1-plots-de-distribuicoes

September 11, 2023

1 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

1.1 Imports

[1]:
```
import seaborn as sns
%matplotlib inline
```

1.2 Dados

Seaborn vem com conjuntos de dados embutidos.

[2]:
```
tips = sns.load_dataset('tips')
```

[3]:
```
tips.head()
```

[3]:

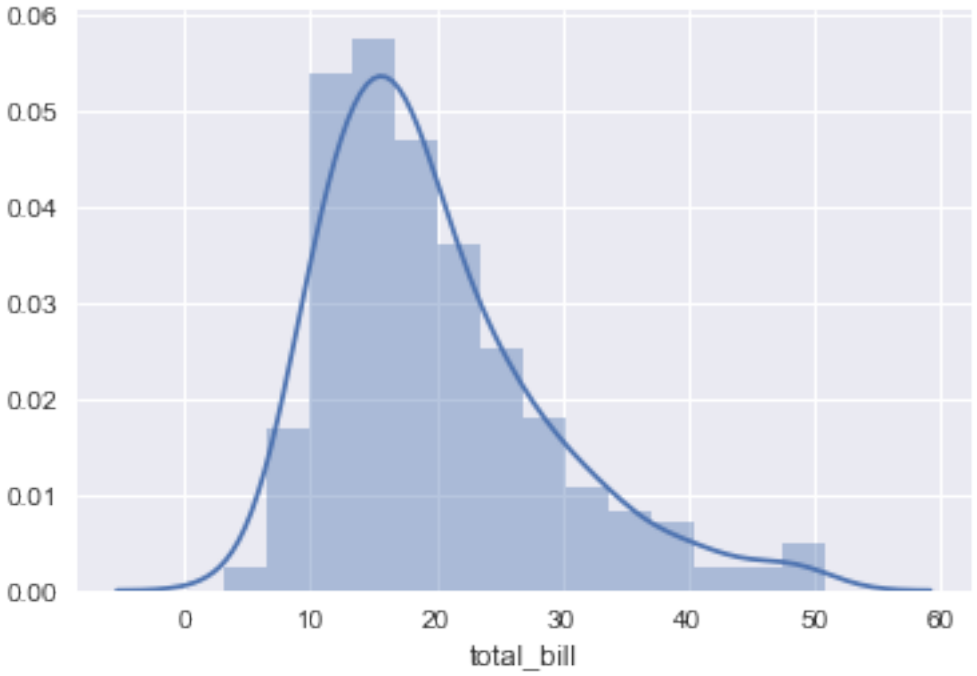
	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

1.3 distplot

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

[5]:
```
sns.distplot(tips['total_bill'])
```

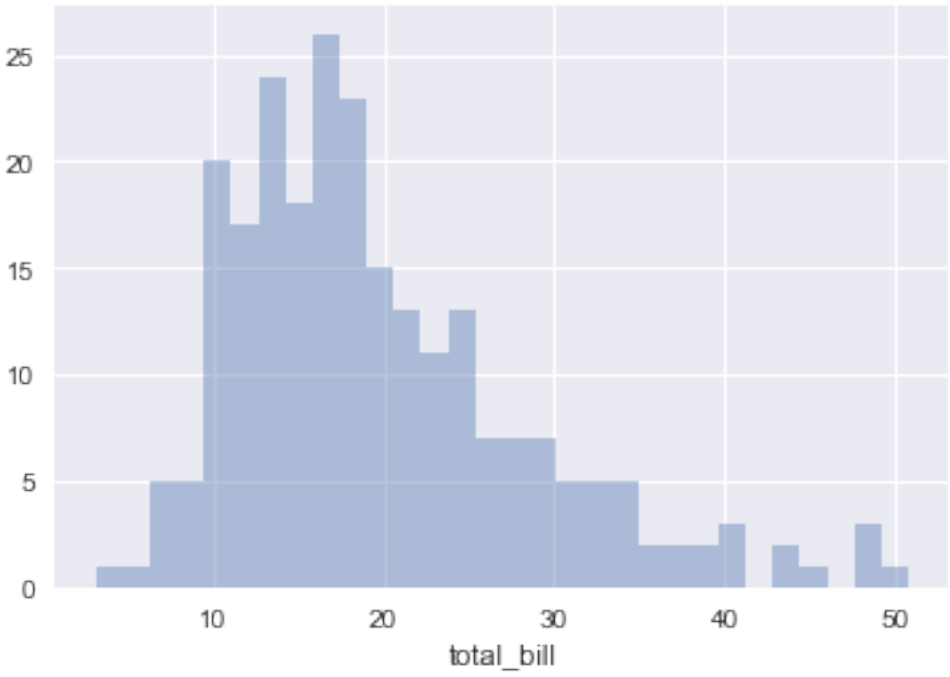
[5]: <matplotlib.axes._subplots.AxesSubplot at 0x158d12401d0>



Para remover a camada kde e apenas usar o histograma:

[6]:
```
sns.distplot(tips['total_bill'],kde=False,bins=30)
```

[6]: <matplotlib.axes._subplots.AxesSubplot at 0x158d142cb00>

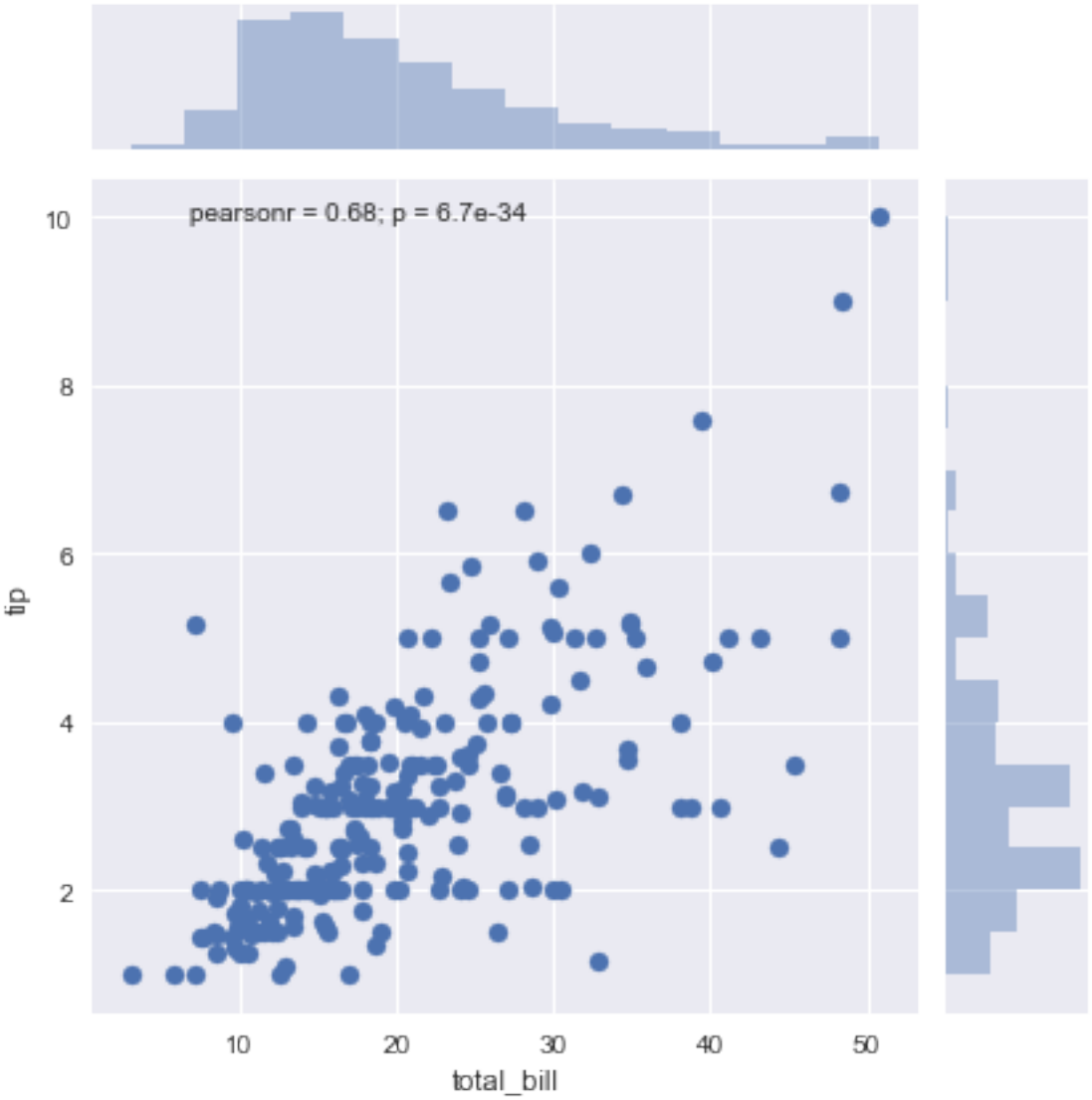


1.4 jointplot

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

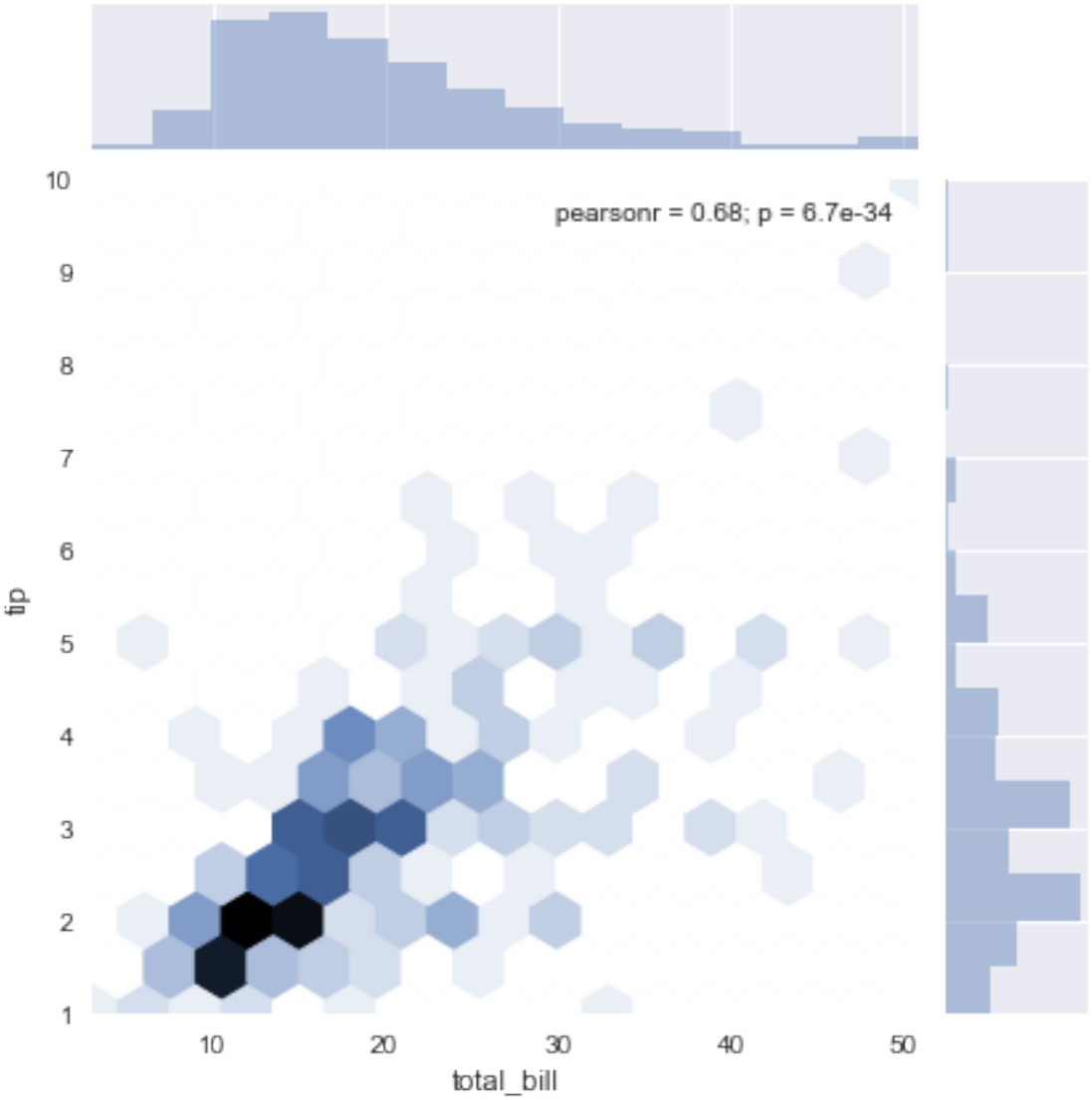
[7]:
```
sns.jointplot(x='total_bill',y='tip',data=tips,kind='scatter')
```

[7]:
```
<seaborn.axisgrid.JointGrid at 0x158d146f748>
```



[8]:
```
sns.jointplot(x='total_bill',y='tip',data=tips,kind='hex')
```

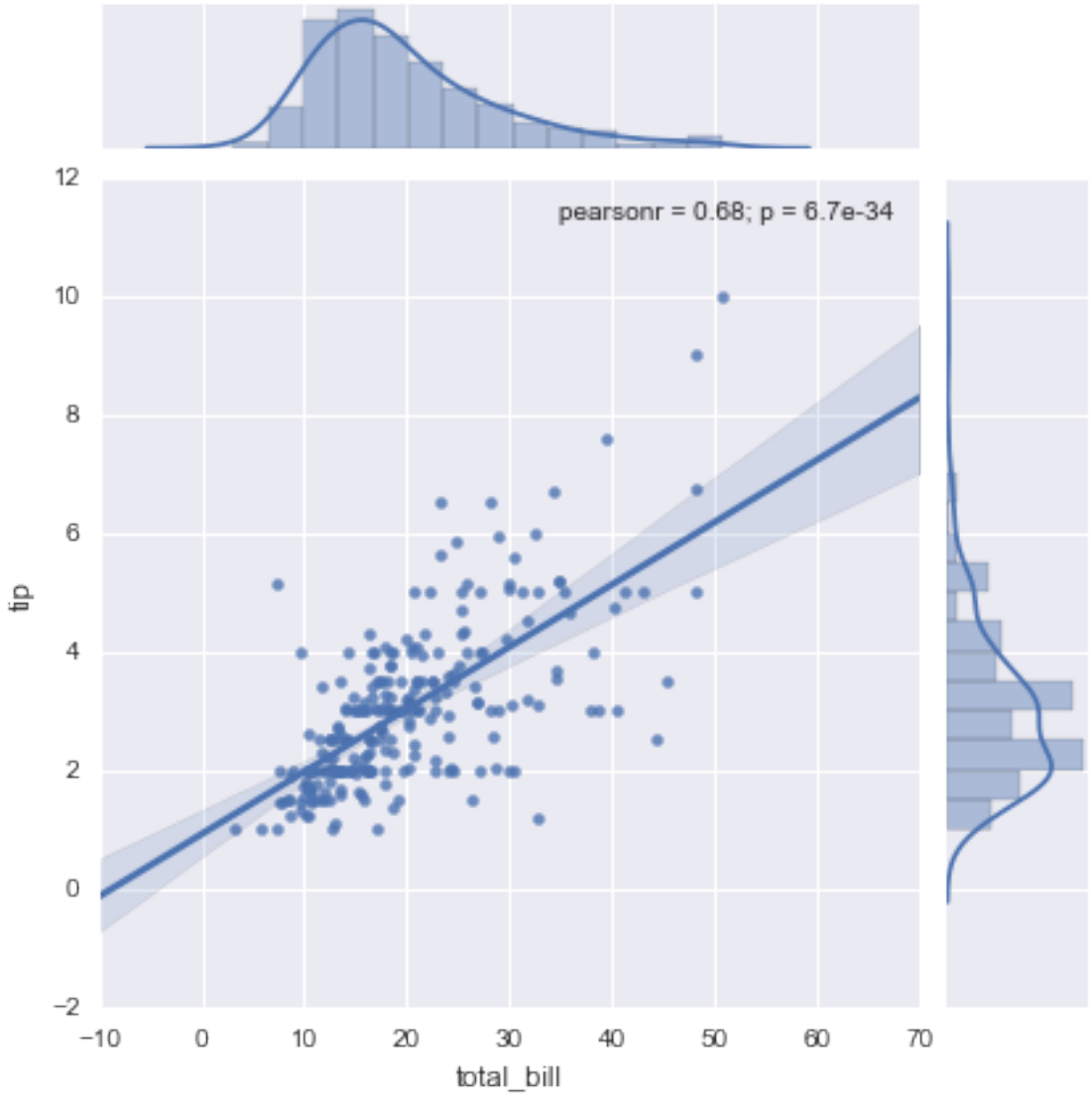
[8]: <seaborn.axisgrid.JointGrid at 0x158d2b0d1d0>



[17]:
```
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
```

[17]:
```
<seaborn.axisgrid.JointGrid at 0x11e0cfba8>
```

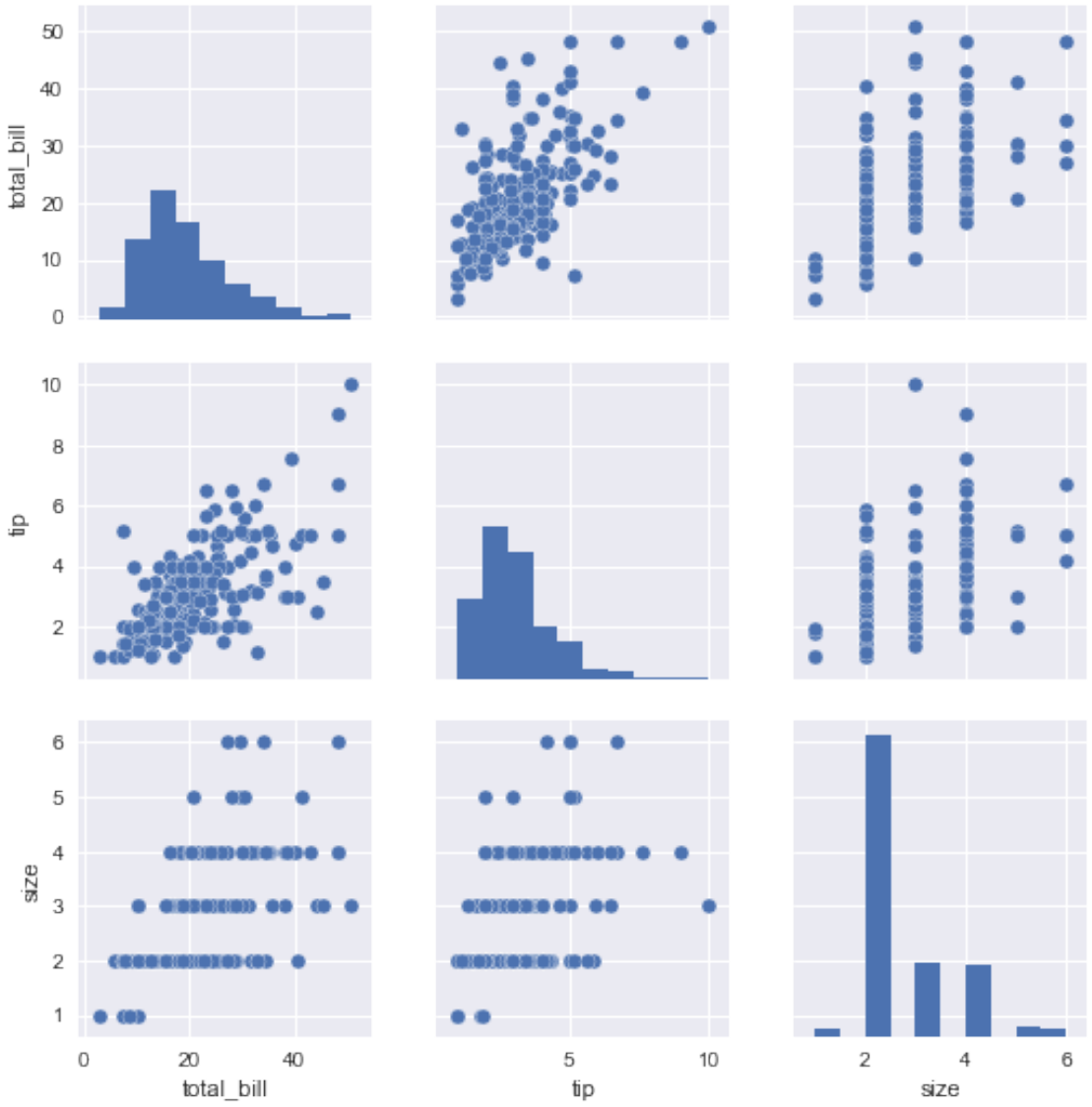


1.5 pairplot

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).

