Pandas 4 - Data Visualization

Data analysis process

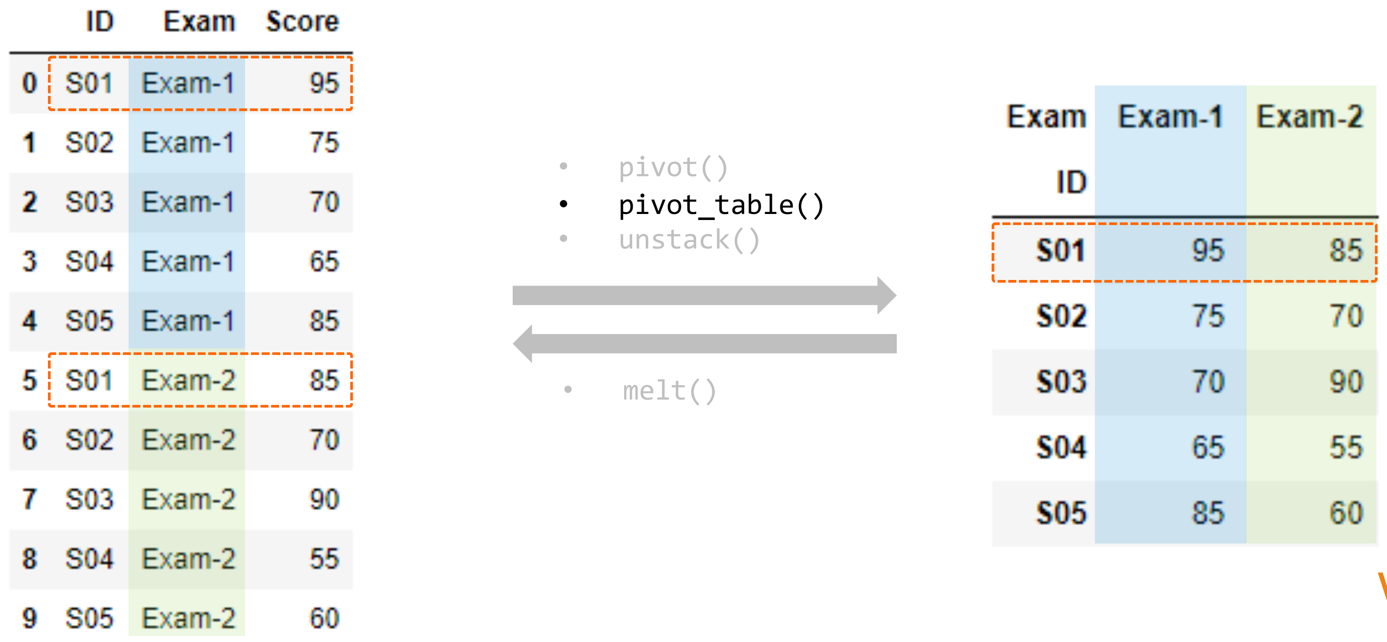
Data Understanding	• Descriptive statistics • Types of data (numerical/categorical)
Data Preprocessing	• Basic operations • Subset selection and data consolidation • Missing data handling
Calculation (Modeling)	• Basic calculation • Data aggregation
Data Visualization	• Univariate chart • Bivariate chart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

Outline

- Reshape DataFrame for visualization
- 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

- `pivot()`
- `pivot_table()`
- `unstack()`

- `melt()`

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

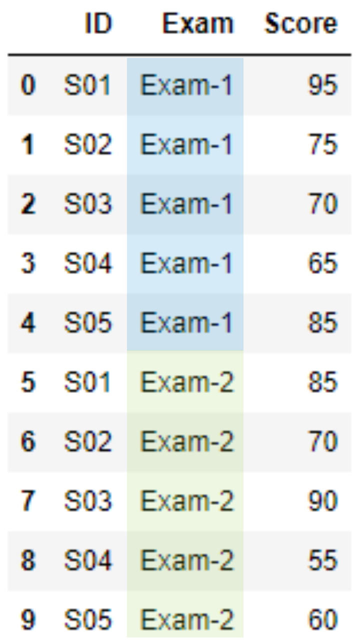
Wide-form

Long-form

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

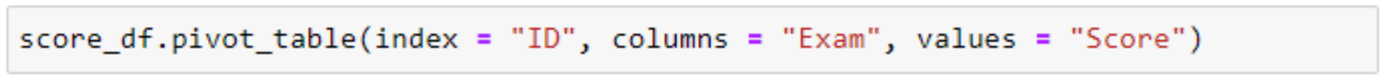
Pivot_table

- Use `pivot_table()` to reshaped a DataFrame by passing arguments: index, columns and values. (By default, aggfunc = mean.)

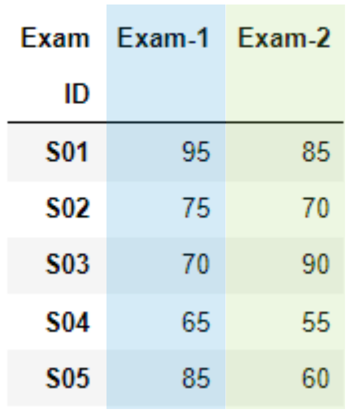


	ID	Exam	Score
0	S01	Exam-1	95
1	S02	Exam-1	75
2	S03	Exam-1	70
3	S04	Exam-1	65
4	S05	Exam-1	85
5	S01	Exam-2	85
6	S02	Exam-2	70
7	S03	Exam-2	90
8	S04	Exam-2	55
9	S05	Exam-2	60

