Pandas 4 - Data Visualization

Data analysis process

Data Understanding	•Descriptive statistics •Types of data (numerical/categorical)
Data Preprocessing	•Basic operations •Subset selection and data consolidation •Missing data handling
Calculation (Modeling)	•Basic calculation •Data aggregation
Data Visualization	•Univariate chart •Bivariate chart •Multivariate chart

Outline

- Pivot
- X-axis with categorical data
  - (Line chart)
  - Bar chart
  - Area chart
  - Pie chart
- Numerical data
  - Histogram
  - Scatter plot
  - Hexagon plot

Wide-form and Long-form

- **Long-form**: Each row is one time point per target. The data of a target can have multiple rows.
- **Wide-form**: A target's repeated responses will be in a single row, and each response is in a separate column.

Target: ID

	ID	Exam	Score
0	S01	Exam-1	95
1	S02	Exam-1	75
2	S03	Exam-1	70
3	S04	Exam-1	65
4	S05	Exam-1	85
5	S01	Exam-2	85
6	S02	Exam-2	70
7	S03	Exam-2	90
8	S04	Exam-2	55
9	S05	Exam-2	60

Long-form

Exam / ID	Exam-1	Exam-2
S01	95	85
S02	75	70
S03	70	90
S04	65	55
S05	85	60

Wide-form

Pivot

- The `pivot()` function is used to reshaped a DataFrame by passing arguments: index, columns and values.

	ID	Exam	Score
0	S01	Exam-1	95
1	S02	Exam-1	75
2	S03	Exam-1	70
3	S04	Exam-1	65
4	S05	Exam-1	85
5	S01	Exam-2	85
6	S02	Exam-2	70
7	S03	Exam-2	90
8	S04	Exam-2	55
9	S05	Exam-2	60

Long-form

```
score_df.pivot(index = "ID", columns = "Exam", values = "Score")
```

Exam	Exam-1	Exam-2
ID		
S01	95	85
S02	75	70
S03	70	90
S04	65	55
S05	85	60

Wide-form

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.pivot.html

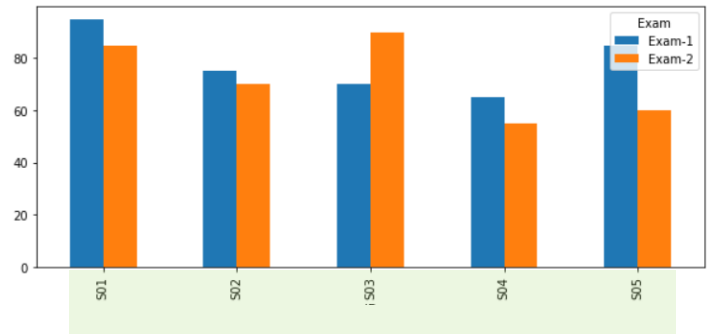
Pivot

- Change the target.

	ID	Exam	Score
0	S01	Exam-1	95
1	S02	Exam-1	75
2	S03	Exam-1	70
3	S04	Exam-1	65
4	S05	Exam-1	85
5	S01	Exam-2	85
6	S02	Exam-2	70
7	S03	Exam-2	90
8	S04	Exam-2	55
9	S05	Exam-2	60

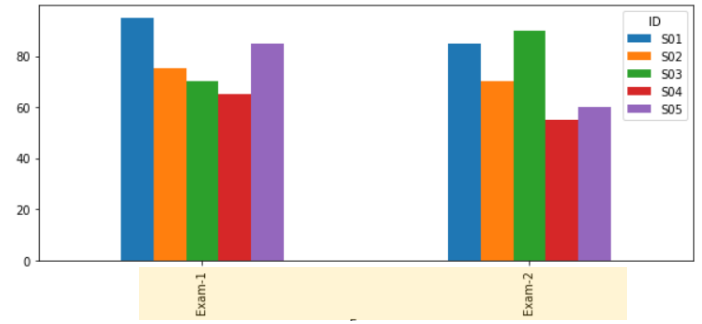
```
score_df.pivot(index = "ID", columns = "Exam", values = "Score")
```

Exam	Exam-1	Exam-2
ID		
S01	95	85
S02	75	70
S03	70	90
S04	65	55
S05	85	60




```
score_df.pivot(index = "Exam", columns = "ID", values = "Score")
```

ID	S01	S02	S03	S04	S05
Exam					
Exam-1	95	75	70	65	85
Exam-2	85	70	90	55	60



Pivot table: https://pandas.pydata.org/docs/reference/api/pandas.pivot_table.html
Melt: https://pandas.pydata.org/docs/reference/api/pandas.melt.html

Pivot

- Example

Target: Month

	Product	Quarter	Month	Sales
0	A	Q1	Jan	67
1	A	Q1	Feb	57
2	A	Q1	Mar	87
3	A	Q2	Apr	50
4	A	Q2	May	97
5	A	Q2	Jun	68
6	B	Q1	Jan	78
7	B	Q1	Feb	102
8	B	Q1	Mar	113
9	B	Q2	Apr	98
10	B	Q2	May	80
11	B	Q2	Jun	84

Long-form

```
sales_df_wide = sales_df.pivot(index = "Month", columns="Product", values = "Sales")
sales_df_wide
```

Product	A	B
Month		
Apr	50	98
Feb	57	102
Jan	67	78
Jun	68	84
Mar	87	113
May	97	80

Wide-form

```
sales_df_wide = sales_df_wide.reindex(["Jan","Feb","Mar","Apr","May","Jun"])
sales_df_wide
```

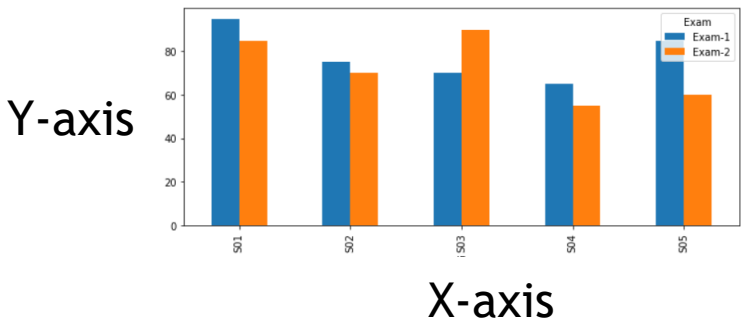
Product	A	B
Month		
Jan	67	78
Feb	57	102
Mar	87	113
Apr	50	98
May	97	80
Jun	68	84

Use reindex() to make the DataFrame conform to the new index.

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.reindex.html

Outline

- Pivot
- X-axis with categorical data
  - (Line chart)
  - Bar chart
  - Area chart
  - Pie chart
- Numerical data
  - Histogram
  - Scatter plot
  - Hexagon plot



Line chart - Series

- Both Series and DataFrame have a `plot()` method to make some basic plot types. By default, plot() makes line charts
- Line chart is usually used to visualize the trend of data over a period of time.