[9]:
```
sns.pairplot(tips)
```

[9]:
```
<seaborn.axisgrid.PairGrid at 0x158d2afc1d0>
```



[10]:
```
sns.pairplot(tips,hue='sex',palette='coolwarm')
```

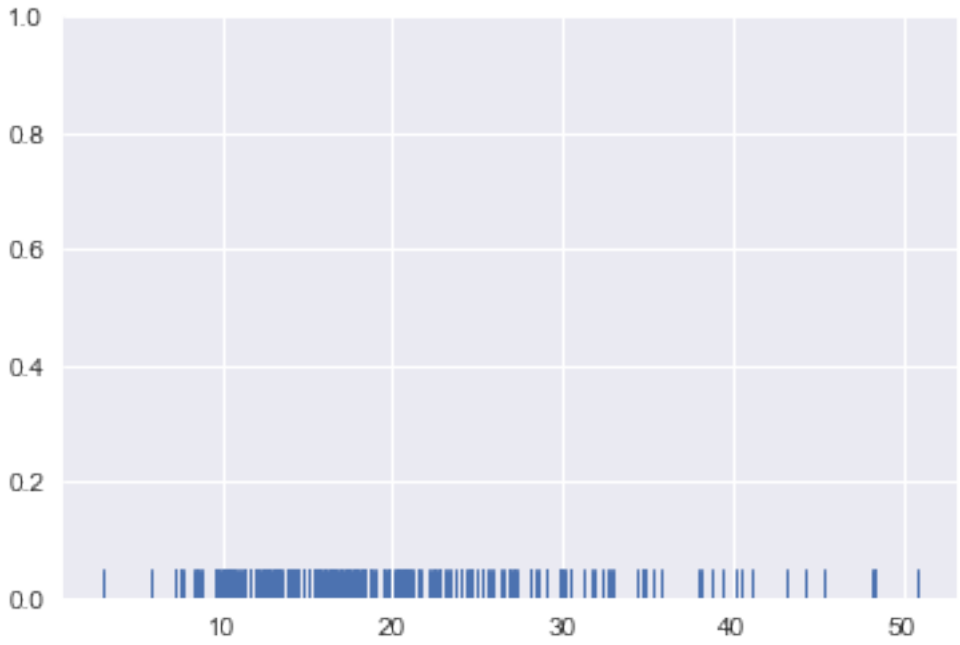
[10]: <seaborn.axisgrid.PairGrid at 0x158d352c5c0>

1.6 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:

[11]:
```
sns.rugplot(tips['total_bill'])
```

[11]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x158d41a3128>
```



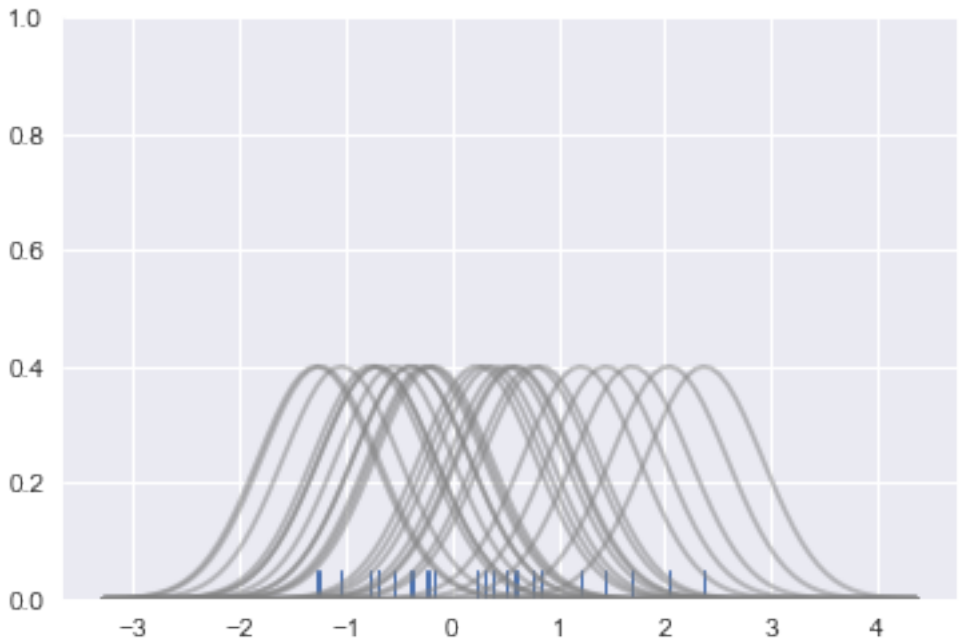
1.7 kdeplot

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:

[12]:
```
# 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:
```

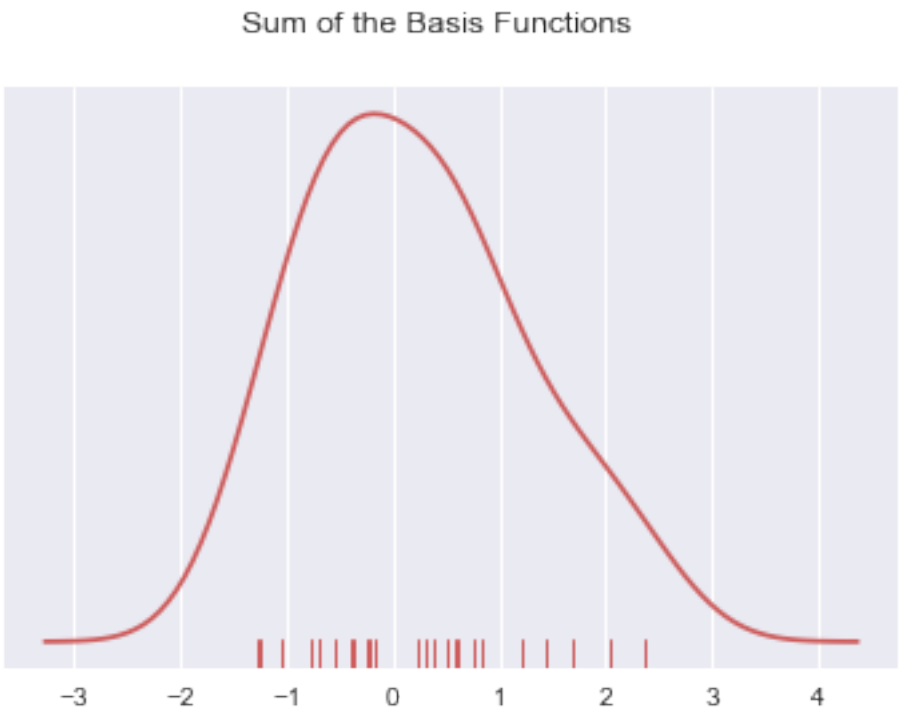
```
url = 'http://en.wikipedia.org/wiki/
 ↪Kernel_density_estimation#Practical_estimation_of_the_bandwidth'

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)
```

[12]: (0, 1)



[13]:
```
# 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")
```

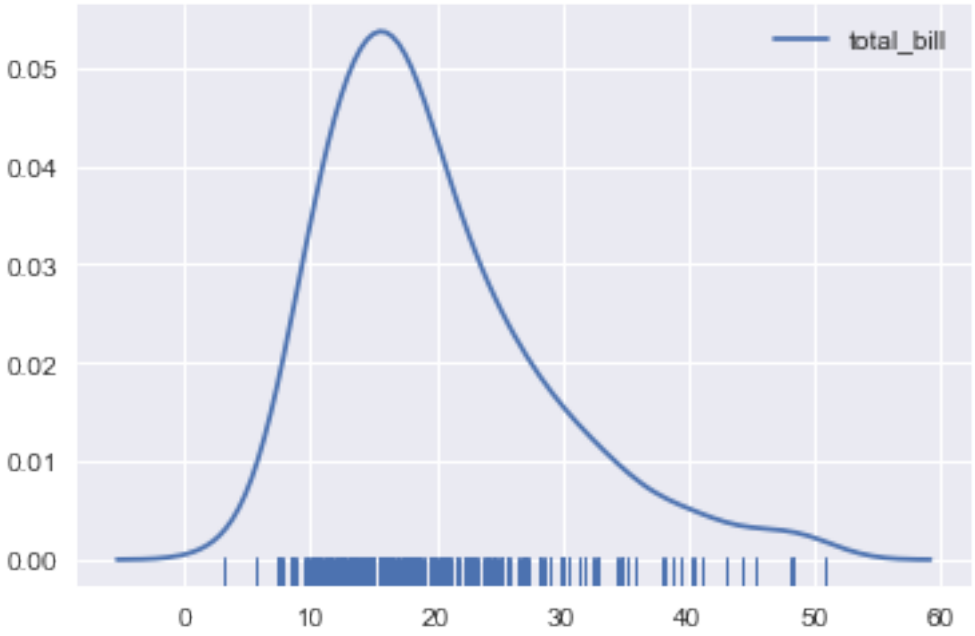
[13]:
```
<matplotlib.text.Text at 0x158d467beb8>
```



Então, com nosso DataFrame tips:

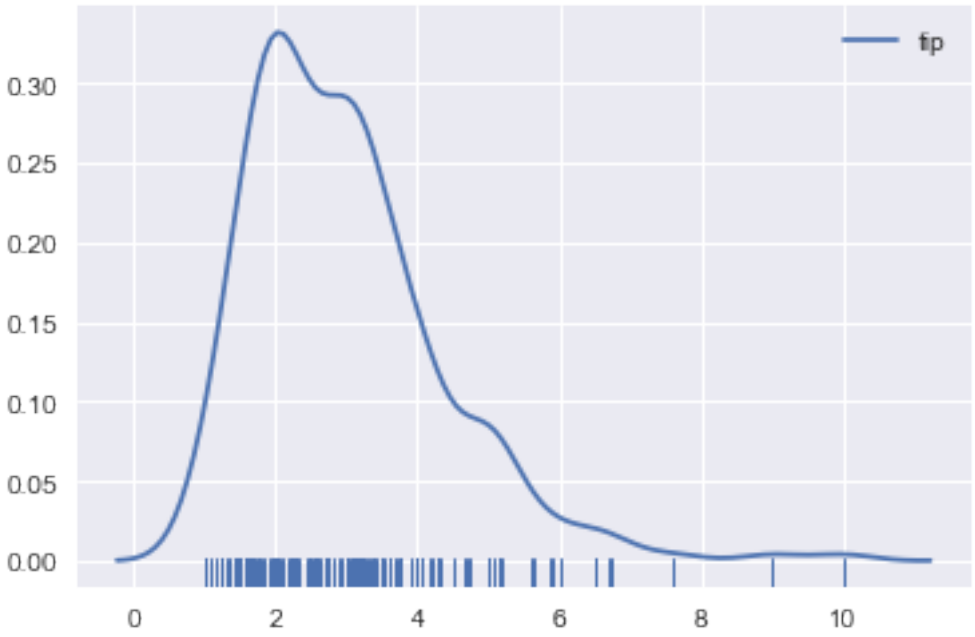
[14]:
```
sns.kdeplot(tips['total_bill'])
sns.rugplot(tips['total_bill'])
```

[14]: <matplotlib.axes._subplots.AxesSubplot at 0x158d467bc18>



[15]:
```
sns.kdeplot(tips['tip'])
sns.rugplot(tips['tip'])
```

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



2-plots-categoricos

September 11, 2023

1 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.

[1]:
```
import seaborn as sns
%matplotlib inline
```

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

[3]:

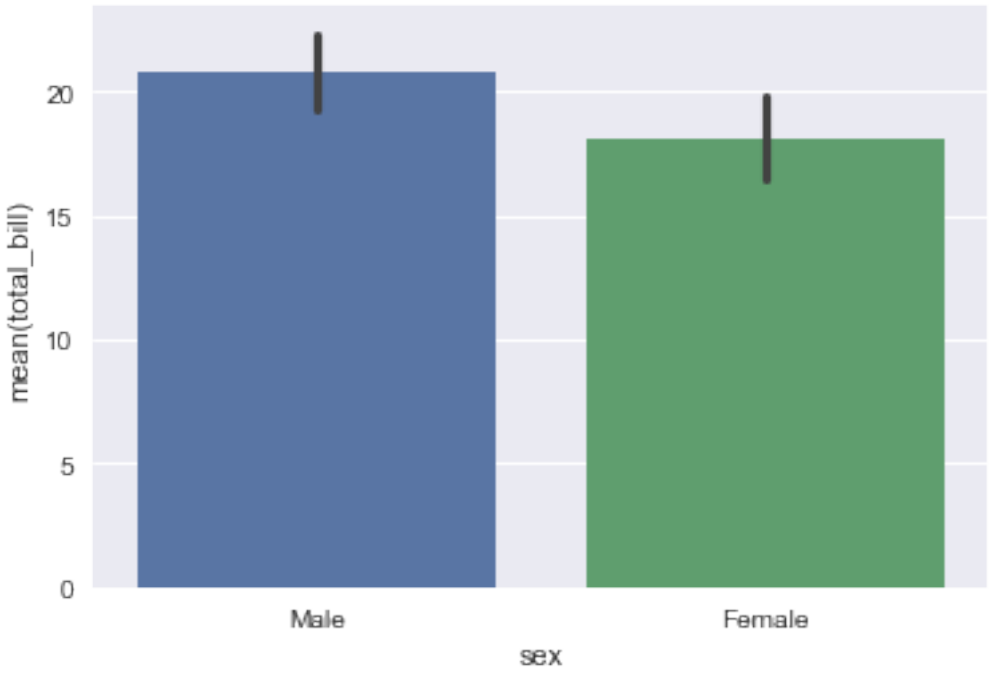
	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

1.1 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:

[4]:
```
sns.barplot(x='sex',y='total_bill',data=tips)
```

[4]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x16ec8082160>
```

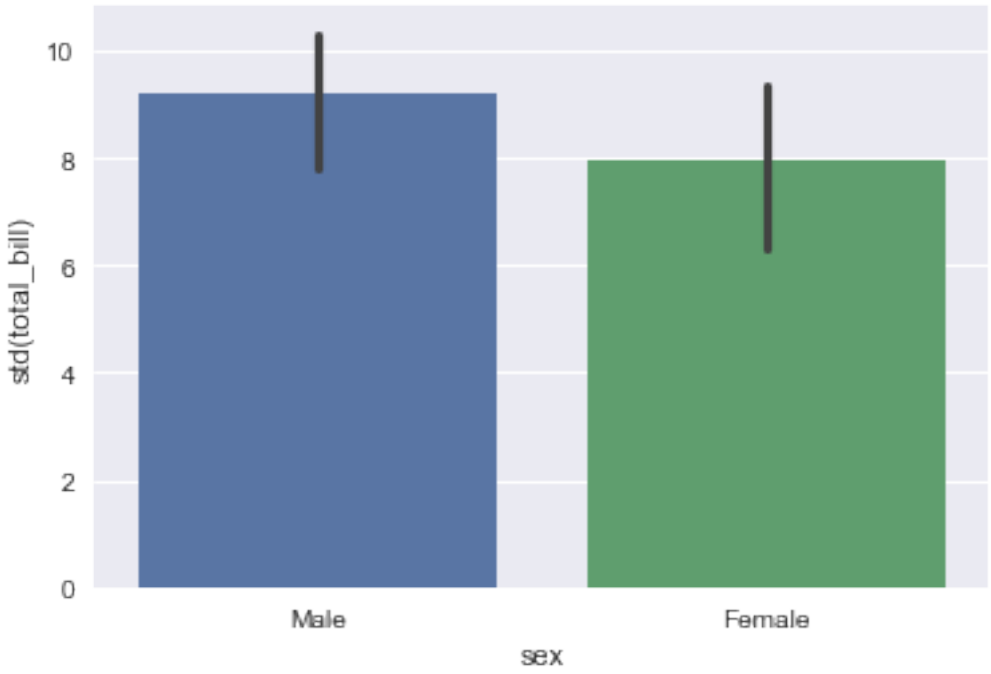


[5]:
```
import numpy as np
```

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

[6]:
```
sns.barplot(x='sex',y='total_bill',data=tips,estimator=np.std)
```

[6]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x16ec89fbcf8>
```

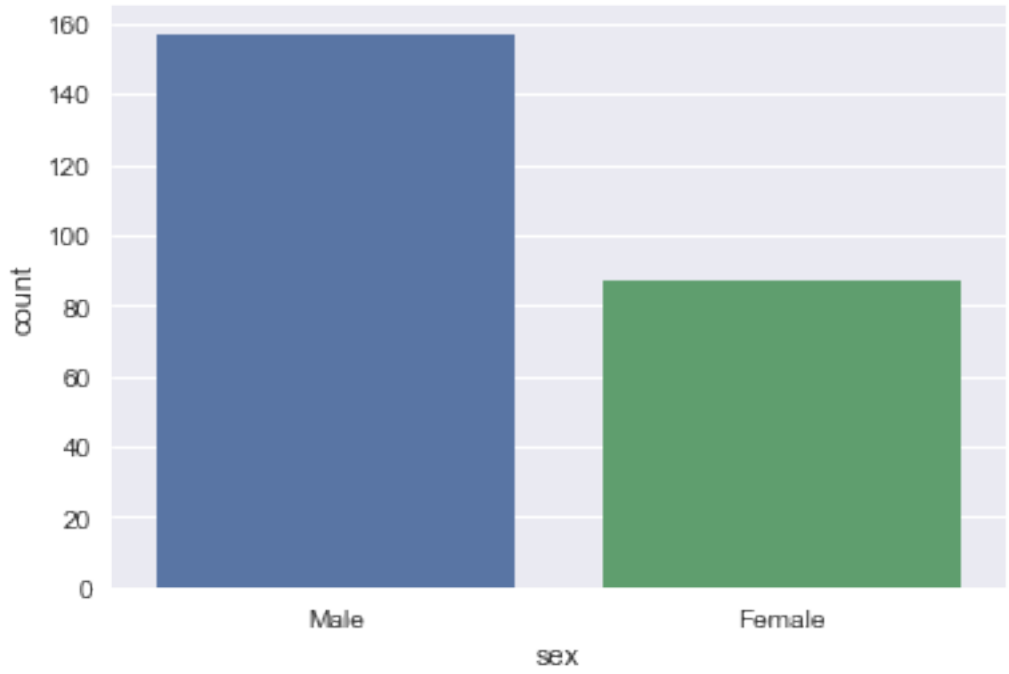


1.1.1 countplot

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:

[7]:
```
sns.countplot(x='sex',data=tips)
```

[7]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x16ec88acbe0>
```

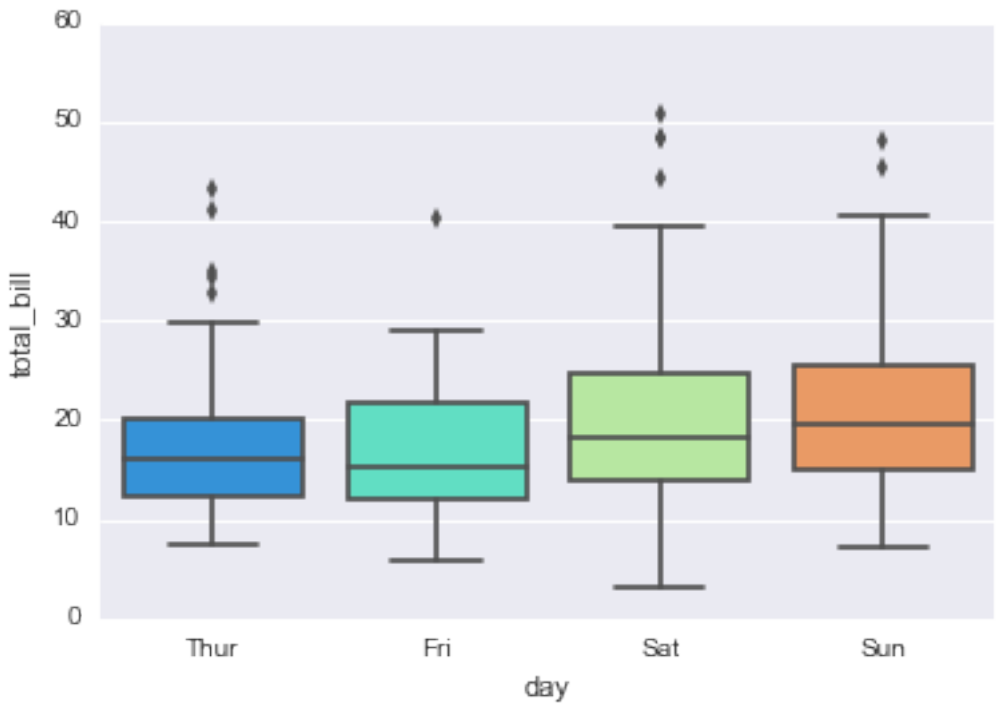


1.2 boxplot and violinplot

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".

