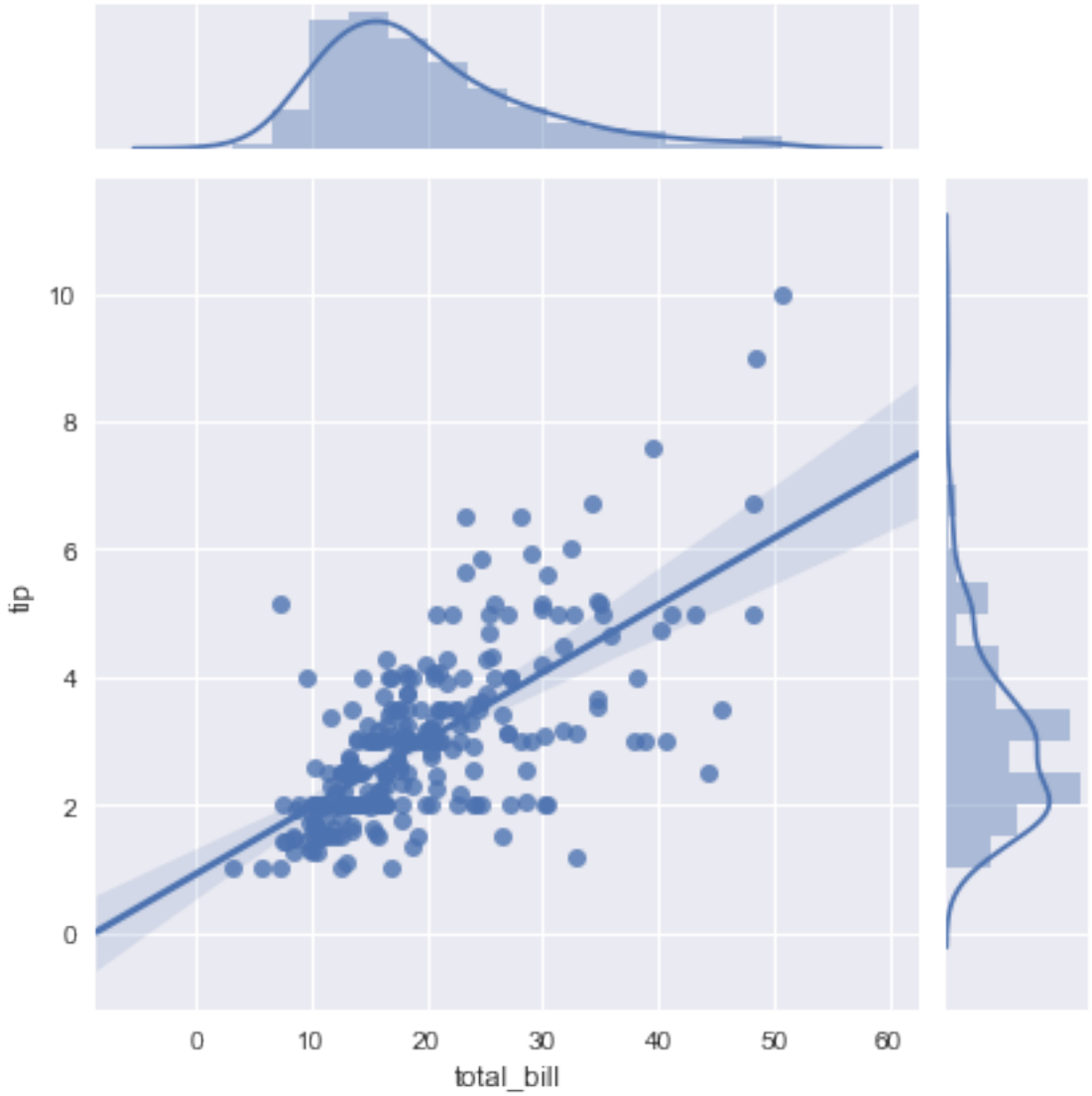
6.4 JointGrid

JointGrid é a versão geral para grades tipo jointplot (), para um exemplo rápido:

[ ]:
```
g = sns.JointGrid(x="total_bill", y="tip", data=tips)
```



[ ]:
```
g = sns.JointGrid(x="total_bill", y="tip", data=tips)
g = g.plot(sns.regplot, sns.distplot)
```



Consulte a documentação conforme necessário para os tipos de grade, mas na maioria das vezes você apenas usará os gráficos mais simples discutidos anteriormente.

