In [1]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:
```
tips = sns.load_dataset('tips')
print("Shape :" ,tips.shape)
tips.sample(10)
```

```
Shape : (244, 7)
```

Out[2]:

	total_bill	tip	sex	smoker	day	time	size
113	23.95	2.55	Male	No	Sun	Dinner	2
194	16.58	4.00	Male	Yes	Thur	Lunch	2
54	25.56	4.34	Male	No	Sun	Dinner	4
132	11.17	1.50	Female	No	Thur	Lunch	2
7	26.88	3.12	Male	No	Sun	Dinner	4
103	22.42	3.48	Female	Yes	Sat	Dinner	2
155	29.85	5.14	Female	No	Sun	Dinner	5
88	24.71	5.85	Male	No	Thur	Lunch	2
133	12.26	2.00	Female	No	Thur	Lunch	2
174	16.82	4.00	Male	Yes	Sun	Dinner	2

In [3]:
```
print ("Name of each columns and their dtype :",tips.columns )
print("\n")
print ("Information about it :" ,tips.info())
print('\n')
print("Descríption :\n",tips.describe(include ='all'))
```

```
Name of each columns and their dtype : Index(['total_bill', 'tip', 'sex', '
smoker', 'day', 'time', 'size'], dtype='object')


<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   total_bill  244 non-null    float64
 1   tip         244 non-null    float64
 2   sex         244 non-null    category
 3   smoker      244 non-null    category
 4   day         244 non-null    category
 5   time        244 non-null    category
 6   size        244 non-null    int64
dtypes: category(4), float64(2), int64(1)
memory usage: 7.4 KB
Information about it : None
```

```
Description :
         total_bill         tip   sex smoker  day    time        size
count   244.000000  244.000000   244    244  244     244  244.000000
unique         NaN         NaN     2      2    4       2         NaN
top            NaN         NaN  Male     No  Sat  Dinner         NaN
freq           NaN         NaN   157    151   87     176         NaN
mean     19.785943    2.998279   NaN    NaN  NaN     NaN    2.569672
std       8.902412    1.383638   NaN    NaN  NaN     NaN    0.951100
min       3.070000    1.000000   NaN    NaN  NaN     NaN    1.000000
25%      13.347500    2.000000   NaN    NaN  NaN     NaN    2.000000
50%      17.795000    2.900000   NaN    NaN  NaN     NaN    2.000000
75%      24.127500    3.562500   NaN    NaN  NaN     NaN    3.000000
```

In [4]:
```
tips.isnull().sum()
```

Out[4]:
```
total_bill    0
tip           0
sex           0
smoker        0
day           0
time          0
size          0
dtype: int64
```

In [5]:
```
pd.DataFrame(tips['sex'].value_counts())
```

Out[5]:

	sex
Male	157
Female	87

In [6]:
```
pd.DataFrame(tips['smoker'].value_counts())
```

Out[6]:

	smoker
No	151
Yes	93

In [7]:
```
pd.DataFrame(tips['day'].value_counts())
```

Out[7]:

	day
Sat	87
Sun	76
Thur	62
Fri	19

In [8]:
```
pd.DataFrame(tips['time'].value_counts())
```

Out[8]:

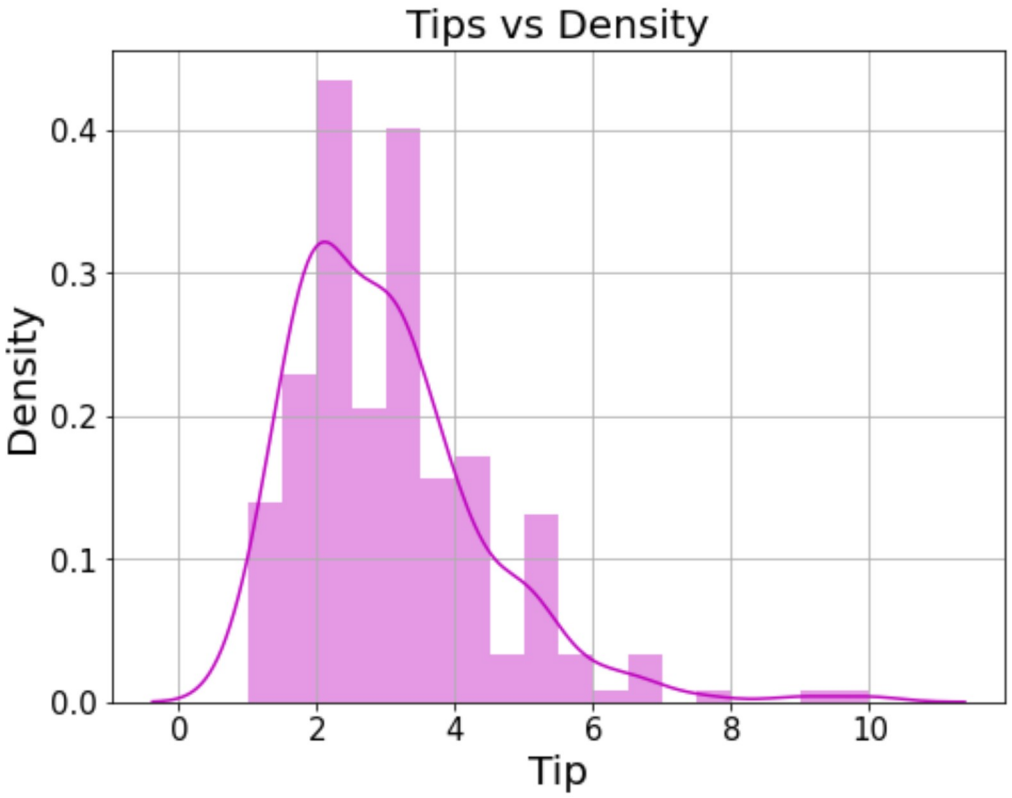
	time
Dinner	176
Lunch	68

Seaborn library

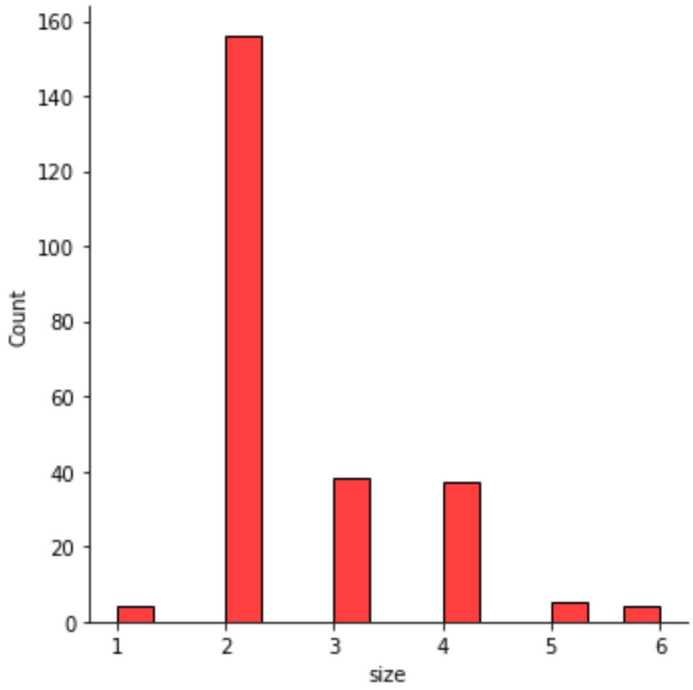
the source library work : https://seaborn.pydata.org/generated/seaborn.distplot.html#seaborn.distplot This function provides access to several approaches for visualizing the univariate or bivariate distribution of data, including subsets of data defined by semantic mapping and faceting across multiple subplots. The kind parameter selects the approach to use:

In [9]:
```
# Histogram plots
figure , grid , graph = plt.figure(figsize=(8,6)),
plt.grid(True), sns.distplot(tips['tip'],color ='m')
#sns.histplot(tips['tip'])
#sns.displot(tips['tip'])
axis_x , axis_y = plt.xticks(fontsize =15), plt.yticks(fontsize
=15)
x_label, y_label = plt.xlabel('Tip' ,fontsize= 20),
plt.ylabel('Density' ,fontsize= 20)
title , visual = plt.title("Tips vs Density",fontsize =20),
plt.show()
```

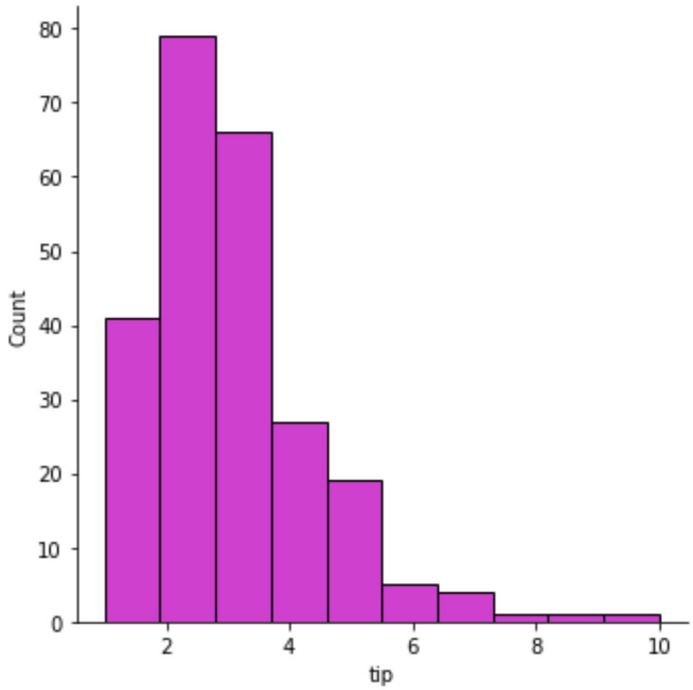
```
C:\Users\User\anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a f
uture version. Please adapt your code to use either `displot` (a figure-lev
el function with similar flexibility) or `histplot` (an axes-level function
for histograms).
  warnings.warn(msg, FutureWarning)
```