Long-form



```
score_df.pivot_table(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

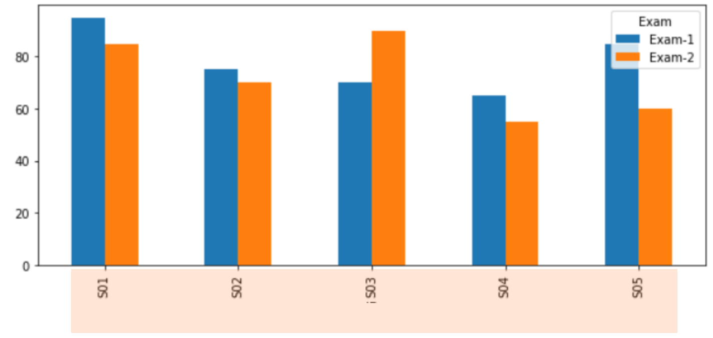
Pivot_table

- 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

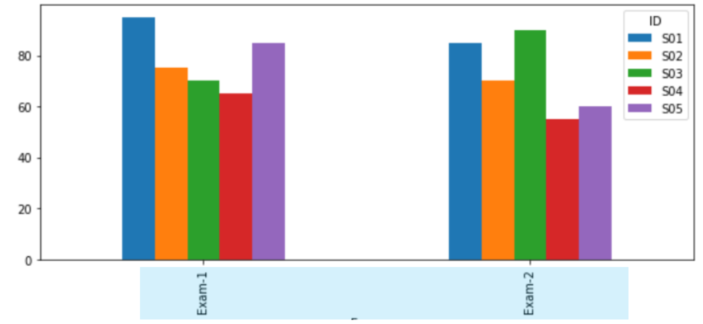
```
score_df.pivot_table(index = "ID", columns = "Exam", values = "Score")
```

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



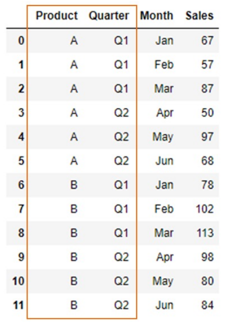
```
score_df.pivot_table(index = "Exam", columns = "ID", values = "Score")
```

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



Unstack

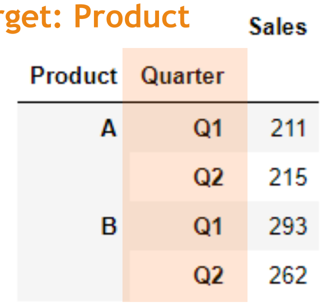
- Use `unstack()` to reshape a dataframe derived from groupby object.
- By default, `unstack()` converts the inner-most row level to column level.



	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.groupby(["Product", "Quarter"]).sum()
```

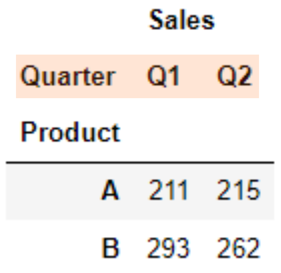
Target: Product



		Sales
Product	Quarter	
A	Q1	211
	Q2	215
B	Q1	293
	Q2	262

Long-form

```
sales_df.groupby(["Product", "Quarter"]).sum().unstack()
```



	Sales	
Quarter	Q1	Q2
Product		
A	211	215
B	293	262

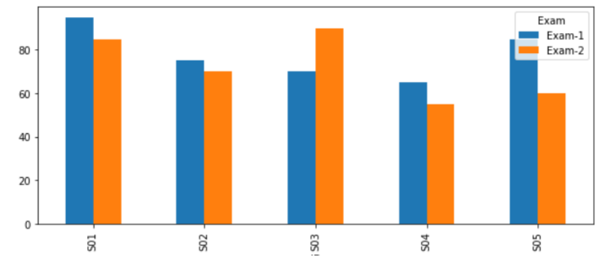
wide-form

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

Outline

- Reshape DataFrame for visualization
- X-axis with categorical data
  - (Line chart)
  - Bar chart
  - Area chart
  - Pie chart
- Numerical data
  - Histogram
  - Scatter plot
  - Hexagon plot

Y-axis



X-axis

Line chart - Series

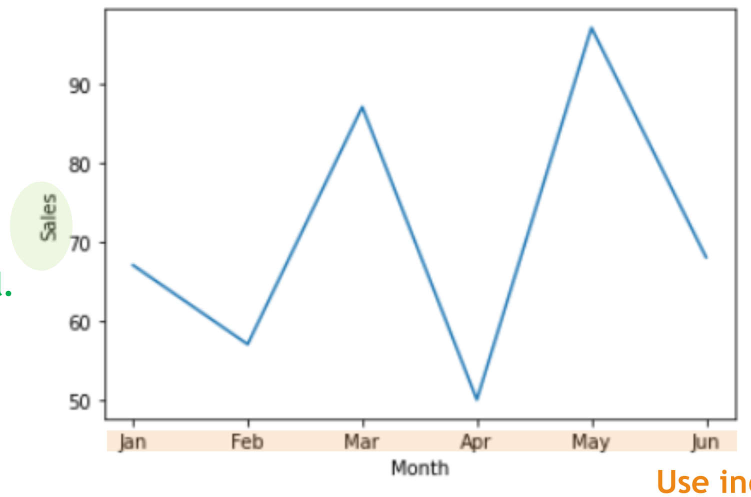
- Both Series and DataFrame have a `plot()` method to make some basic plot types. By default, plot() makes line charts.
- Line chart is usually used to visualize the trend of data over a period of time.

```
series_A = pd.Series([67, 57, 87, 50, 97, 68],
                     index = ["Jan","Feb","Mar","Apr","May","Jun"])
series_A
```

```
Jan    67
Feb    57
Mar    87
Apr    50
May    97
Jun    68
dtype: int64
```

```
series_A.plot(xlabel ="Month", ylabel ="Sales")
```

```
<AxesSubplot:xlabel='Month', ylabel='Sales'>
```



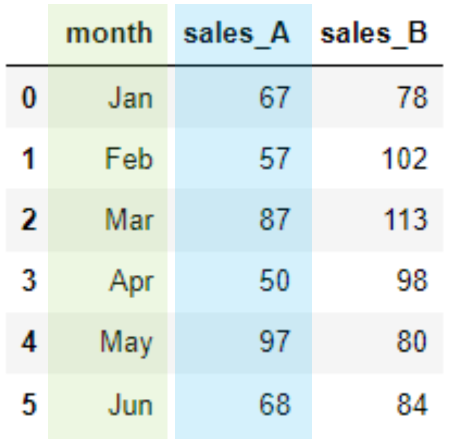
Use “Sales” as the y-axis label.

Use index as ticks for x-axis.