7 Estilos e cores

Nós mostramos anteriormente como controlar a estética da figura em Seaborn, mas vamos agora examiná-lo formalmente:

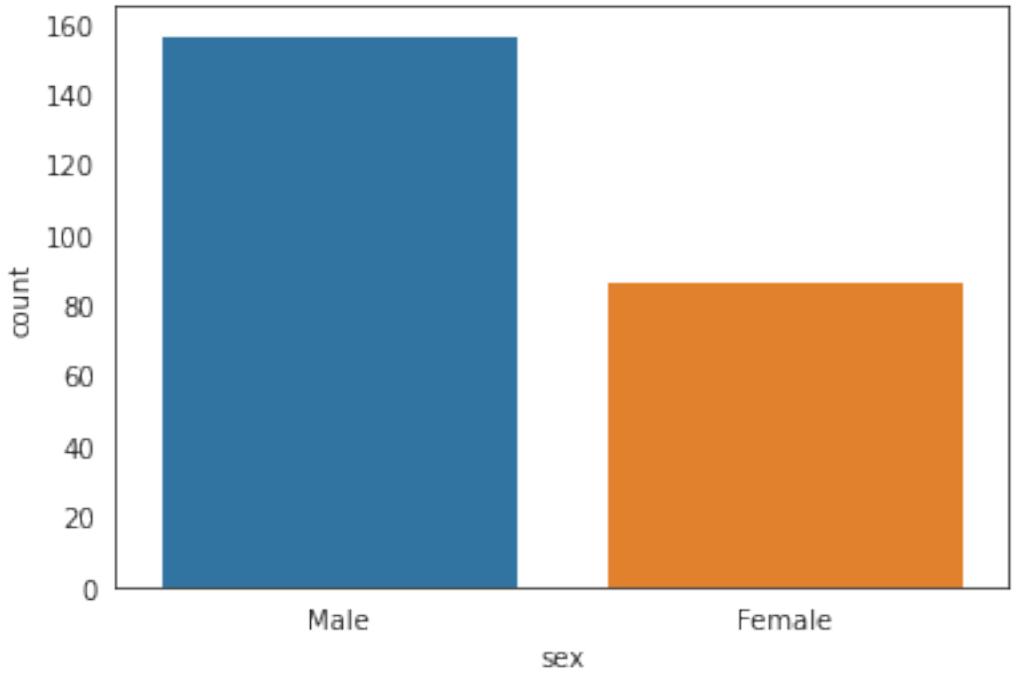
[65]:
```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
tips = sns.load_dataset('tips')
```

7.1 Styles

Você pode definir um estilo específico.

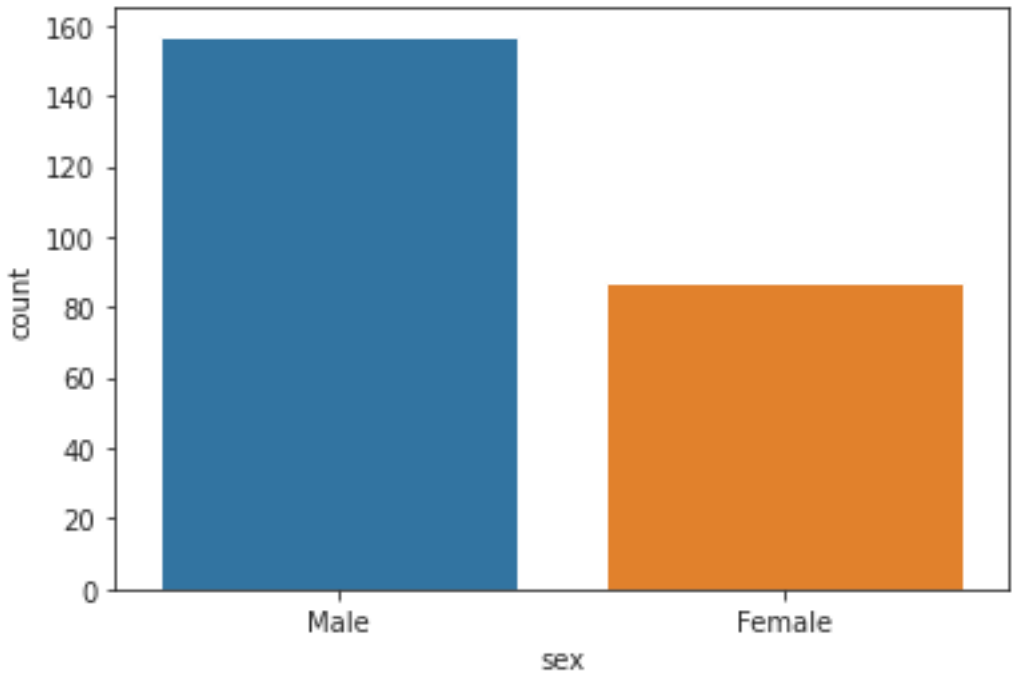
[66]:
```
sns.countplot(x='sex',data=tips)
```