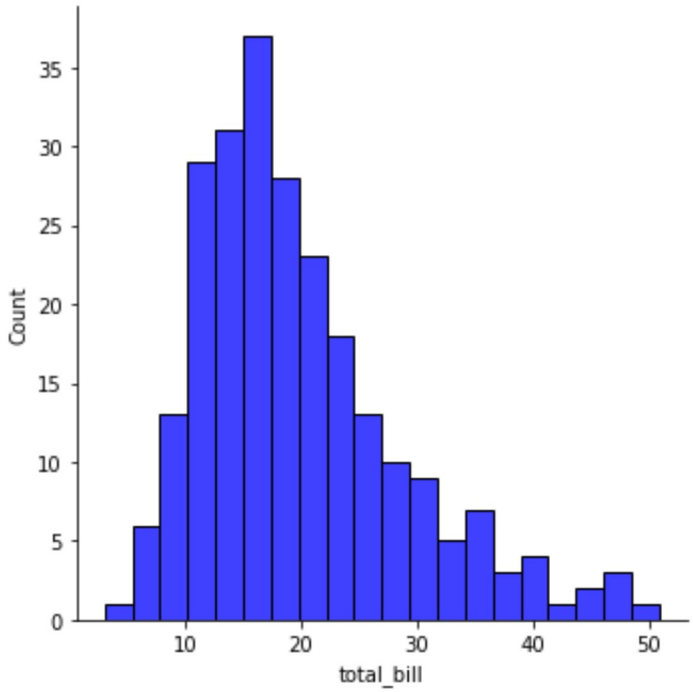
In [10]:
```
#sns.distplot(tips['tip'], bins=10)
sns.displot(tips['size'], color='red', bins =15) #
plt.hist(tips['size'])
plt.show()
```



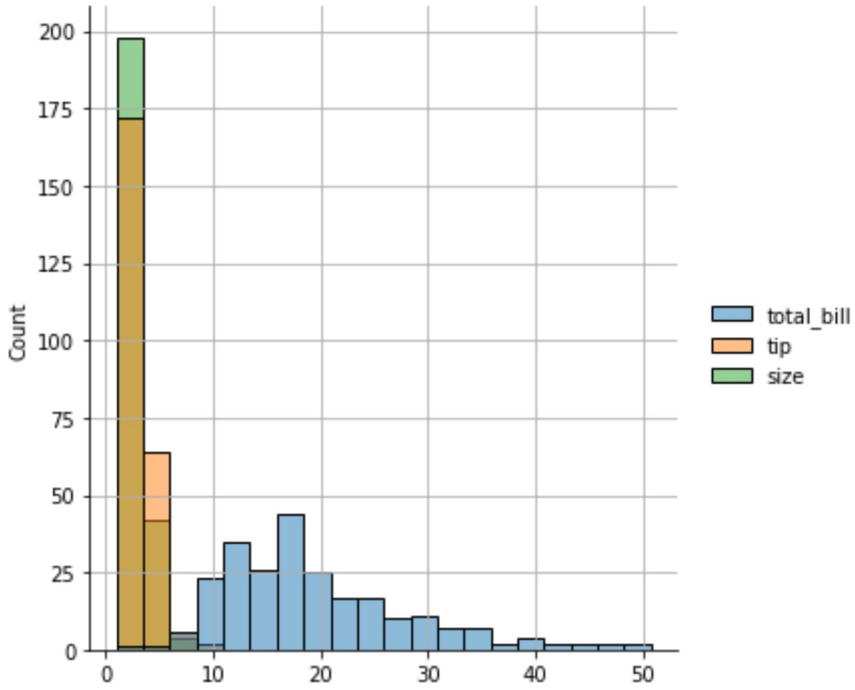
In [11]:
```
sns.displot(tips['tip'], color='m',bins =10) #
plt.hist(tips['tip'])
plt.show()
```



In [12]:
```
sns.displot(tips['total_bill'], color='b', bins =20) #
plt.hist(tips['total_bill'])
plt.show()
```



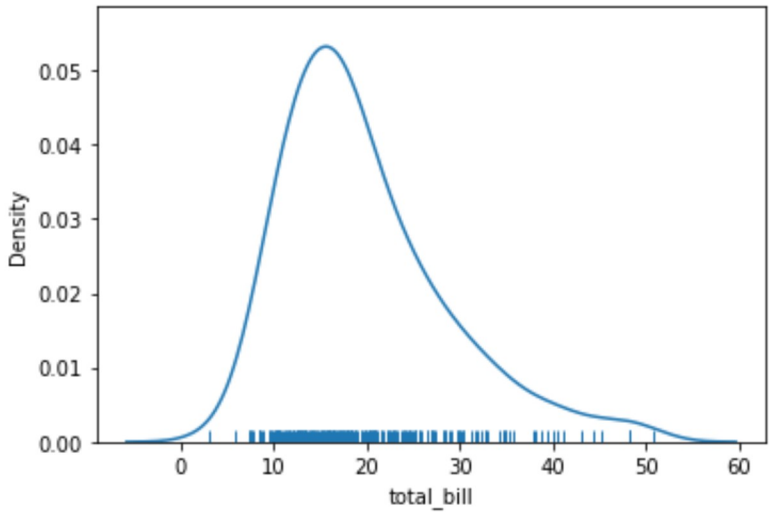
In [13]:
```
sns.displot(data= tips) # sns.displot(tips)
plt.grid()
plt.show()
sns.distplot(tips['total_bill'],rug= True, hist =False)
```




```
C:\Users\User\anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a f
uture version. Please adapt your code to use either `displot` (a figure-lev
el function with similar flexibility) or `kdeplot` (an axes-level function
for kernel density plots).
```

```
  warnings.warn(msg, FutureWarning)
C:\Users\User\anaconda3\lib\site-packages\seaborn\distributions.py:2103: Fu
tureWarning: The `axis` variable is no longer used and will be removed. Ins
tead, assign variables directly to `x` or `y`.
  warnings.warn(msg, FutureWarning)
```

Out[13]: <AxesSubplot:xlabel='total_bill', ylabel='Density'>

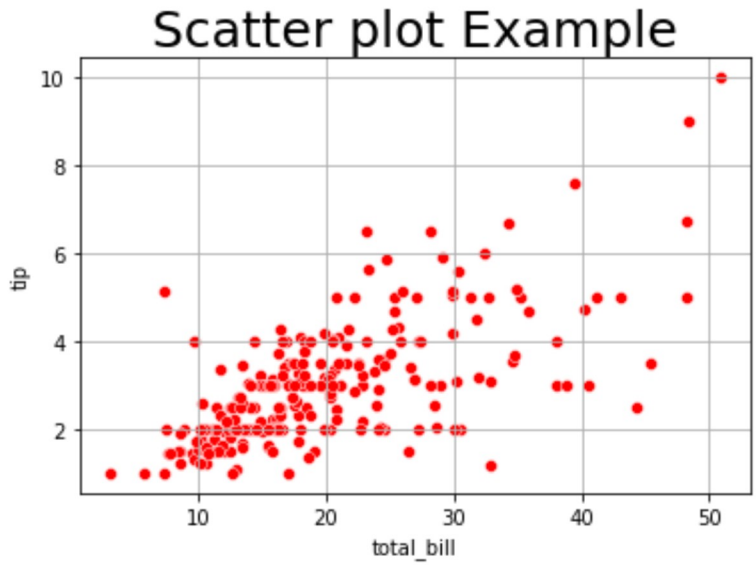


Scatter Plot

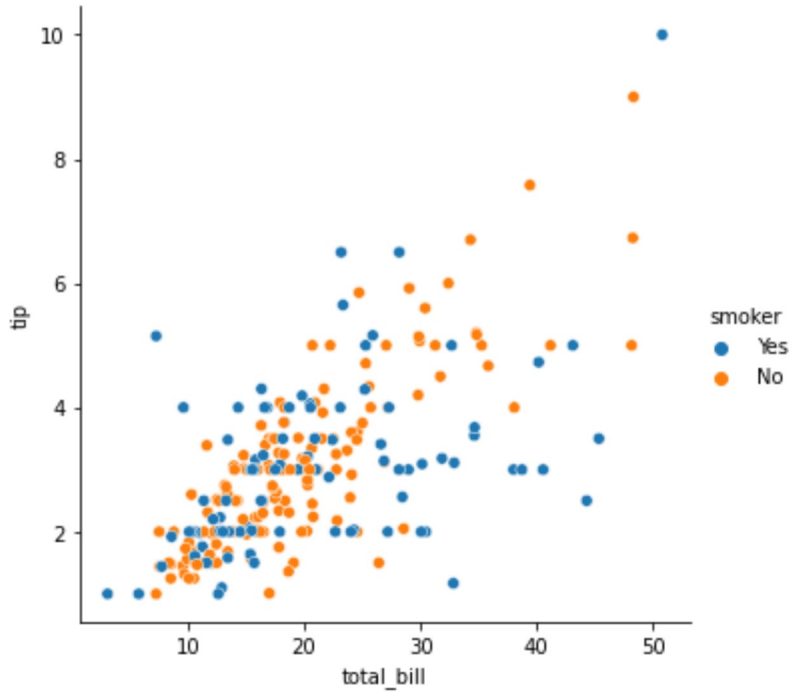
The relationship between x and y can be shown for different subsets of the data using the hue, size, and style parameters. These parameters control what visual semantics are used to identify the different subsets. It is possible to show up to three dimensions independently by using all three semantic types, but this style of plot can be hard to interpret and is often ineffective. Using redundant semantics (i.e. both hue and style for the same variable) can be helpful for making graphics more accessible.

In [14]:

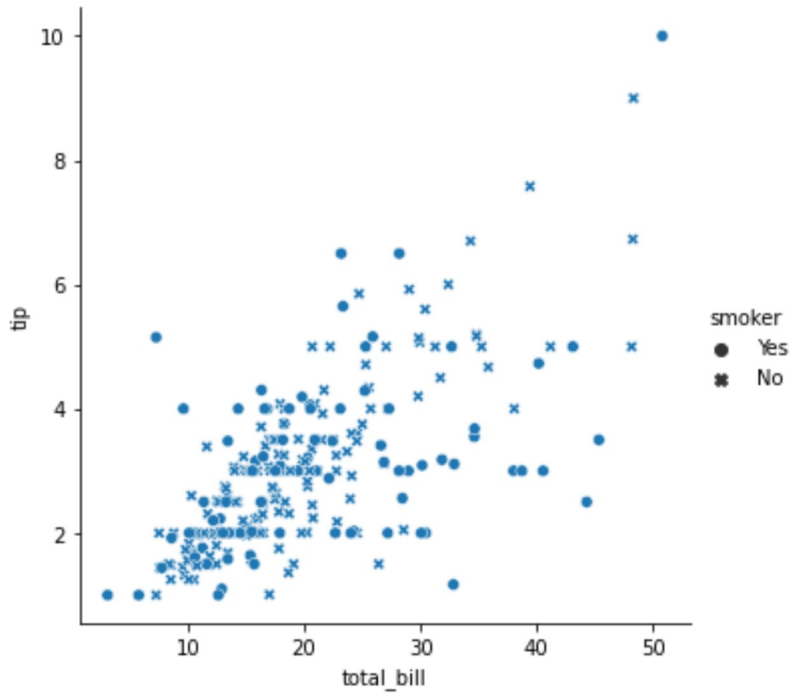
```
sns.scatterplot(x='total_bill', y='tip', data=tips, color
='red')
# data  = pandas.DataFrame, numpy.ndarray, mapping, or sequence
# x, yvectors or keys in data
plt.title("Scatter plot Example", fontsize =25)
plt.grid()
plt.show()
```