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

Line chart - DataFrame

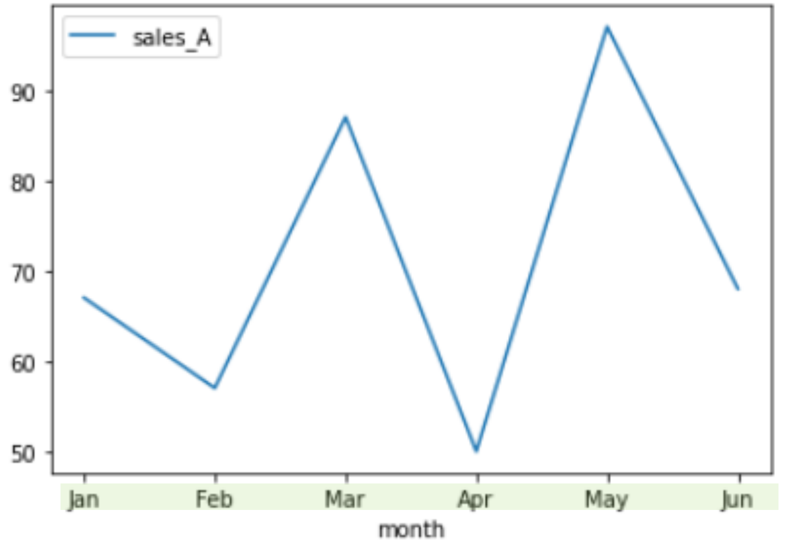
- Use the arguments x and y to specify the columns used for plotting.



	month	sales_A	sales_B
0	Jan	67	78
1	Feb	57	102
2	Mar	87	113
3	Apr	50	98
4	May	97	80
5	Jun	68	84

```
product_df.plot(x = "month", y = "sales_A")
```

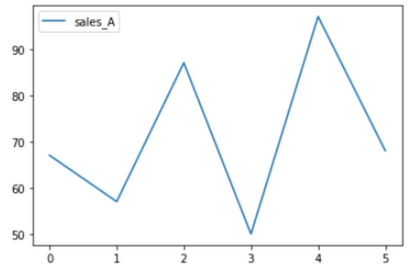
```
<AxesSubplot:xlabel='month'>
```



If "x" is not specified, the index of the DataFrame is used.

```
product_df.plot(y = "sales_A")
```

```
<AxesSubplot:>
```



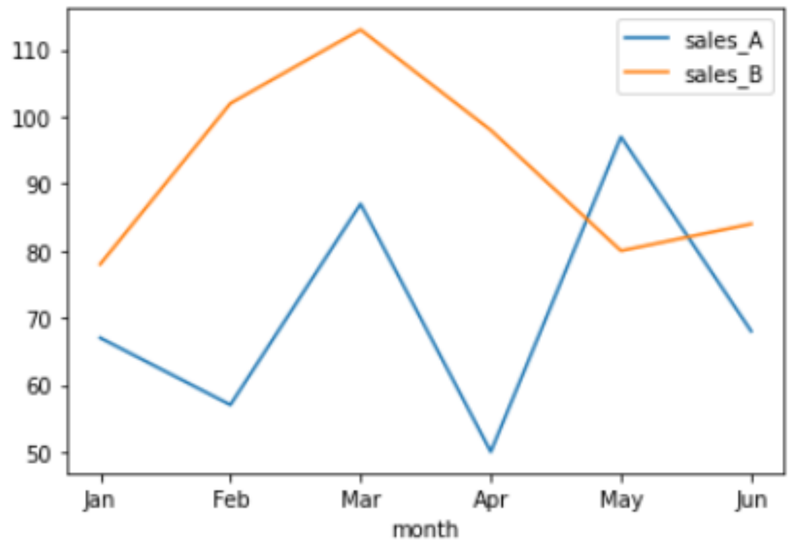
Line chart - DataFrame

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

	month	sales_A	sales_B
0	Jan	67	78
1	Feb	57	102
2	Mar	87	113
3	Apr	50	98
4	May	97	80
5	Jun	68	84

```
product_df.plot(x = "month", y= ["sales_A","sales_B"])
```

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



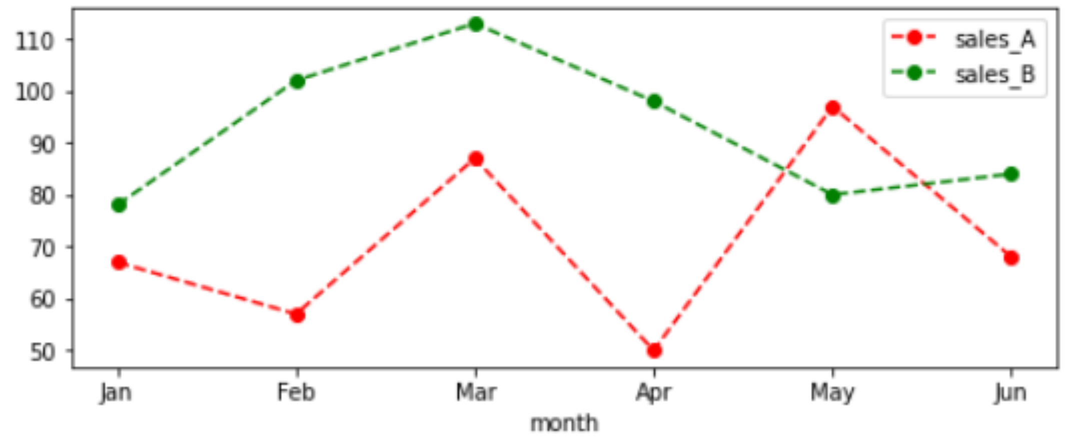
Line chart – Custom style

- Use some arguments to change the style.