[66]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8cd7d438d0>
```



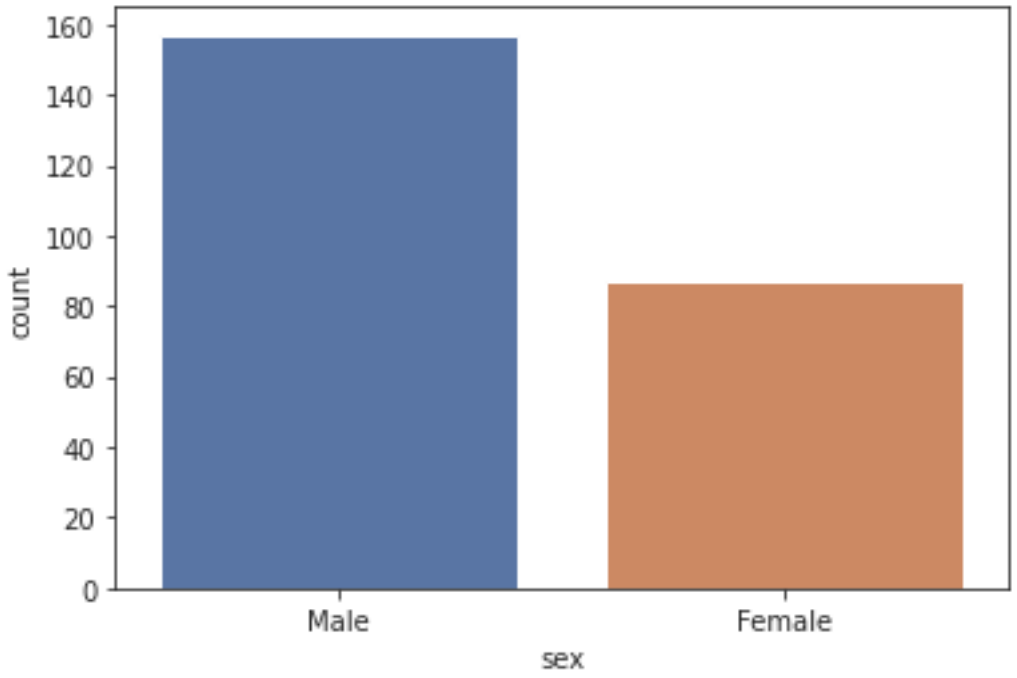
[70]:
```
sns.set_style('ticks')
sns.countplot(x='sex',data=tips)
```

[70]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8cd7b8b890>
```