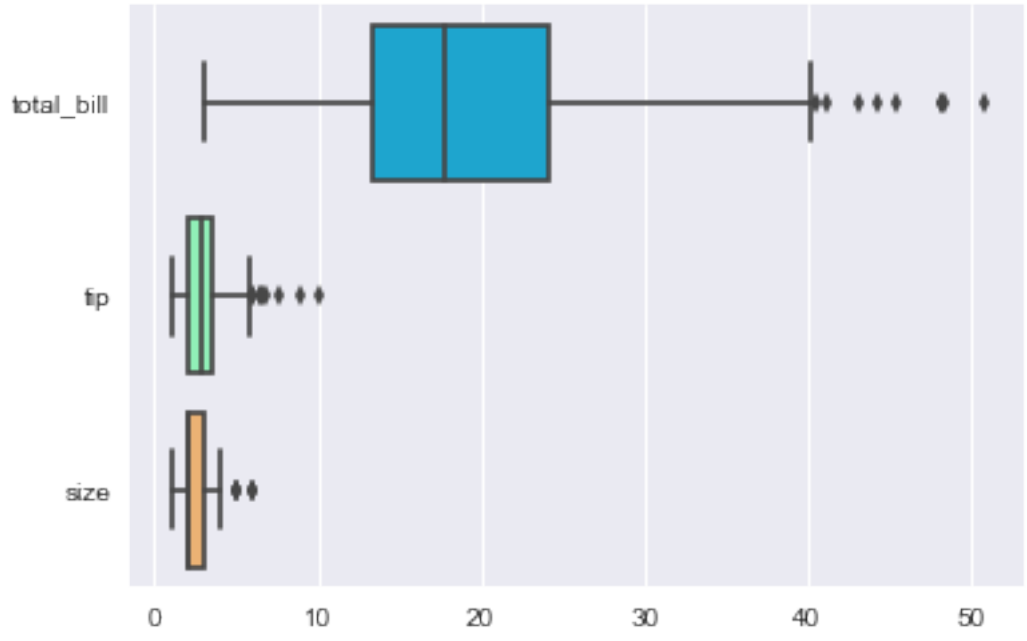
[22]:
```
sns.boxplot(x="day", y="total_bill", data=tips,palette='rainbow')
```

[22]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x11db81630>
```



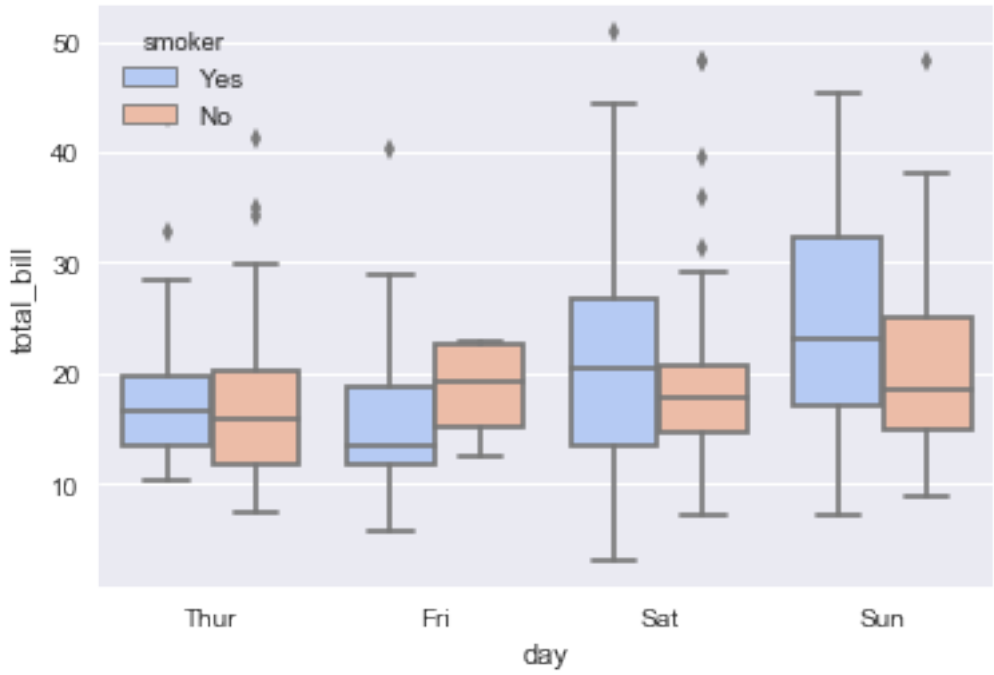
[8]:
```
# Podemos orientar os dados para aparecerem na horizontal
sns.boxplot(data=tips,palette='rainbow',orient='h')
```

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



[9]:
```
sns.boxplot(x="day", y="total_bill", hue="smoker",data=tips, palette="coolwarm")
```

[9]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x16ec9d1df28>
```

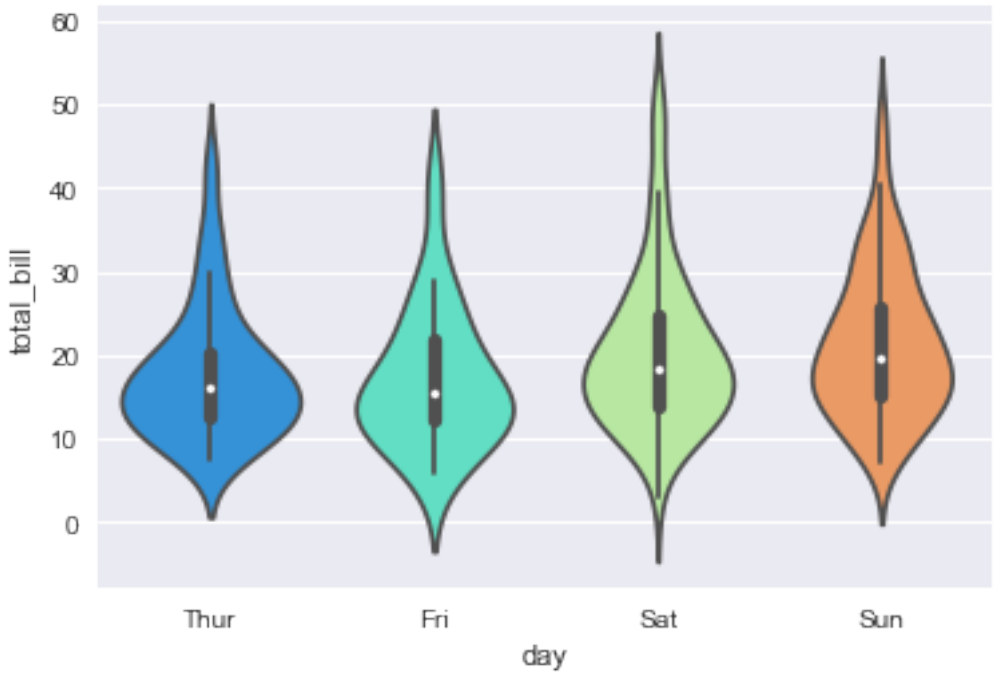


1.2.1 violinplot

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.

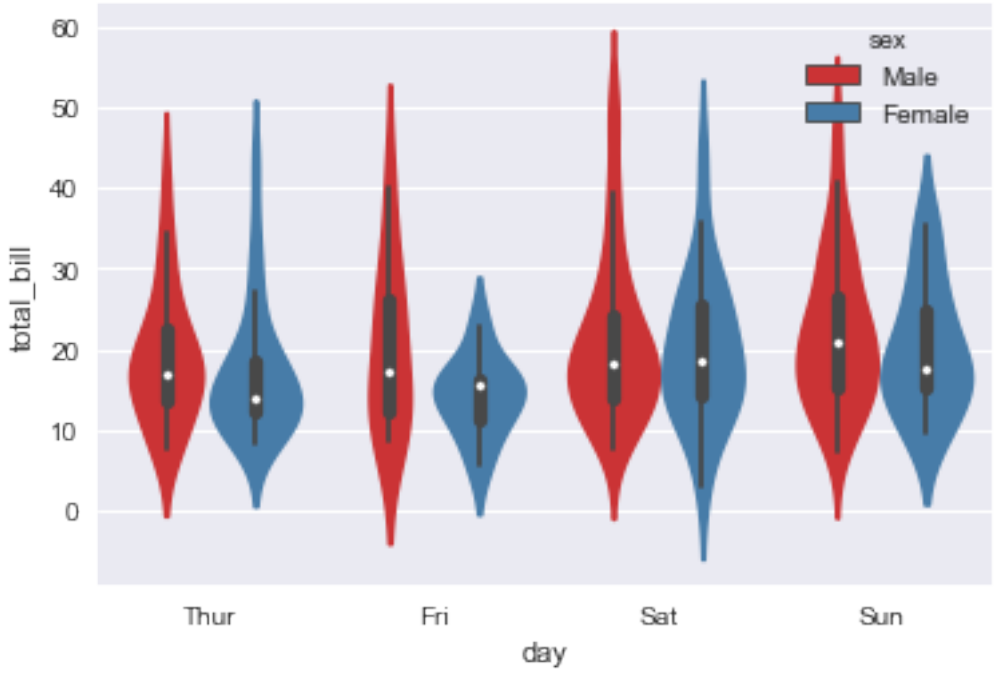
[10]:
```
sns.violinplot(x="day", y="total_bill", data=tips,palette='rainbow')
```

[10]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x16ec9eb9f60>
```



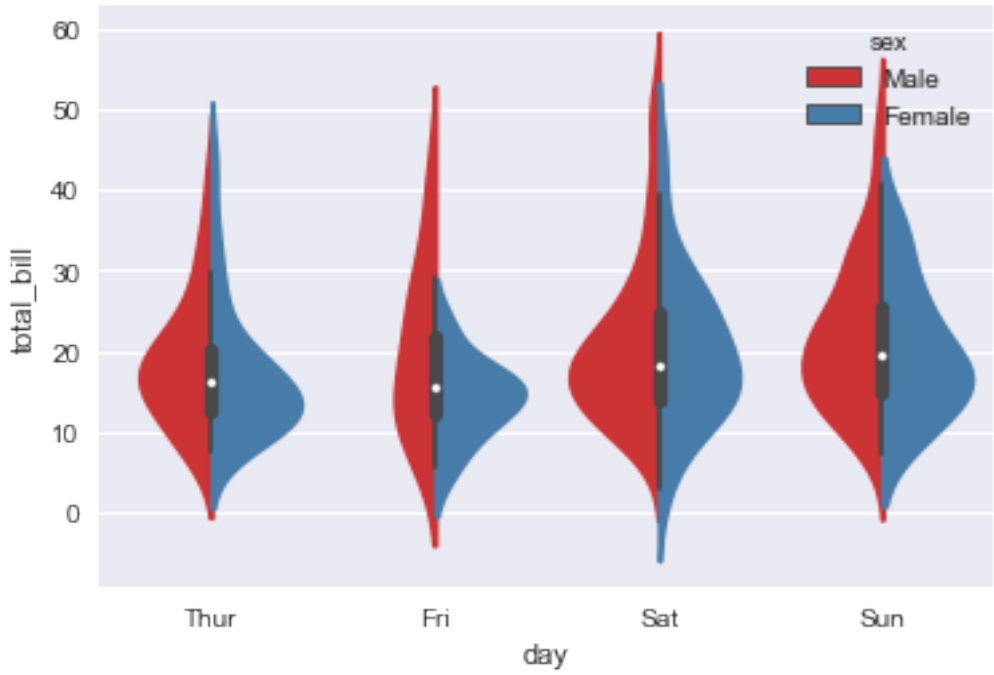
[11]:
```
sns.violinplot(x="day", y="total_bill", data=tips,hue='sex',palette='Set1')
```

[11]: <matplotlib.axes._subplots.AxesSubplot at 0x16ec9eba6a0>



[12]:
```
sns.violinplot(x="day", y="total_bill",␣
↪data=tips,hue='sex',split=True,palette='Set1')
```

[12]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x16eca086cf8>
```



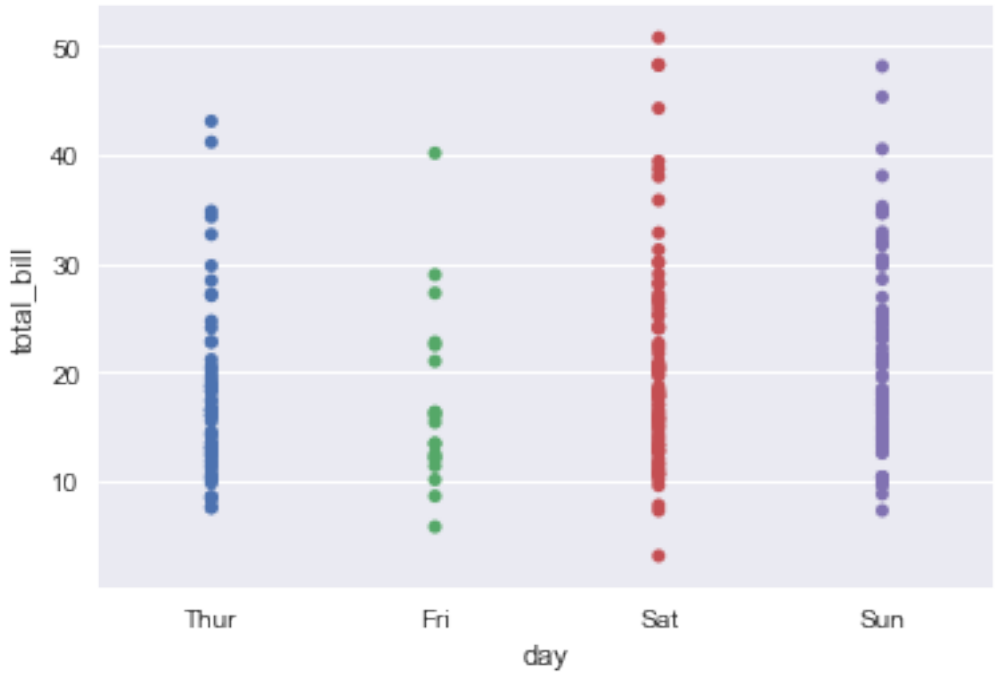
1.3 stripplot e swarmplot

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).

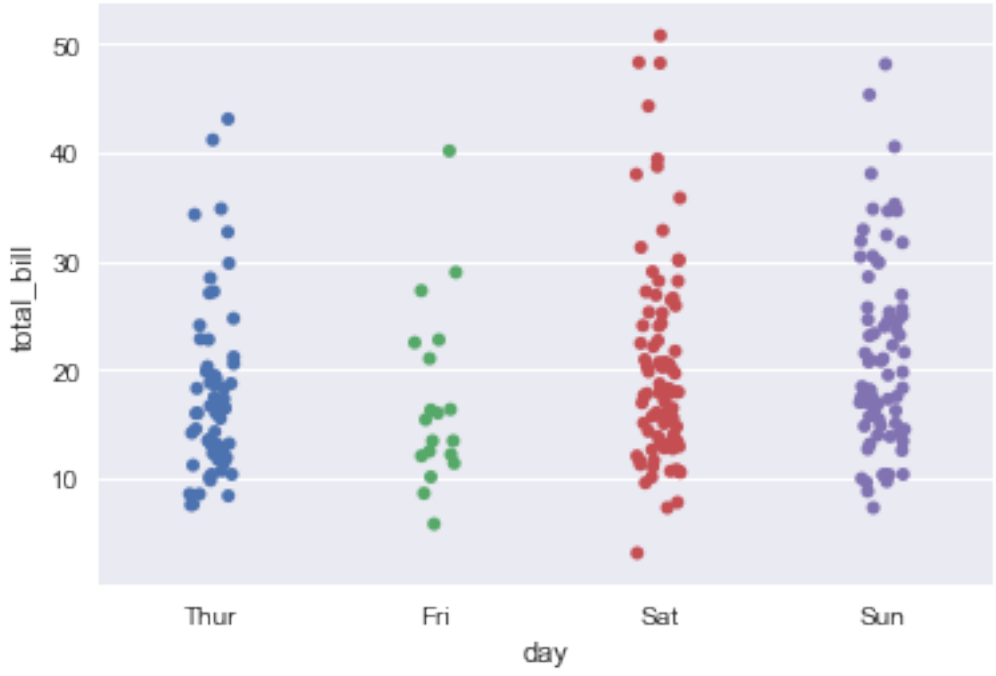
[13]:
```
sns.stripplot(x="day", y="total_bill", data=tips)
```

[13]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x16eca0f50f0>
```



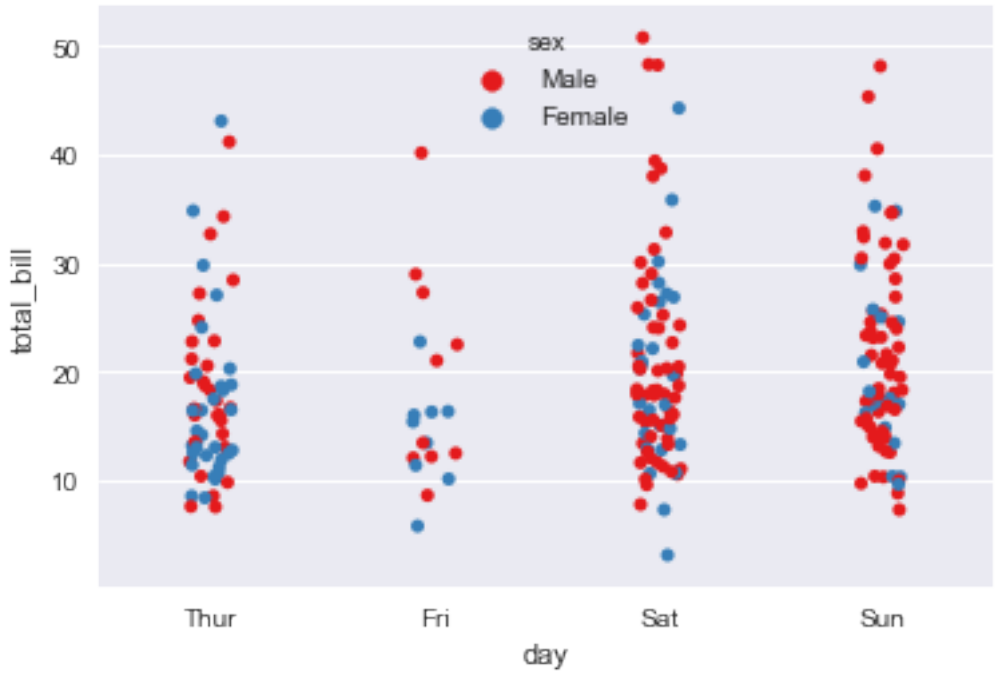
[14]:
```
sns.stripplot(x="day", y="total_bill", data=tips,jitter=True)
```

[14]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x16eca1db668>
```



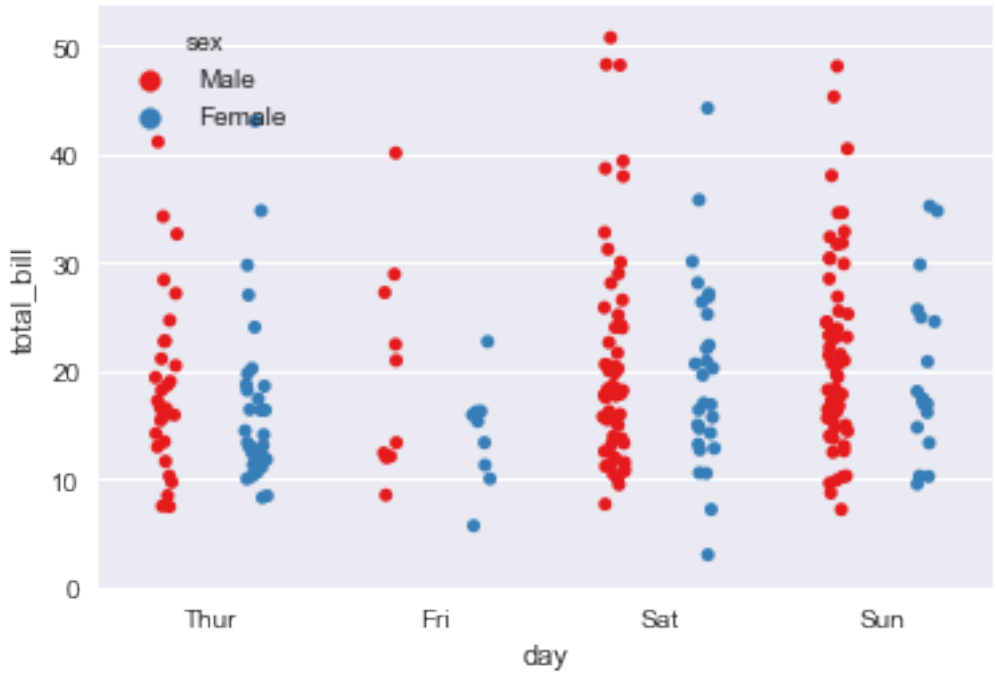
[15]:
```
sns.stripplot(x="day", y="total_bill",␣
↪data=tips,jitter=True,hue='sex',palette='Set1')
```

