Lliurament tasca 6A: Visualització gràfica

Multiples variables : - Exercici 1

\begin{align*}Cristiane\:de Souza \end{align*} \begin{align*}Date : Febrer\hspace{2mm}2021\end{align*}

EXAMINING NUMERICAL DATA

We will be introduced to techniques for exploring and summarizing numerical variables, working with the dataset : '$tips$'.

In [32]:
```
# importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings

warnings.filterwarnings('ignore')
```

EXPLORING BIVARIATE VARIABLES WITH SCATTERPLOTS

In [33]:
```
# Open the choosen file
tips = pd.read_csv('tips.csv')
tips.head()
```

Out[33]:

	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

In [34]:
```
tips.shape
```

Out[34]:
```
(244, 7)
```

In [35]:
```
tips.columns
```

Out[35]:
```
Index(['total_bill', 'tip', 'sex', 'smoker', 'day', 'time', 'size'], dtype=
'object')
```

In [36]:
```
tips.sex.unique()
```

Out[36]:
```
array(['Female', 'Male'], dtype=object)
```

In [37]:
```
tips.sex.nunique()
```

Out[37]: 2

In [38]:
```
tips.describe().round(3)
```

Out[38]:

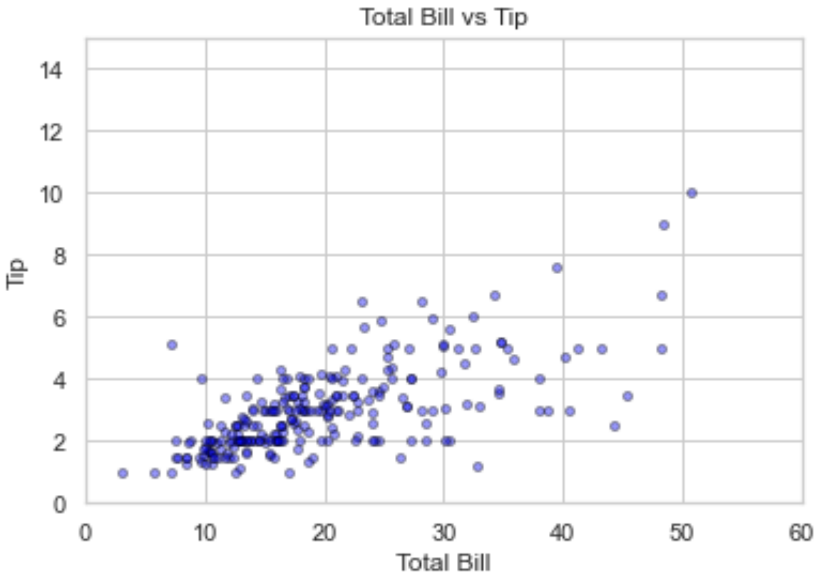
	total_bill	tip	size
count	244.000	244.000	244.000
mean	19.786	2.998	2.570
std	8.902	1.384	0.951
min	3.070	1.000	1.000
25%	13.348	2.000	2.000
50%	17.795	2.900	2.000
75%	24.127	3.562	3.000
max	50.810	10.000	6.000

In [39]:
```
# Create data
x = tips.total_bill
y = tips.tip
colors = 'Blue'
area = np.pi*5

plt.axis([0, 60, 0, 15])

# Plot
plt.scatter(x, y, s=area, c=colors, alpha=0.4, edgecolors='black')

plt.title('Total Bill vs Tip')
plt.ylabel('Tip')
plt.xlabel('Total Bill')
plt.show()
```



In [40]:
```
# Checking dataset variables
tips.dtypes
```

Out[40]:
```
total_bill    float64
tip           float64
sex            object
smoker         object
day            object
time           object
size            int64
dtype: object
```