	month	sales_A	sales_B
0	Jan	67	78
1	Feb	57	102
2	Mar	87	113
3	Apr	50	98
4	May	97	80
5	Jun	68	84

```
product_df.plot(x = "month",
                y = ["sales_A","sales_B"],
                marker = "o",
                color = ["red","green"],
                linestyle = 'dashed',
                figsize = (8,3))
```

```
<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

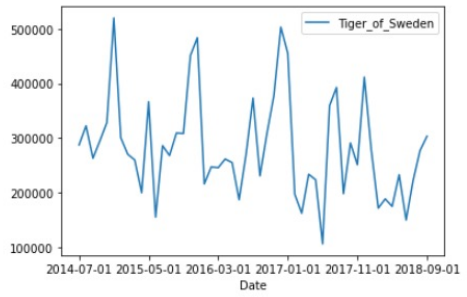
Exercise

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

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

(A.4) Import dataset `fashion.csv`. Show the first five rows.

(A.4) Show the sales trends of `Tiger_of_Sweden` with a line chart.



(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)`.

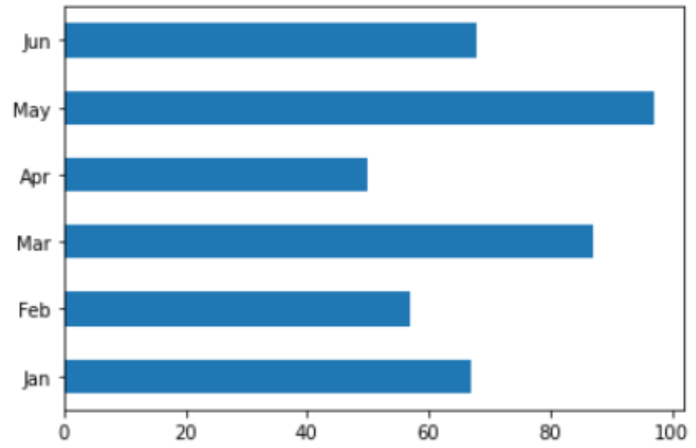
Bar chart - Series

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

```
Jan     67
Feb     57
Mar     87
Apr     50
May     97
Jun     68
dtype: int64
```

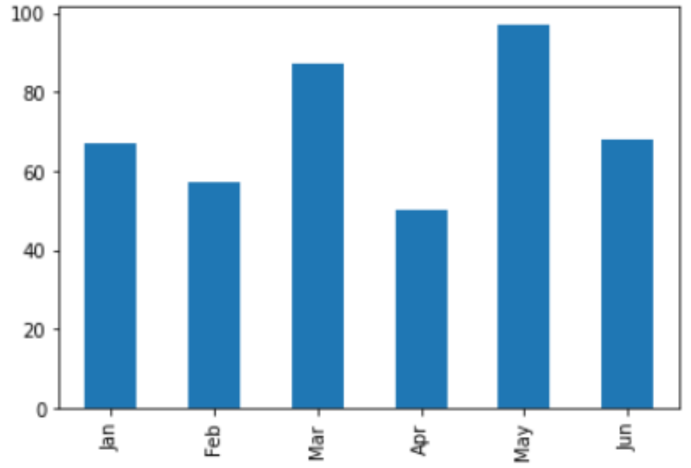
```
series_A.plot(kind = 'barh')
```

```
<AxesSubplot:>
```



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

```
<AxesSubplot:>
```



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

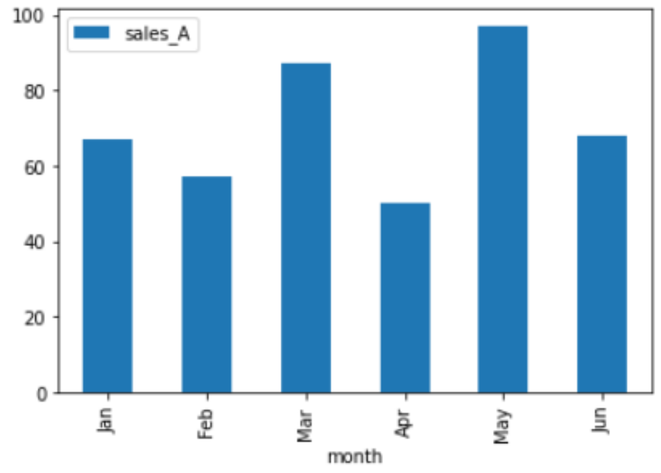
Bar chart - DataFrame

- Use the arguments x and y to specify the columns used for plotting.
- Pass a list of column names to the argument y to plot multiple lines.

	month	sales_A	sales_B
0	Jan	67	78
1	Feb	57	102
2	Mar	87	113
3	Apr	50	98
4	May	97	80
5	Jun	68	84

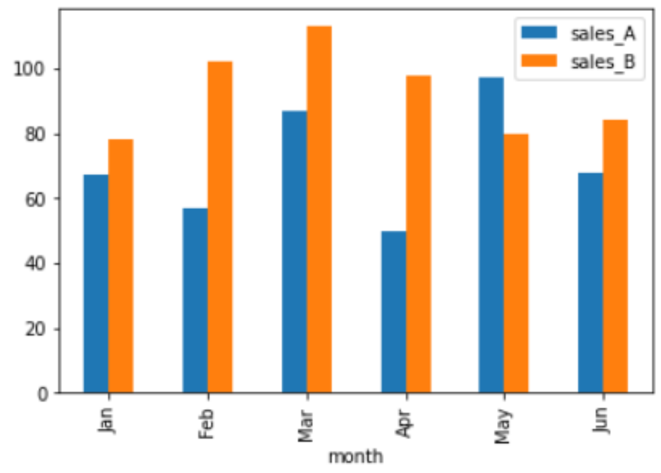
```
product_df.plot(kind = "bar", x= "month", y = "sales_A")
```

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



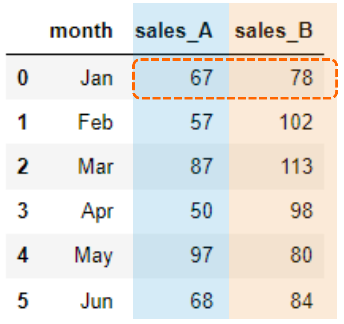
```
product_df.plot(kind = "bar", x = "month", y= ["sales_A","sales_B"])
```

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



Bar chart – Stacked bar chart

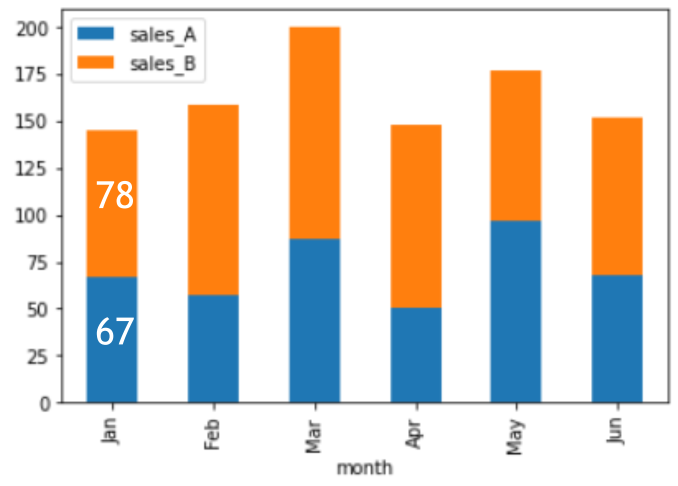
- Stacked bar chart
  - Each bar is stacked by multiple data series.
  - Stacked bar charts can be used to break down and compare parts of the whole.



	month	sales_A	sales_B
0	Jan	67	78
1	Feb	57	102
2	Mar	87	113
3	Apr	50	98
4	May	97	80
5	Jun	68	84