[15]: <matplotlib.axes._subplots.AxesSubplot at 0x16eca1d5400>



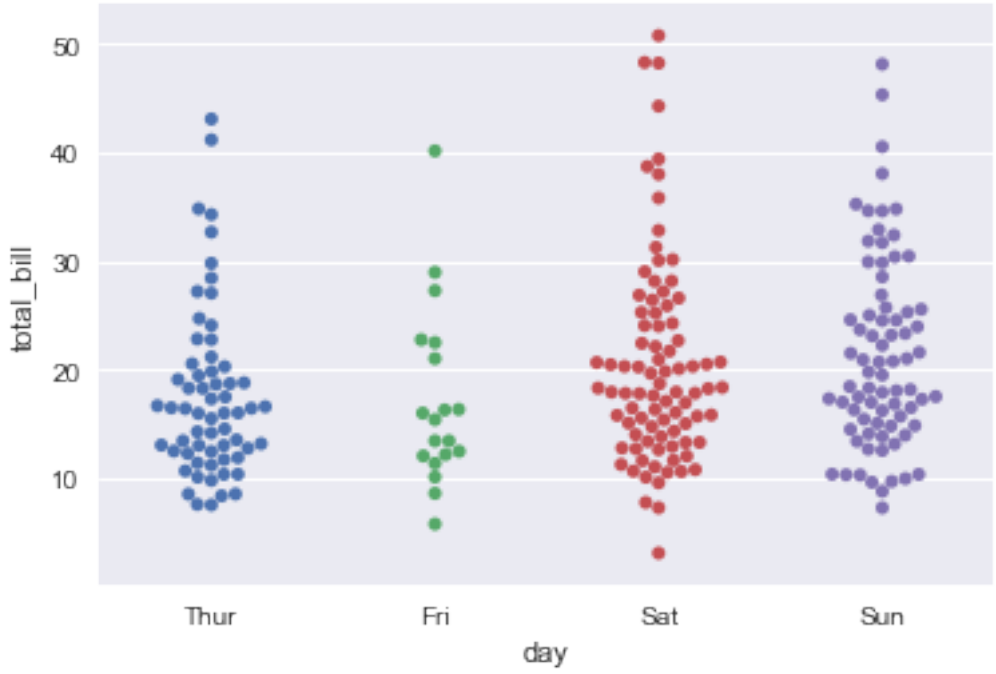
[16]:
```
sns.stripplot(x="day", y="total_bill",␣
↪data=tips,jitter=True,hue='sex',palette='Set1',split=True)
```

[16]: <matplotlib.axes._subplots.AxesSubplot at 0x16eca19eb00>



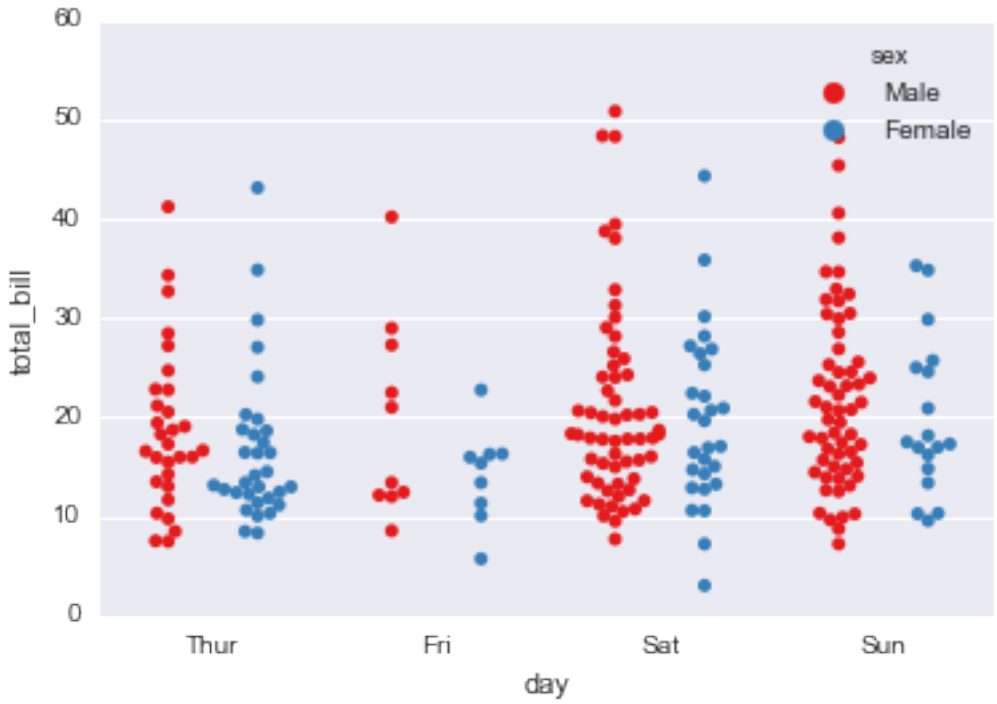
[17]:
```
sns.swarmplot(x="day", y="total_bill", data=tips)
```