In [41]:
```
# Categorical Variables
tips.day.unique()
```

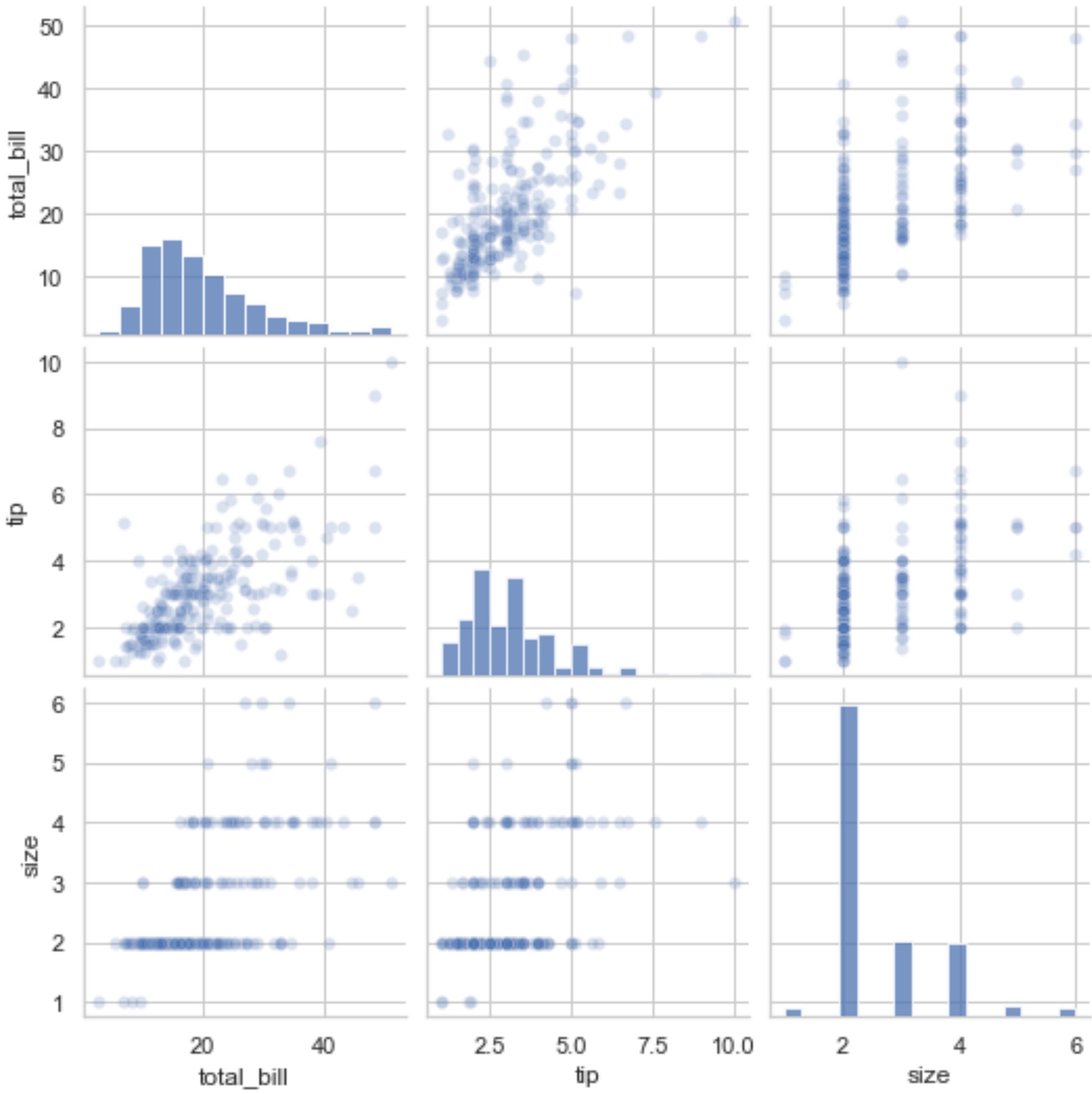
Out[41]:
```
array(['Sun', 'Sat', 'Thur', 'Fri'], dtype=object)
```

The relationship is evidently nonlinear.

MATRIX PLOTS

In [42]:
```
# Matrix Plot
sns.pairplot(tips, diag_kind='hist', plot_kws={'alpha': 0.2})
```

Out[42]: <seaborn.axisgrid.PairGrid at 0x7fd34fb9dfa0>

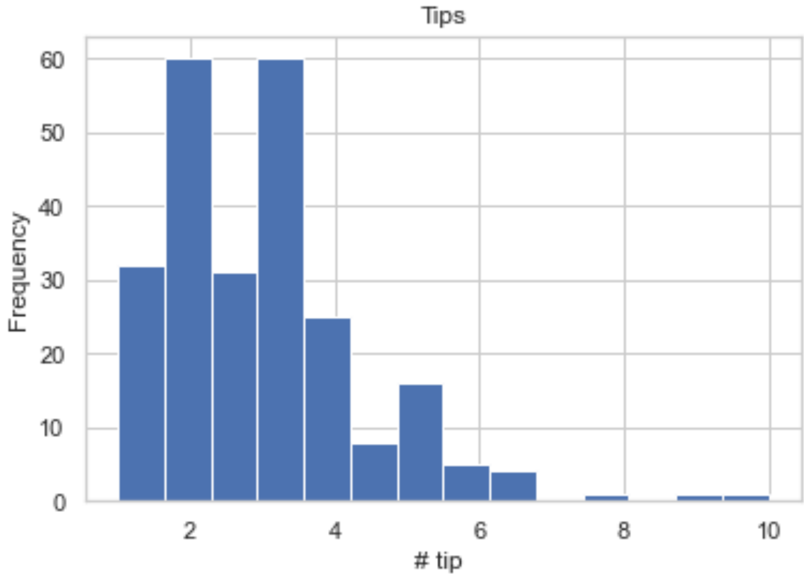


HISTOGRAMS

In [43]:

```
tips.hist(['tip'], bins=14)
plt.title('Tips')
plt.ylabel('Frequency')
plt.xlabel('# tip ')
```

Out[43]: Text(0.5, 0, '# tip ')



Long tails to identify skew When data trail off in one direction, the distribution has a **long tail**. If a distribution has a long left tail, it is **Left Skewed**. If a distribution has a long right tail, it is **Right Skewed**.