In [15]:
```
sns.relplot(x='total_bill', y='tip', data = tips, hue ='smoker')
#hue = Grouping variable that will produce points with different
colors. Can be either categorical or numeric,
#although color mapping will behave differently in latter case.
plt.show()
```

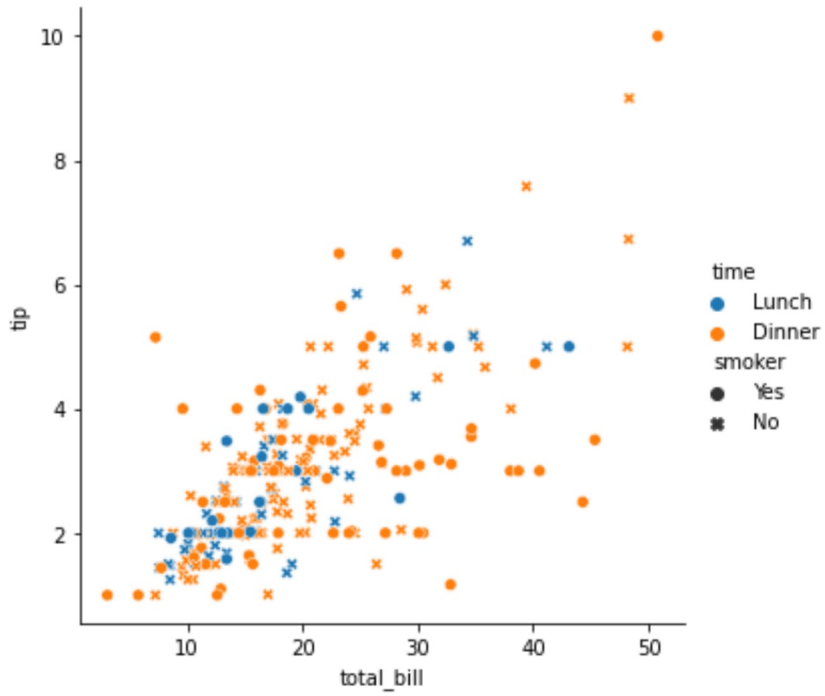


In [16]:
```
sns.relplot(x='total_bill', y='tip', data = tips,
style='smoker')
#style = Grouping variable that will produce points with
different markers. Can have a numeric dtype
#but will always be treated as categorical.
plt.show()
```

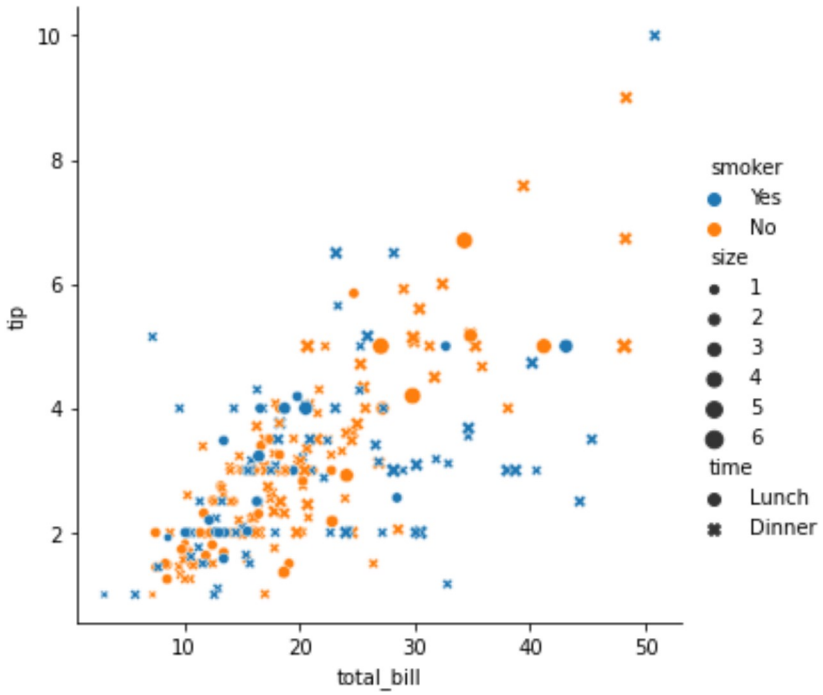


In [17]:

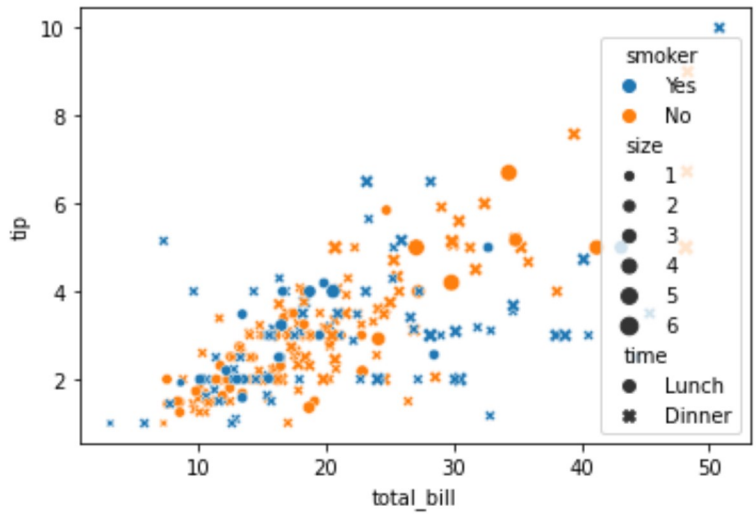
```
sns.relplot(x='total_bill', y='tip', data = tips,hue ='time',
style='smoker')
plt.show()
```



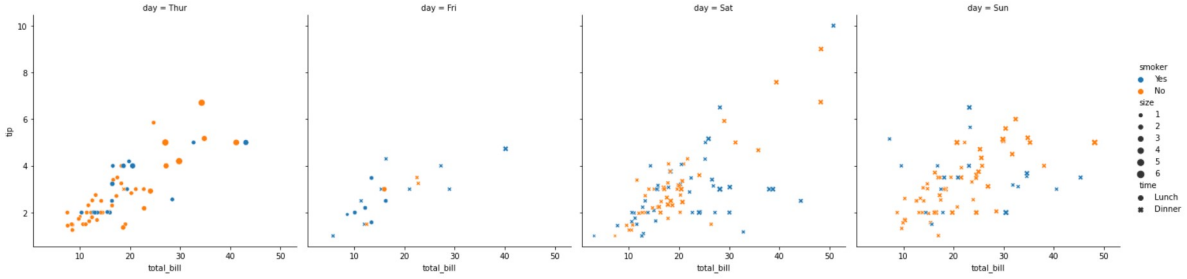
In [18]:
```
sns.relplot(x='total_bill', y='tip', data=tips, hue='smoker',
style='time',size='size')
#size =sizeslist, dict, or tuple An object that determines how
sizes are chosen when size is used. It can always be a list of
size values or a dict mapping levels of the size variable to
sizes. When size is numeric, it can also be a tuple specifying
the minimum and maximum size to use such that other values are
normalized within this range.
#
#size_orderlist = Specified order for appearance of the size
variable levels, otherwise they are determined from the data.
Not relevant when the size variable is numeric.

plt.show()
```



In [19]:
```
sns.scatterplot(x='total_bill', y='tip', data=tips,
hue='smoker', style='time',size='size')
plt.show()
```

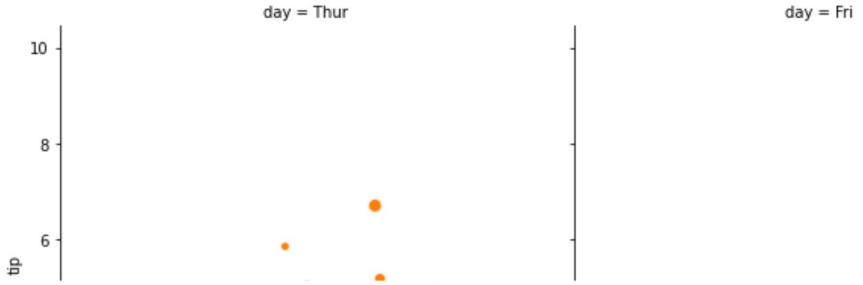


In [20]:
```
sns.relplot(x='total_bill', y='tip', data=tips, hue='smoker',
style='time',size='size',col='day')
plt.show()
```

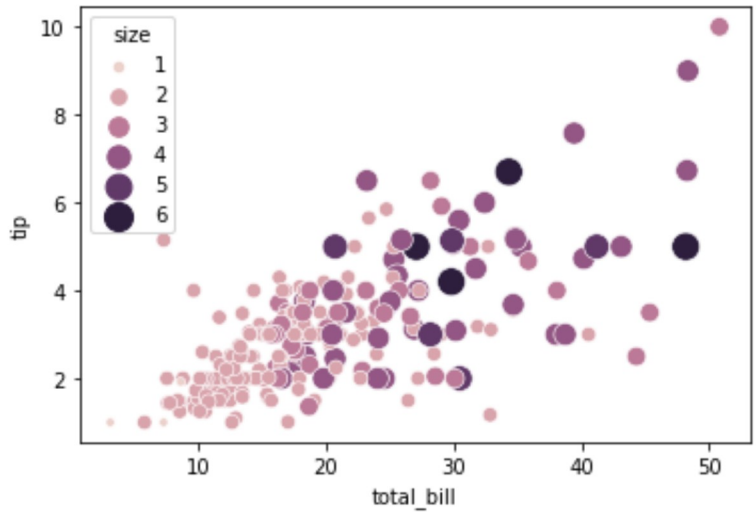


In [21]:
```
sns.relplot(x='total_bill', y='tip', data=tips,
hue='smoker',size='size',style='time', col='day',col_wrap=2)
```