```
product_df.plot(kind = "bar", x = "month", y = ["sales_A","sales_B"], stacked = True)
```

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

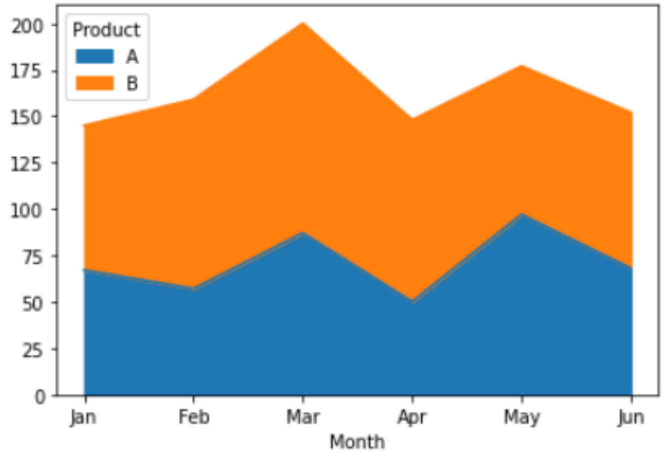


Area chart

- 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`.

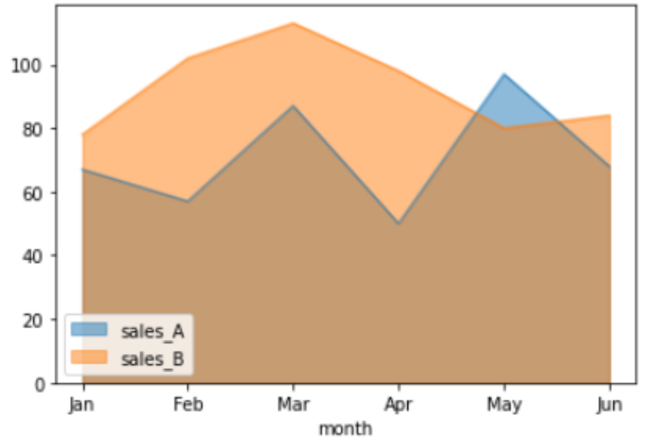
```
product_df.plot(kind = "area",
                x = "month",
                y = ["sales_A","sales_B"])
```

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



```
product_df.plot(kind = "area",
                x = "month",
                y = ["sales_A","sales_B"],
                stacked = False)
```

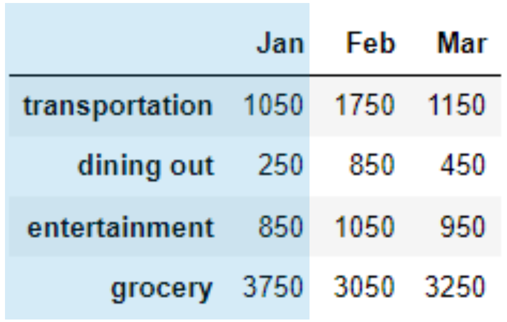
```
<AxesSubplot:xlabel='month'>
```



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

Pie chart

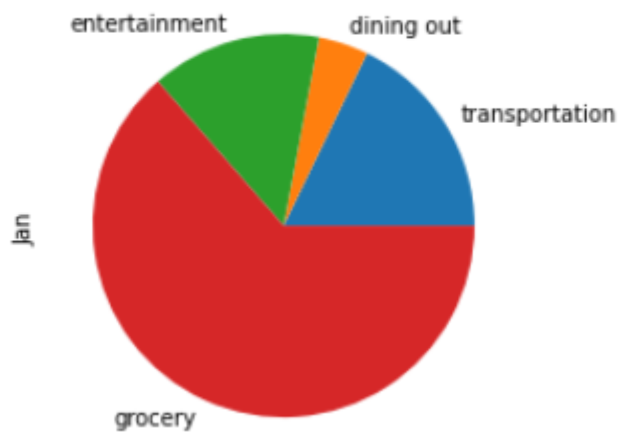
- The pieces of the pie chart are proportional to the fraction of the whole in each category.



	Jan	Feb	Mar
transportation	1050	1750	1150
dining out	250	850	450
entertainment	850	1050	950
grocery	3750	3050	3250