Modal Distribution

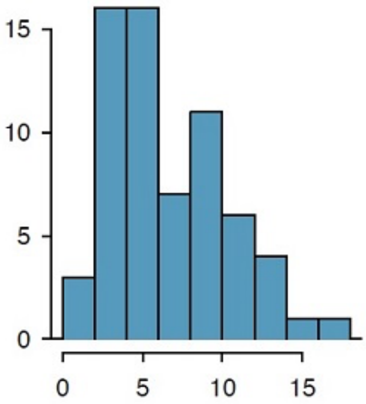
In addition to looking at whether a distribution is Skewed or Symmetric, histograms can be used to identify **Modes**.
$%$
A **mode** is the *value with the* ***most occurrences***.
$%$
However, It is common to have **no** observations with the same value in a dataset, which makes, **mode**, useless for many real datasets.

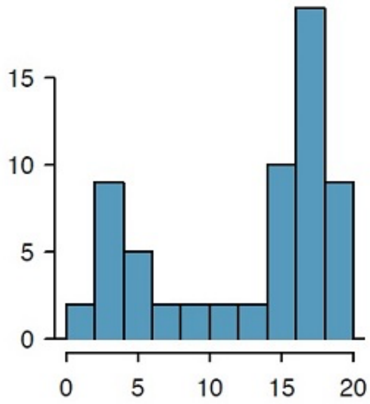
A **mode** is represented by a prominent peak in the distribution. There is only one prominent peak in the histogram of **num_char**.

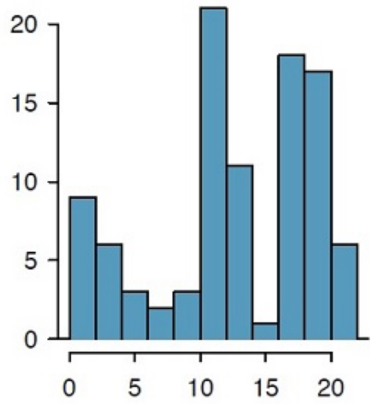
Histogram that have one, two, or three prominent peaks are called **Unimodal,$\:$ Bimodal,$\:$ and $\:$ Multimodal,** respectively.

Any distribution with more than 2 prominent peaks is called Multimodal.

Notice that there was **one prominent peak** in the Unimodal distribution with a **second less prominent peak** that was **not counted** since it only differs from its neighboring **bins** by a few observations.







Looking for modes Looking for **modes** isn't about finding a clear and correct answer about the number of **modes** in a **distribution**. The important part of this examination is to better understand your data and how it might be structured.

Statistical Foundations for Data Scientist

\begin{align*}Alex\:Kumenius\end{align*}

\begin{align*}Business\hspace{2mm}Intelligence\hspace{2mm}and\hspace{2mm}Data\hspa
$%$ \begin{align*}Date : Desembre\hspace{2mm}2020\end{align*}

SUMMARY STATISTICS

Mean - Average

In [44]:
```
tips.head()
```

Out[44]:

	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

In [45]:
```
tips.describe()
```

Out[45]:

	total_bill	tip	size
count	244.000000	244.000000	244.000000
mean	19.785943	2.998279	2.569672
std	8.902412	1.383638	0.951100
min	3.070000	1.000000	1.000000
25%	13.347500	2.000000	2.000000
50%	17.795000	2.900000	2.000000
75%	24.127500	3.562500	3.000000
max	50.810000	10.000000	6.000000

In [46]:

```
tips.info()
```

```
<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    object
 3   smoker      244 non-null    object
 4   day         244 non-null    object
 5   time        244 non-null    object
 6   size        244 non-null    int64
dtypes: float64(2), int64(1), object(4)
memory usage: 13.5+ KB
```

In [47]:

```
#dbe.num_char.mean().round(3)

#changed to:
round(tips.tip.mean(),3)
```

Out[47]:

```
2.998
```

The mean of tip is **$2.99**.

Mean The **sample mean** **$\bar{x}$** of a *numerical variable* is computed as the **sum** of all of the *observations* **divided** by the number of *observations*: $%$ \begin{align*}\bar{x}\:=\:\frac{x_1\:+\:x_2\:+\:· · ·\:+\:x_n}{n}\end{align*} $%$ where **x_1, x_2, . . . ,x_n** represent the **n** observed values. $%$ It is useful to think of the **mean** as the balancing point of the **distribution**.