Out[21]: <seaborn.axisgrid.FacetGrid at 0x1a20a0aee80>

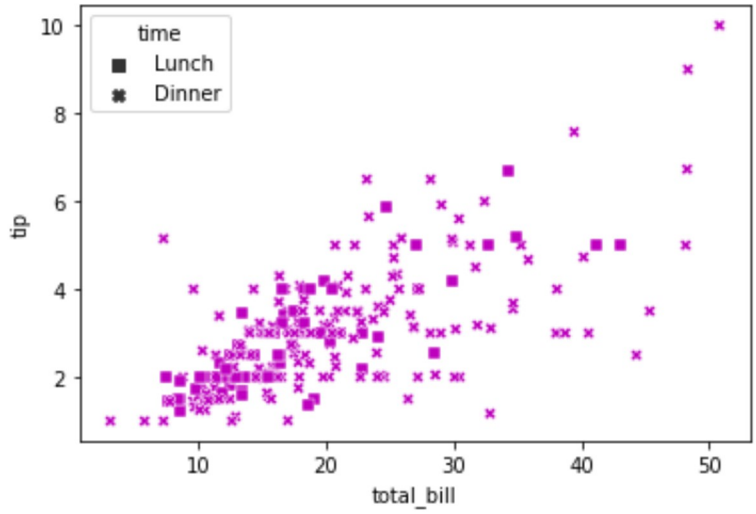


In [22]:
```
sns.scatterplot(
    data=tips, x="total_bill", y="tip", hue="size", size="size",
    sizes=(20, 200), legend="full"
)
plt.show()
```



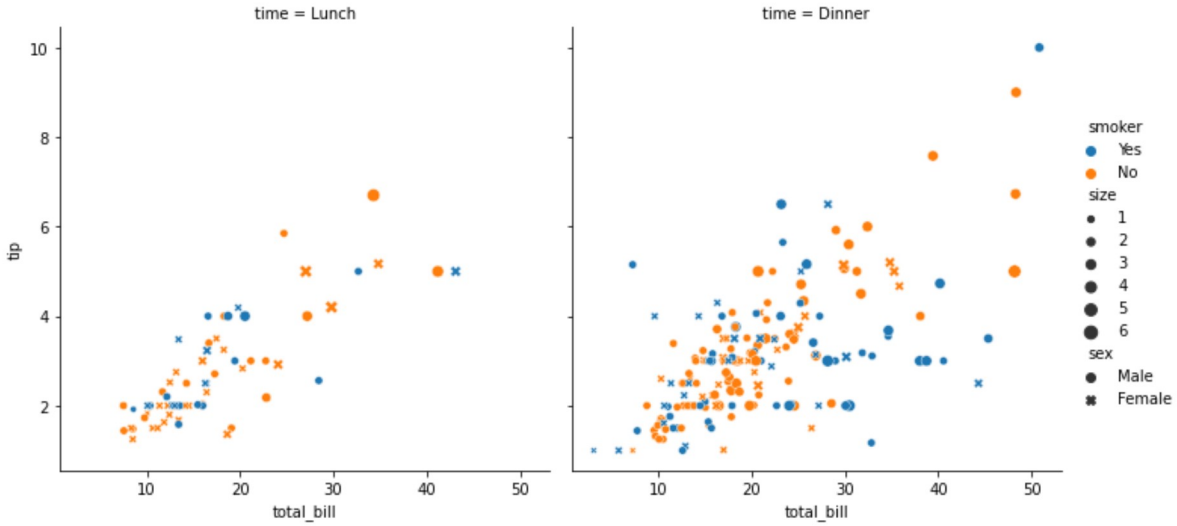
In [23]:
```
markers = {"Lunch": "s", "Dinner": "X"}
sns.scatterplot(data=tips, x="total_bill", y="tip",
style="time", markers=markers,color='m')
```

Out[23]: <AxesSubplot:xlabel='total_bill', ylabel='tip'>



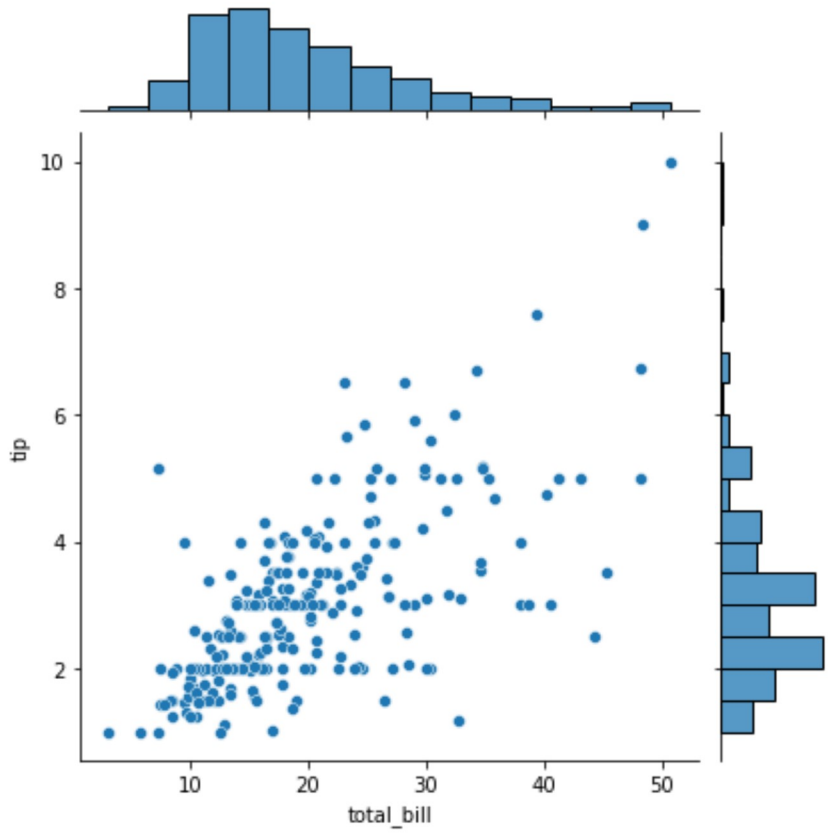
In [24]:
```
sns.relplot(x='total_bill', y='tip',
            data=tips, hue='smoker',size='size', style=
'sex',col='time')
```

Out[24]: <seaborn.axisgrid.FacetGrid at 0x1a20bb7c790>

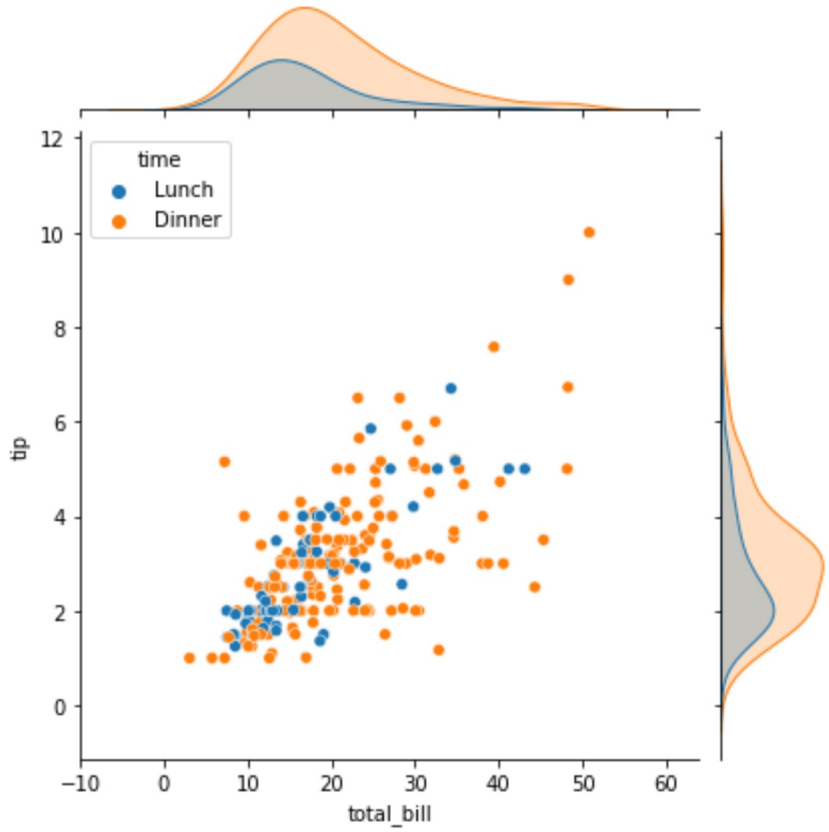


Join Plot

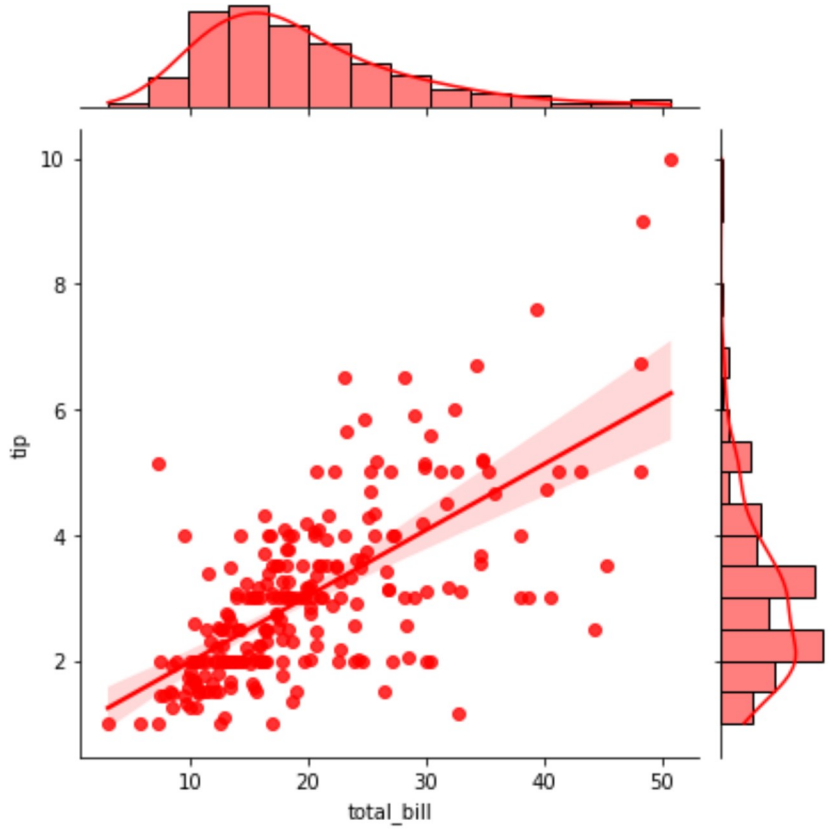
In [25]:
```
sns.jointplot(y='tip', x='total_bill', data=tips)
# data  = pandas.DataFrame, numpy.ndarray, mapping, or sequence
# x, yvectors or keys in data
plt.show()
```



In [26]:
```
sns.jointplot(y='tip', x='total_bill', data=tips, hue ='time')
plt.show()
```