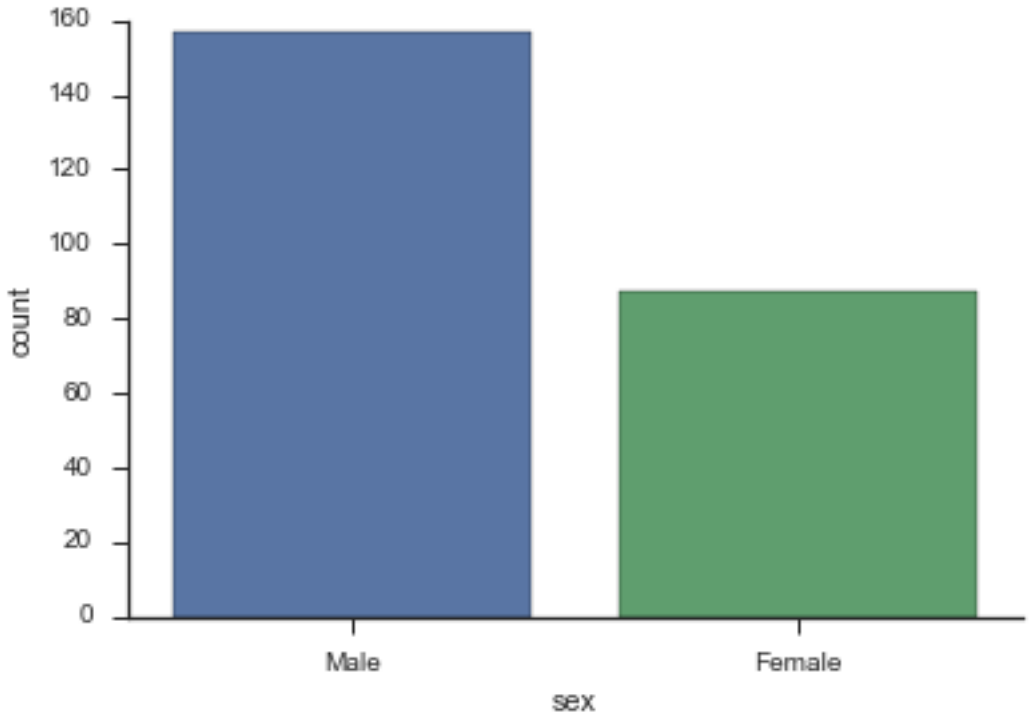
[68]:
```
sns.set_style('ticks')
sns.countplot(x='sex',data=tips,palette='deep')
```