EXERCISE - 3.1

Compare both Equations above.

- What does x_1 correspond to ?,
- and **x_2** ?
- Can you infer a general meaning to what x_i might represent?
- What was n in this sample?

SOLUTION - 3.1

- x_1 corresponds to the tip of the first total bill ,
- x_2 corresponds to tip of the second total bill, , and
- x_i corresponds to the tips in the i^{th} total bills in the dataset.
- The sample size was n = **244**.

> **Population Mean** The **Population mean** has a special label : μ. The symbol μ is the $Greek$ letter mu and represents the average/mean of all observations in the **Population**. $%$ Sometimes a subscript, such as **$_x$**, is used to represent which variable the **population mean** refers to, e.g. **μ_x**

EXERCISE - 3.2

The average number of tips (**population**) can be estimated using the **sample data**.

Based on the sample of **244** $tips$, what would be a reasonable estimate of **μ_x**, the **mean** value of tips?

Variance and Standard Deviation

In [48]:
```
tips.tip.mean()- tips.tip.std()
```

Out[48]: 1.6146404995234076

Variance

The **mean** was introduced as a method to describe the center of a data set, but the **variability in the data** is also **important**.

We introduce **two measures of variability**: the **Variance** and the **Standard Deviation**. Both are very useful in data analysis.

The **Standard Deviation** describes **how far away** the typical **observation** is from the mean.

We call the *distance of an observation from its mean* its **Deviation**.

Below are the **deviations** for the 1st, 2nd, 3rd, and 50th observations in the **num_char** variable. For computational convenience, the number of characters is listed in the thousands and rounded to the first decimal.

In [49]:
```
tips.tip.iloc[[1], ]
```

Out[49]:
```
1    1.66
Name: tip, dtype: float64
```

Sample Variance s^2 We divide by $n\:–\:1$, rather than dividing by n, when computing the **Variance**. $%$ squaring the deviations does two things: - First, it makes large values much larger, seen by comparing 10.1^2, $(–4.6)^2$, $(–11.0)^2$, and 4.2^2. - Second, it gets rid of any negative signs. $%$ The **variance** is roughly the average squared distance from the **mean**.

Standard Deviation

Standard Deviation Formulas and methods used to compute the **Variance** and **Standard Deviation** for a **Population** are similar to those used for a **sample** *(The only difference is that the **Population Variance** has a division by n instead of $n - 1$)*. $%$ However, like the **Mean**, the **Population** values have special symbols : - σ^2 for the **Variance** and - σ for the **Standard Deviation**. The symbol σ is the $Greek$ letter $sigma$.

In [50]:
```
round(tips.tip.std(),2)
```

Out[50]:
```
1.38
```

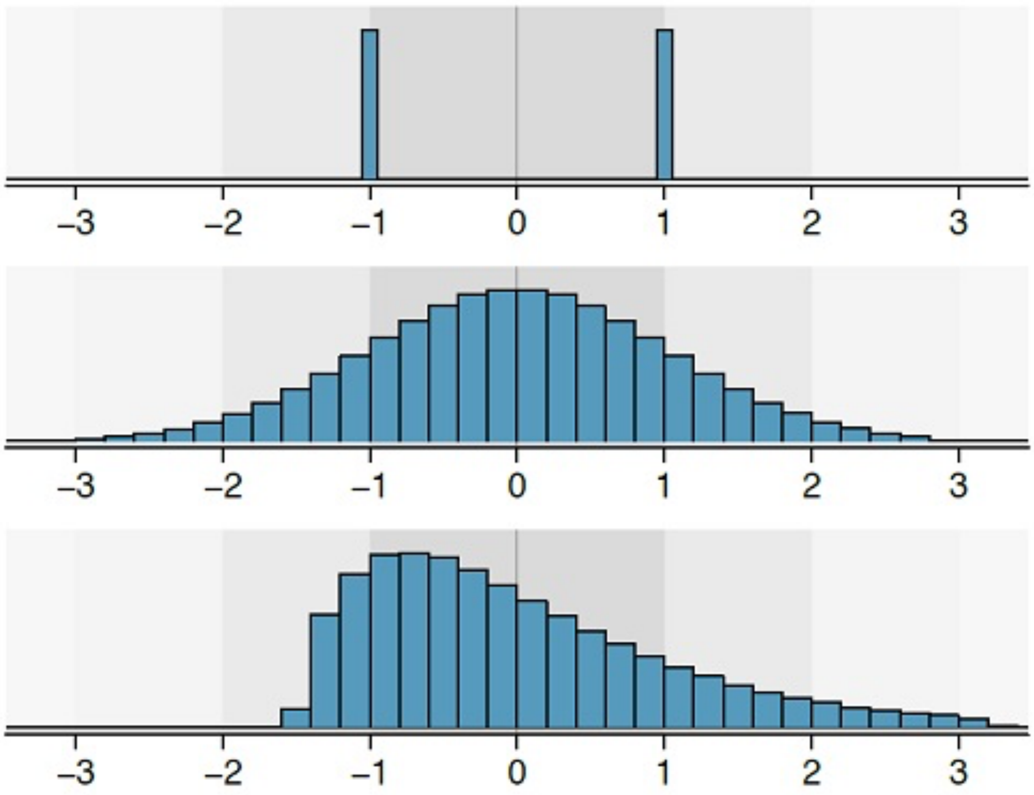
Standard Deviation describes Variability, so focus on the conceptual meaning of the Standard Deviation as a descriptor of Variability rather than the formulas.