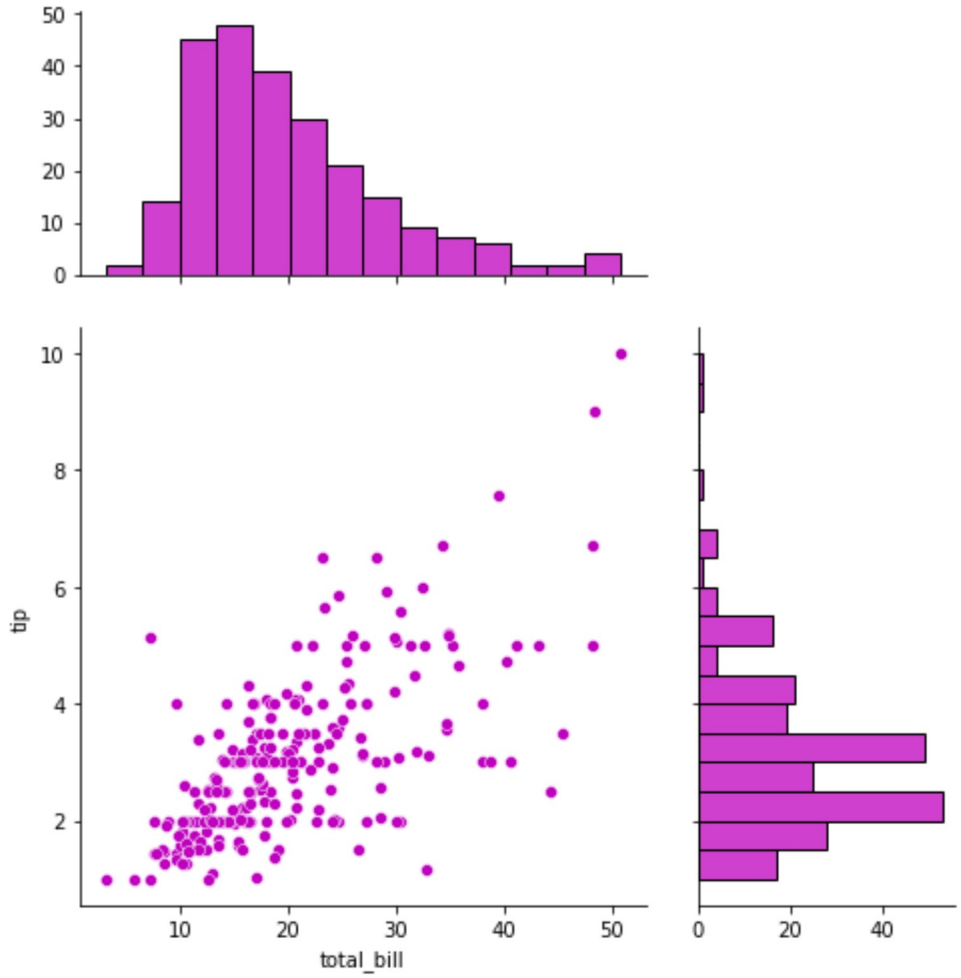


In [27]:
```
sns.jointplot(y='tip', x='total_bill', data=tips,kind
='reg',color ='red')
#kind{ "scatter" | "kde" | "hist" | "hex" | "reg" | "resid" }
plt.show()
```



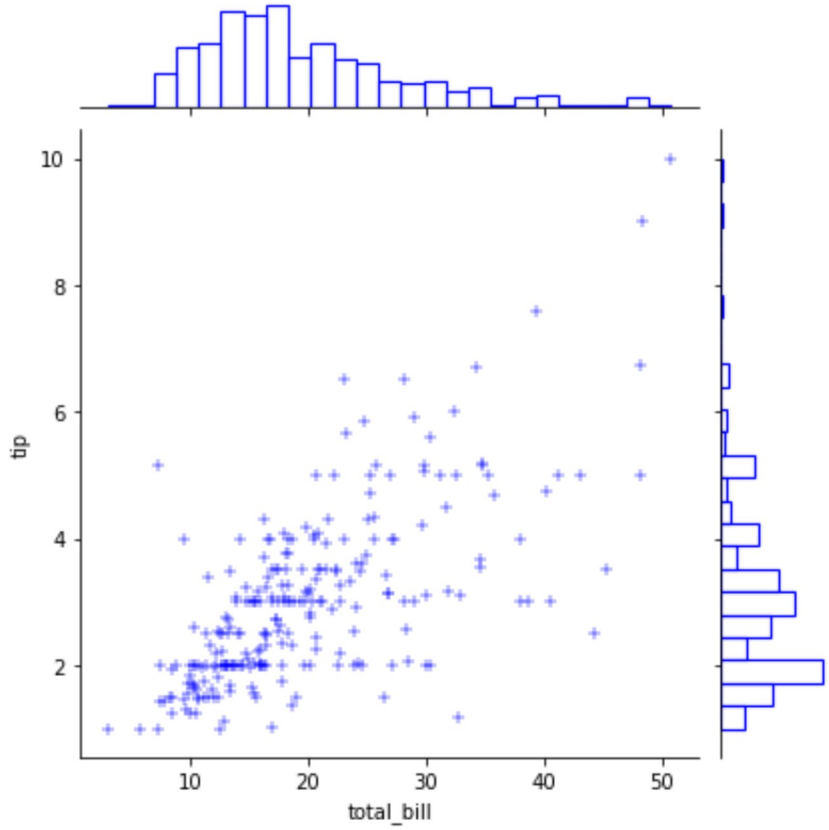
In [28]:

```
sns.jointplot(y='tip', x='total_bill', data=tips,height=7,
ratio=2, marginal_ticks=True, color='m')
#height = Size of the figure (it will be square).
#ratio = Ratio of joint axes height to marginal axes height
plt.show()
```



In [29]:

```
sns.jointplot(y='tip', x='total_bill', data=tips,color ='b',
marker = "+", marginal_kws=dict(bins=25, fill=False),)
#{joint, marginal}_kwsdicts Additional keyword arguments for the
plot components.

plt.show()
```



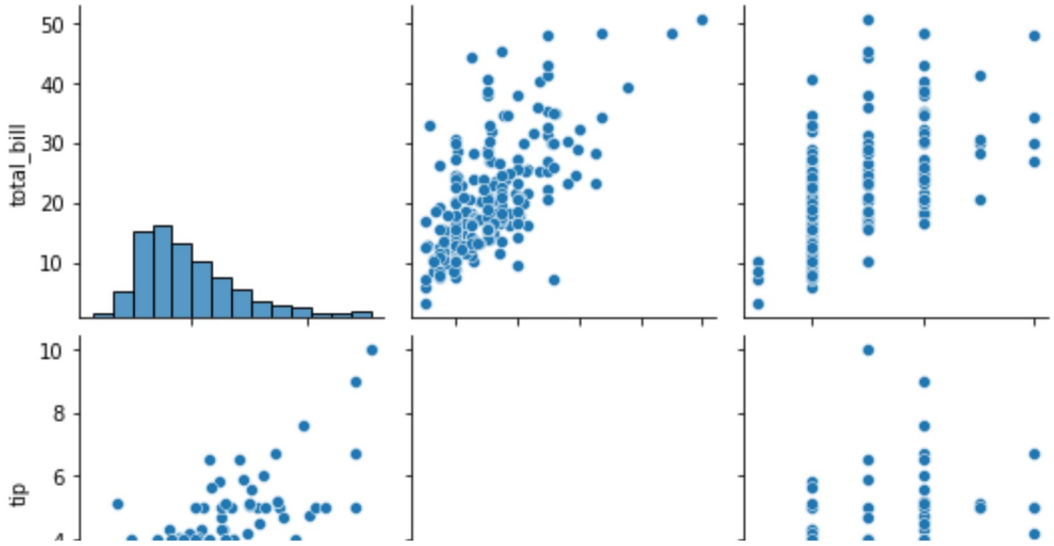
Pair Plot

Plot pairwise relationships in a dataset.

By default, this function will create a grid of Axes such that each numeric variable in data will by shared across the y-axes across a single row and the x-axes across a single column. The diagonal plots are treated differently: a univariate distribution plot is drawn to show the marginal distribution of the data in each column.

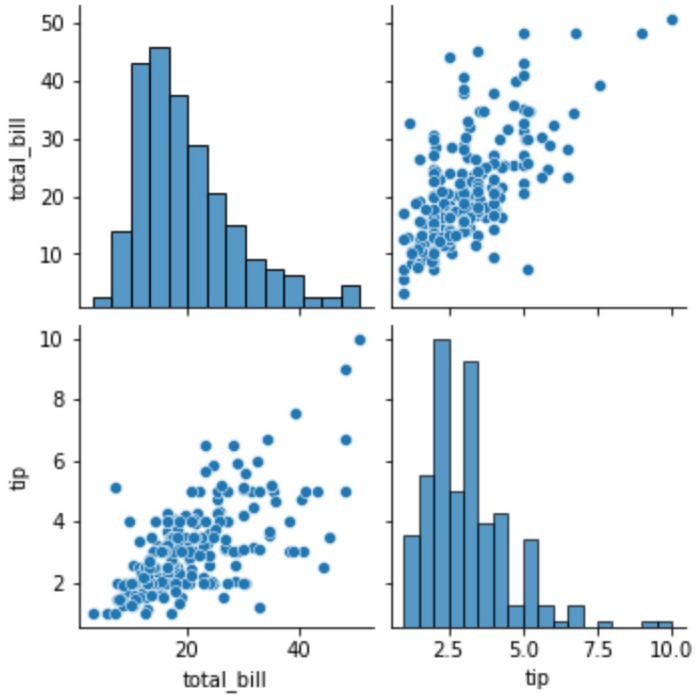
In [30]:
```
sns.pairplot(data =tips)
# data : pandas.DataFrame .Tidy (long-form) dataframe where each
column is a variable and each row is an observation.
```