[68]: <matplotlib.axes._subplots.AxesSubplot at 0x7f8cd7ca1190>

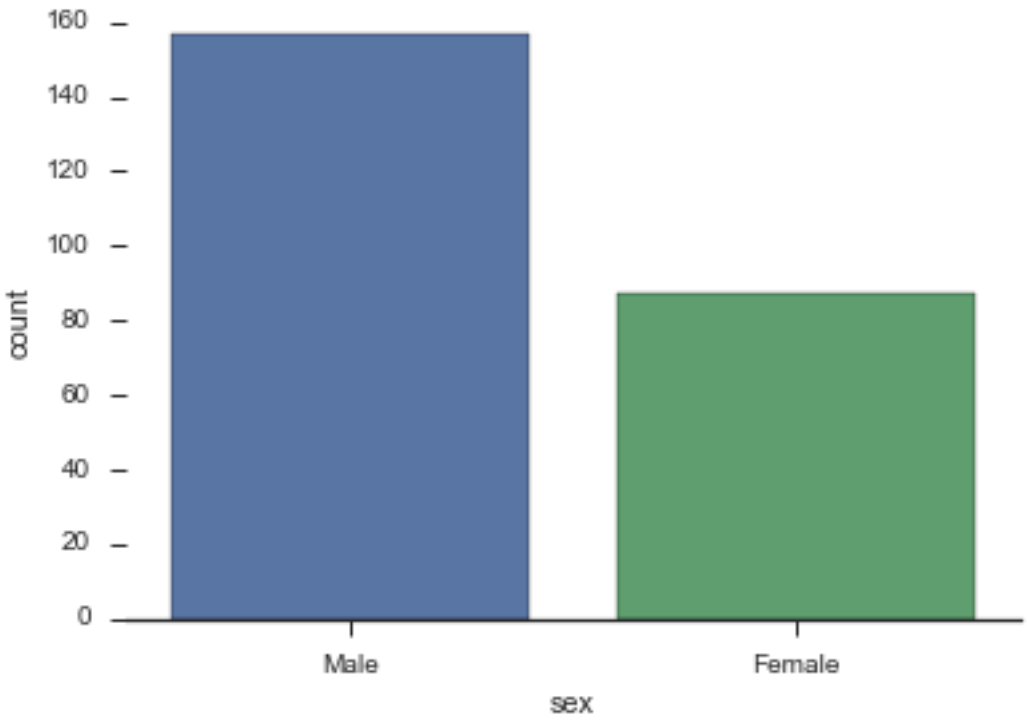


7.2 Remoção dos limites

[ ]:
```
sns.countplot(x='sex',data=tips)
sns.despine()
```



[ ]:
```
sns.countplot(x='sex',data=tips)
sns.despine(left=True)
```



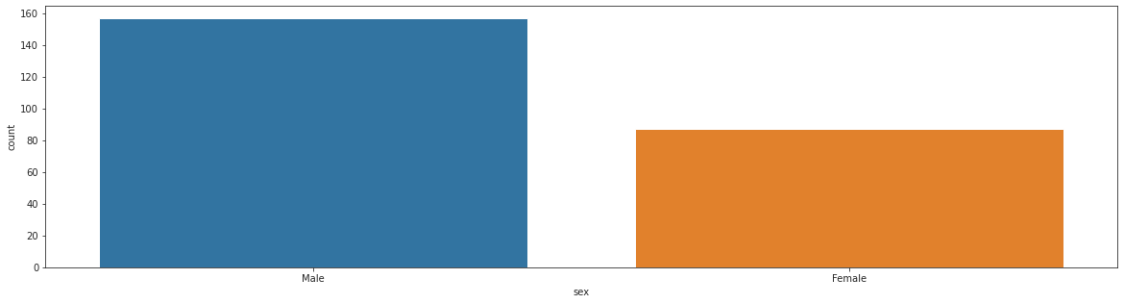
7.3 Tamanho e aspecto

Você pode usar o ** plt.figure do matplotlib (figsize = (width, height) ** para alterar o tamanho da maioria dos gráficos do seaborn.

Você pode controlar a proporção de tamanho e aspecto da maioria dos plots do seaborn passando parâmetros: size e aspect. Por exemplo:

[73]:
```
# Plot não gradeado
plt.figure(figsize=(20,5))
sns.countplot(x='sex',data=tips)
```