Usually **70%** of the data will be within one standard deviation of the mean and about **95%** will be within two standard deviations two standard deviations. However, these **percentages are not strict rules**.

EXERCISE - 3.6

A good description of the shape of a distribution should include modality and whether the distribution is symmetric or skewed to one side.

Explore the figure as an example, explain why such a description is important :



SOLUTION - 3.6

Figure shows three distributions that look quite different, but all have the same **Mean, Variance**, and **Standard Deviation**.

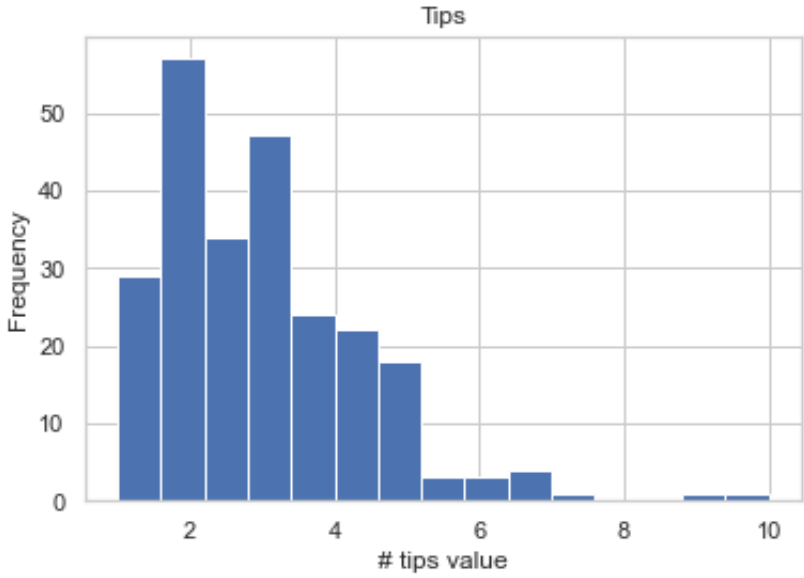
Using **Modality**, we can distinguish between the first plot (**bimodal**) and the last two (**unimodal**).

Using **Skewness**, we can distinguish between the last plot (**right skewed**) and the first two.

While a picture, like a **histogram**, tells a more **complete** story, we can use **Modality** and shape (**Symmetry/Skew**) to characterize basic information about a **distribution**.

In [51]:
```
tips.hist(['tip'], bins=15)
plt.title('Tips')
plt.ylabel('Frequency')
plt.xlabel('# tips value')
```

Out[51]: Text(0.5, 0, '# tips value')



In [52]:
```
round(tips.tip.std(),2)
```

Out[52]: 1.38

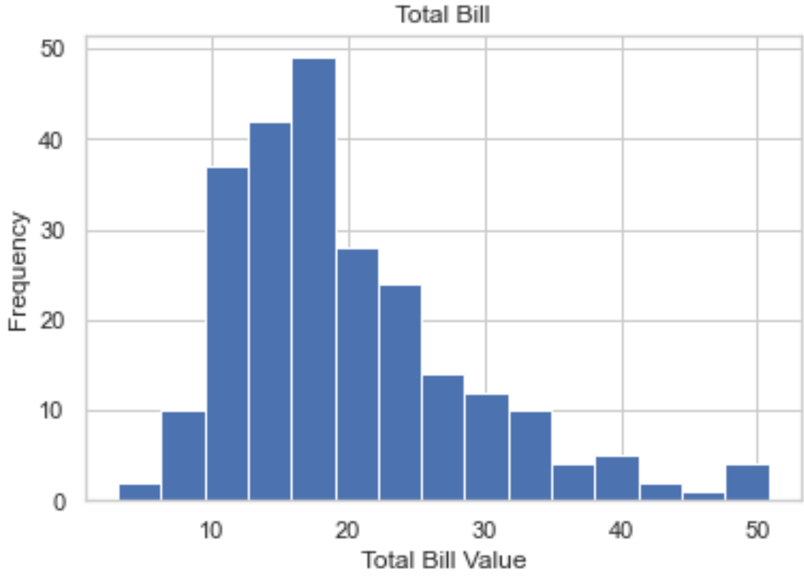
We will use the **Variance** and **Standard Deviation** to **assess how close** the Sample Mean ($\bar{x}$) is to the Population Mean (μ).