[17]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x16eca38beb8>
```



[47]:
```
sns.swarmplot(x="day", y="total_bill",hue='sex',data=tips, palette="Set1",␣
↪split=True)
```

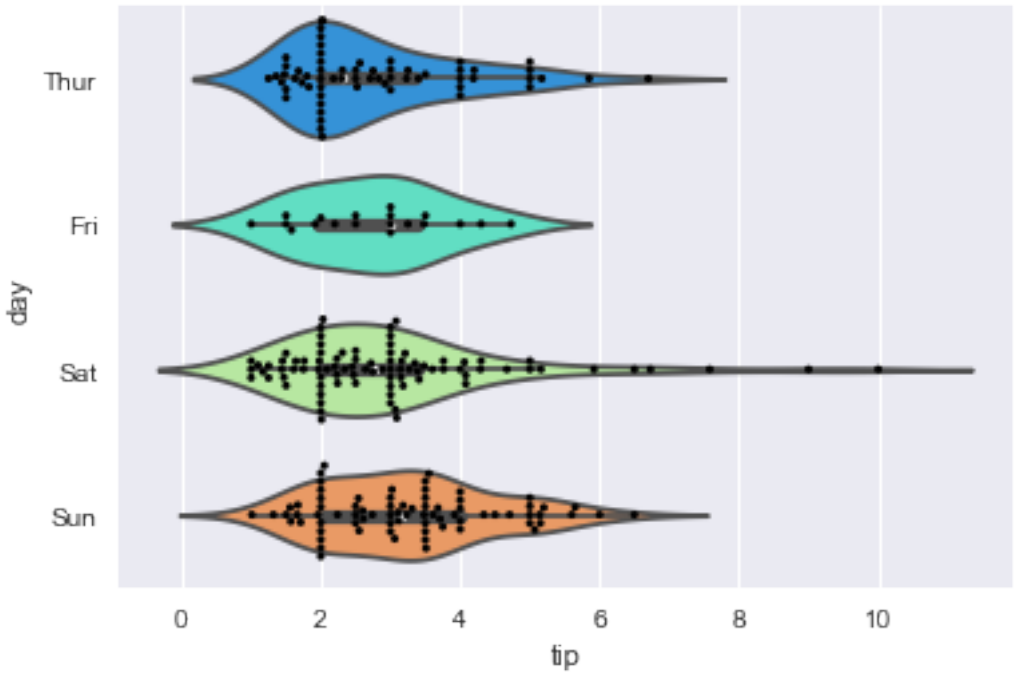
[47]: <matplotlib.axes._subplots.AxesSubplot at 0x1211b6da0>



1.3.1 Combinando plots categóricos

[18]:
```
sns.violinplot(x="tip", y="day", data=tips,palette='rainbow')
sns.swarmplot(x="tip", y="day", data=tips,color='black',size=3)
```

[18]: <matplotlib.axes._subplots.AxesSubplot at 0x16eca0e7240>

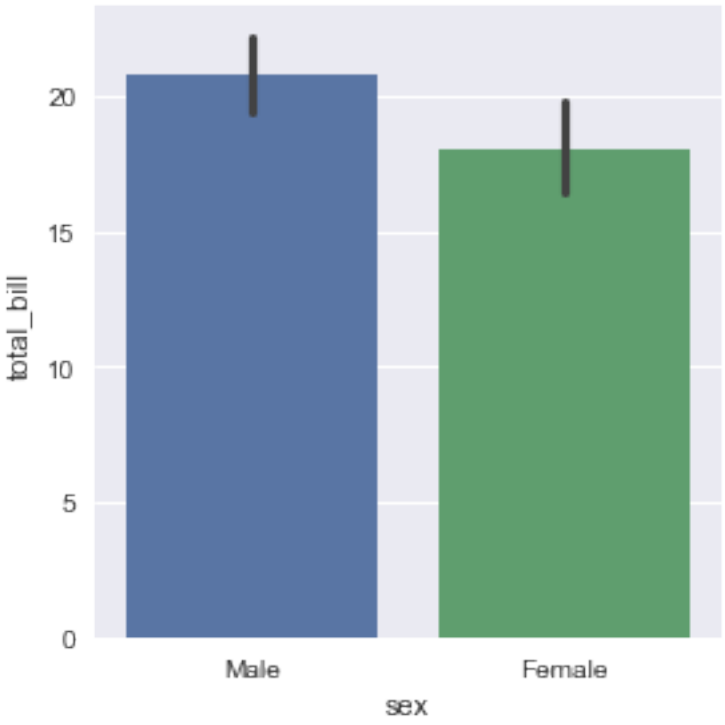


1.4 factorplot

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

[19]:
```
sns.factorplot(x='sex',y='total_bill',data=tips,kind='bar')
```

[19]:
```
<seaborn.axisgrid.FacetGrid at 0x16eca52e240>
```


20
15
10
5
0
total_bill
Male
Female
sex

3-plots-de-regressoes

September 11, 2023

1 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:

[1]:
```
import seaborn as sns
%matplotlib inline
```

[2]:
```
tips = sns.load_dataset('tips')
```

[3]:
```
tips.head()
```

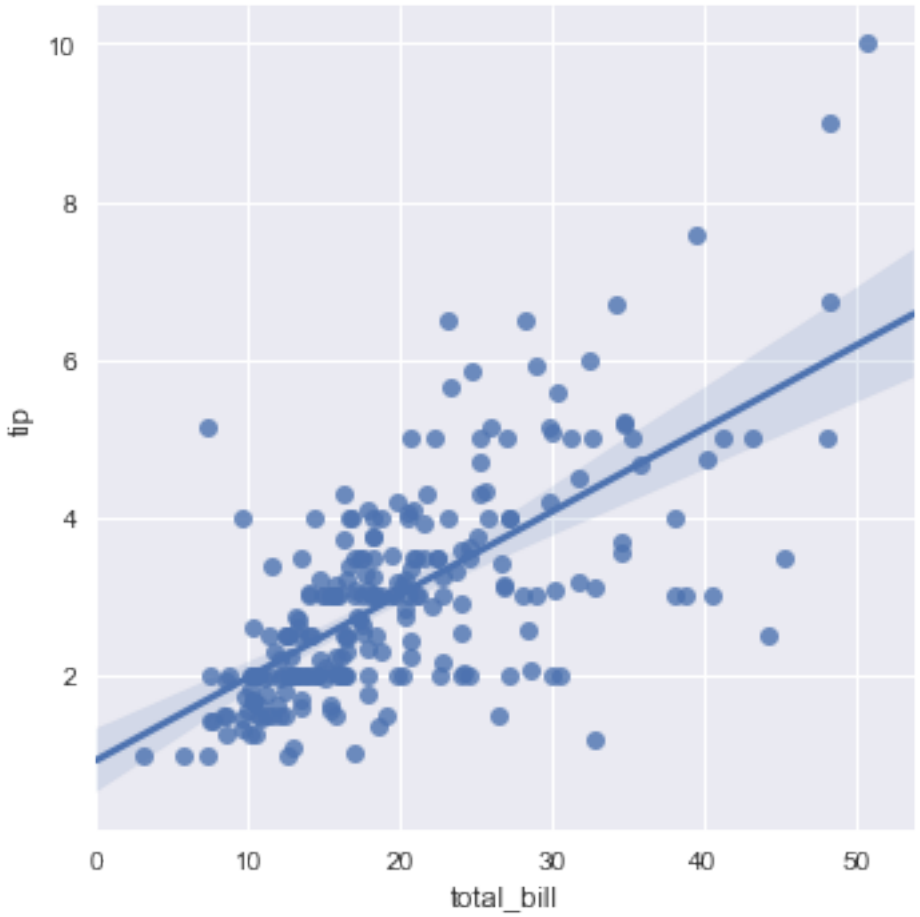
[3]:

	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

1.1 lmplot()

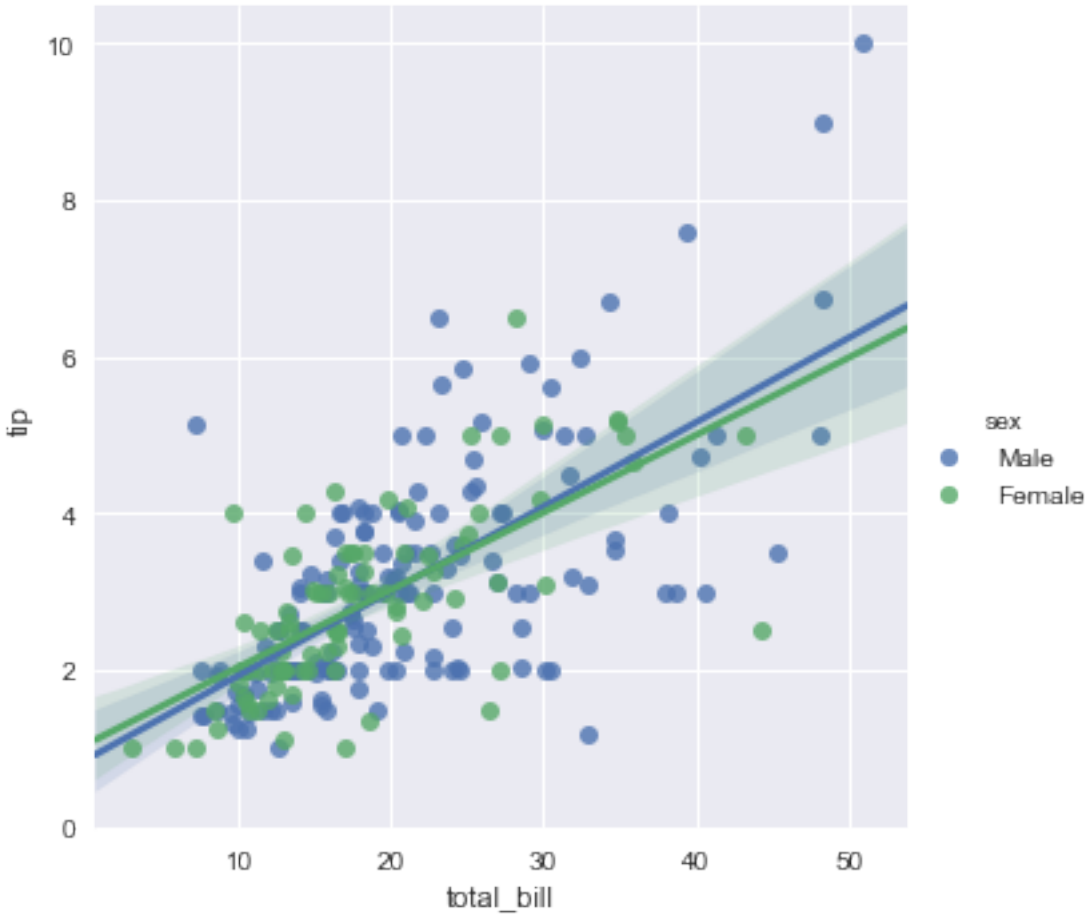
[4]:
```
sns.lmplot(x='total_bill',y='tip',data=tips)
```

[4]:
```
<seaborn.axisgrid.FacetGrid at 0x1ea69106b70>
```



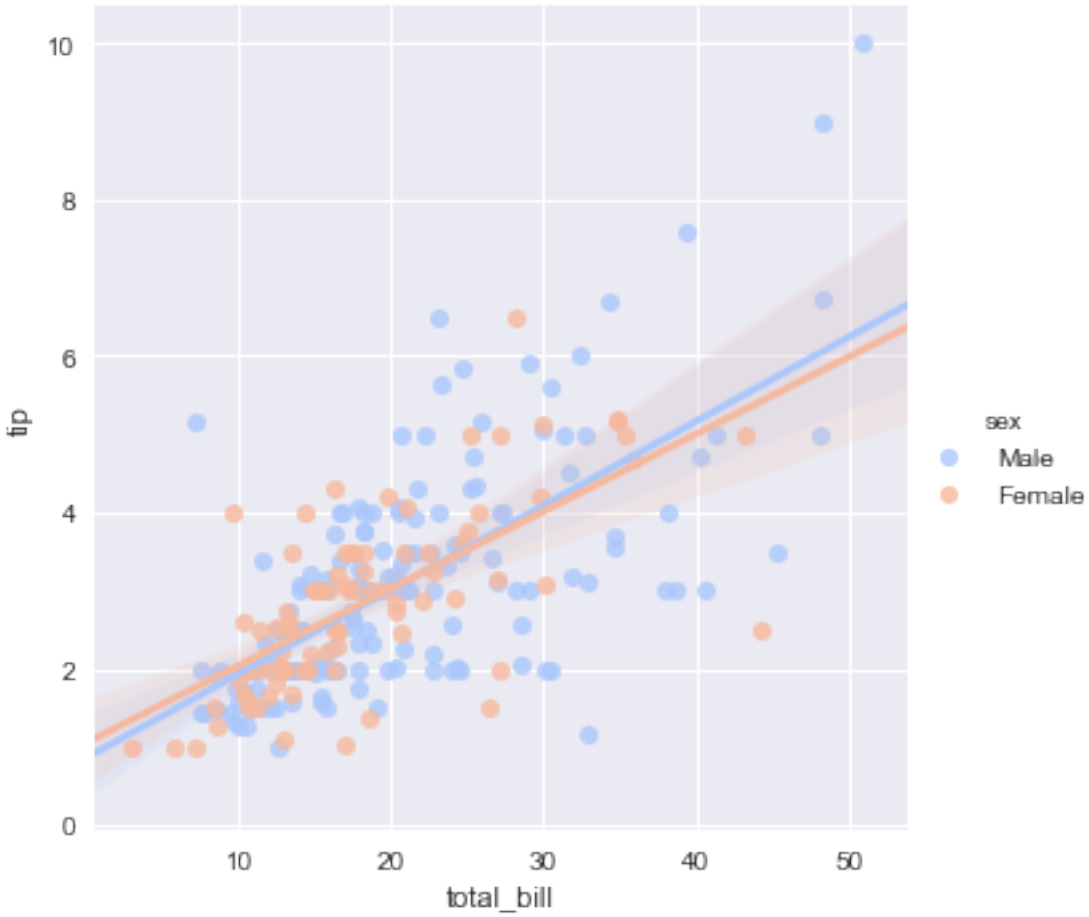
[5]:
```
sns.lmplot(x='total_bill',y='tip',data=tips,hue='sex')
```

[5]:
```
<seaborn.axisgrid.FacetGrid at 0x1ea68c74c88>
```



[6]:
```
sns.lmplot(x='total_bill',y='tip',data=tips,hue='sex',palette='coolwarm')
```

[6]: <seaborn.axisgrid.FacetGrid at 0x1ea69442128>

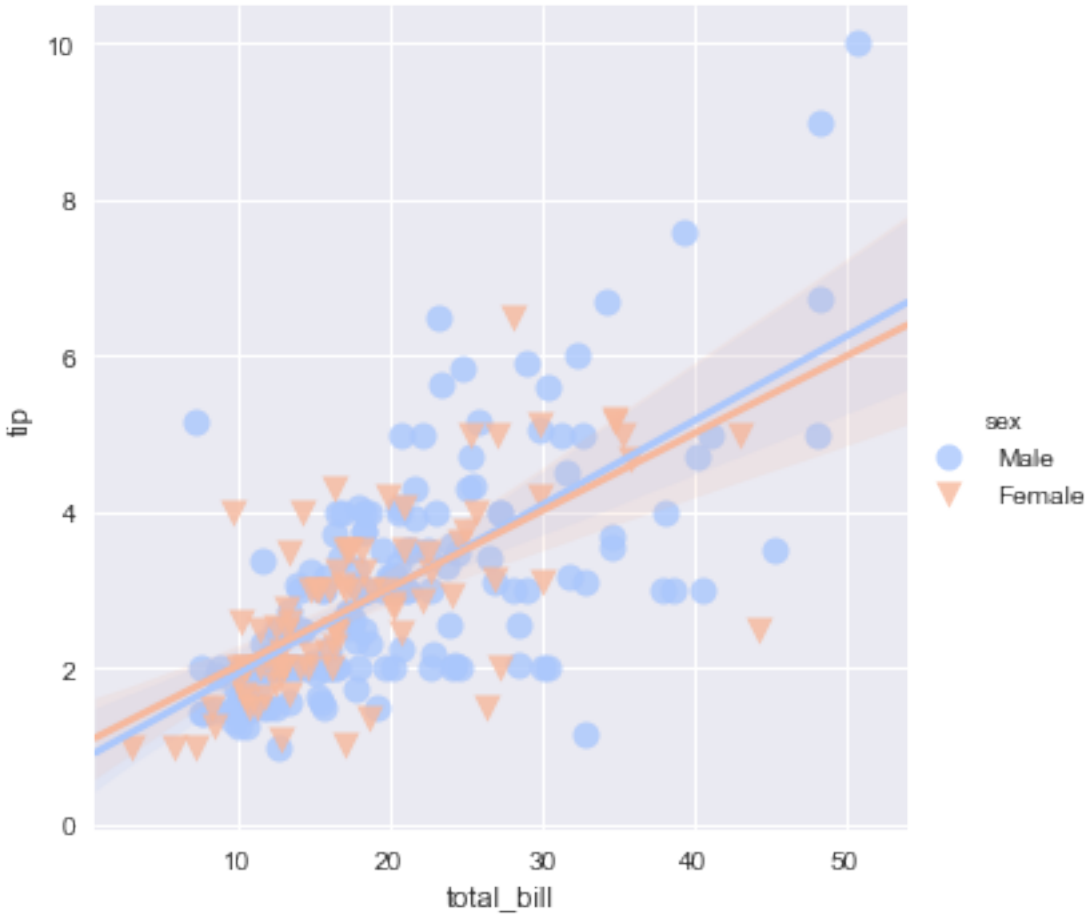


1.1.1 Trabalhando com marcadores

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.

[7]:
```
# http://matplotlib.org/api/markers_api.html
sns.lmplot(x='total_bill',y='tip',data=tips,hue='sex',palette='coolwarm',
           markers=['o','v'],scatter_kws={'s':100})
```

[7]:
```
<seaborn.axisgrid.FacetGrid at 0x1ea692d84a8>
```

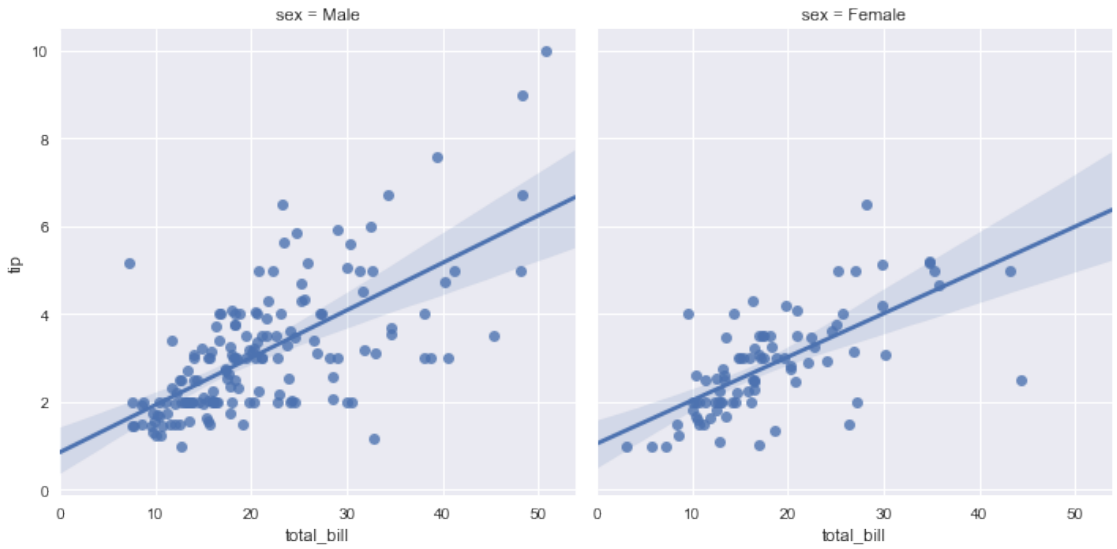


1.2 Usando grades

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:

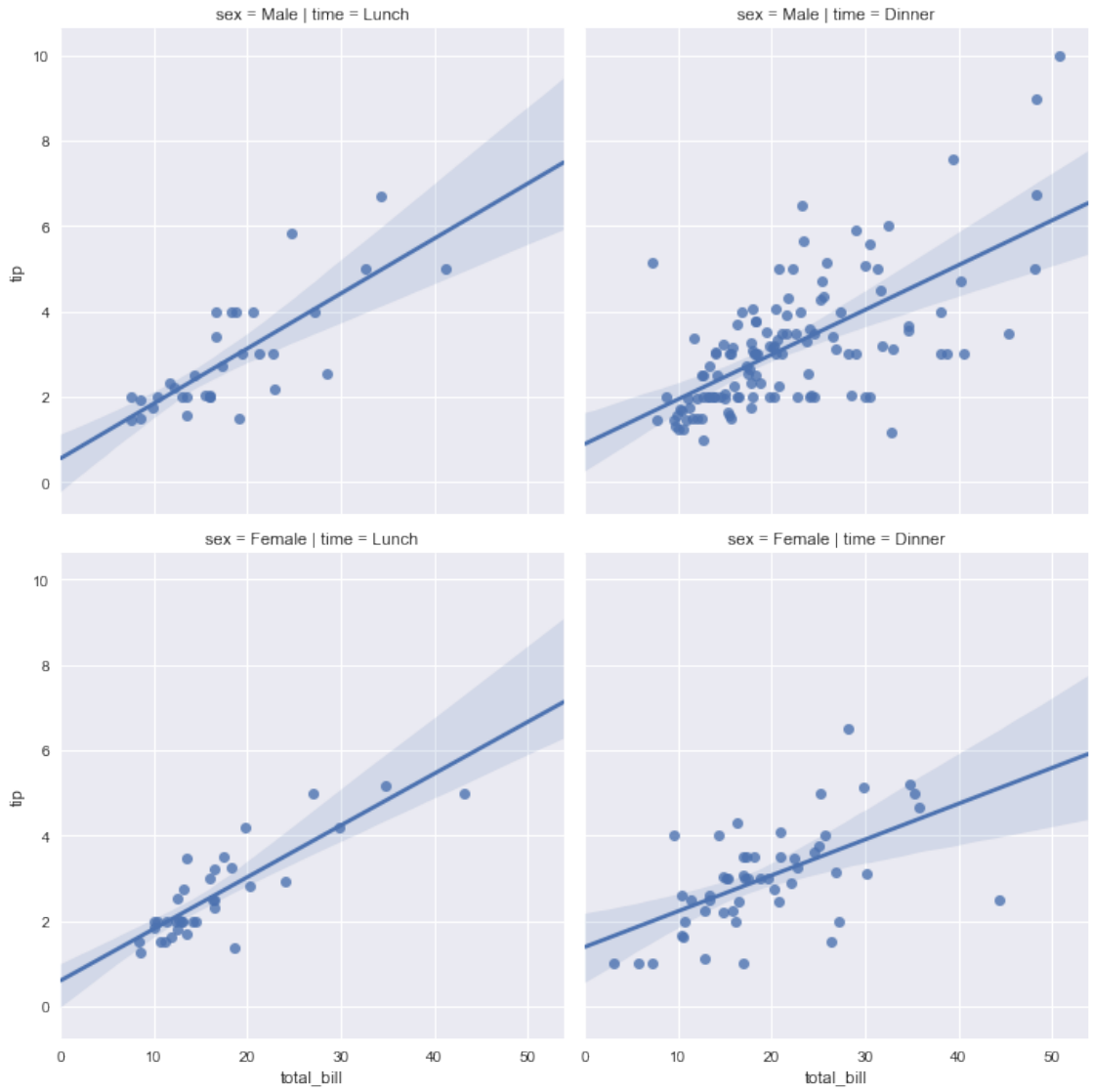
[8]:
```
sns.lmplot(x='total_bill',y='tip',data=tips,col='sex')
```

[8]:
```
<seaborn.axisgrid.FacetGrid at 0x1ea6a95d0f0>
```



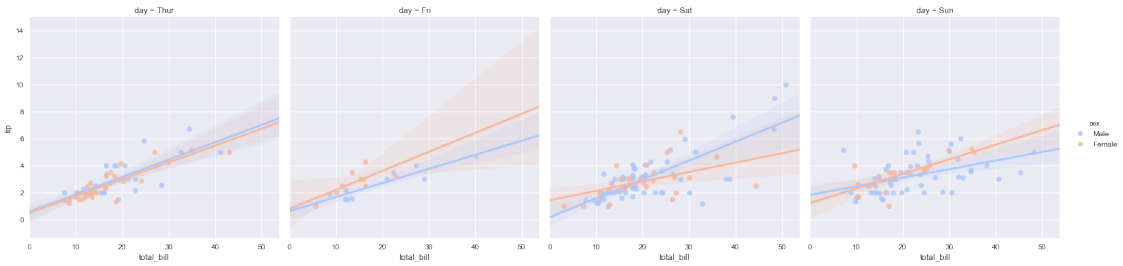
[9]:
```
sns.lmplot(x="total_bill", y="tip", row="sex", col="time",data=tips)
```

[9]: <seaborn.axisgrid.FacetGrid at 0x1ea6aac3ba8>



[10]:
```
sns.
↪lmplot(x='total_bill',y='tip',data=tips,col='day',hue='sex',palette='coolwarm')
```

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

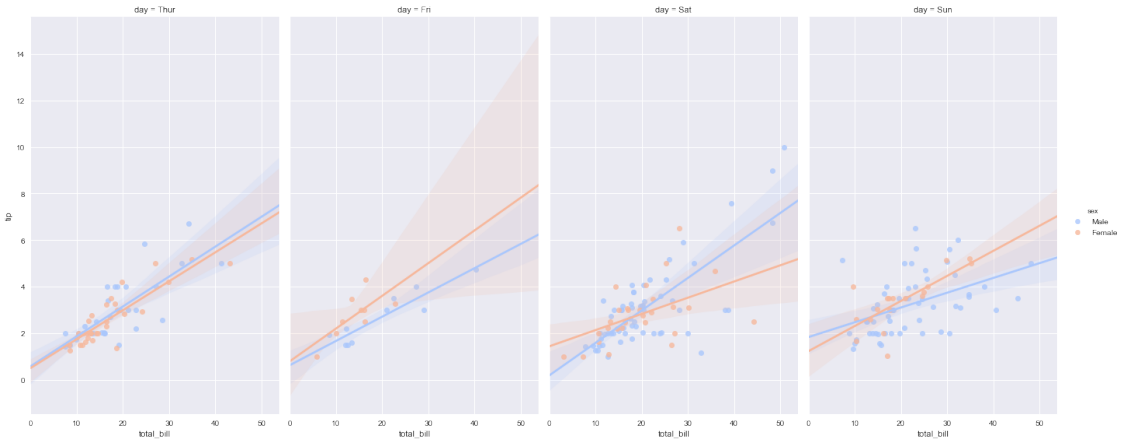


1.3 Aspecto e tamanho

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

[11]:
```
sns.
↪lmplot(x='total_bill',y='tip',data=tips,col='day',hue='sex',palette='coolwarm',
        aspect=0.6,size=8)
```

[11]:
```
<seaborn.axisgrid.FacetGrid at 0x1ea6af5b0f0>
```



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

4-plots-matriciais

September 11, 2023

1 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:

[1]:
```
import seaborn as sns
%matplotlib inline
```

[2]:
```
flights = sns.load_dataset('flights')
```

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

[4]:
```
tips.head()
```

[4]:

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

[5]:

	year	month	passengers
0	1949	January	112
1	1949	February	118
2	1949	March	132
3	1949	April	129
4	1949	May	121

1.1 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:

[6]:
```
tips.head()
```

[6]:

	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

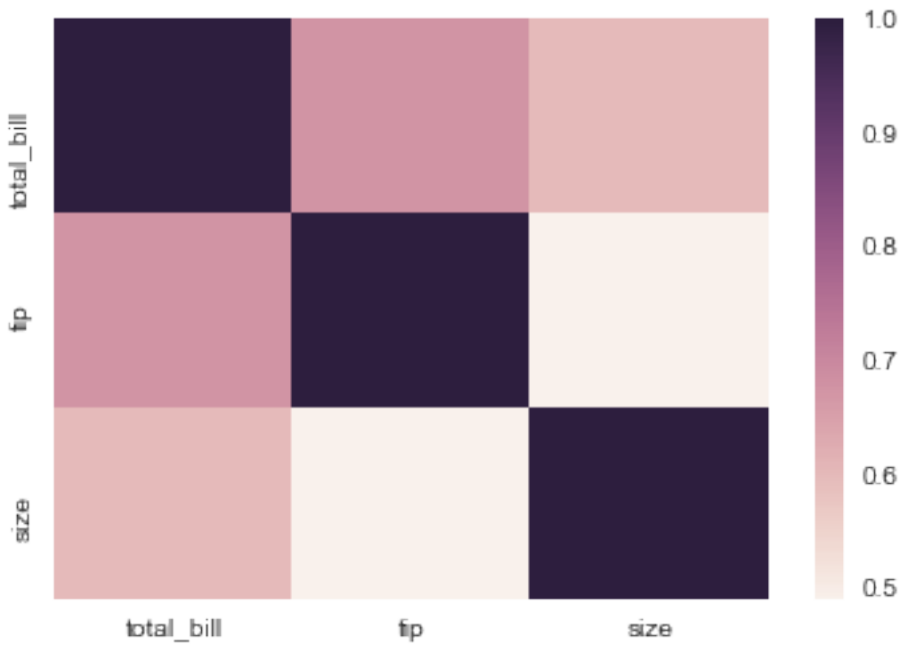
[7]:
```
# Correlograma
tips.corr()
```

[7]:

	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

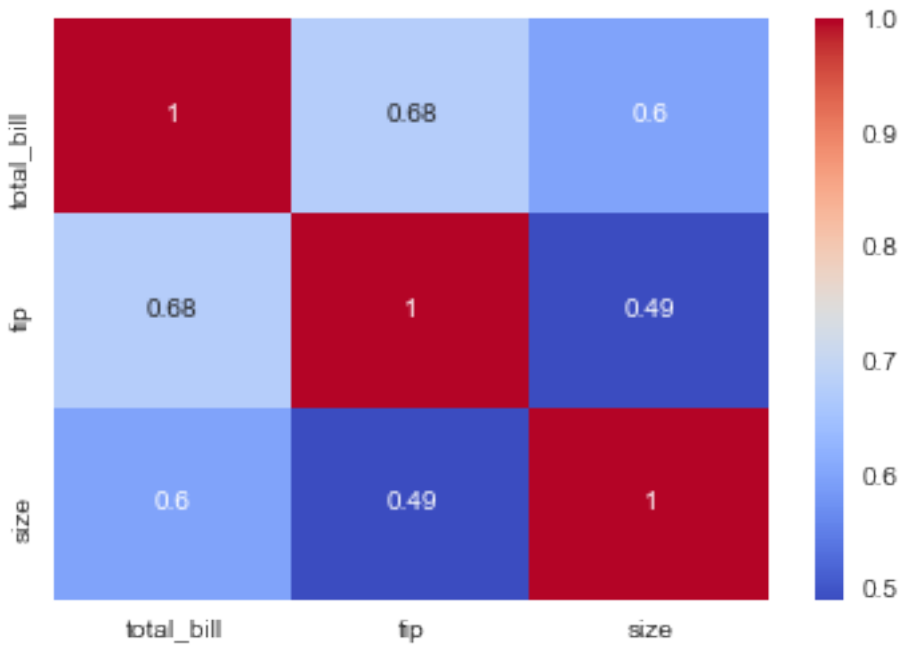
[8]:
```
sns.heatmap(tips.corr())
```

[8]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1ed137966a0>
```



[9]:
```
sns.heatmap(tips.corr(),cmap='coolwarm',annot=True)
```

[9]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1ed135556d8>
```



Ou para os dados dos vôos:

[10]:
```
flights.pivot_table(values='passengers',index='month',columns='year')
```

[10]:
```
year        1949  1950  1951  1952  1953  1954  1955  1956  1957  1958  1959  \
month
January      112   115   145   171   196   204   242   284   315   340   360
February     118   126   150   180   196   188   233   277   301   318   342
March        132   141   178   193   236   235   267   317   356   362   406
April        129   135   163   181   235   227   269   313   348   348   396
May          121   125   172   183   229   234   270   318   355   363   420
June         135   149   178   218   243   264   315   374   422   435   472
July         148   170   199   230   264   302   364   413   465   491   548
August       148   170   199   242   272   293   347   405   467   505   559
September    136   158   184   209   237   259   312   355   404   404   463
October      119   133   162   191   211   229   274   306   347   359   407
November     104   114   146   172   180   203   237   271   305   310   362
December     118   140   166   194   201   229   278   306   336   337   405

year        1960
month
January      417
February     391
March        419
April        461
May          472
```

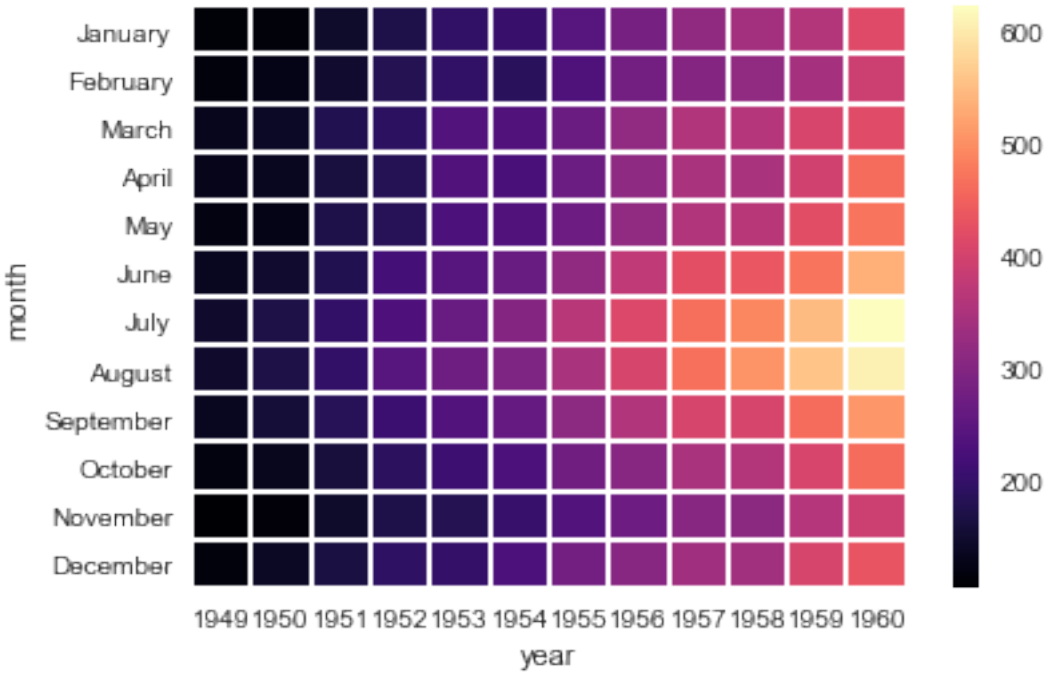
```
June        535
July        622
August      606
September   508
October     461
November    390
December    432
```

[11]:
```
pvflights = flights.
  ↪pivot_table(values='passengers',index='month',columns='year')
sns.heatmap(pvflights)
```

[11]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1ed13d0ac88>
```