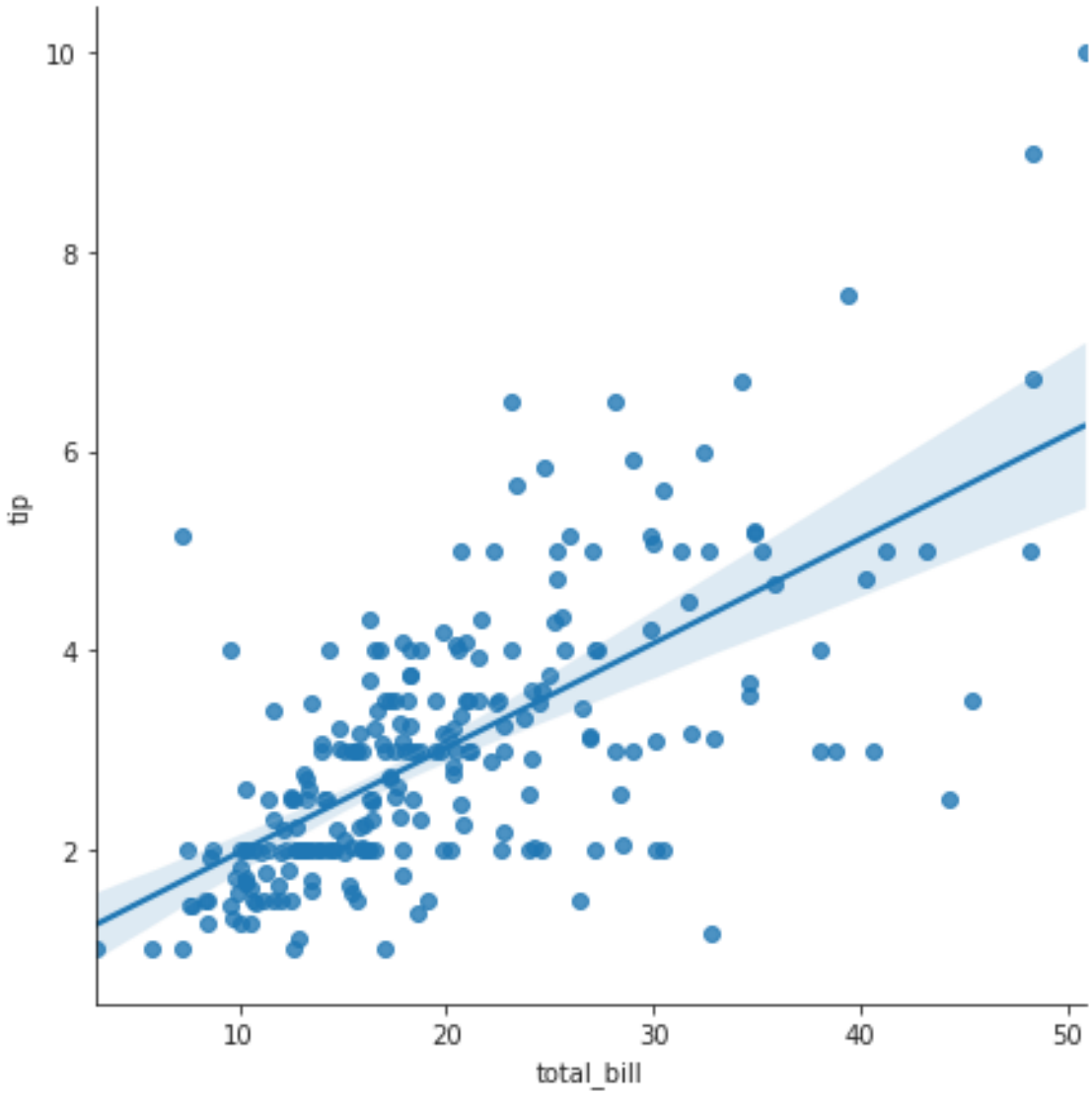
[73]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8cd7a4b090>
```



[77]:
```
# Plot tipo grade
sns.lmplot(x='total_bill',y='tip',size=6,aspect=1,data=tips)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/regression.py:580: UserWarning:
The `size` parameter has been renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
```

[77]: <seaborn.axisgrid.FacetGrid at 0x7f8cd78ddb50>

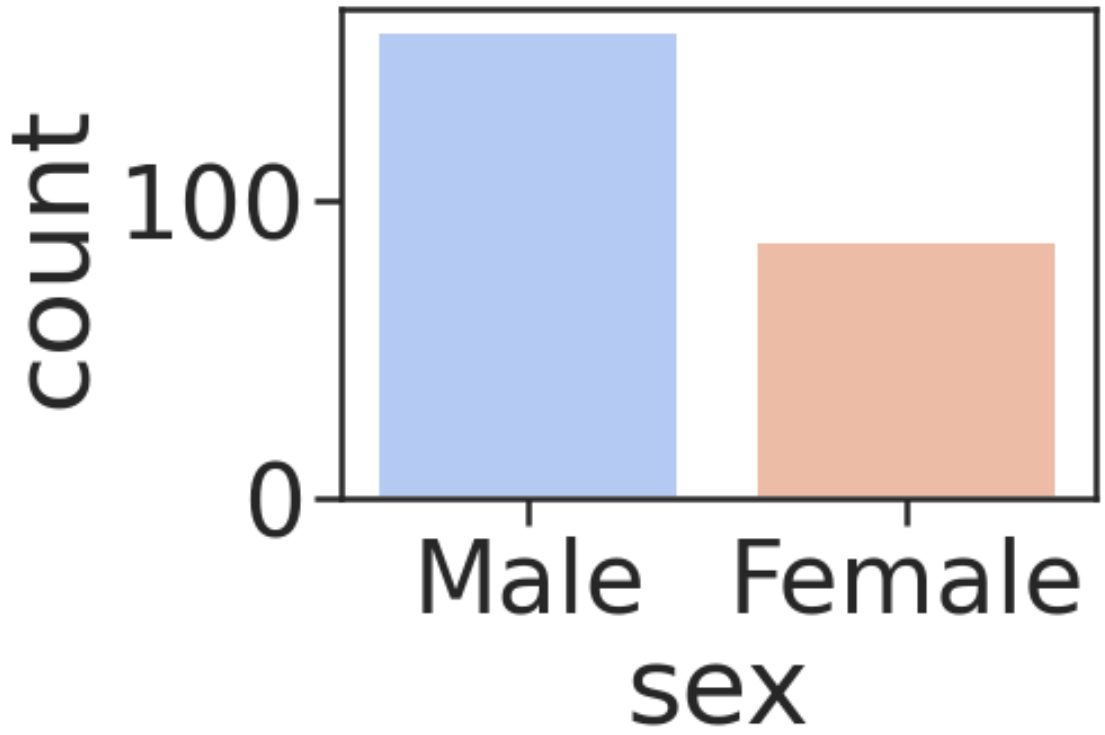


7.4 Escala e Contexto

O set_context () permite que você substitua parâmetros padrão:

[78]:
```
sns.set_context('poster',font_scale=2)
sns.countplot(x='sex',data=tips,palette='coolwarm')
```

[78]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8cd7855490>
```



Confira a página de documentação para obter mais informações sobre esses tópicos: https://stanford.edu/~mwaskom/software/seaborn/tutorial/aesthetics.html