variable	description
`name`	County name
`state`	State where the county resides (also including the District of Columbia)
`pop2000`	Population in 2000
`pop2010`	Population in 2010
`fed_spend`	Federal spending per capita
`poverty`	Percent of the population in poverty
`homeownership`	Percent of the population that lives in their own home or lives with the owner (e.g. children living with parents who own the home)
`multiunit`	Percent of living units that are in multi-unit structures (e.g. apartments)
`income`	Income per capita
`med_income`	Median household income for the county, where a household's income equals the total income of its occupants who are 15 years or older
`smoking_ban`	Type of county-wide smoking ban in place at the end of 2011, which takes one of three values: `none`, `partial`, or `comprehensive`, where a `comprehensive` ban means smoking was not permitted in restaurants, bars, or workplaces, and `partial` means smoking was banned in at least one of those three locations

In [53]:
```
fig = plt.figure(figsize=(10,8))

tips.hist(['total_bill'], bins=15)
plt.title('Total Bill')
plt.ylabel('Frequency')
plt.xlabel('Total Bill Value')
plt.show()
```

```
<Figure size 720x576 with 0 Axes>
```



BOX PLOTS

A Box Plot summarizes a dataset using ***five statistics*** while also plotting unusual observations - **Anomalies or Outliers**.

Quartiles, and the Median

In [54]:
```python
(tips['tip']).describe()
```

Out[54]:
```
count    244.000000
mean       2.998279
std        1.383638
min        1.000000
25%        2.000000
50%        2.900000
75%        3.562500
max       10.000000
Name: tip, dtype: float64
```

The median (6,890), splits the data into the bottom 50% and the top 50%, marked in the dot plot by horizontal dashes and open circles, respectively.

In [55]:
```python
round((tips['tip']).median(), 3)
```

Out[55]:
```
2.9
```

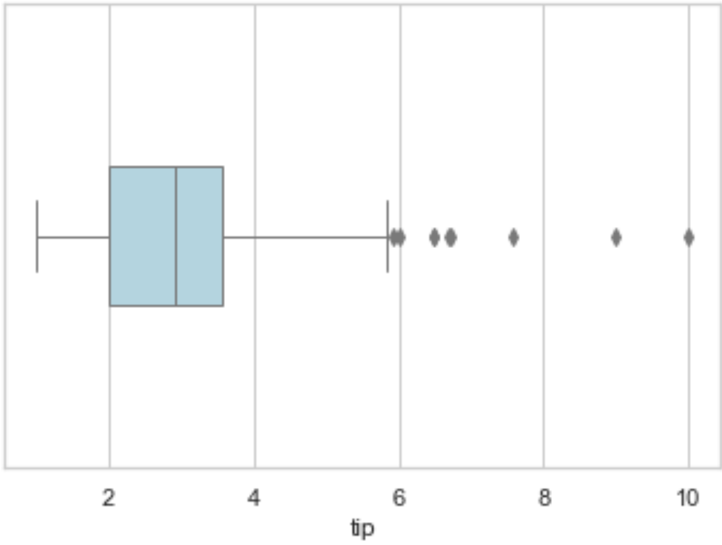
The first step in building a **box plot** is drawing a dark line denoting the **median**, which **splits** the data in half. **50%** of the data falling below the **median** and other **50%** falling above the **median**.

There are 50 character counts in the **dataset** (an even number) so the data are perfectly split into two groups of 25. We take the **median** in this case to be the **average** of the two observations closest to the 50th percentile:

$(6,768+7,012)/2 = 6,890$.

When there are an odd number of observations, there will be exactly one observation that splits the data into two halves, and in such a case that observation is the **median** (no average needed).

In [56]:
```
sns.set(style="whitegrid")
ax = sns.boxplot(x=tips["tip"], color='lightblue', fliersize=5,  orient='v
```



Median If the data are **ordered from smallest to largest**, the **median** is the **observation** right in the **middle**. If there are an even number of observations, there will be two values in the middle, and the **median** is taken as their average.