Out[30]: <seaborn.axisgrid.PairGrid at 0x1a20b2e2e50>



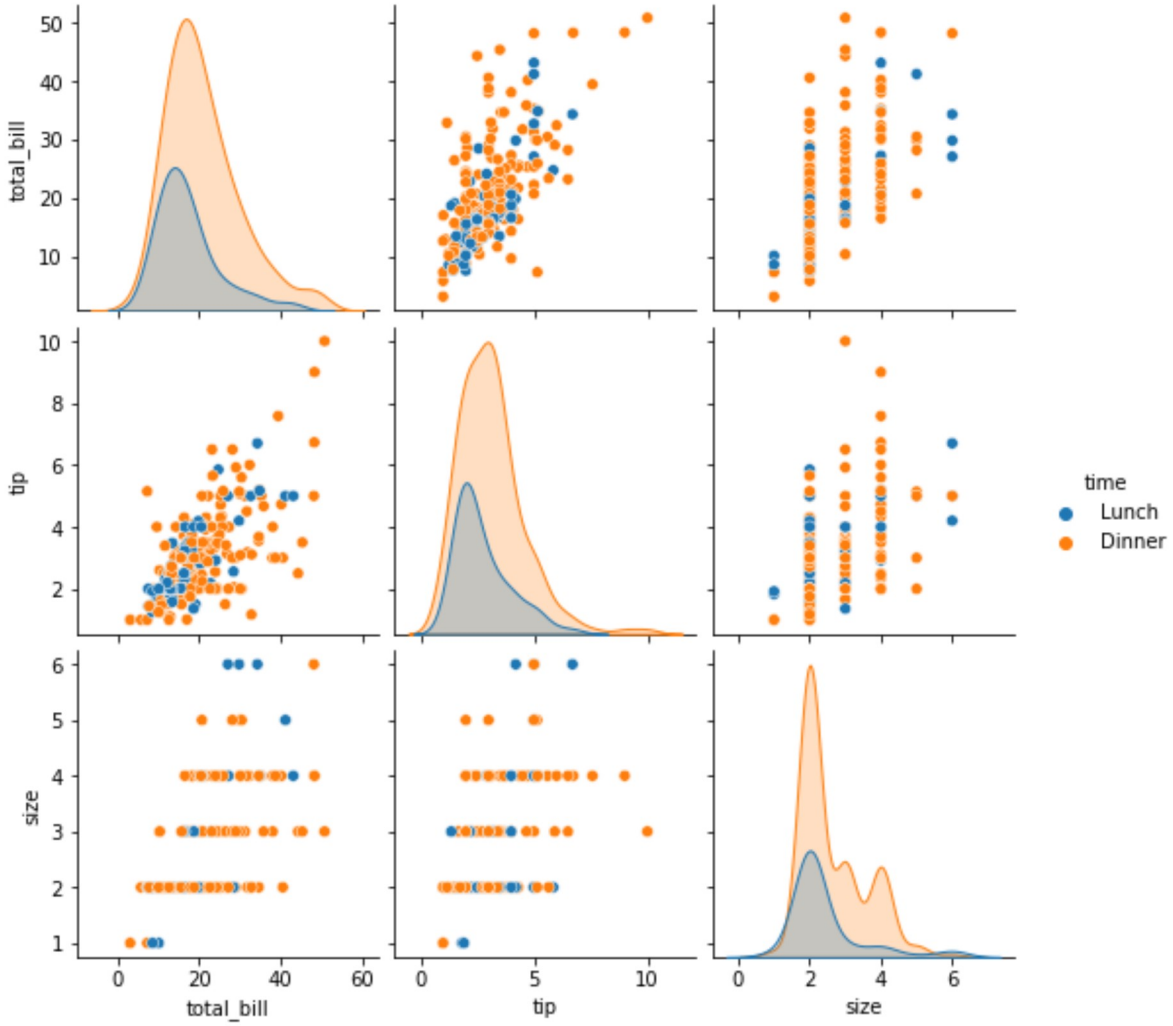
In [31]:

```
sns.pairplot(tips, vars=['total_bill','tip'],kind ='scatter')
#vars = list of variable names Variables within data to use,
otherwise use every column with a numeric datatype.
# kind = kind{'scatter', 'kde', 'hist', 'reg'} Kind of plot to
make.

plt.show()
```



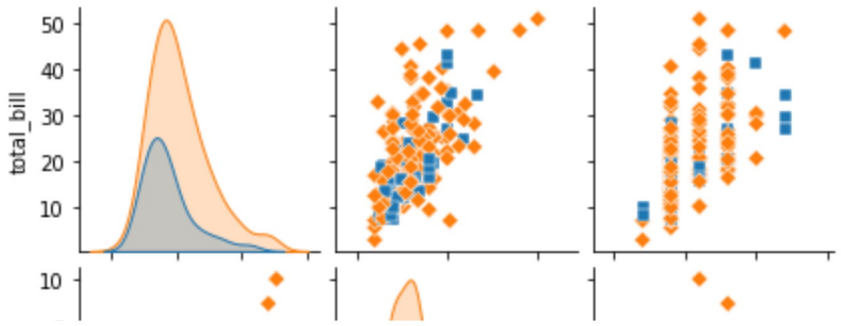
In [32]:

```
sns.pairplot(tips, hue='time')
# name of variable in data Variable in data to map plot aspects
to different colors.
plt.show()
```



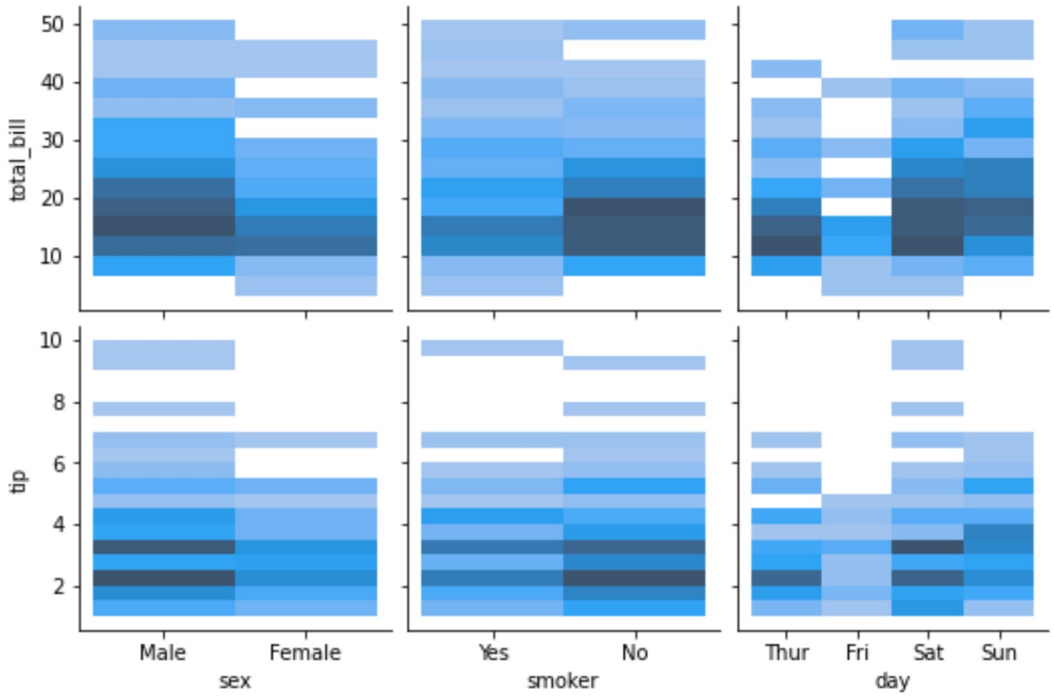
In [33]:
```
sns.pairplot(data=tips, hue='time', markers=[ "s", "D"],
height=2)
#As with other figure-level functions, the size of the figure is
controlled by setting the height of each individual subplot:
#he markers parameter applies a style mapping on the off-
diagonal axes. Currently, it will be redundant with the hue
variable:
```

Out[33]: <seaborn.axisgrid.PairGrid at 0x1a209e3e6a0>



In [34]:
```
sns.pairplot(data=tips,
             x_vars= ['sex','smoker','day'],
             y_vars= ['total_bill','tip'],
             kind ='hist'
             )
```

Out[34]: <seaborn.axisgrid.PairGrid at 0x1a20c75b4f0>



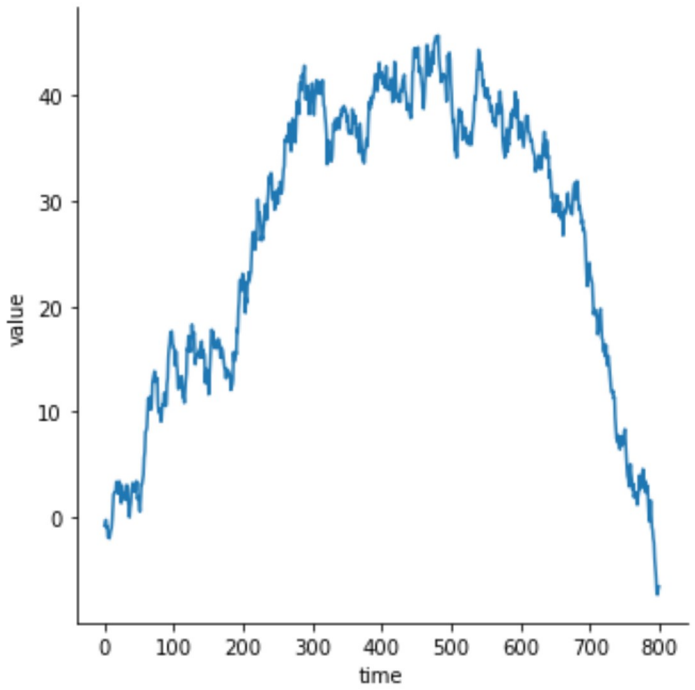
In [36]:
```
import numpy as np
from numpy.random import randn
df = pd.DataFrame(dict(time = np.arange(800), value =
randn(800).cumsum()))
df.head()
```

Out[36]:

	time	value
0	0	-0.795094
1	1	-0.588394
2	2	-0.232454
3	3	-0.982390

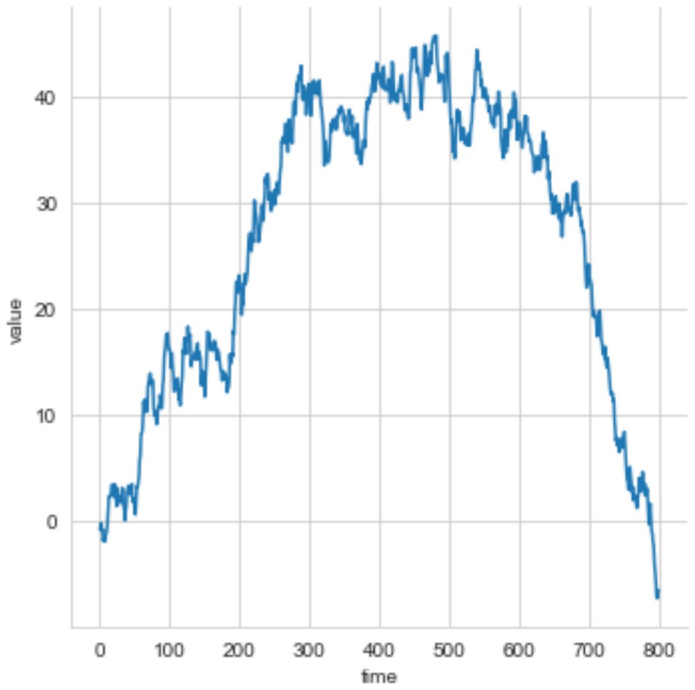
time value

In [37]:
```
sns.relplot(x='time', y='value', kind='line', data=df)
plt.show()
```