```
spend_df.plot(kind = "pie", y = "Jan", legend = False)
```

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

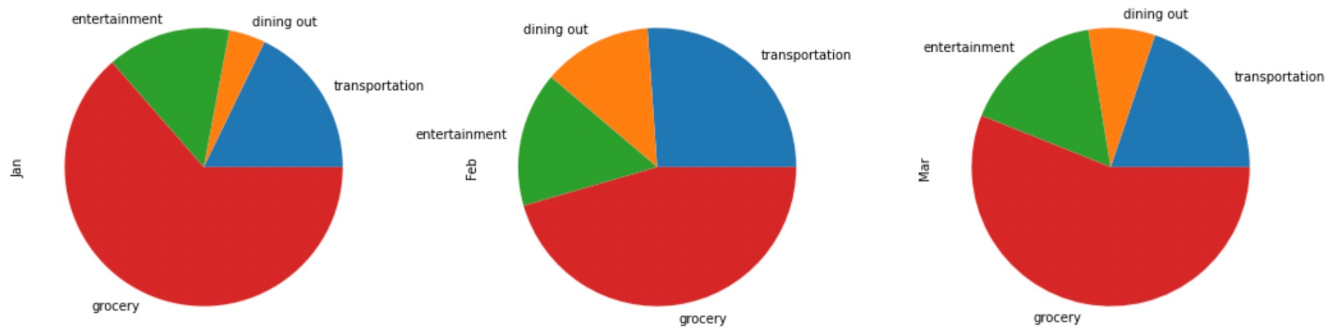


Pie chart

- Use `subplots=True` to plot a pie chart for each numerical column.

	Jan	Feb	Mar
transportation	1050	1750	1150
dining out	250	850	450
entertainment	850	1050	950
grocery	3750	3050	3250