[12]:
```
sns.heatmap(pvflights,cmap='magma',linecolor='white',linewidths=1)
```

[12]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1ed15083c88>
```



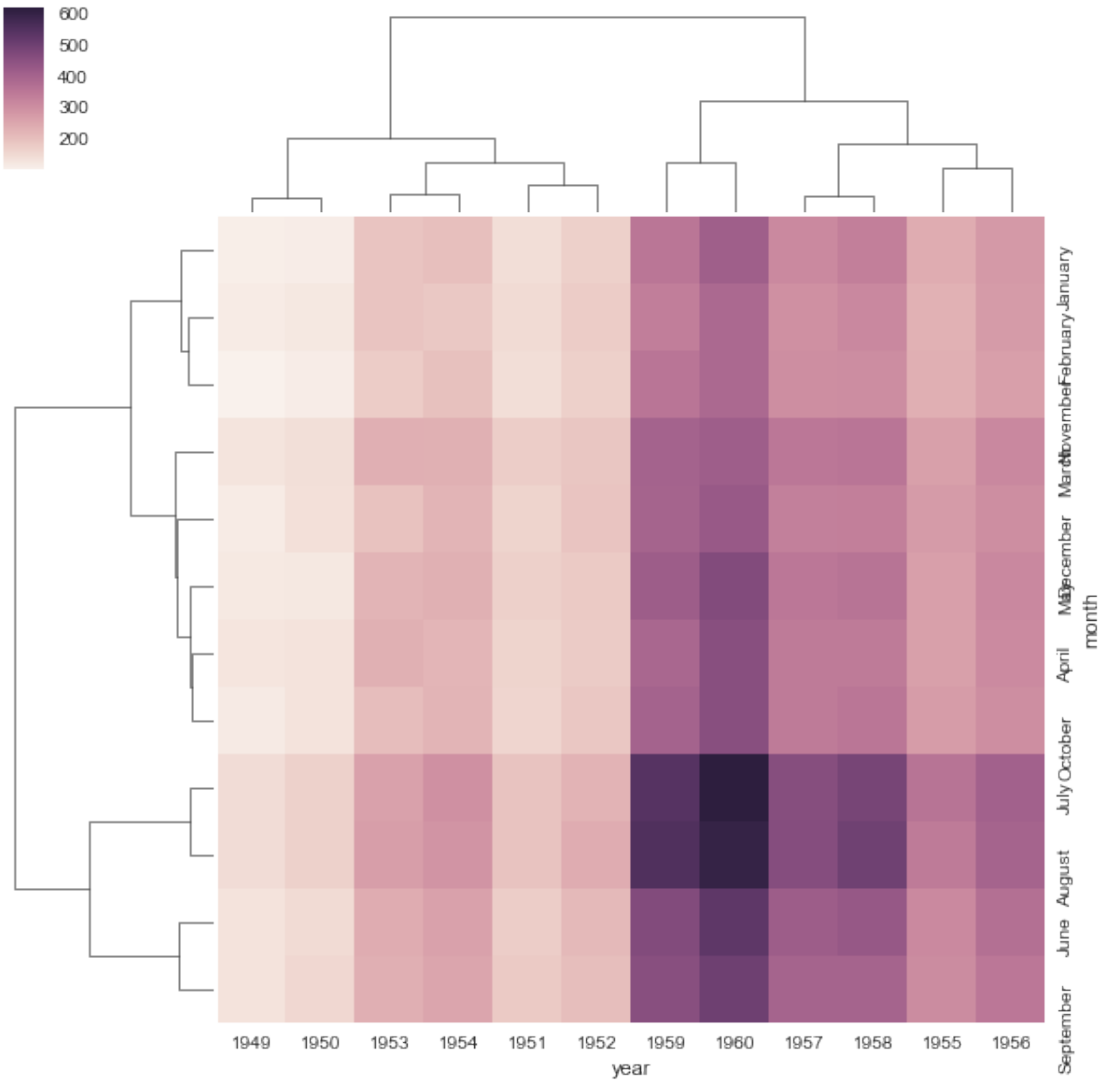
1.2 clustermap

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

[13]:
```
sns.clustermap(pvflights, 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)
```

[13]:
```
<seaborn.matrix.ClusterGrid at 0x1ed13d0f668>
```

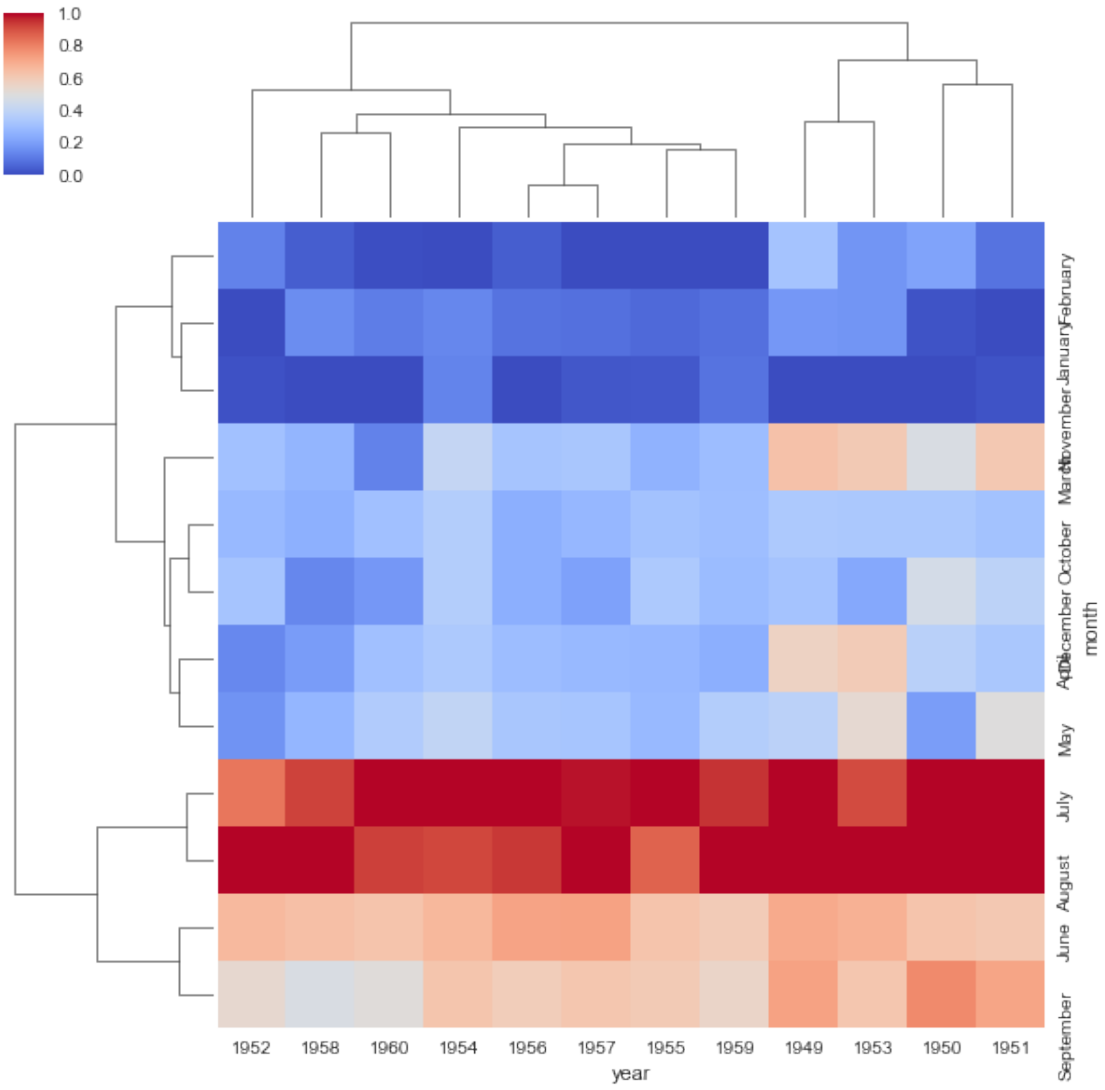


Observe agora como os anos e os meses não estão mais em ordem, em vez disso, eles são agrupados por similaridade em valor (contagem de passageiros). Isso significa que podemos começar a inferir coisas desse plot, como agosto e julho sendo semelhantes (faz sentido, uma vez que são ambos os meses de viagem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

[14]:
```
# 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)
```

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


1.0
0.8
0.6
0.4
0.2
0.0
1952
1958
1960
1954
1956
1957
1955
1959
1949
1953
1950
1951
year
February
January
November
March
October
December
April
May
July
August
June
September
month

5-estilos-e-cores

September 11, 2023

1 Estilos e cores

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

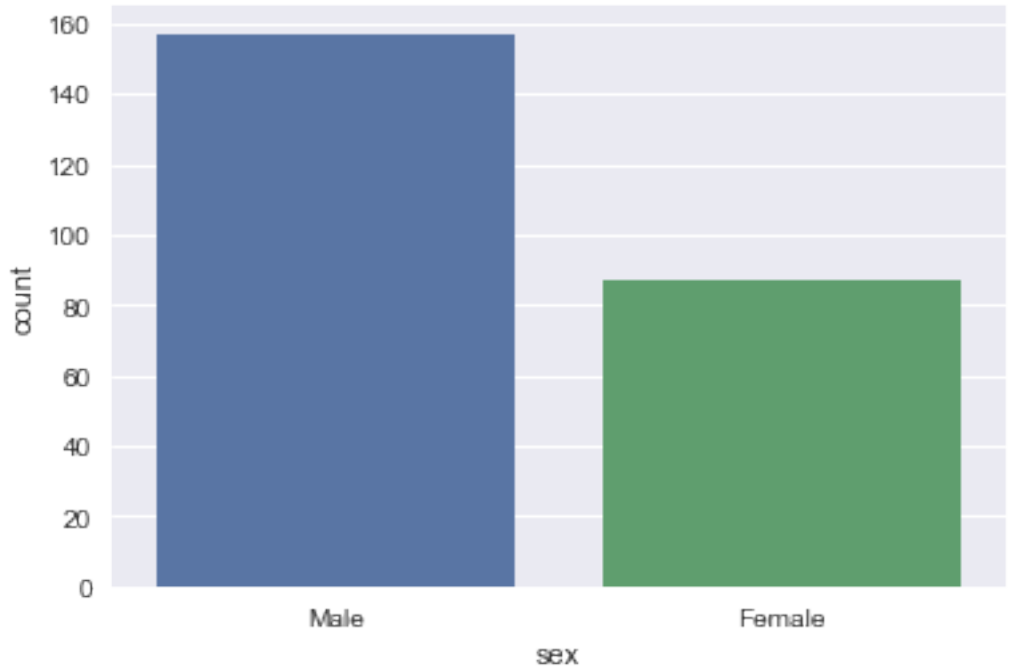
[1]:
```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
tips = sns.load_dataset('tips')
```

1.1 Styles

Você pode definir um estilo específico.

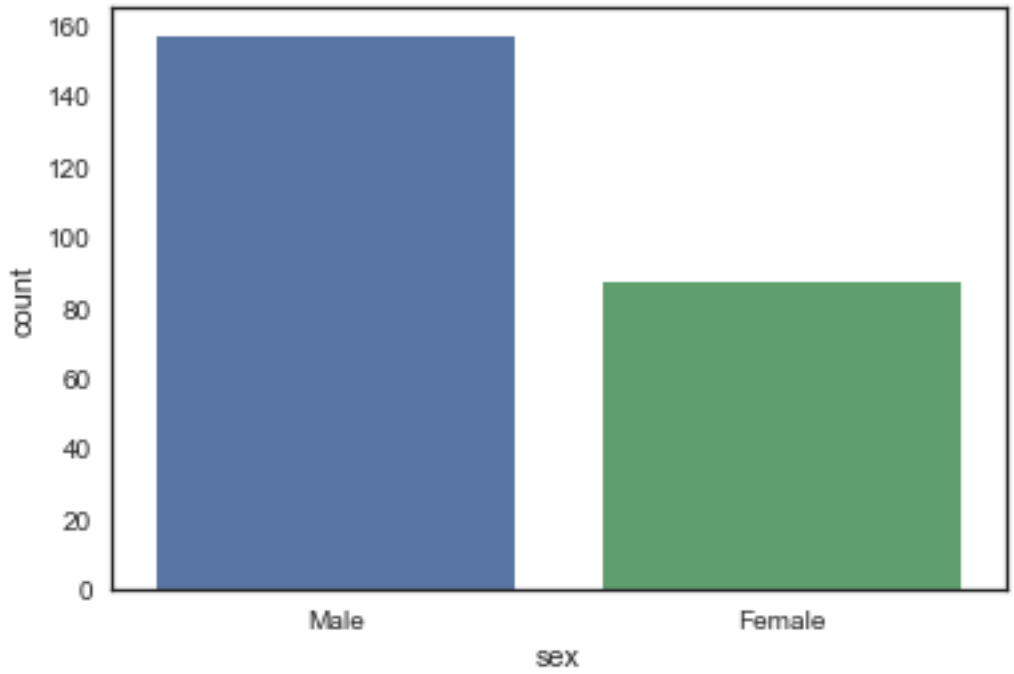
[2]:
```
sns.countplot(x='sex',data=tips)
```

[2]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1f6de000a58>
```



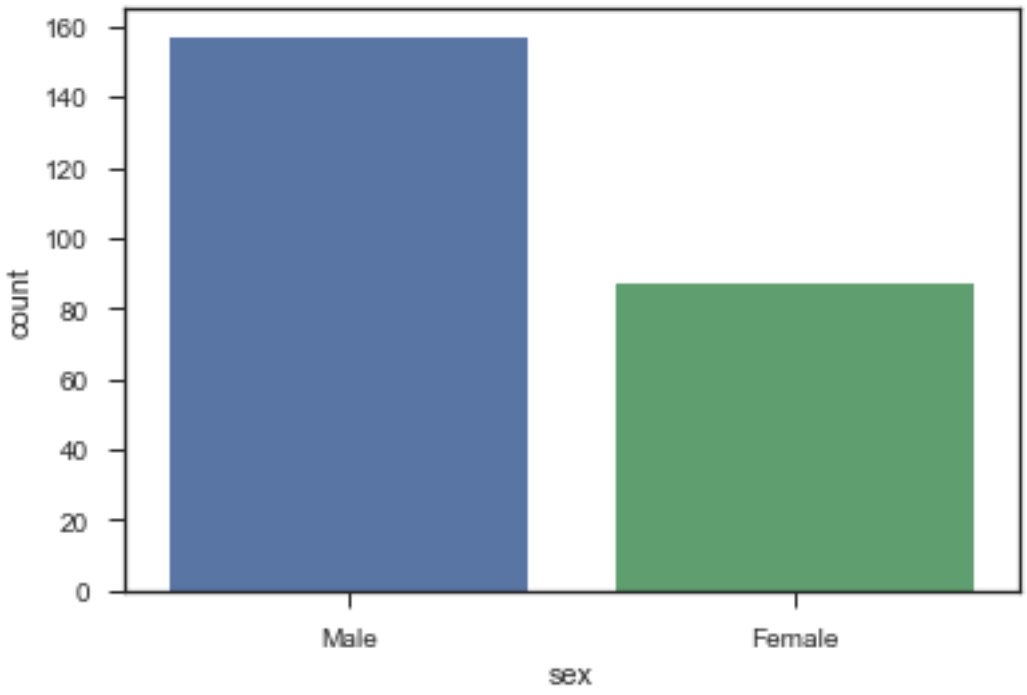
[3]:
```
sns.set_style('white')
sns.countplot(x='sex',data=tips)
```

[3]: <matplotlib.axes._subplots.AxesSubplot at 0x1f6dde90898>



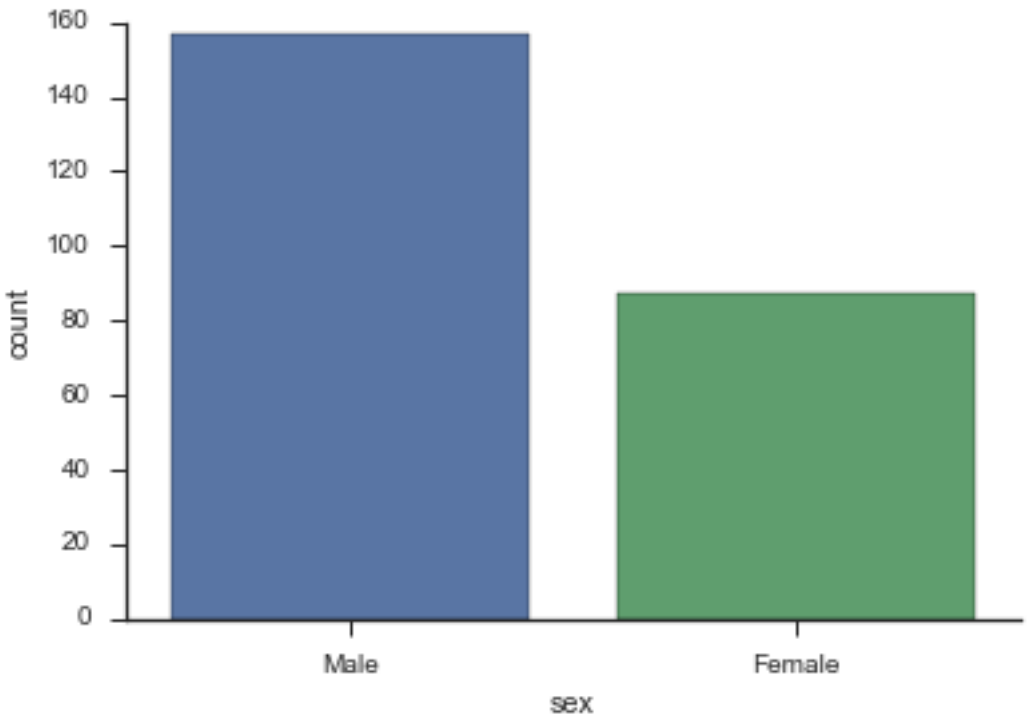
[4]:
```
sns.set_style('ticks')
sns.countplot(x='sex',data=tips,palette='deep')
```

[4]: <matplotlib.axes._subplots.AxesSubplot at 0x1f6de10c1d0>

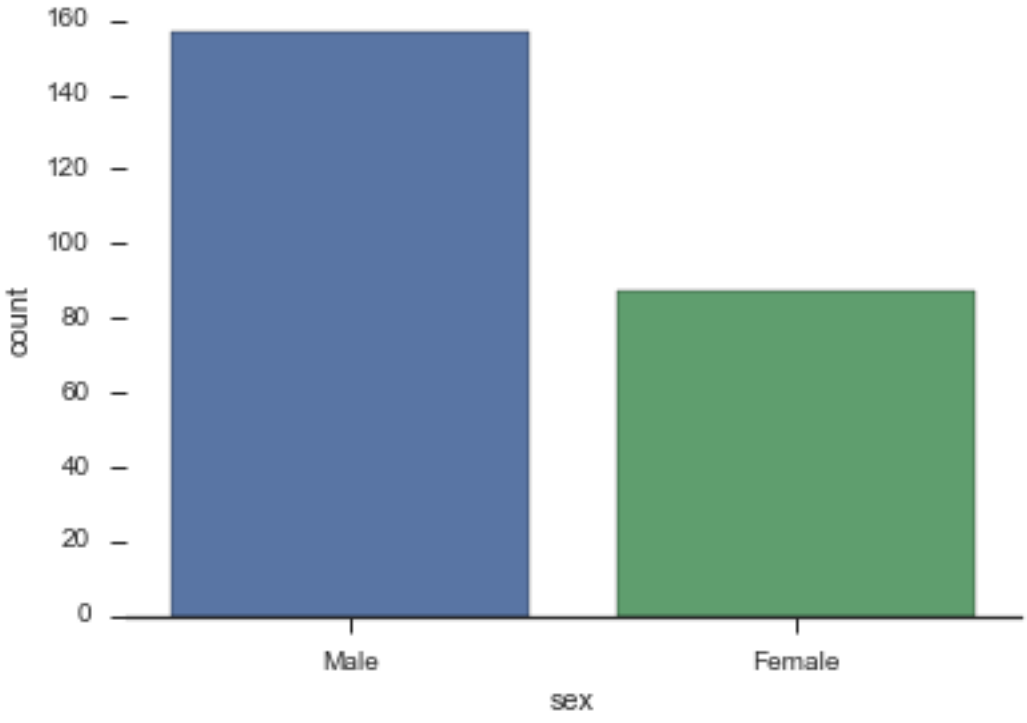


1.2 Remoção dos limites

[5]:
```
sns.countplot(x='sex',data=tips)
sns.despine()
```



[6]:
```
sns.countplot(x='sex',data=tips)
sns.despine(left=True)
```



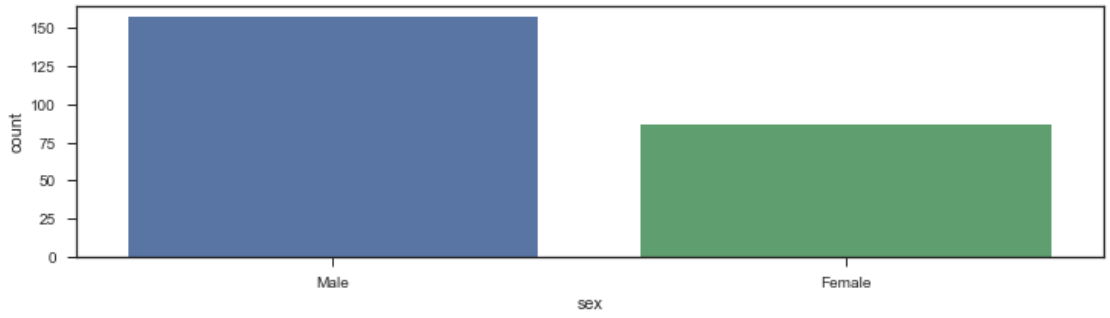
1.3 Tamanho e aspecto

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:

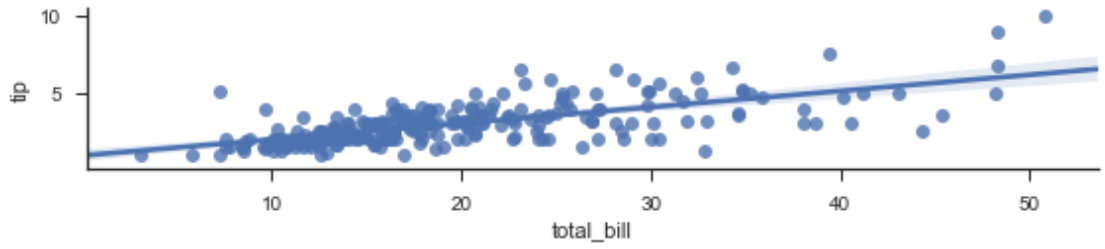
[5]:
```
# Plot não gradeado
plt.figure(figsize=(12,3))
sns.countplot(x='sex',data=tips)
```

[5]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1f6de473208>
```



[6]:
```
# Plot tipo grade
sns.lmplot(x='total_bill',y='tip',size=2,aspect=4,data=tips)
```

[6]:
```
<seaborn.axisgrid.FacetGrid at 0x1f6de49e6d8>
```

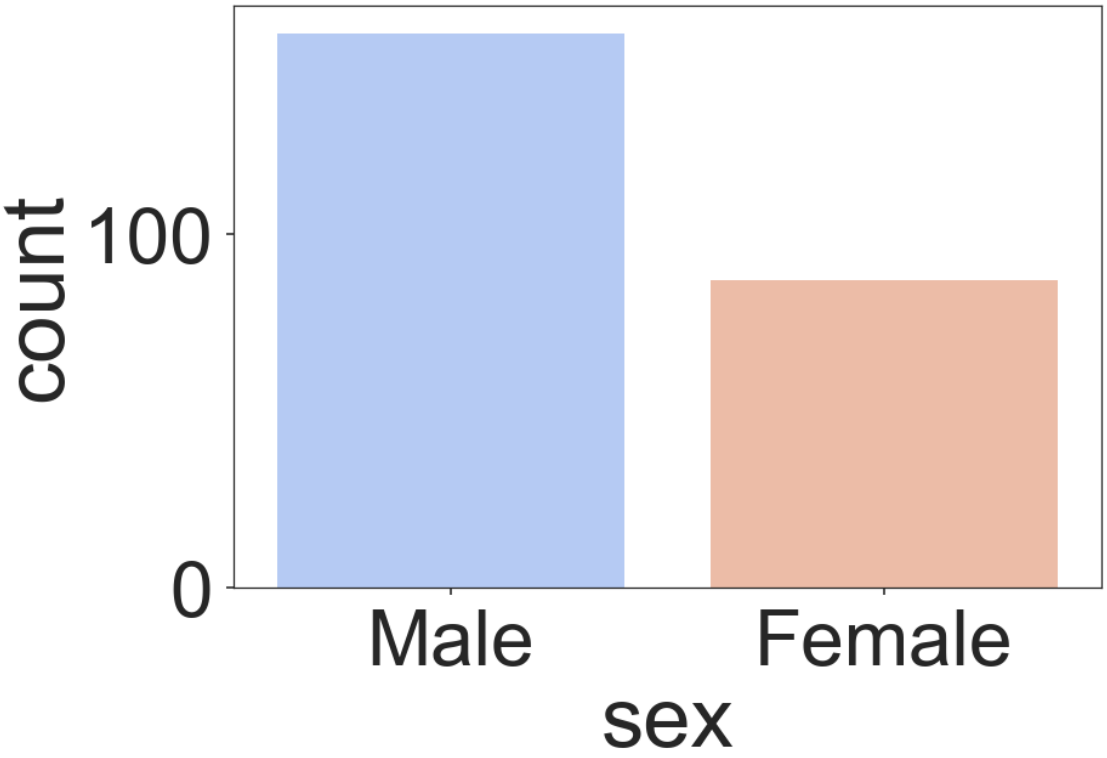


1.4 Escala e Contexto

O set_context () permite que você substitua parâmetros padrão:

[7]:
```
sns.set_context('poster',font_scale=4)
sns.countplot(x='sex',data=tips,palette='coolwarm')
```

[7]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1f6de58a0b8>
```



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

[8]:
```
sns.puppyplot()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\bs4\__init__.py:181: UserWarning: No
parser was explicitly specified, so I'm using the best available HTML parser for
this system ("lxml"). This usually isn't a problem, but if you run this code on
another system, or in a different virtual environment, it may use a different
parser and behave differently.

The code that caused this warning is on line 193 of the file
C:\ProgramData\Anaconda3\lib\runpy.py. To get rid of this warning, change code
that looks like this:

 BeautifulSoup(YOUR_MARKUP})

to this:
```

```
      BeautifulSoup(YOUR_MARKUP, "lxml")

       markup_type=markup_type))
```

[8]:
```
<IPython.core.display.HTML object>
```

6-pairgrid

September 11, 2023

1 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.