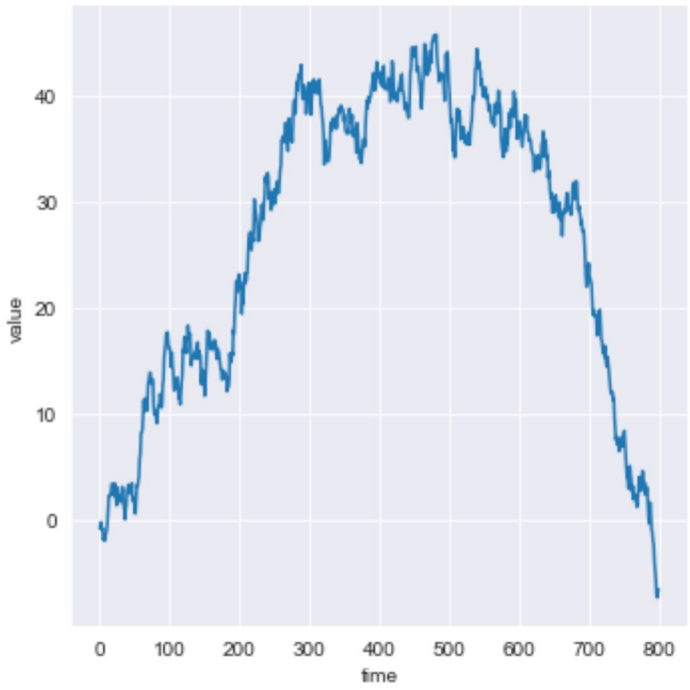
In [38]:
```
sns.set_style('whitegrid')
sns.relplot(x='time', y='value', kind='line', data=df)
```

Out[38]: <seaborn.axisgrid.FacetGrid at 0x1a20c95d3d0>



In [39]:
```
sns.set_style('darkgrid')
sns.relplot(x='time', y='value', kind='line', data=df)
```

Out[39]: <seaborn.axisgrid.FacetGrid at 0x1a20b27fca0>



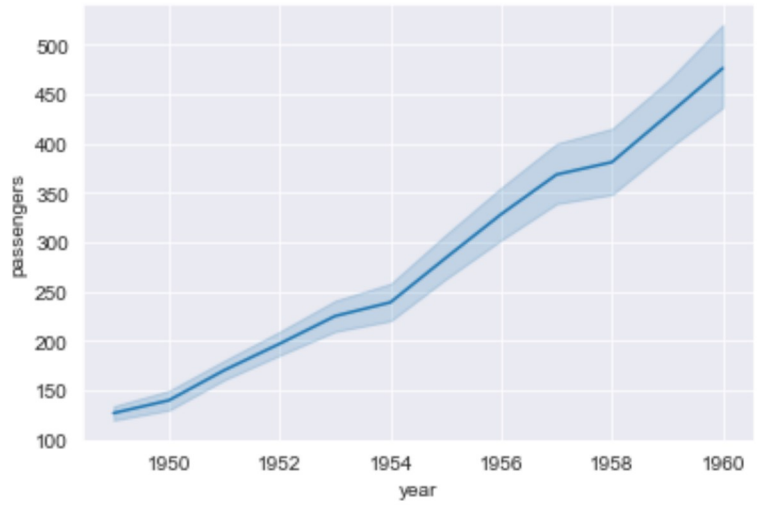
In [41]:
```
flights = sns.load_dataset("flights")
flights.head()
```

Out[41]:

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

In [42]:
```
sns.lineplot(data=flights, x="year", y="passengers")
```

Out[42]: <AxesSubplot:xlabel='year', ylabel='passengers'>



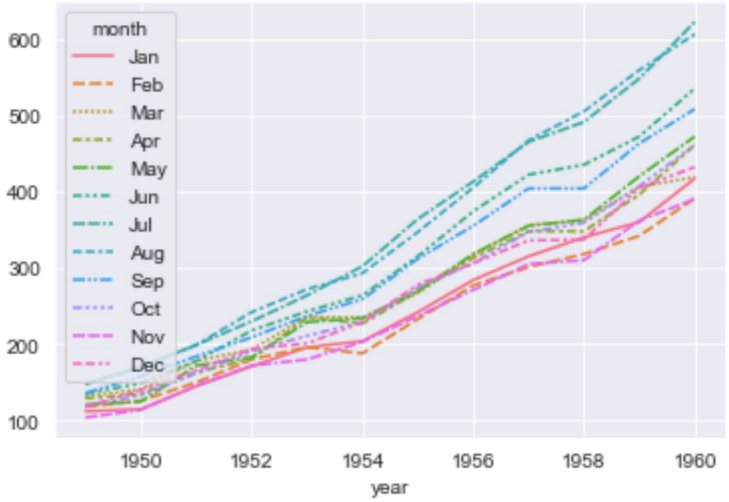
In [43]:
```
flights_wide = flights.pivot("year", "month", "passengers")
#Pivot the dataframe to a wide-form representation:
flights_wide.head()
```

Out[43]:

year	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1949	112	118	132	129	121	135	148	148	136	119	104	118
1950	115	126	141	135	125	149	170	170	158	133	114	140
1951	145	150	178	163	172	178	199	199	184	162	146	166
1952	171	180	193	181	183	218	230	242	209	191	172	194
1953	196	196	236	235	229	243	264	272	237	211	180	201

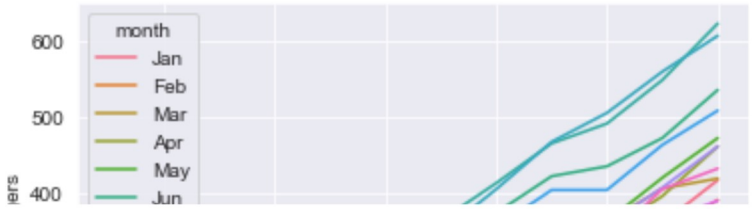
In [44]:
```
sns.lineplot(data=flights_wide)
#To plot a single vector, pass it to data. If the vector is a pandas.Series, it will be plotted against its index:
#Passing the entire wide-form dataset to data plots a separate line for each column:
```

Out[44]:
```
<AxesSubplot:xlabel='year'>
```



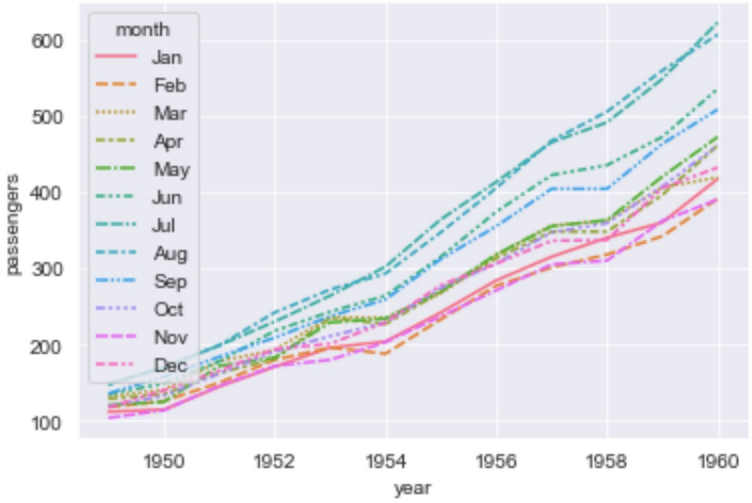
In [45]:
```
sns.lineplot(data=flights, x="year", y="passengers",
hue="month")
```

Out[45]:
```
<AxesSubplot:xlabel='year', ylabel='passengers'>
```



In [46]:
```
sns.lineplot(data=flights, x="year", y="passengers",
hue="month", style="month")
```

Out[46]: <AxesSubplot:xlabel='year', ylabel='passengers'>



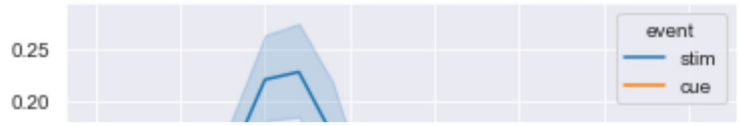
In [50]:
```
fmri = sns.load_dataset("fmri") # Taking a complex dataset
fmri.head()
```

Out[50]:

	subject	timepoint	event	region	signal
0	s13	18	stim	parietal	-0.017552
1	s5	14	stim	parietal	-0.080883
2	s12	18	stim	parietal	-0.081033
3	s11	18	stim	parietal	-0.046134
4	s10	18	stim	parietal	-0.037970

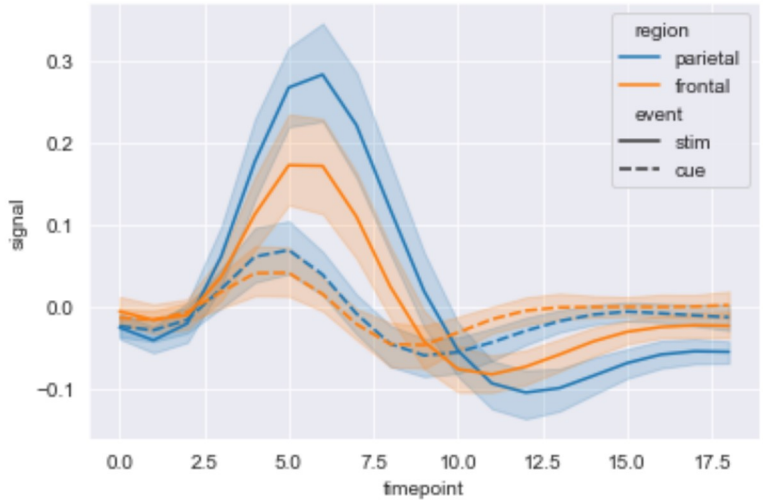
In [51]:
```
sns.lineplot(data=fmri, x="timepoint", y="signal", hue="event")
```

Out[51]: <AxesSubplot:xlabel='timepoint', ylabel='signal'>



In [52]:
```
sns.lineplot(data=fmri, x="timepoint", y="signal", hue="region",
style="event")
```