	Product	Quarter	Month	Sales
0	A	Q1	Jan	67
1	A	Q1	Feb	57
2	A	Q1	Mar	87
3	A	Q2	Apr	50
4	A	Q2	May	97
5	A	Q2	Jun	68
6	B	Q1	Jan	78
7	B	Q1	Feb	102
8	B	Q1	Mar	113
9	B	Q2	Apr	98
10	B	Q2	May	80
11	B	Q2	Jun	84

(1) Select a series

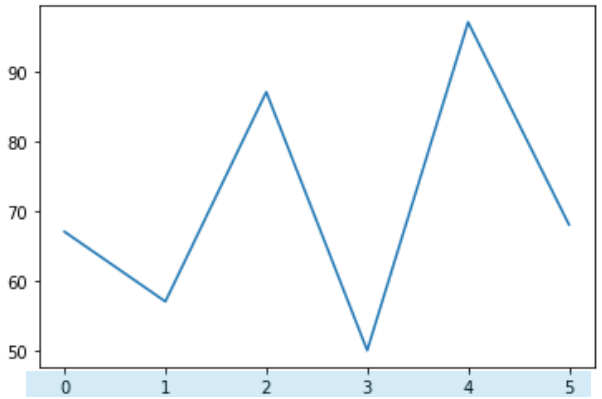
```
series_A = sales_df[sales_df.Product == "A"].Sales
series_A
```

```
0    67
1    57
2    87
3    50
4    97
5    68
Name: Sales, dtype: int64
```

(2) Call plot() function

```
series_A.plot()
```

```
<AxesSubplot:>
```



Use index as xlabel on x-axis.

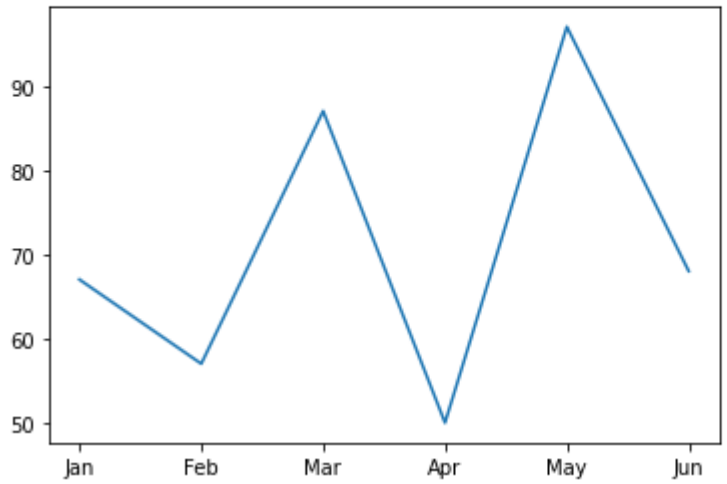
https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.plot.html

Line chart - Series

- Assign a new index.

```
series_A.index = ["Jan","Feb","Mar","Apr","May","Jun"]
series_A.plot()
```

```
<AxesSubplot:>
```



Line chart - DataFrame

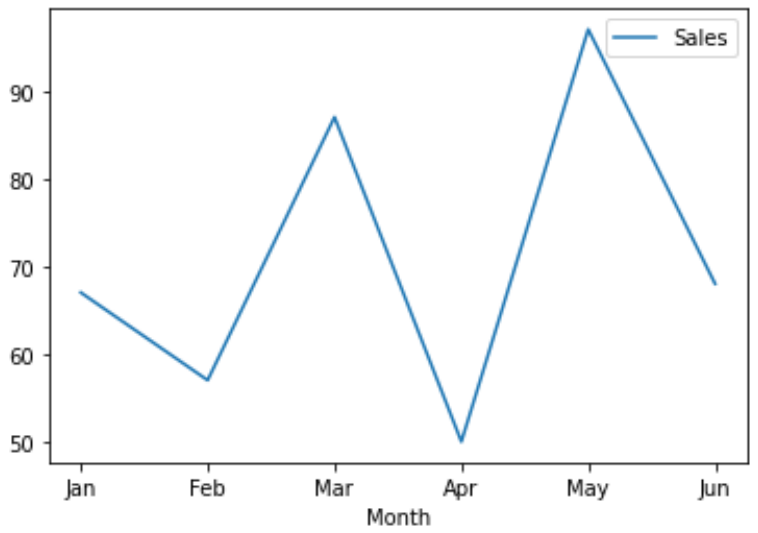
- Use the arguments x and y to specify the columns used for plotting.

	Product	Quarter	Month	Sales
0	A	Q1	Jan	67
1	A	Q1	Feb	57
2	A	Q1	Mar	87
3	A	Q2	Apr	50
4	A	Q2	May	97
5	A	Q2	Jun	68
6	B	Q1	Jan	78
7	B	Q1	Feb	102
8	B	Q1	Mar	113
9	B	Q2	Apr	98
10	B	Q2	May	80
11	B	Q2	Jun	84

Long-form

```
# Select the rows of product A
sales_df[sales_df.Product == "A"].plot(x = "Month", y = "Sales")
```

```
<AxesSubplot:xlabel='Month'>
```



Line chart - DataFrame

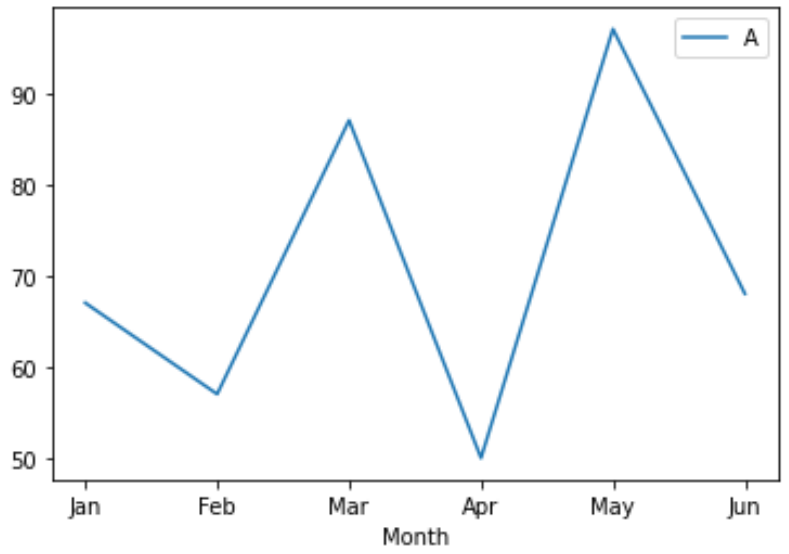
- Use wide-form.