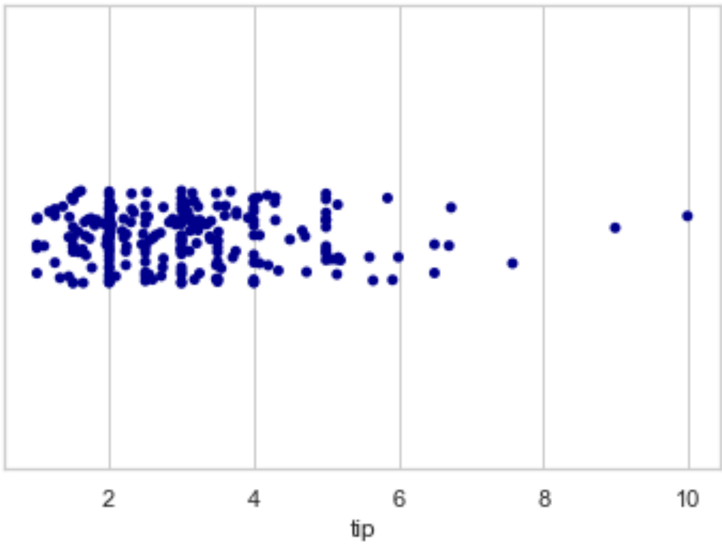
The second step in building a box plot is drawing a rectangle to represent the middle $50%$ of the data. The total length of the box, is called the **interquartile range (IQR)**. It, like the **Standard Deviation**, is a measure of **Variability** in data. The more variable the data, the larger the **Standard Deviation** and **IQR**.

The **two boundaries** of the box are called the **first quartile** (the 25^{th} percentile), i.e. $25%$ of the data fall below this value and the **third quartile** (the 75^{th} percentile), and these are often labeled $Q1$ and $Q3$, respectively.

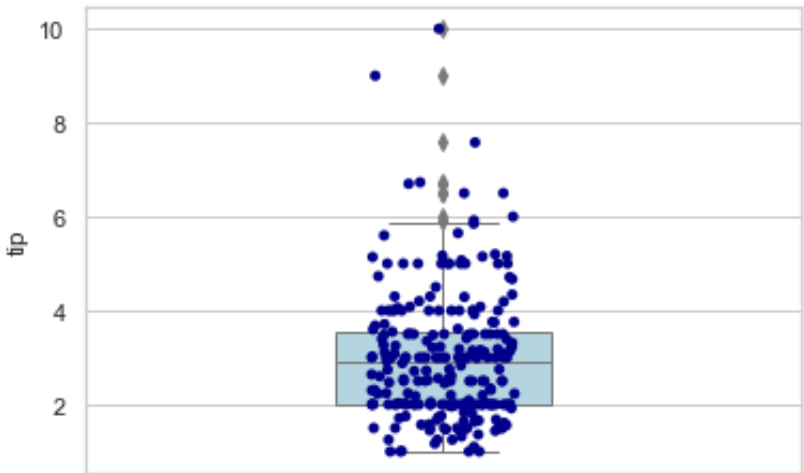
Interquartile range (IQR) The IQR is the length of the box in a box plot. It is computed as $$IQR = Q3 – Q1$$ where $Q1$ and $Q3$ are the 25^{th} and 75^{th} percentiles.

In [57]:
```
sns.stripplot(x=tips["tip"], orient='v', color='darkblue')
```

Out[57]: <AxesSubplot:xlabel='tip'>



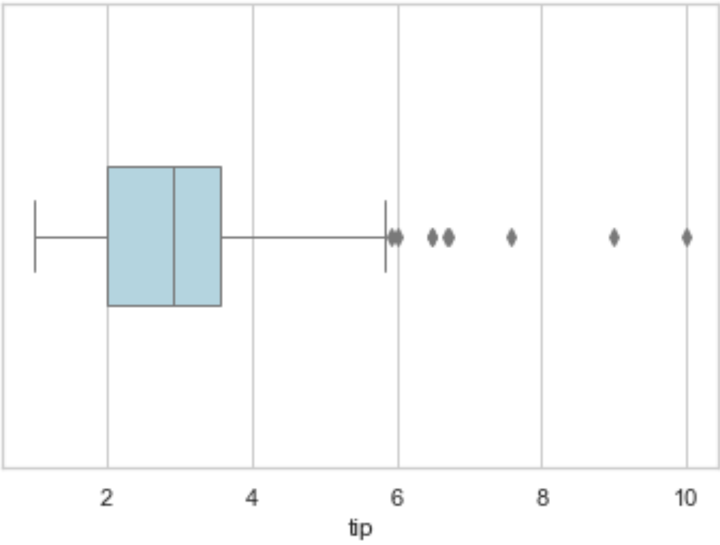
In [58]:
```
ax = sns.boxplot(y="tip", data=tips,  color='lightblue', fliersize=5,  orie
ax = sns.stripplot(y=tips["tip"], orient='v', color='darkblue')
```