[1]:
```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

[2]:
```
iris = sns.load_dataset('iris')
```

[3]:
```
iris.head()
```

[3]:

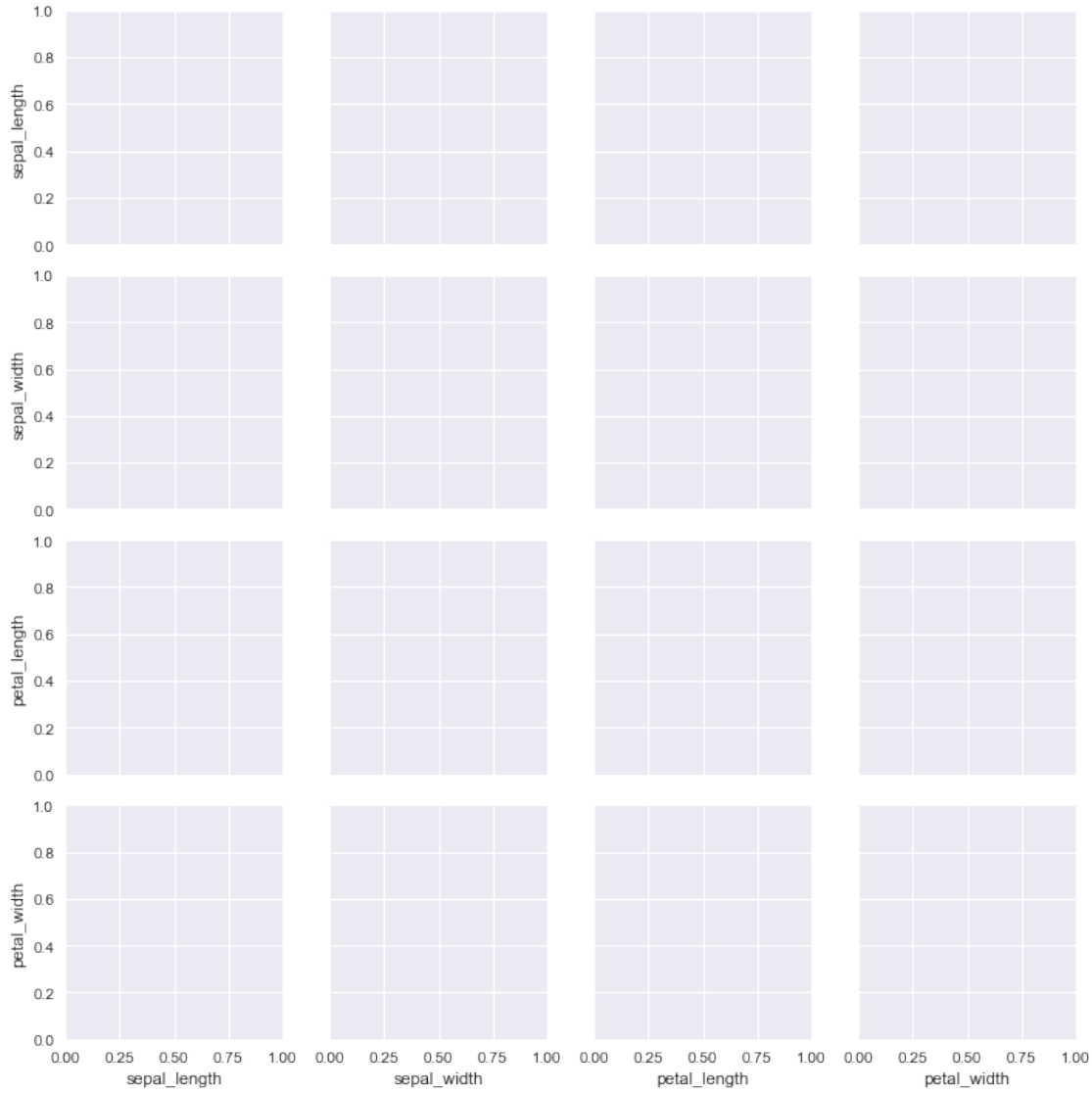
	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

1.1 PairGrid

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

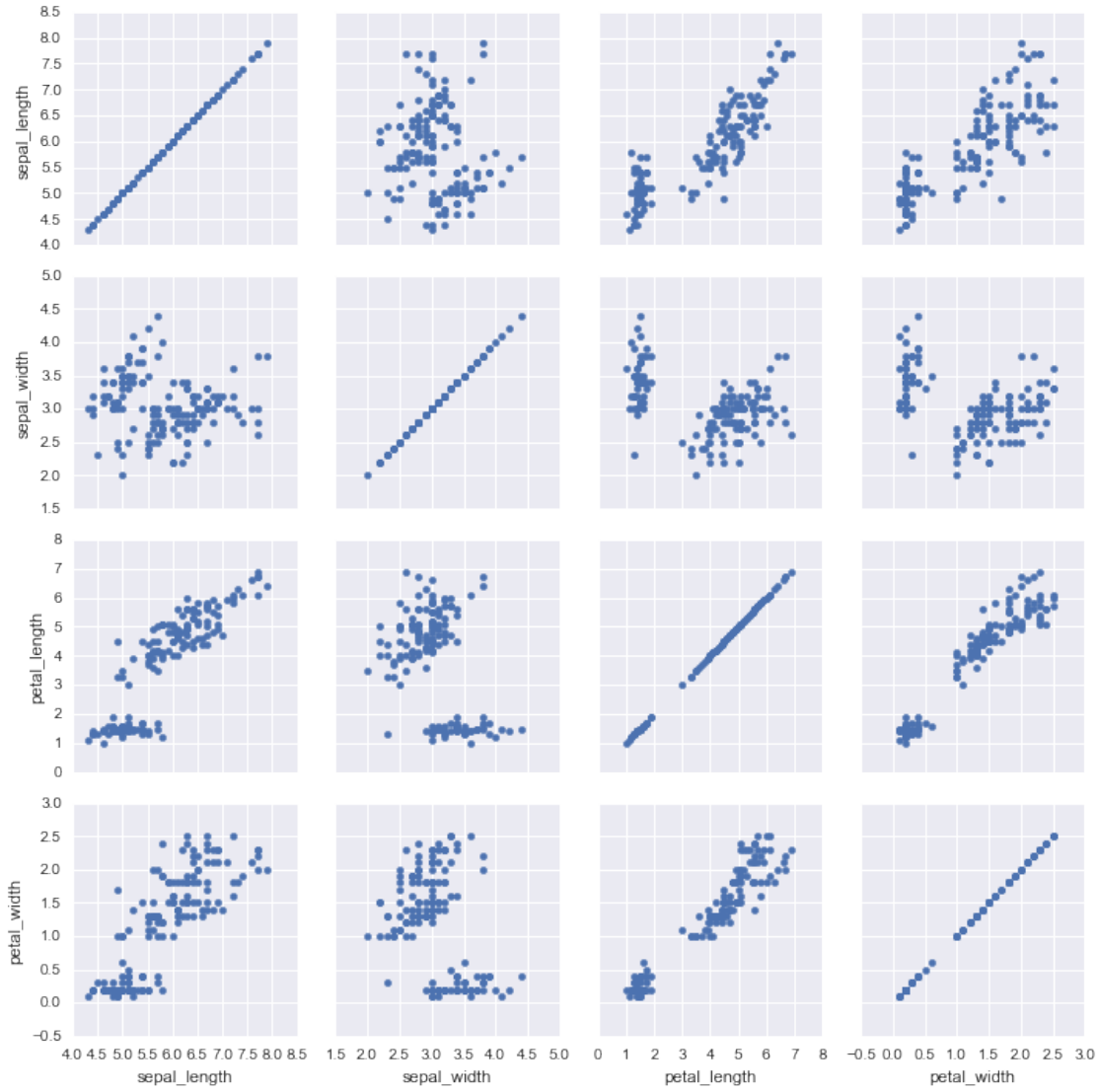
[11]:
```
# Just the Grid
sns.PairGrid(iris)
```

[11]:
```
<seaborn.axisgrid.PairGrid at 0x25b9a0927f0>
```



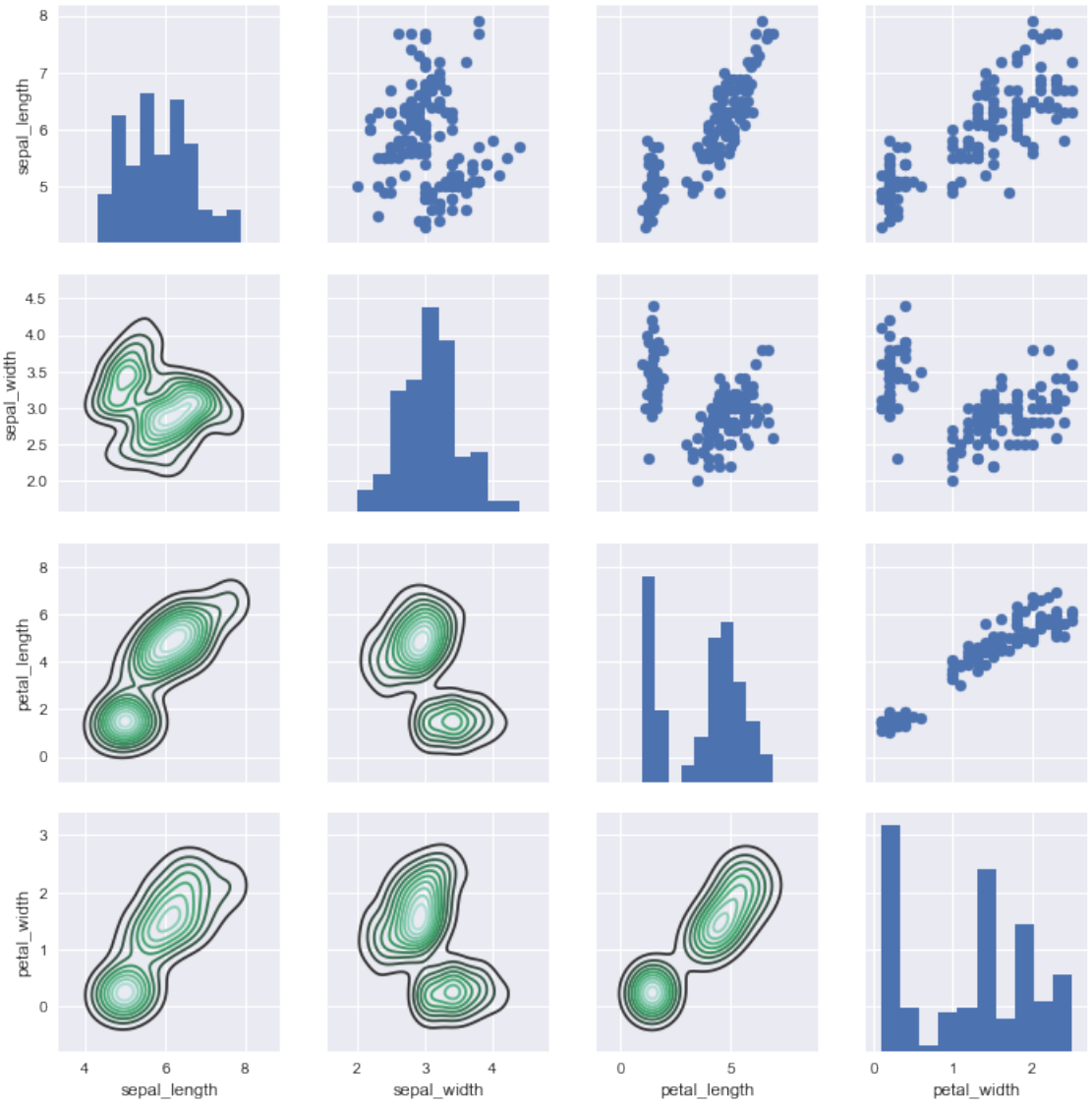
[26]:
```
# Then you map to the grid
g = sns.PairGrid(iris)
g.map(plt.scatter)
```

[26]: <seaborn.axisgrid.PairGrid at 0x11f431208>



[8]:
```
# 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)
```

[8]: <seaborn.axisgrid.PairGrid at 0x25b96a1eeb8>

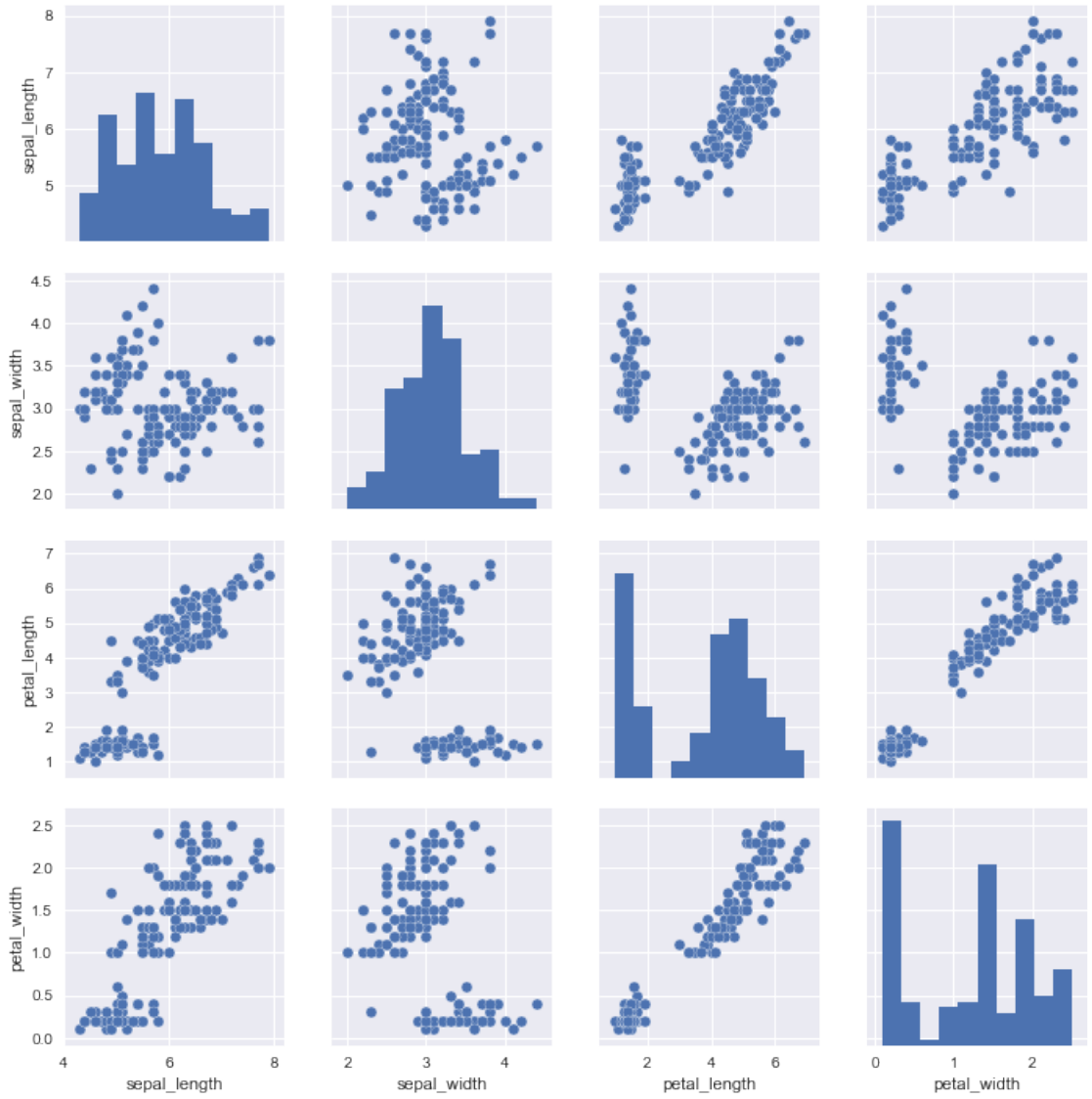


1.2 pairplot

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

[12]:
```
sns.pairplot(iris)
```

[12]:
```
<seaborn.axisgrid.PairGrid at 0x25b9a4c6128>
```



[13]:
```
sns.pairplot(iris,hue='species',palette='rainbow')
```

[13]:
```
<seaborn.axisgrid.PairGrid at 0x25b9b9a59b0>
```

1.3 FacetGrid

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

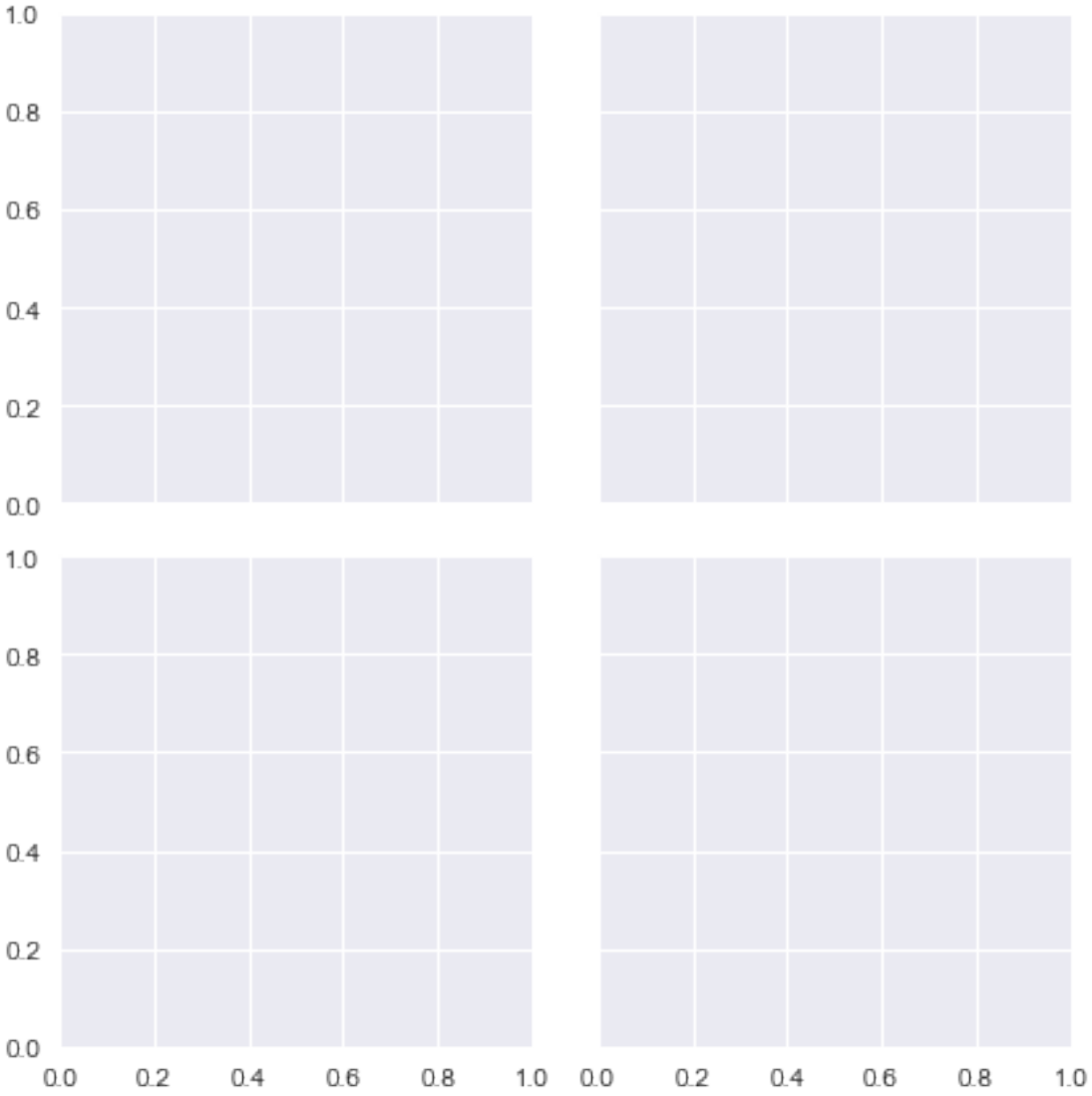
[14]:
```
tips = sns.load_dataset('tips')
```

[15]:
```
tips.head()
```

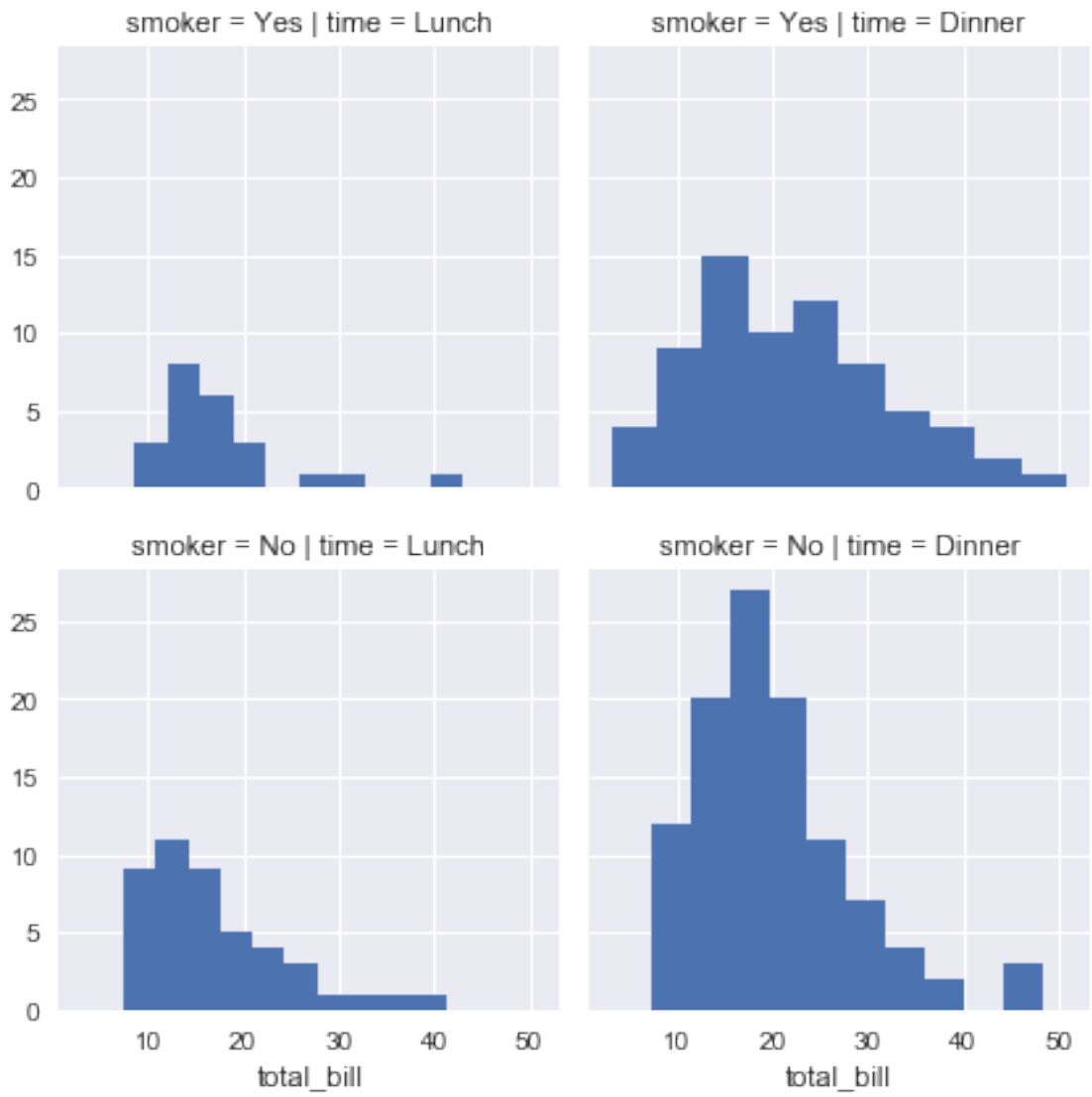
[15]:

	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

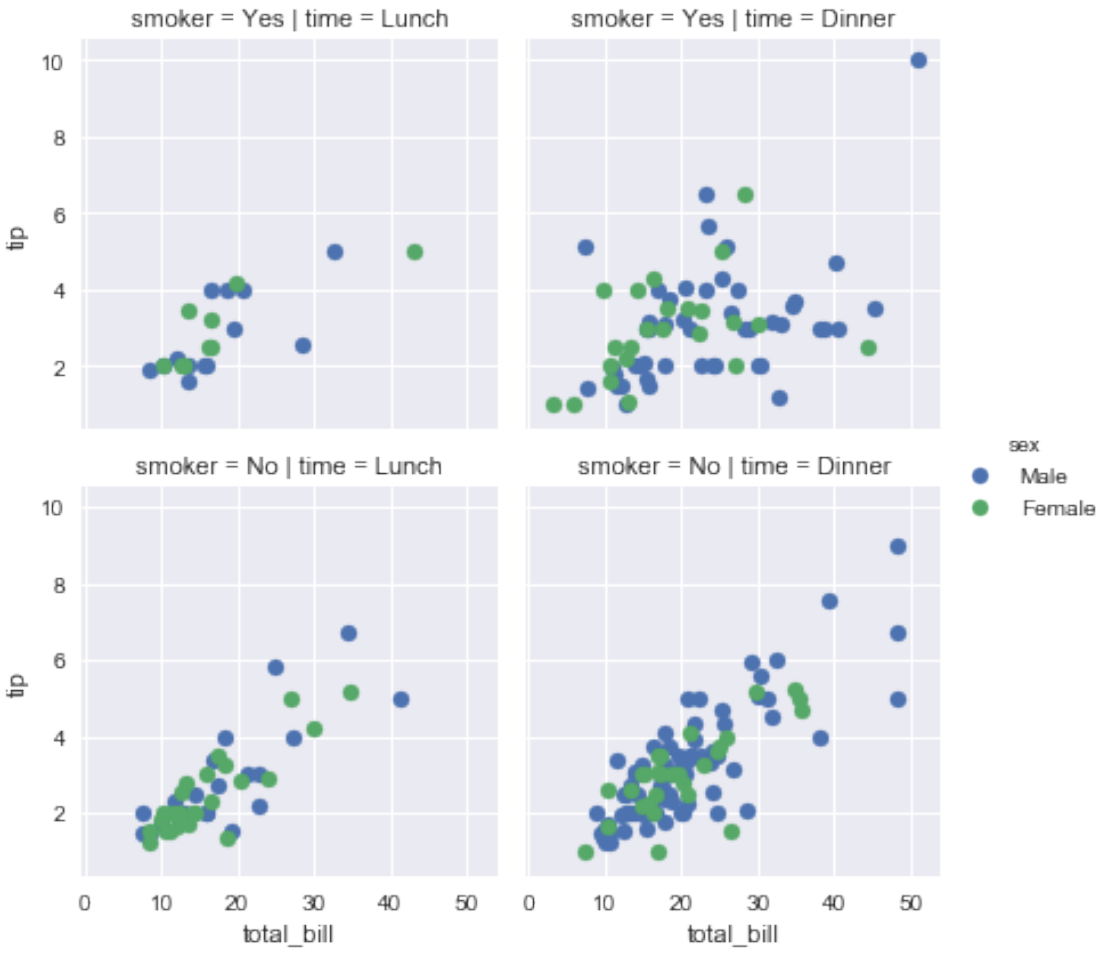
[16]:
```
# Só a grade
g = sns.FacetGrid(tips, col="time", row="smoker")
```



[17]:
```
g = sns.FacetGrid(tips, col="time",  row="smoker")
g = g.map(plt.hist, "total_bill")
```



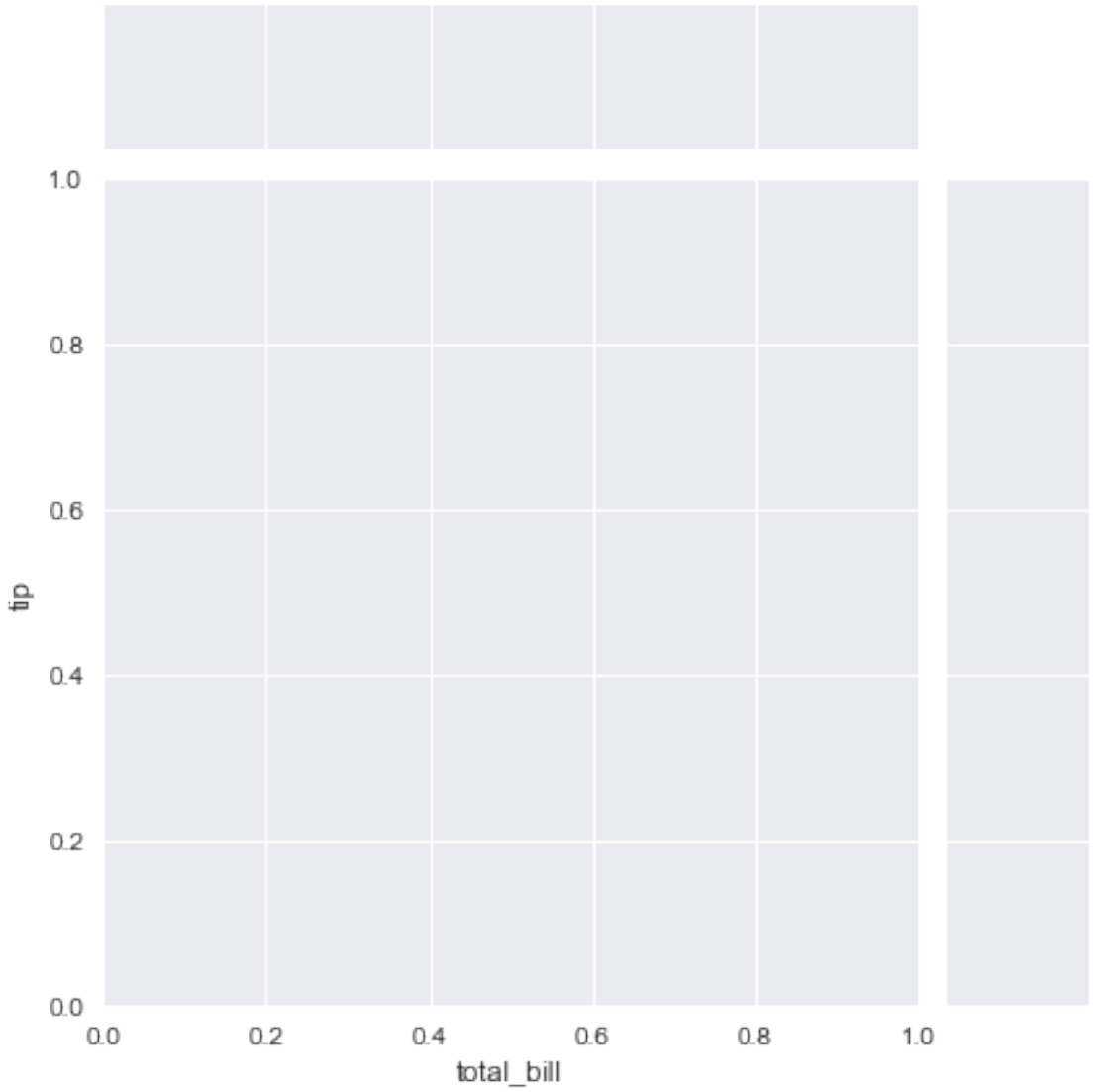
[18]:
```
g = sns.FacetGrid(tips, col="time",  row="smoker",hue='sex')
# Observe como os argumentos vêm após a chamada do plt.scatter
g = g.map(plt.scatter, "total_bill", "tip").add_legend()
```



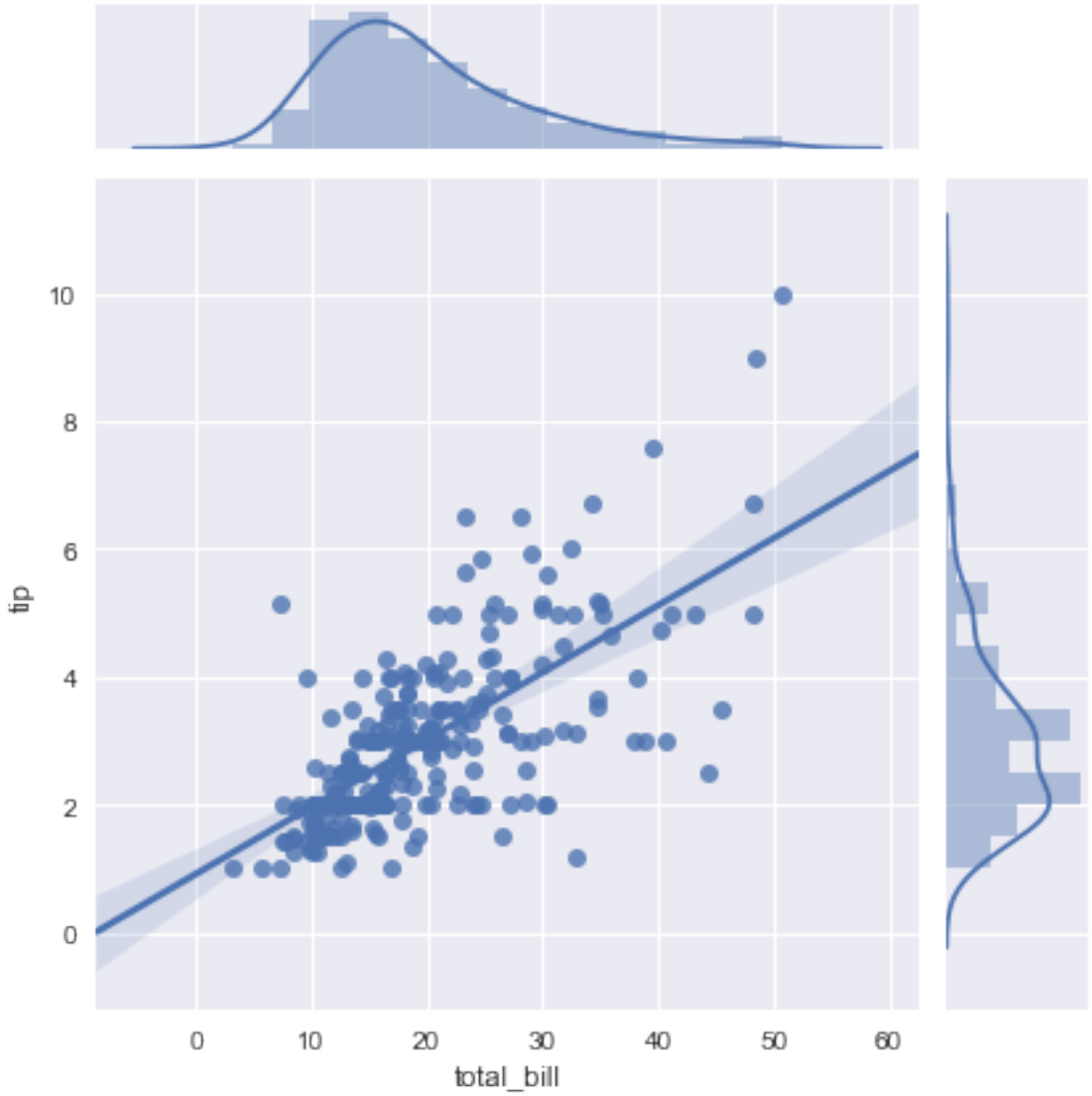
1.4 JointGrid

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

[19]:
```
g = sns.JointGrid(x="total_bill", y="tip", data=tips)
```



[20]:
```
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.