Product	A	B
Month		
Jan	67	78
Feb	57	102
Mar	87	113
Apr	50	98
May	97	80
Jun	68	84

Wide-form

```
# Select the column of product A
sales_df_wide.plot(y = "A")
```

```
<AxesSubplot:xlabel='Month'>
```



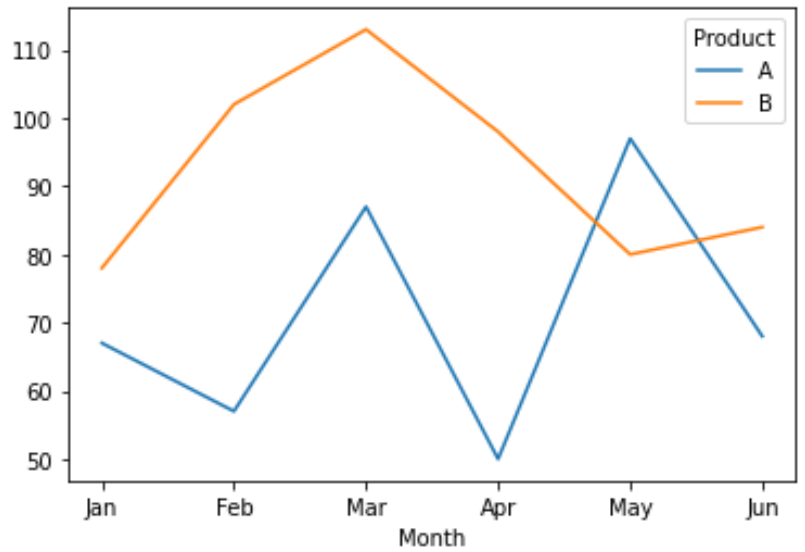
Line chart – Multiple lines

- Pass a list of column names to the argument y to plot multiple lines.

Product	A	B
Month		
Jan	67	78
Feb	57	102
Mar	87	113
Apr	50	98
May	97	80
Jun	68	84

```
sales_df_wide.plot(y= ["A","B"])
```

```
<AxesSubplot:xlabel='Month'>
```



Use index as xlabel on x-axis.

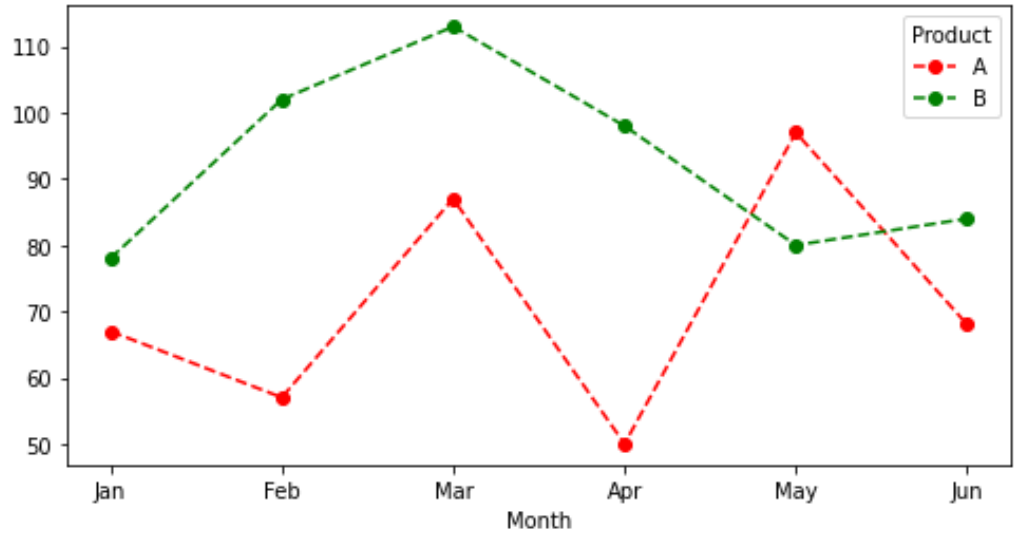
Line chart – Custom style

- Use some arguments to change the style.

Product Month	A	B
Jan	67	78
Feb	57	102
Mar	87	113
Apr	50	98
May	97	80
Jun	68	84

```
sales_df_wide.plot(y= ["A","B"],
                   marker = "o",
                   color = ["red","green"],
                   linestyle = 'dashed',
                   figsize = (8,4))
```

```
<AxesSubplot:xlabel='Month'>
```



- Marker: https://matplotlib.org/stable/api/markers_api.html#module-matplotlib.markers
- Color: https://matplotlib.org/stable/gallery/color/named_colors.html
- Linestyle: https://matplotlib.org/stable/gallery/lines_bars_and_markers/linestyles.html
- Others: https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.plot.html

Exercise

(A.1) Given the dataframe `expense_df`. Convert the data frame to the following format (wide-form) and store the result in a new variable named `expense_df_wide`.

```
expense_df = pd.DataFrame({'month':['Jan','Feb','Mar','Apr','May','Jun', 'Jan','Feb','Mar','Apr','May','Jun'],
                           'expense':[3050, 2800, 2750, 2300, 3150, 2900,1050, 900, 1150, 1850, 1250, 950],
                           'category':['grocery', 'grocery', 'grocery', 'grocery', 'grocery', 'grocery', 'transportation','trans
```

	month	expense	category
0	Jan	3050	grocery
1	Feb	2800	grocery
2	Mar	2750	grocery
3	Apr	2300	grocery
4	May	3150	grocery
5	Jun	2900	grocery
6	Jan	1050	transportation
7	Feb	900	transportation
8	Mar	1150	transportation
9	Apr	1850	transportation
10	May	1250	transportation
11	Jun	950	transportation

	grocery	transportation
Jan	3050	1050
Feb	2800	900
Mar	2750	1150
Apr	2300	1850
May	3150	1250
Jun	2900	950

(A.2) Use reindex() to make the dataframe `expense_df_wide` conform to the new index: ['Jan','Feb','Mar','Apr','May','Jun'].

(A.3) Use the dataframe `expense_df_wide` obtained in (A.2). Draw a multiple line chart to show the monthly groceries and transportation expenses.

(A.4) Import dataset `fashion.csv` and set the first column as the index.
Hint: Use argument `index_col=[0]`.