```
spend_df.plot(kind = "pie", subplots = True, figsize=(18,5), legend = False);
```

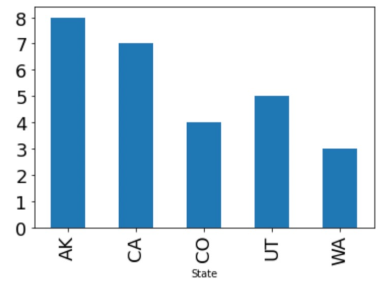


Exercise

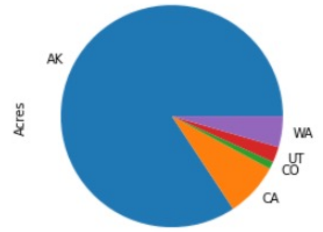
(B.1) Import dataset `parks.csv`. Show the first five rows.

(B.2) Select the national parks in the following five states and keep columns `Park Name`, `State`, and `Acres`. Use this subset to answer the following questions.
State: CA, CO, UT, AK, WA

(B.3) Count the number of national parks in each state. Display the result using a bar graph.
Hint: (1) Group data using column "State". (2) The x-axis shows each state, and each bar is the number of national parks in each state.



(B.4) Calculate the total area of national parks in each state. Display the result using a pie chart.

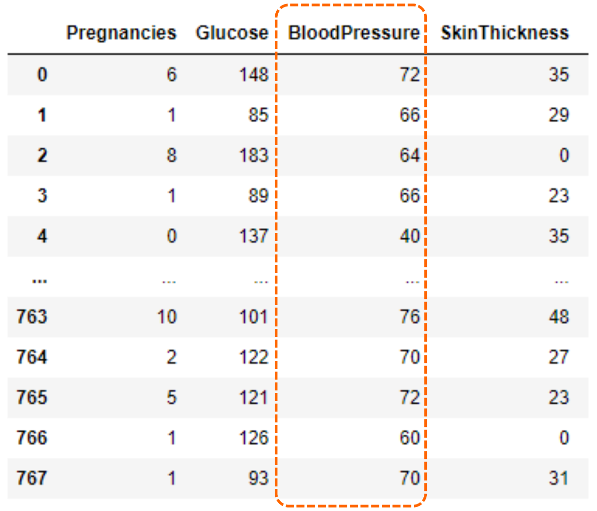


Outline

- Reshape DataFrame for visualization
- X-axis with categorical data
  - (Line chart)
  - Bar chart
  - Area chart
  - Pie chart
- Numerical data
  - Histogram
  - Scatter plot
  - Hexagon plot

Histogram

- 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.



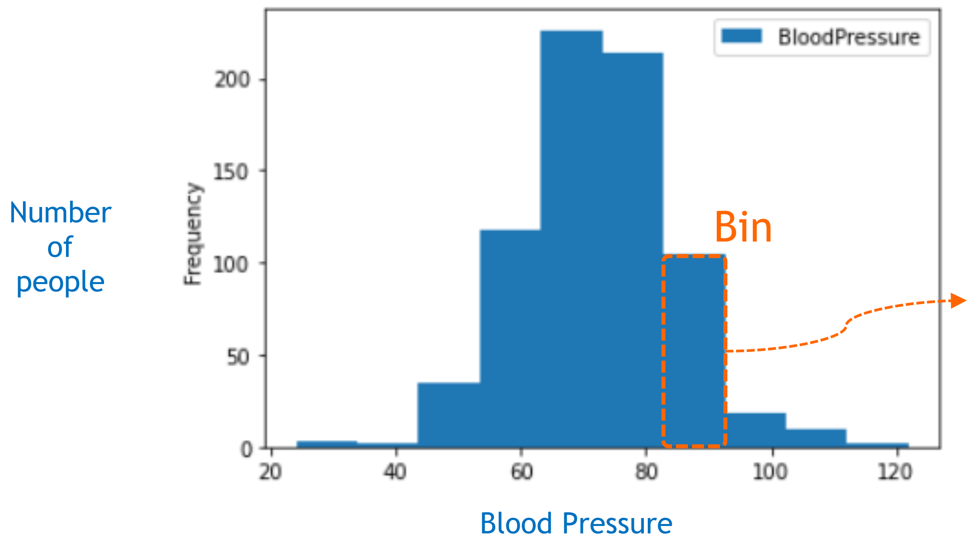
	Pregnancies	Glucose	BloodPressure	SkinThickness
0	6	148	72	35
1	1	85	66	29
2	8	183	64	0
3	1	89	66	23
4	0	137	40	35
...	...	...	...	...
763	10	101	76	48
764	2	122	70	27
765	5	121	72	23
766	1	126	60	0
767	1	93	70	31

By default, the number of bins is 10.

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

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

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



105 people have blood pressure between 82.8 and 92.6

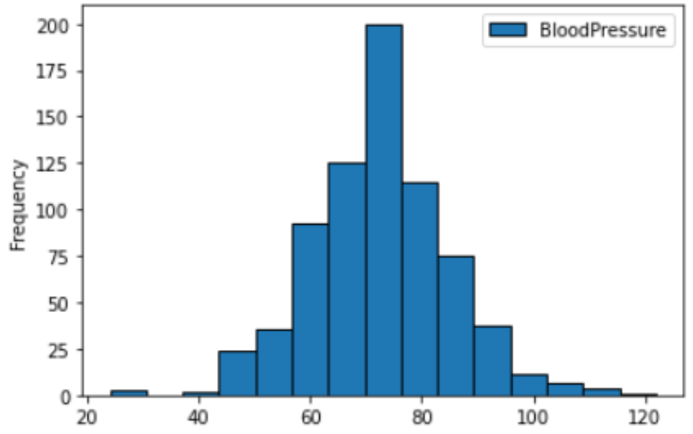
https://pandas.pydata.org/docs/reference/api/pandas.Series.hist.html

Histogram – custom bins

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

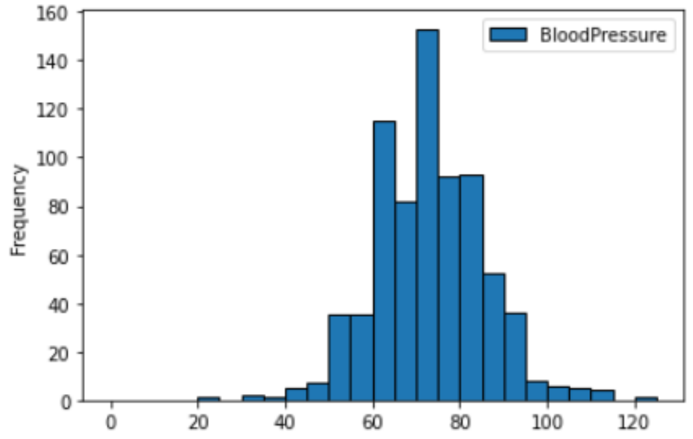
```
diabetes_df.plot(kind = "hist",