In [59]:
```
tips.tip
```

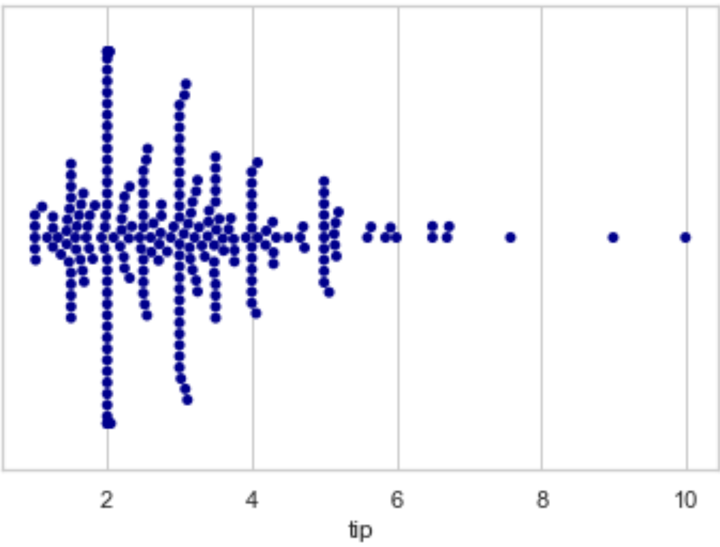
Out[59]:
```
0       1.01
1       1.66
2       3.50
3       3.31
4       3.61
        ...
239     5.92
240     2.00
241     2.00
242     1.75
243     3.00
Name: tip, Length: 244, dtype: float64
```

In [60]:
```
sns.set(style="whitegrid")
ax = sns.boxplot(x=tips["tip"], color='lightblue', fliersize=5,  orient='v
```

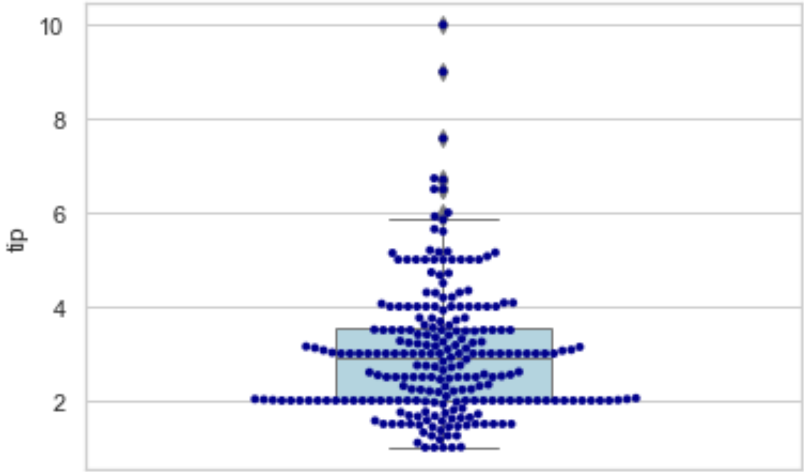


In [61]:
```
sns.swarmplot(x=tips["tip"], orient='v', color='darkblue')
```

Out[61]: <AxesSubplot:xlabel='tip'>



In [62]:
```
ax = sns.boxplot(y="tip", data=tips,  color='lightblue', fliersize=5,  orie
ax = sns.swarmplot(y="tip", data=tips, color="darkblue", orient="v", size=4
```



EXERCISE - 3.8

1. What percent of the data fall between Q1 and the median?
2. What percent is between the median and Q3?

SOLUTION - 3.8

1. Since $Q1$ and $Q3$ capture the middle **50%** of the data and the median splits the data in the middle,
2. **25%** of the data fall between $Q1$ and the median, and another **25%** falls between the median and $Q3$.

Extending out from the box, the whiskers attempt to capture the data outside of the box, however, their reach is never allowed to be more than

$1.5\hspace{2mm}x\hspace{2mm}IQR$

They capture everything within this reach. The **upper whisker** does not extend to the last three points, which is beyond $Q3\hspace{2mm}+\hspace{2mm}1.5\hspace{2mm}x\hspace{2mm}IQR$, and so it extends only to the last point below this limit.

The **lower whisker** stops at the lowest value, **33**, since there is no additional data to reach; the **lower whisker's limit** is not shown in the figure because the plot does not extend down to $Q1\hspace{2mm}-\hspace{2mm}1.5\hspace{2mm}x\hspace{2mm}IQR$. In a sense, the box is like the body of the box plot and the whiskers are like its arms trying to reach the rest of the data.

EXERCISE - 3.9

estimate the following values for **tip** in the $tips$ dataset:

a).- $Q1$,
b).- $Q3$, and
c).- IQR

SOLUTION - 3.9

These visual estimates will vary a little from one person to the next: Q1 = 2, Q3 = 2.9, IQR = Q3 - Q1 = 0.9.

In [ ]:

```
```