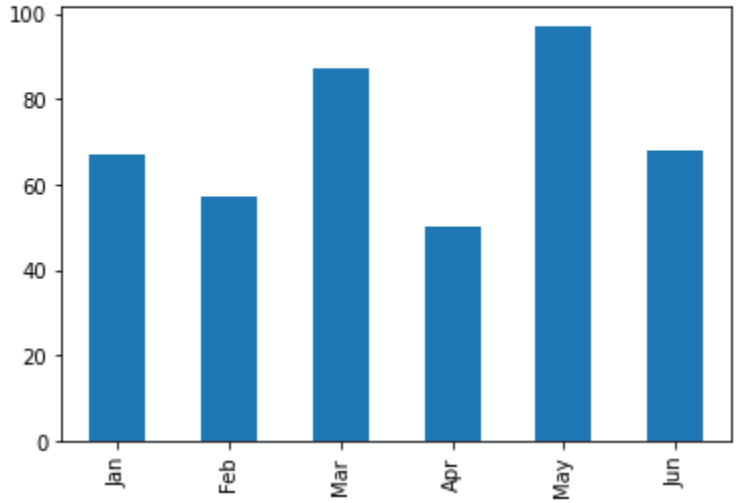
(A.5) Show the sales trends of `Eton`, `Levi_s`, and `Tiger_of_Sweden` with a multiple line chart.
Settings: Use marker = "D", figsize = (12,4).

Bar chart - Series

- Use `kind = “bar”` to plot vertical bar chart.
- Use `kind = “barh”` to plot horizontal bar chart.

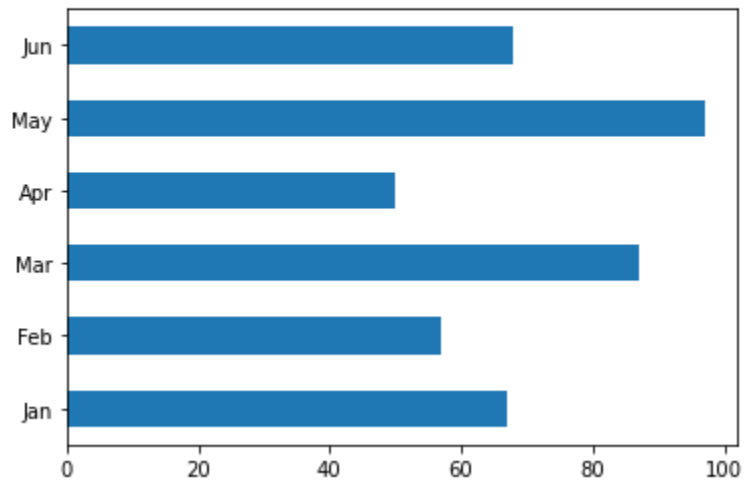
```
series_A.plot(kind = 'bar')
```

```
<AxesSubplot:>
```




```
series_A.plot(kind = 'barh')
```

```
<AxesSubplot:>
```



https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.plot.bar.html

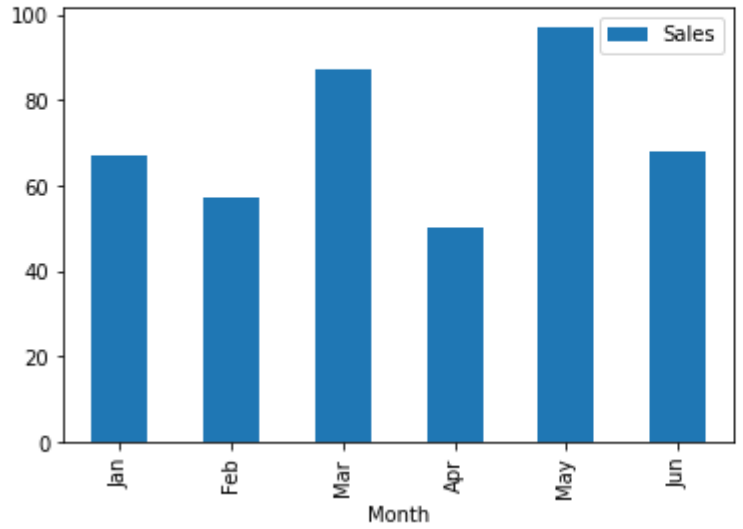
Bar chart - DataFrame

- Use the arguments x and y to specify the columns used for plotting.

	Product	Quarter	Month	Sales
0	A	Q1	Jan	67
1	A	Q1	Feb	57
2	A	Q1	Mar	87
3	A	Q2	Apr	50
4	A	Q2	May	97
5	A	Q2	Jun	68
6	B	Q1	Jan	78
7	B	Q1	Feb	102
8	B	Q1	Mar	113
9	B	Q2	Apr	98
10	B	Q2	May	80
11	B	Q2	Jun	84

```
sales_df[sales_df.Product == "A"].plot(kind = "bar", x = "Month", y = "Sales")
```

```
<AxesSubplot:xlabel='Month'>
```



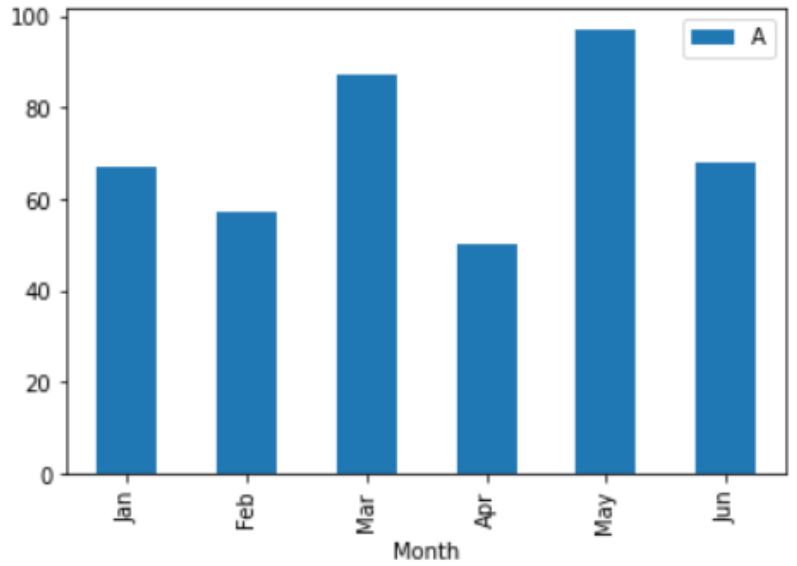
Bar chart - DataFrame

- Use the arguments x and y to specify the columns used for plotting.

Product	A	B
Month		
Apr	50	98
Feb	57	102
Jan	67	78
Jun	68	84
Mar	87	113
May	97	80

```
sales_df_wide.plot(kind = "bar", y = "A")
```

```
<AxesSubplot:xlabel='Month'>
```



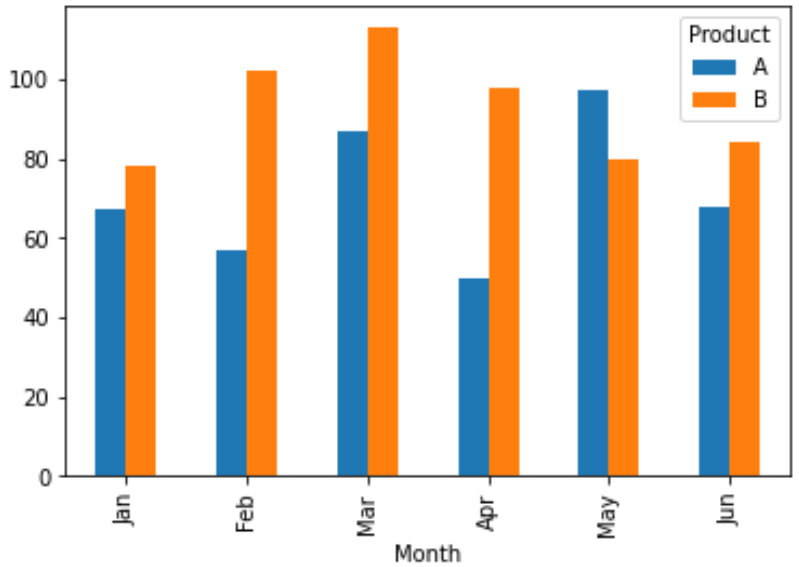
Bar chart – Multiple bars

- Pass a list of column names to the argument y to plot multiple lines.

Product	A	B
Month		
Jan	67	78
Feb	57	102
Mar	87	113
Apr	50	98
May	97	80
Jun	68	84

```
sales_df_wide.plot(kind = "bar", y= ["A","B"])
```

```
<AxesSubplot:xlabel='Month'>
```



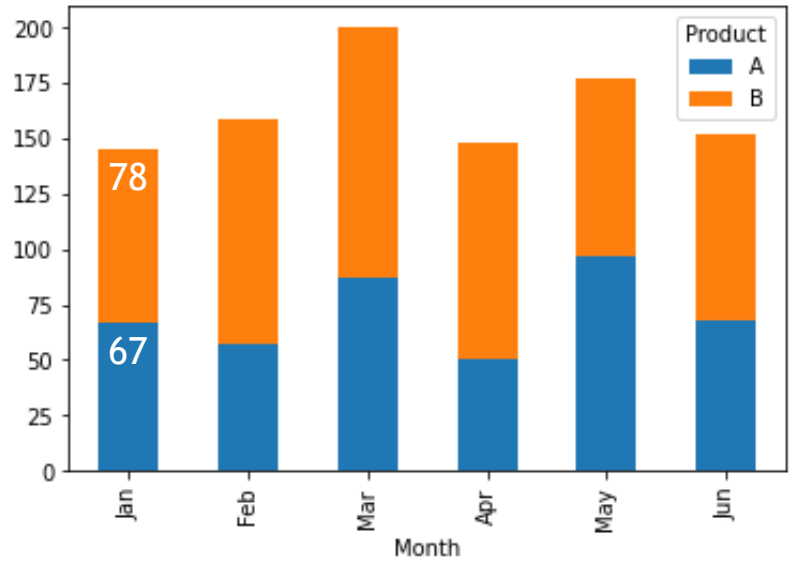
Bar chart – Stacked bar chart

- Stacked bar chart
  - Each bar is stacked by multiple data series.
  - Stacked bar charts can be used to break down and compare parts of the whole.

Product	A	B
Month		
Jan	67	78
Feb	57	102
Mar	87	113
Apr	50	98
May	97	80
Jun	68	84

```
sales_df_wide.plot(kind = "bar", y= ["A","B"], stacked = True)
```

```
<AxesSubplot:xlabel='Month'>
```

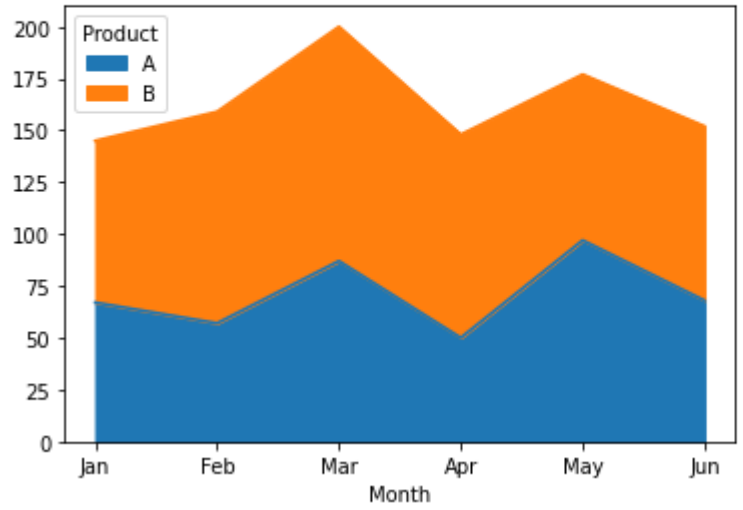


Area chart

- An area chart is similar to a line chart, except that the area between the drawn line and the x-axis is shaded with color.
- Use `kind = "area"` to plot area chart. By default, `stacked = True`.

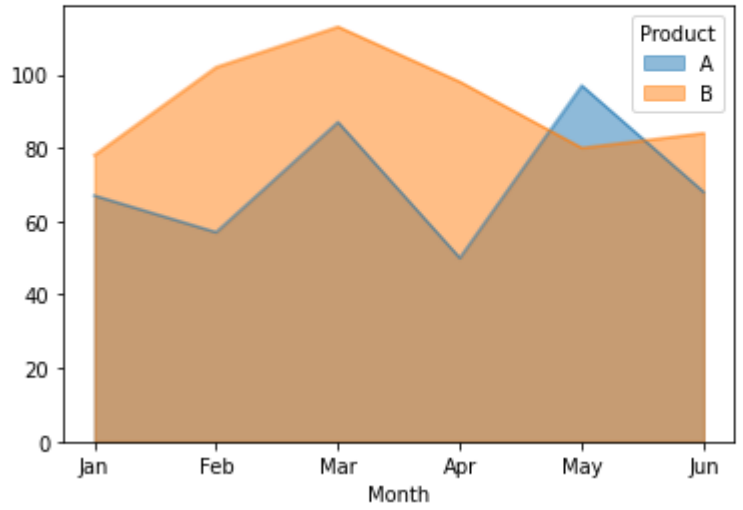
```
sales_df_wide.plot(kind = "area", y= ["A","B"])
```

```
<AxesSubplot:xlabel='Month'>
```




```
sales_df_wide.plot(kind = "area", y= ["A","B"], stacked = False)
```

```
<AxesSubplot:xlabel='Month'>
```



https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.plot.area.html

Pie chart

- The pieces of the pie chart are proportional to the fraction of the whole in each category.

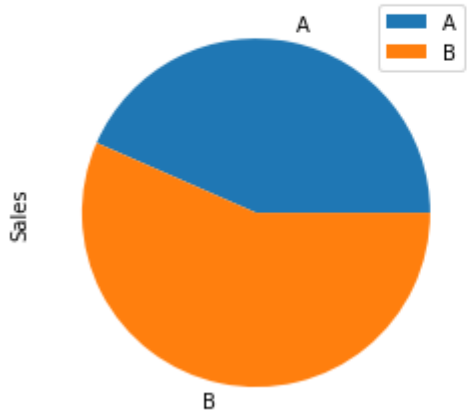
Add up the total sales of each product.

```
sales_df.groupby("Product").sum()
```

Product	Sales
A	426
B	555

```
sales_df.groupby("Product").sum().plot(kind = "pie", y = "Sales")
```

```
<AxesSubplot:ylabel='Sales'>
```



https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.plot.pie.html

Exercise

Exercise.B

(B.1) Use the dataframe `expense_df_wide` obtained in (A.2). Draw a multiple bar chart to show the monthly groceries and transportation expenses.

(B.2) Use the dataframe `expense_df_wide` obtained in (A.2). Draw a stacked area chart to show the monthly groceries and transportation expenses.

(B.3) Import dataset `melbourne.csv`. Select the properties (rows) located in the following suburbs: Reservoir, Richmond, Bentleigh East, Preston.
Hint: DataFrame.Column.isin()

(B.4) Group the data by the column `Suburb` and count the number of properties in each suburb.
Hint: size()

(B.5) Display the results obtained in (B.2) with a pie chart.

Outline

- Pivot
- X-axis with categorical data
  - (Line chart)
  - Bar chart
  - Area chart
  - Pie chart
- Numerical data
  - Histogram
  - Scatter plot
  - Hexagon plot

Histogram - Series

- A histogram is an approximate representation of the distribution of numerical data
  - Step1: Divide the entire range of values into a series of intervals.
  - Step2: Count how many values fall into each interval.

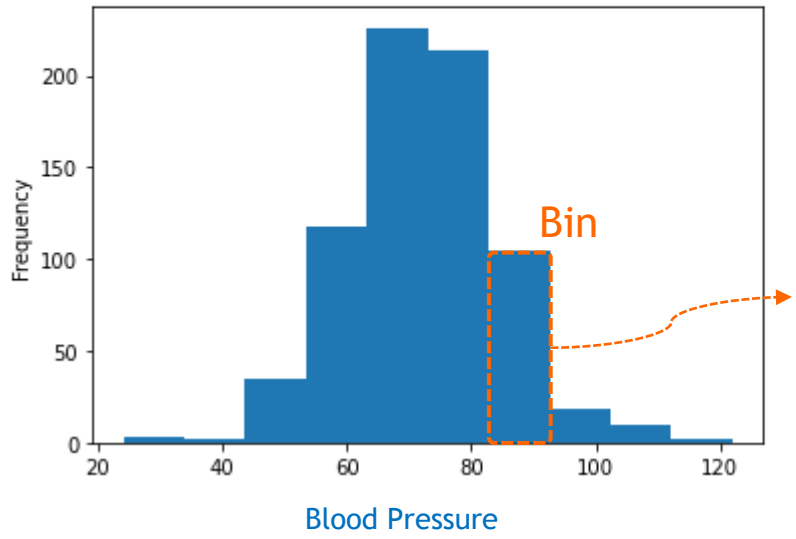
	Pregnancies	Glucose	BloodPressure	SkinThickness
0	6	148	72	35
1	1	85	66	29
2	8	183	64	0
3	1	89	66	23
4	0	137	40	35
...	...	...	...	...
763	10	101	76	48
764	2	122	70	27
765	5	121	72	23
766	1	126	60	0
767	1	93	70	31

By default, the number of bins is 10.

width of bins
= (max – min)/number of bins
= (122-24)/10 = 9.8

```
diabetes_df.BloodPressure.plot(kind = "hist")
```

```
<AxesSubplot:ylabel='Frequency'>
```



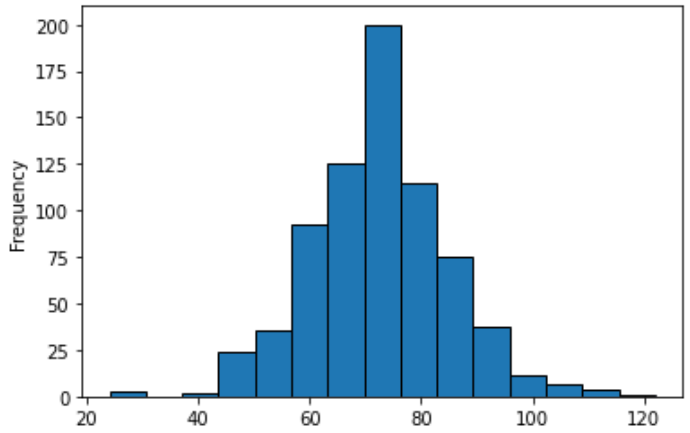
https://pandas.pydata.org/docs/reference/api/pandas.Series.hist.html

Histogram - Series

- Use the arugument "`bins`" to customize the number of bins.
  - Integer: `bins = 15`.
  - A sequence of bin edges: `bins = [0,5,10,..,130]`

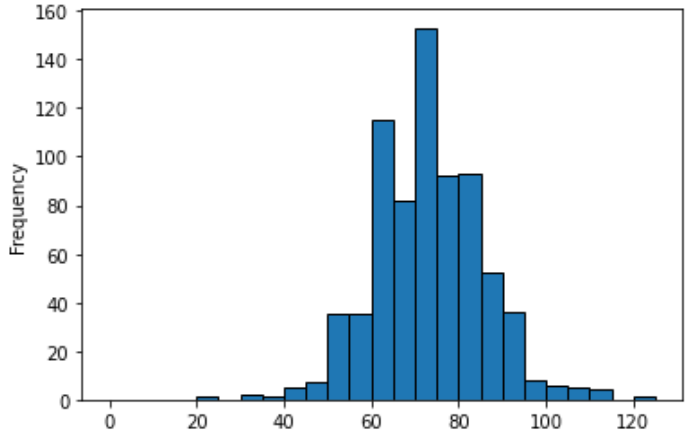
```
diabetes_df.BloodPressure.plot(kind = "hist",
                               bins = 15,
                               edgecolor = "black")
```

```
<AxesSubplot:ylabel='Frequency'>
```




```
diabetes_df.BloodPressure.plot(kind = "hist",
                               bins = range(0,130,5),
                               edgecolor = "black")
```

```
<AxesSubplot:ylabel='Frequency'>
```



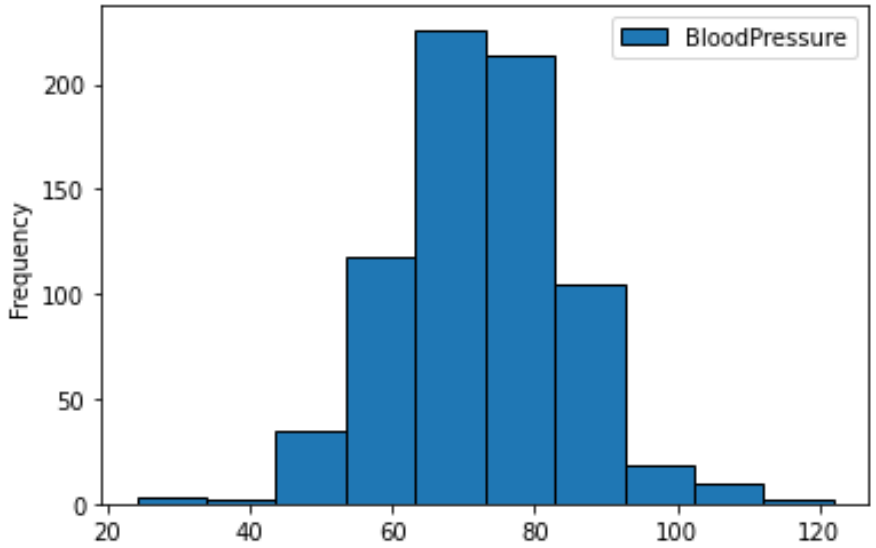
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.hist.html

Histogram - DataFrame

- Use the arguments `y` to specify the column used for plotting.

```
diabetes_df.plot(kind = "hist", y = "BloodPressure", edgecolor = "black")
```

```
<AxesSubplot:ylabel='Frequency'>
```

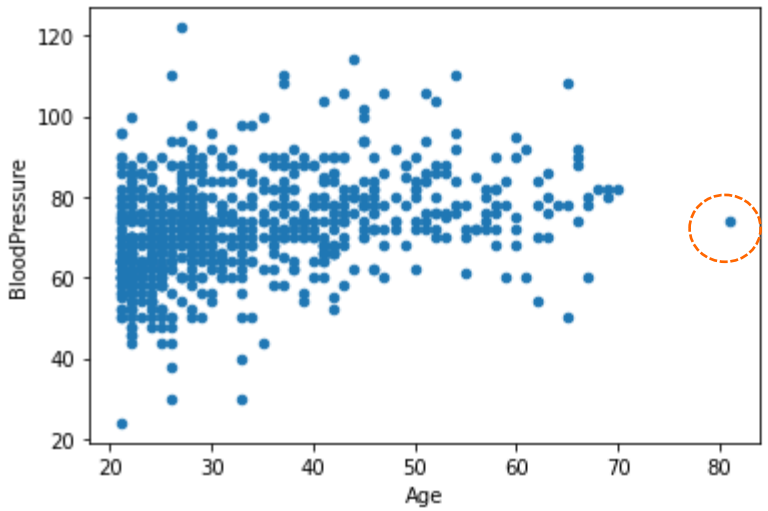


Scatter plot

- Scatter plots are used to observe the relationship between two variables.
  - Each dot indicates an individual data point (observation).

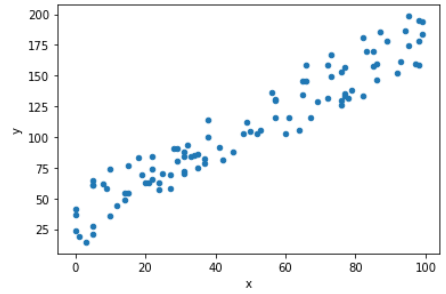
```
diabetes_df.plot(kind = "scatter", x ="Age", y= "BloodPressure")
```

```
<AxesSubplot:xlabel='Age', ylabel='BloodPressure'>
```

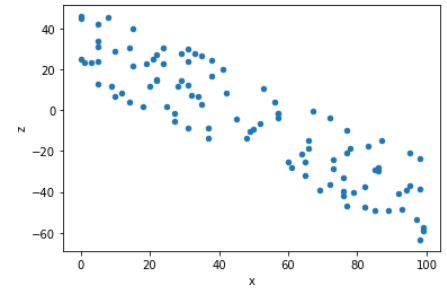


Age = 81,
BloodPressure = 74

Positive correlation



Negative correlation



https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.plot.scatter.html

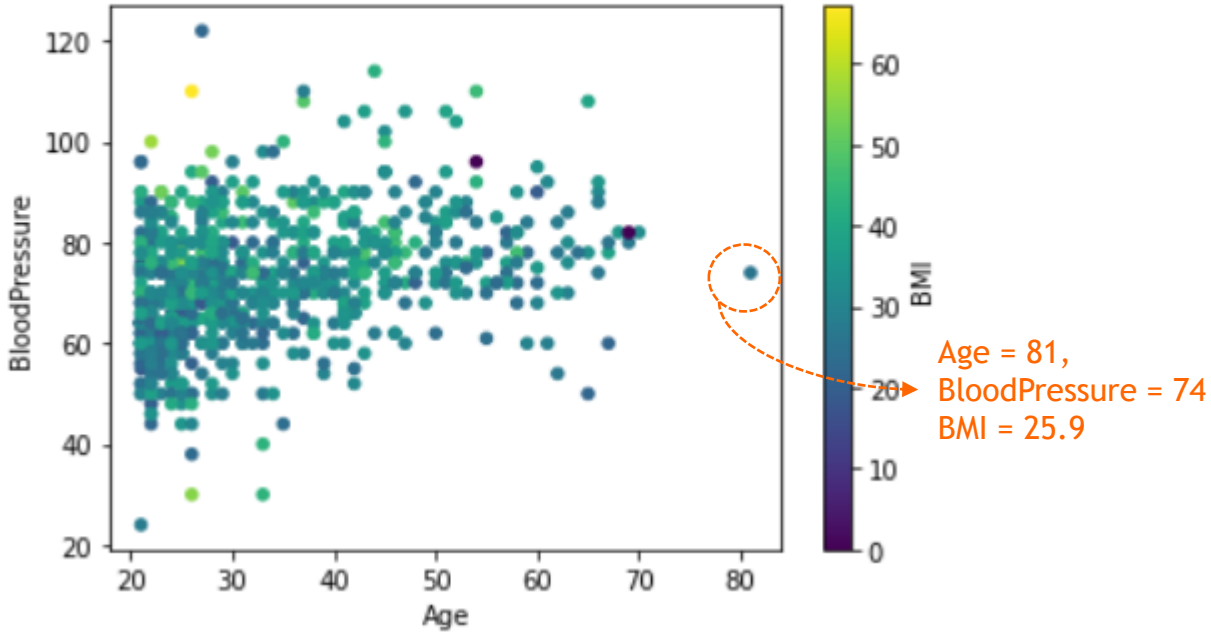
Scatter plot

- Color points based on the third variable.

```
diabetes_df.plot(kind = "scatter", x ="Age", y= "BloodPressure", c= "BMI", cmap = "viridis", sharex=False)
```

```
<AxesSubplot:xlabel='Age', ylabel='BloodPressure'>
```

Color points according to the column BMI



Age = 81,
BloodPressure = 74
BMI = 25.9

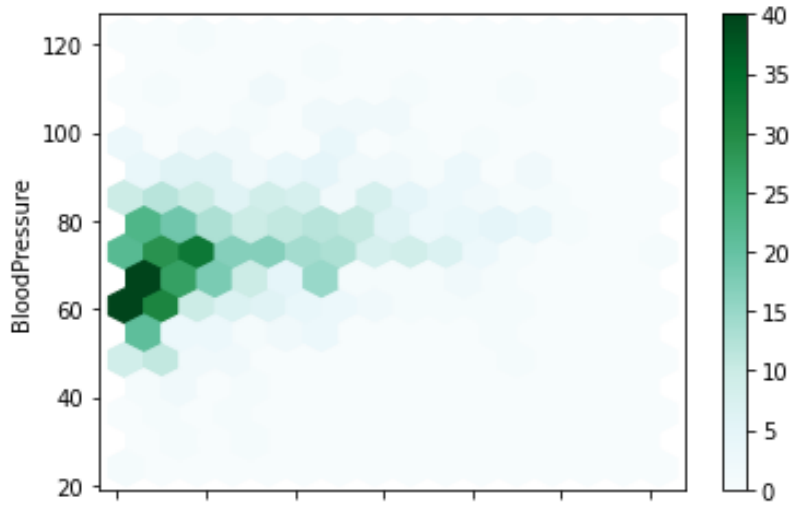
https://matplotlib.org/stable/tutorials/colors/colormaps.html

Hexagon plot

- A hexagon plot combines nearby data points into a hexagon, and then displays the density (the number of data points) in color.
- Hexagon plots can solve the problem that many points begin to overlap.

```
diabetes_df.plot(kind = "hexbin", x ="Age", y= "BloodPressure", gridsize = 15)
```

```
<AxesSubplot:xlabel='Age', ylabel='BloodPressure'>
```



Gridsize: The number of hexagons in the x-direction.

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.plot.hexbin.html

Exercise

(C.1) Import dataset `wine.csv` and set the first column as the index. Display the first 5 rows.

(C.2) Use the following criteria to select a subset.

- Select wines (rows) from Spain, Italy or France (use column `country`).
- Select wines (rows) with a price of less than 200(use column `price`).

(C.3) Use a histogram to show the cost distribution of French wines.
Hint: Use column `price`.

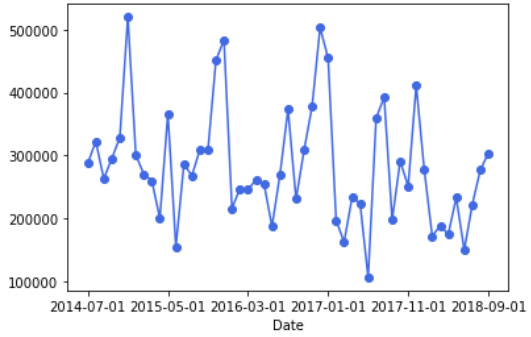
(C.4) Use a scatter plot to show the relationship between wine cost and the number of points recieved in the review.
Hint: Use column `price` and `points`.

(C.5) Use a hexagon plot to show the relationship between wine cost and the number of points recieved in the review.
Hint: Use gridsize = 20.

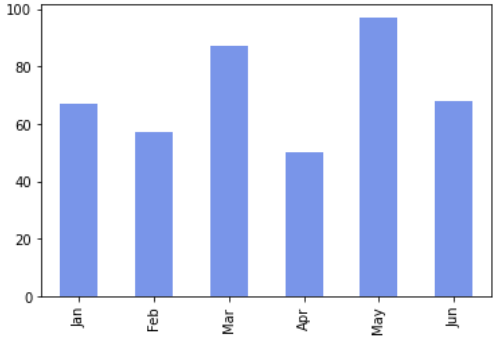
Summary

Univariate chart

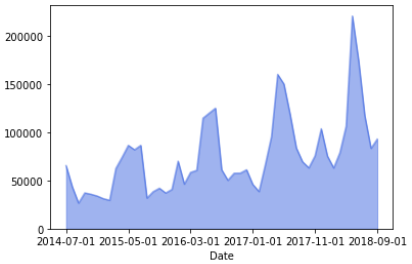
Line chart



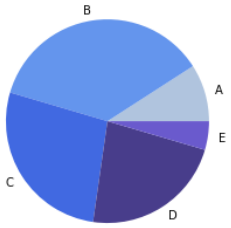
Bar chart



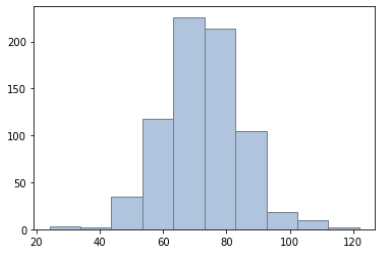
Area chart



Pie chart

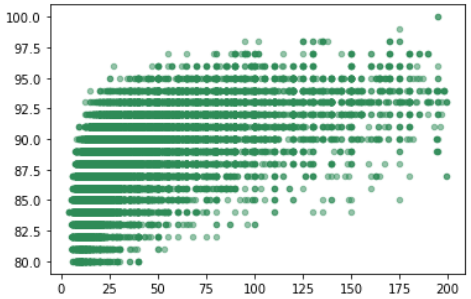


Histogram

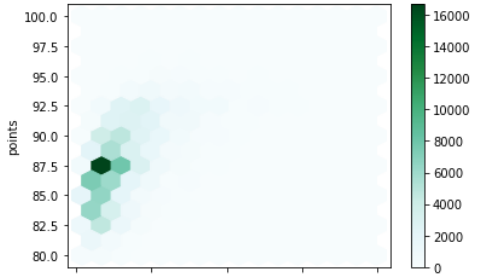


Bivariate chart

Scatter plot

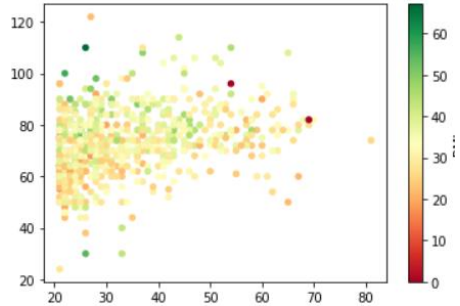


Hexagon plot



Multivariate chart

Scatter plot



Heatmap