                 y = "BloodPressure",
                 bins = 15,
                 edgecolor = "black")
```

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



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

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



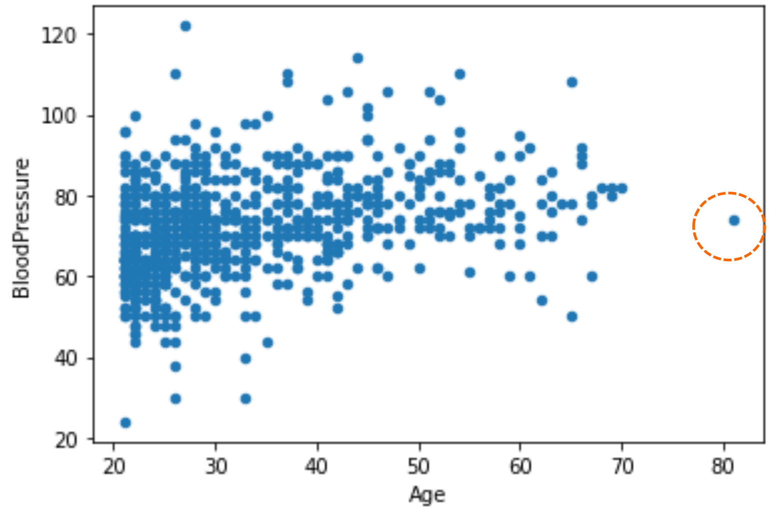
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.hist.html

Scatter plot

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

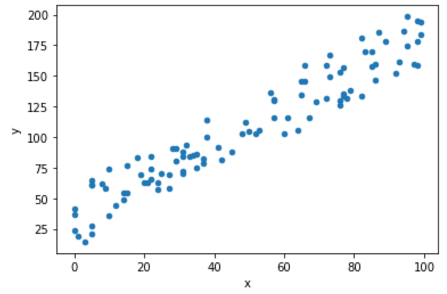
```
diabetes_df.plot(kind = "scatter", x ="Age", y= "BloodPressure")
```

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

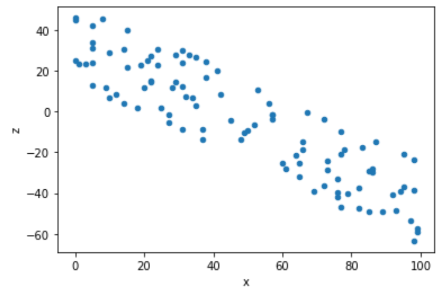


Age = 81,
BloodPressure = 74

Positive correlation



Negative correlation



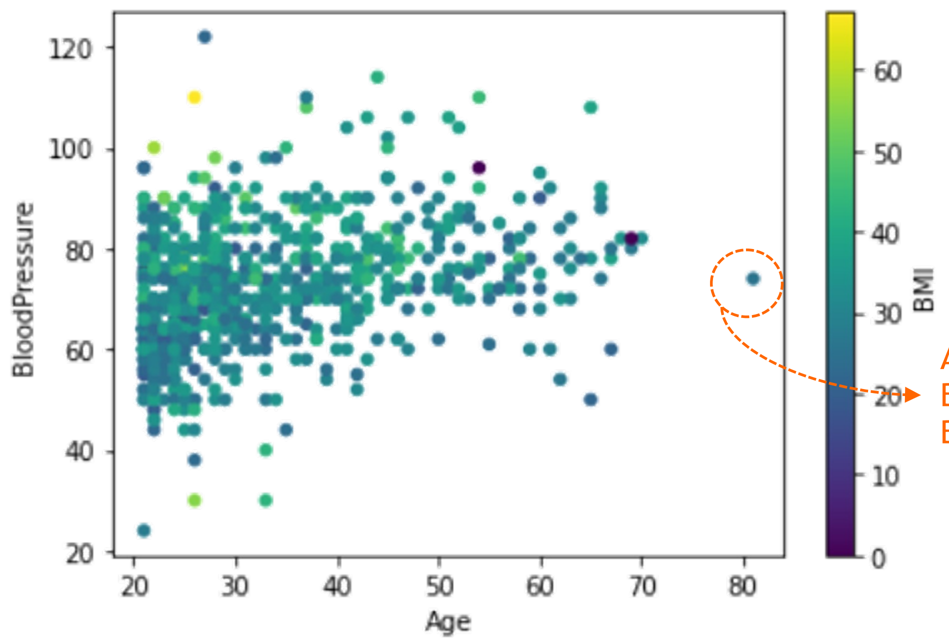
Scatter plot

- 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

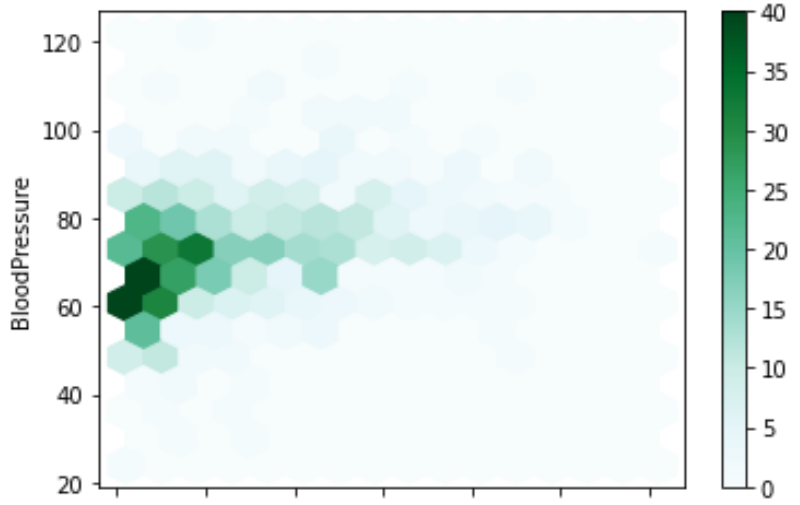
https://matplotlib.org/stable/tutorials/colors/colormaps.html

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.

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`.

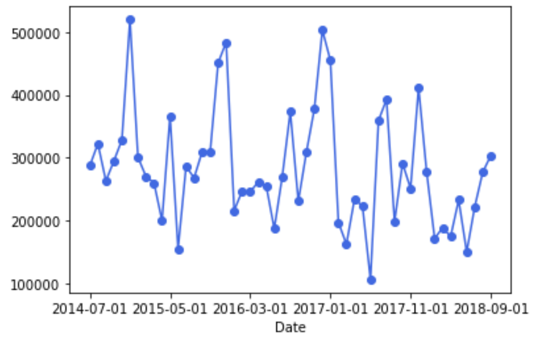
(C.4) Use a scatter plot to show the relationship between wine cost and the number of points recieved in the review.
Hint: Use column `price` and `points`.

(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.

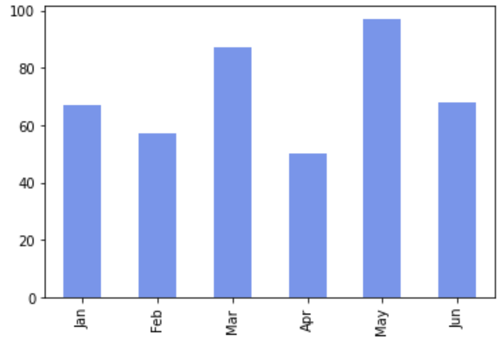
Summary

Univariate chart

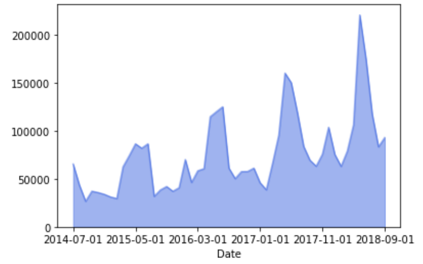
Line chart



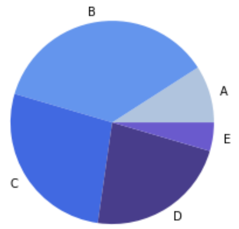
Bar chart



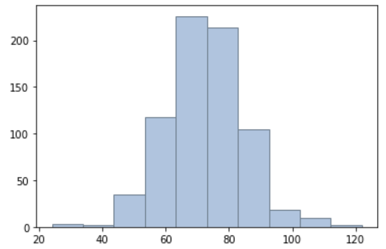
Area chart



Pie chart

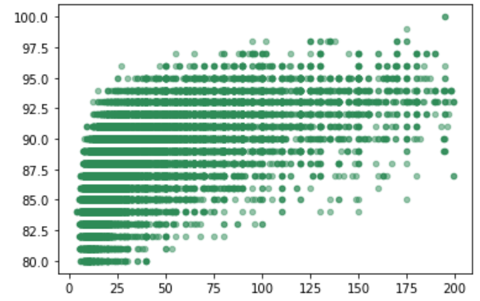


Histogram

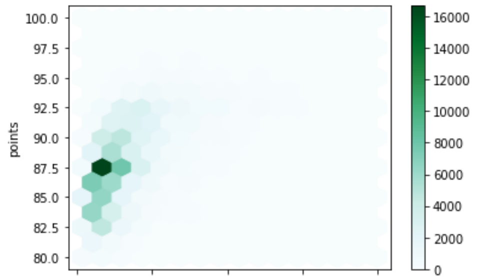


Bivariate chart

Scatter plot

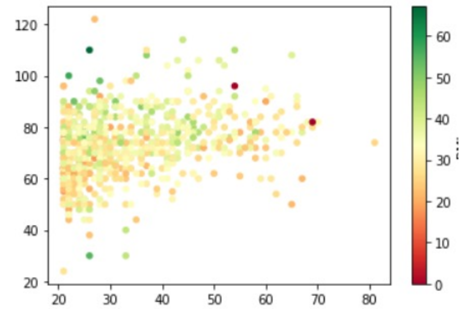


Hexagon plot

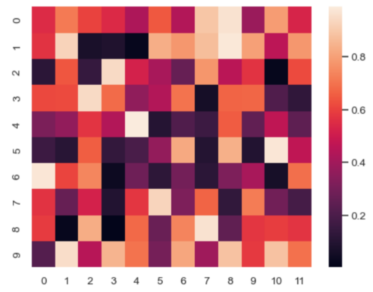


Multivariate chart

Scatter plot



Heatmap


0
1
2
3
4
5
6
7
8
9
0
1
2
3
4
5
6
7
8
9
10
11
0.8
0.6
0.4
0.2