Out[52]: <AxesSubplot:xlabel='timepoint', ylabel='signal'>



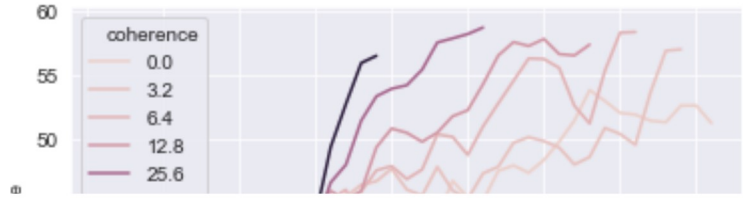
In [53]:
```
dots = sns.load_dataset("dots").query("align == 'dots'")
dots.head()
```

Out[53]:

	align	choice	time	coherence	firing_rate
0	dots	T1	-80	0.0	33.189967
1	dots	T1	-80	3.2	31.691726
2	dots	T1	-80	6.4	34.279840
3	dots	T1	-80	12.8	32.631874
4	dots	T1	-80	25.6	35.060487

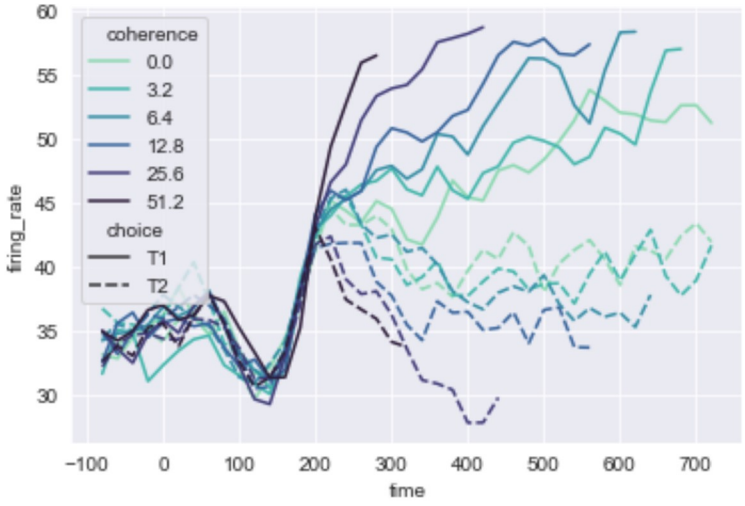
In [54]:
```
sns.lineplot(
    data=dots, x="time", y="firing_rate", hue="coherence",
style="choice",
)
```

Out[54]: <AxesSubplot:xlabel='time', ylabel='firing_rate'>



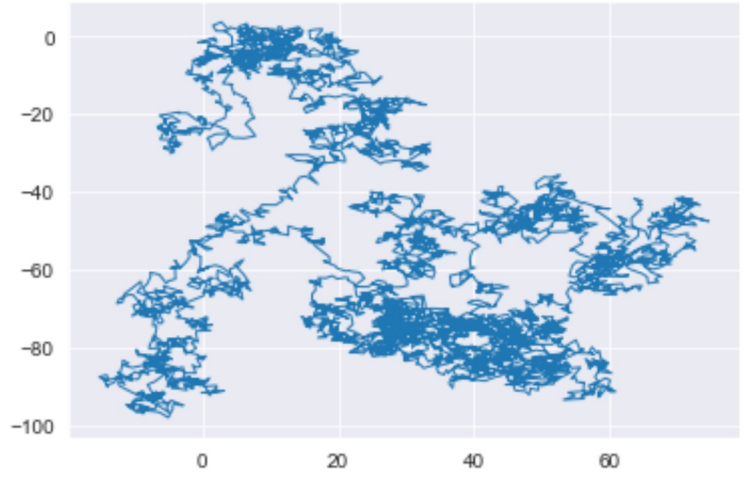
In [55]:
```
palette = sns.color_palette("mako_r", 6)
sns.lineplot(
    data=dots, x="time", y="firing_rate",
    hue="coherence", style="choice",
    palette=palette
)
```

Out[55]: <AxesSubplot:xlabel='time', ylabel='firing_rate'>



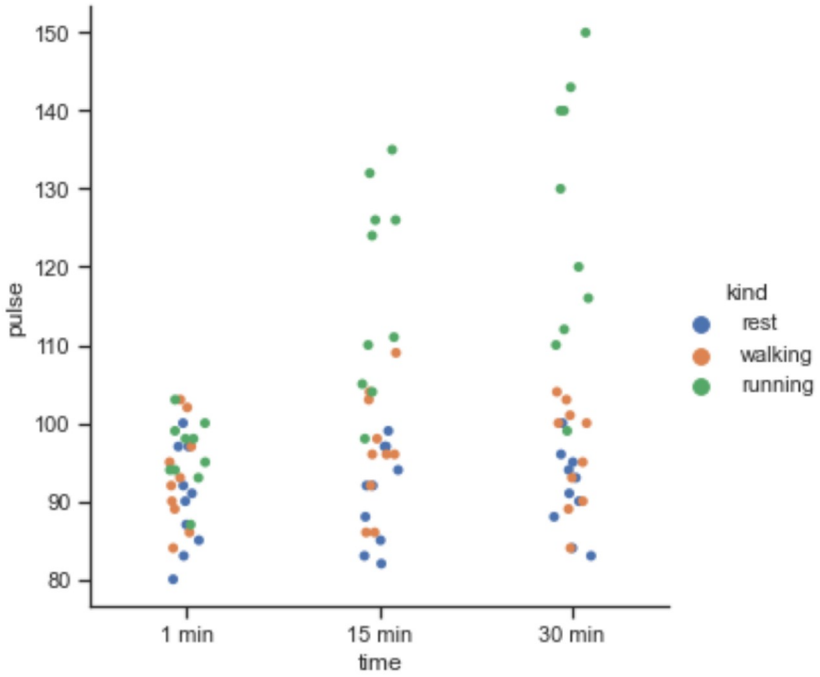
In [56]:
```
x, y = np.random.normal(size=(2, 5000)).cumsum(axis=1)
sns.lineplot(x=x, y=y, sort=False, lw=1)
```

Out[56]: <AxesSubplot:>



Cat plot

In [57]:
```
sns.set_theme(style="ticks")
exercise = sns.load_dataset("exercise")
g = sns.catplot(x="time", y="pulse", hue="kind", data=exercise)
```

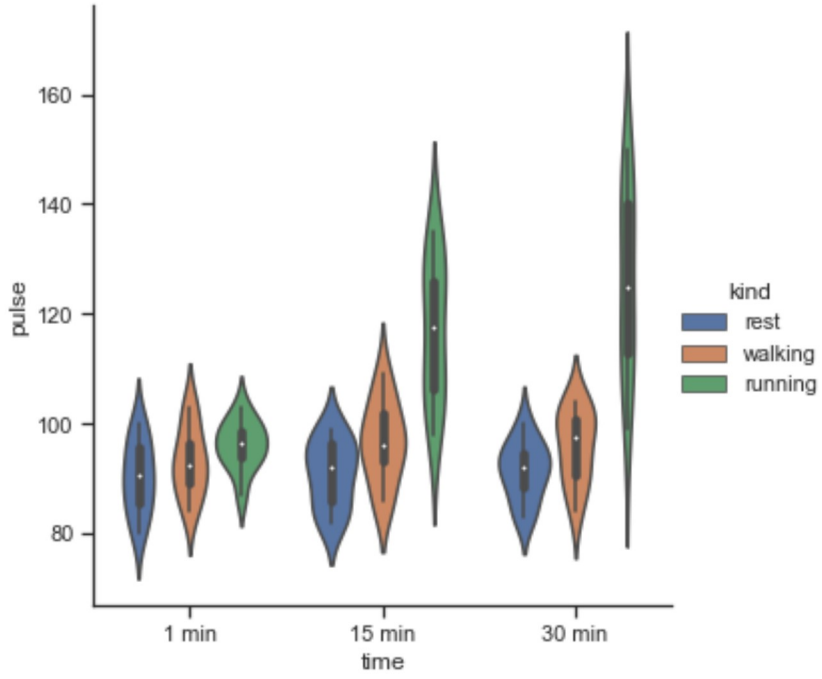


In [58]:
```
exercise.head(5)
```

Out[58]:

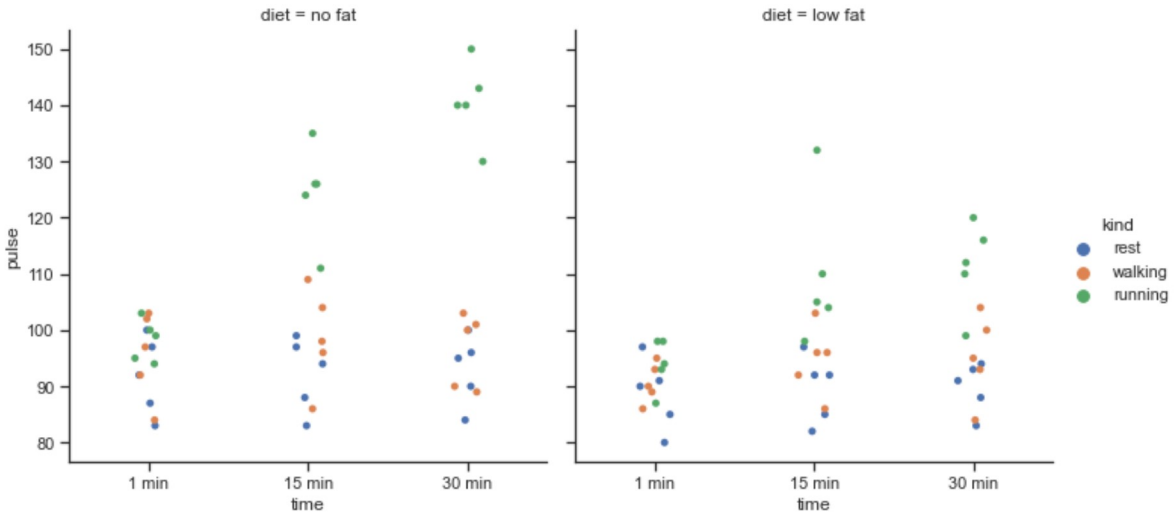
	Unnamed: 0	id	diet	pulse	time	kind
0	0	1	low fat	85	1 min	rest
1	1	1	low fat	85	15 min	rest
2	2	1	low fat	88	30 min	rest
3	3	2	low fat	90	1 min	rest
4	4	2	low fat	92	15 min	rest

In [59]:
```
g = sns.catplot(x="time", y="pulse", hue="kind",

                data=exercise, kind="violin")
```



In [60]:

```
g = sns.catplot(x="time", y="pulse", hue="kind",

                col="diet", data=exercise)
```



In [ ]: