Storytelling with Data

Peng Xiong, DAO, NUS Business School

Contents

Preprocessing with Pandas

- Handling missing data

```
gdp = pd.read_csv('gdp.csv')

cols = ['Country Name', '1980', '1981', '1982', '1983']
gdp_subset = gdp.loc[:5, cols]
gdp_subset
```

	Country Name	1980	1981	1982	1983
0	Aruba	NaN	NaN	NaN	NaN
1	Afghanistan	3.641723e+09	3.478788e+09	NaN	NaN
2	Angola	5.930503e+09	5.550483e+09	5.550483e+09	5.784342e+09
3	Albania	NaN	NaN	NaN	NaN
4	Andorra	4.464161e+08	3.889587e+08	3.758960e+08	3.278618e+08
5	United Arab Emirates	4.359875e+10	4.933342e+10	4.662272e+10	4.280332e+10

Missing data are indicated by `NaN` or "Not a number"

Preprocessing with Pandas

- Handling missing data
  - Detecting missing data: `isnull()` and `notnull()`

```
gdp_subset.isnull()
```

	Country Name	1980	1981	1982	1983
0	False	True	True	True	True
1	False	False	False	True	True
2	False	False	False	False	False
3	False	True	True	True	True
4	False	False	False	False	False
5	False	False	False	False	False

Return `True` if the data item is missing and `False` otherwise

Preprocessing with Pandas

- Handling missing data
  - Detecting missing data: `isnull()` and `notnull()`

```
gdp_subset.notnull()
```

	Country Name	1980	1981	1982	1983
0	True	False	False	False	False
1	True	True	True	False	False
2	True	True	True	True	True
3	True	False	False	False	False
4	True	True	True	True	True
5	True	True	True	True	True

Return `False` if the data item is missing and `True` otherwise

Preprocessing with Pandas

- Handling missing data
  - Detecting missing data: `isnull()` and `notnull()`

```
year = '1981'

bool_series = gdp_subset[year].notnull()
gdp_subset.loc[bool_series, ['Country Name', year]]
```

	Country Name	1980	1981	1982	1983
0	Aruba	NaN	NaN	NaN	NaN
1	Afghanistan	3.641723e+09	3.478788e+09	NaN	NaN
2	Angola	5.930503e+09	5.550483e+09	5.550483e+09	5.784342e+09
3	Albania	NaN	NaN	NaN	NaN
4	Andorra	4.464161e+08	3.889587e+08	3.758960e+08	3.278618e+08
5	United Arab Emirates	4.359875e+10	4.933342e+10	4.662272e+10	4.280332e+10

Preprocessing with Pandas

- Handling missing data
  - Detecting missing data: `isnull()` and `notnull()`

```
year = '1981'

bool_series = gdp_subset[year].notnull()
gdp_subset.loc[bool_series, ['Country Name', year]]
```

`gdp_subset[year]`

```
0            NaN
1    3.478788e+09
2    5.550483e+09
3            NaN
4    3.889587e+08
5    4.933342e+10
Name: 1981, dtype: float64
```

`bool_series`

```
0    False
1     True
2     True
3    False
4     True
5     True
Name: 1981, dtype: bool
```

Preprocessing with Pandas

- Handling missing data
  - Detecting missing data: `isnull()` and `notnull()`

```
year = '1981'

bool_series = gdp_subset[year].notnull()
gdp_subset.loc[bool_series, ['Country Name', year]]
```

	Country Name	1981
1	Afghanistan	3.478788e+09
2	Angola	5.550483e+09
4	Andorra	3.889587e+08
5	United Arab Emirates	4.933342e+10

Boolean indexing for selecting valid GDP data

Label based indexing for selecting columns

Preprocessing with Pandas

- Handling missing data
  - Dropping missing data: `dropna()`

	Country Name	1980	1981	1982	1983
0	Aruba	NaN	NaN	NaN	NaN
1	Afghanistan	3.641723e+09	3.478788e+09	NaN	NaN
2	Angola	5.930503e+09	5.550483e+09	5.550483e+09	5.784342e+09
3	Albania	NaN	NaN	NaN	NaN
4	Andorra	4.464161e+08	3.889587e+08	3.758960e+08	3.278618e+08
5	United Arab Emirates	4.359875e+10	4.933342e+10	4.662272e+10	4.280332e+10

Preprocessing with Pandas

- Handling missing data
  - Dropping missing data: `dropna()`

```
output = gdp_subset.dropna()
output
```

	Country Name	1980	1981	1982	1983
2	Angola	5.930503e+09	5.550483e+09	5.550483e+09	5.784342e+09
4	Andorra	4.464161e+08	3.889587e+08	3.758960e+08	3.278618e+08
5	United Arab Emirates	4.359875e+10	4.933342e+10	4.662272e+10	4.280332e+10

The default setting of the `dropna()` method makes a copy of the data frame with rows of missing values dropped. The original data frame is unchanged

Preprocessing with Pandas

- Handling missing data
  - Dropping missing data: `dropna()`

```
output = gdp_subset.dropna(inplace=True)
print(output)
```

```
None
```

In the case of `inplace=True`, the `dropna()` method gives no output (`None`).

```
gdp_subset
```

	Country Name	1980	1981	1982	1983
2	Angola	5.930503e+09	5.550483e+09	5.550483e+09	5.784342e+09
4	Andorra	4.464161e+08	3.889587e+08	3.758960e+08	3.278618e+08
5	United Arab Emirates	4.359875e+10	4.933342e+10	4.662272e+10	4.280332e+10

The original data frame is overwritten with rows of missing values dropped.

Preprocessing with Pandas

- Handling missing data
  - Replacing `NaN` by other values: `fillna()`

```
cols = ['Country Name', '1980', '1981', '1982', '1983']
gdp_subset = gdp.loc[:5, cols]
gdp_subset
```

	Country Name	1980	1981	1982	1983
0	Aruba	NaN	NaN	NaN	NaN
1	Afghanistan	3.641723e+09	3.478788e+09	NaN	NaN
2	Angola	5.930503e+09	5.550483e+09	5.550483e+09	5.784342e+09
3	Albania	NaN	NaN	NaN	NaN
4	Andorra	4.464161e+08	3.889587e+08	3.758960e+08	3.278618e+08
5	United Arab Emirates	4.359875e+10	4.933342e+10	4.662272e+10	4.280332e+10

Preprocessing with Pandas

- Handling missing data
  - Replacing `NaN` by other values: `fillna()`

```
output = gdp_subset.fillna(0)
output
```

	Country Name	1980	1981	1982	1983
0	Aruba	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
1	Afghanistan	3.641723e+09	3.478788e+09	0.000000e+00	0.000000e+00
2	Angola	5.930503e+09	5.550483e+09	5.550483e+09	5.784342e+09
3	Albania	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
4	Andorra	4.464161e+08	3.889587e+08	3.758960e+08	3.278618e+08
5	United Arab Emirates	4.359875e+10	4.933342e+10	4.662272e+10	4.280332e+10

The default setting of the `fillna()` method makes a copy of the data frame with missing values to be replaced. The original data frame is unchanged

Preprocessing with Pandas

- Handling missing data
  - Replacing `NaN` by other values: `fillna()`

```
output = gdp_subset.fillna(0, inplace=True)
output
```

```
None
```

In the case of `inplace=True`, the `fillna()` method gives no output (`None`).

Preprocessing with Pandas

- Handling missing data
  - Replacing `NaN` by other values: `fillna()`

```
gdp_subset
```

	Country Name	1980	1981	1982	1983
0	Aruba	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
1	Afghanistan	3.641723e+09	3.478788e+09	0.000000e+00	0.000000e+00
2	Angola	5.930503e+09	5.550483e+09	5.550483e+09	5.784342e+09
3	Albania	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
4	Andorra	4.464161e+08	3.889587e+08	3.758960e+08	3.278618e+08
5	United Arab Emirates	4.359875e+10	4.933342e+10	4.662272e+10	4.280332e+10

click to scroll output; double click to hide

The original data frame is overwritten with the missing values being replaced.

Preprocessing with Pandas

- Row indexes of row indexes
  - Specifying row indexes: `set_index()`

> **Example 1:** Visualize the GDP trend of three countries: China, Japan, and Germany, from 1985 to 2015.

```
gdp_chn = gdp.loc[gdp['Country Code'] == 'CHN', '1980':'2015']
gdp_jpn = gdp.loc[gdp['Country Code'] == 'JPN', '1980':'2015']
gdp_deu = gdp.loc[gdp['Country Code'] == 'DEU', '1980':'2015']
```

Preprocessing with Pandas

- Row indexes of row indexes
  - Specifying row indexes: `set_index()`

```
gdp_new = gdp.set_index('Country Code')
gdp_new
```

Indexes are set to be country code

Country Code	Country Name	1960	1961	...	2018	2019	2020
ABW	Aruba	NaN	NaN	...	3.202189e+09	NaN	NaN
AFG	Afghanistan	5.377778e+08	5.488889e+08	...	1.805323e+10	1.879945e+10	2.011614e+10
AGO	Angola	NaN	NaN	...	1.010000e+11	8.941719e+10	5.837598e+10
...	...	...	...	...	...	...	...
ZAF	South Africa	7.575397e+09	7.972997e+09	...	4.050000e+11	3.880000e+11	3.350000e+11
ZMB	Zambia	7.130000e+08	6.962857e+08	...	2.631159e+10	2.330867e+10	1.811063e+10
ZWE	Zimbabwe	1.052990e+09	1.096647e+09	...	1.811554e+10	1.928429e+10	1.805117e+10

Preprocessing with Pandas

- Row indexes of row indexes
  - Specifying row indexes: `set_index()`

```
gdp_chn = gdp_new.loc['CHN', '1980':'2015']
gdp_jpn = gdp_new.loc['JPN', '1980':'2015']
gdp_deu = gdp_new.loc['DEU', '1980':'2015']
```

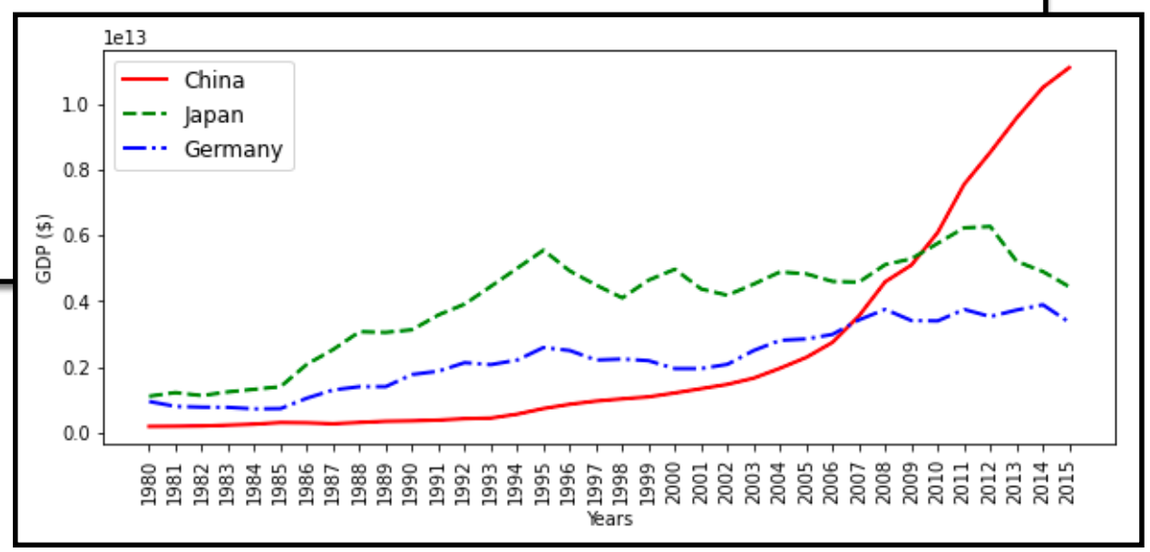
Indexes are set to be country code

Country Code	Country Name	1960	1961	...	2018	2019	2020
ABW	Aruba	NaN	NaN	...	3.202189e+09	NaN	NaN
AFG	Afghanistan	5.377778e+08	5.488889e+08	...	1.805323e+10	1.879945e+10	2.011614e+10
AGO	Angola	NaN	NaN	...	1.010000e+11	8.941719e+10	5.837598e+10
...	...	...	...	...	...	...	...
ZAF	South Africa	7.575397e+09	7.972997e+09	...	4.050000e+11	3.880000e+11	3.350000e+11
ZMB	Zambia	7.130000e+08	6.962857e+08	...	2.631159e+10	2.330867e+10	1.811063e+10
ZWE	Zimbabwe	1.052990e+09	1.096647e+09	...	1.811554e+10	1.928429e+10	1.805117e+10

Preprocessing with Pandas

- Row indexes of row indexes
  - Specifying row indexes: `set_index()`

```
plt.figure(figsize=(10, 4))
plt.plot(gdp_chn.index, gdp_chn,
         linewidth=2, linestyle='-', color='r', label='China')
plt.plot(gdp_jpn.index, gdp_jpn,
         linewidth=2, linestyle='--', color='g', label='Japan')
plt.plot(gdp_deu.index, gdp_deu,
         linewidth=2, linestyle='-.', color='b', label='Germany')
plt.legend(fontsize=12)
plt.xlabel('Years', fontsize=10)
plt.ylabel('GDP ($)', fontsize=10)
plt.xticks(rotation=90)
plt.show()
```



Preprocessing with Pandas

- Row indexes of row indexes
  - Reset row indexes to default: `reset_index()`

```
gdp_new.reset_index()
```

Indexes are reset to be default integers

	Country Code	Country Name	1960	...	2018	2019	2020
0	ABW	Aruba	NaN	...	3.202189e+09	NaN	NaN
1	AFG	Afghanistan	5.377778e+08	...	1.805323e+10	1.879945e+10	2.011614e+10
2	AGO	Angola	NaN	...	1.010000e+11	8.941719e+10	5.837598e+10
...	...	...	...	...	...	...	...
212	ZAF	South Africa	7.575397e+09	...	4.050000e+11	3.880000e+11	3.350000e+11
213	ZMB	Zambia	7.130000e+08	...	2.631159e+10	2.330867e+10	1.811063e+10
214	ZWE	Zimbabwe	1.052990e+09	...	1.811554e+10	1.928429e+10	1.805117e+10

Preprocessing with Pandas

- Row indexes of row indexes
  - Reset row indexes to default: `reset_index()`

```
gdp_new.reset_index()
```

	Country Code	Country Name	1960	...	2018	2019	2020
0	ABW	Aruba	NaN	...	3.202189e+09	NaN	NaN
1	AFG	Afghanistan	5.377778e+08	...	1.805323e+10	1.879945e+10	2.011614e+10
2	AGO	Angola	NaN	...	1.010000e+11	8.941719e+10	5.837598e+10
...	...	...	...	...	...	...	...
212	ZAF	South Africa	7.575397e+09	...	4.050000e+11	3.880000e+11	3.350000e+11
213	ZMB	Zambia	7.130000e+08	...	2.631159e+10	2.330867e+10	1.811063e+10
214	ZWE	Zimbabwe	1.052990e+09	...	1.811554e+10	1.928429e+10	1.805117e+10

The previous label based indexes are turned into a column

Preprocessing with Pandas

- Row indexes of row indexes
  - Reset row indexes to default: `reset_index()`

`set_index('Country Name')`

	Country Code	Country Name	1960	...
0	ABW	Aruba	NaN	...
1	AFG	Afghanistan	5.377778e+08	...
2	AGO	Angola	NaN	...
...	...	...	...	...
212	ZAF	South Africa	7.575397e+09	...
213	ZMB	Zambia	7.130000e+08	...
214	ZWE	Zimbabwe	1.052990e+09	...

	Country Name	1960	1961	...
Country Code				
ABW	Aruba	NaN	NaN	...
AFG	Afghanistan	5.377778e+08	5.488889e+08	...
AGO	Angola	NaN	NaN	...
...	...	...	...	...
ZAF	South Africa	7.575397e+09	7.972997e+09	...
ZMB	Zambia	7.130000e+08	6.962857e+08	...
ZWE	Zimbabwe	1.052990e+09	1.096647e+09	...

`reset_index()`

Preprocessing with Pandas

- Row indexes of row indexes
  - ▸ Reset row indexes to default: `reset_index()`
    - ✓ Reset indexes after filtering or dropping missing values

```
cols = ['Country Name', '1980', '1981', '1982', '1983']
gdp_subset = gdp.loc[:5, cols]
gdp_subset
```

	Country Name	1980	1981	1982	1983
0	Aruba	NaN	NaN	NaN	NaN
1	Afghanistan	3.641723e+09	3.478788e+09	NaN	NaN
2	Angola	5.930503e+09	5.550483e+09	5.550483e+09	5.784342e+09
3	Albania	NaN	NaN	NaN	NaN
4	Andorra	4.464161e+08	3.889587e+08	3.758960e+08	3.278618e+08
5	United Arab Emirates	4.359875e+10	4.933342e+10	4.662272e+10	4.280332e+10

Preprocessing with Pandas

- Row indexes of row indexes
  - Reset row indexes to default: `reset_index()`
    - ✓ Reset indexes after filtering or dropping missing values

```
output = gdp_subset.dropna()
output
```

	Country Name	1980	1981	1982	1983
2	Angola	5.930503e+09	5.550483e+09	5.550483e+09	5.784342e+09
4	Andorra	4.464161e+08	3.889587e+08	3.758960e+08	3.278618e+08
5	United Arab Emirates	4.359875e+10	4.933342e+10	4.662272e+10	4.280332e+10

Disordered indexes

Preprocessing with Pandas

- Row indexes of row indexes
  - Reset row indexes to default: `reset_index()`
    - ✓ Reset indexes after filtering or dropping missing values

```
output.reset_index()
```

	index	Country Name	1980	1981	1982	1983
0	2	Angola	5.930503e+09	5.550483e+09	5.550483e+09	5.784342e+09
1	4	Andorra	4.464161e+08	3.889587e+08	3.758960e+08	3.278618e+08
2	5	United Arab Emirates	4.359875e+10	4.933342e+10	4.662272e+10	4.280332e+10

Default indexes

Disordered indexes become a new column

Preprocessing with Pandas

- Row indexes of row indexes
  - Reset row indexes to default: `reset_index()`
    - ✓ Reset indexes after filtering or dropping missing values

```
output.reset_index(drop=True)
```

Drop the previous indexes

	Country Name	1980	1981	1982	1983
0	Angola	5.930503e+09	5.550483e+09	5.550483e+09	5.784342e+09
1	Andorra	4.464161e+08	3.889587e+08	3.758960e+08	3.278618e+08
2	United Arab Emirates	4.359875e+10	4.933342e+10	4.662272e+10	4.280332e+10

Default indexes

Preprocessing with Pandas

- Row indexes of row indexes
  - Change in-place for `set_index()` and `reset_index()`
    - ✓ Argument `inplace=Flase` (default setting): return a new data frame with changed indexes and/or column data; the original data frame is unchanged
    - ✓ Argument `inplace=True` : overwrite the original data frame with changed indexes and/or column data; return no output.

Preprocessing with Pandas

- Operations on series
  - Element-wise arithmetic operations

```
gdp_new['1980'] / 1000000
```

```
Country Code
ABW              NaN
AFG      3641.723322
AGO      5930.503401
ALB              NaN
AND       446.416106
             ...
WSM              NaN
YEM              NaN
ZAF     82980.483388
ZMB      3829.500000
ZWE      6678.868200
Name: 1980, Length: 215, dtype: float64
```

Preprocessing with Pandas

- Operations on series
  - Element-wise arithmetic operations

```
(gdp_new['1981'] - gdp_new['1980']) / 1000000
```

```
Country Code
ABW             NaN
AFG     -162.935413
AGO     -380.020365
ALB             NaN
AND      -57.457375
            ...
WSM             NaN
YEM             NaN
ZAF     2473.937112
ZMB       43.166667
ZWE     1332.505600
Length: 215, dtype: float64
```

Preprocessing with Pandas

- Operations on series
  - Element-wise arithmetic operations

```
(gdp_new['1981'] - gdp_new['1980']) / gdp_new['1980'] * 100
```

```
Country Code
ABW           NaN
AFG     -4.474129
AGO     -6.407894
ALB           NaN
AND    -12.870811
           ...
WSM           NaN
YEM           NaN
ZAF      2.981348
ZMB      1.127214
ZWE     19.951069
Length: 215, dtype: float64
```

Preprocessing with Pandas

- Operations on series
  - Vectorized string operations

```
prop = pd.read_csv('properties.csv')
prop
```

	project	street	type	...	area	level	date
0	PARC CLEMATIS	JALAN LEMPENG	Apartment	...	1496	11 to 15	Dec-21
1	ONE PEARL BANK	PEARL BANK	Apartment	...	1281	16 to 20	Dec-21
2	ONE PEARL BANK	PEARL BANK	Apartment	...	700	21 to 25	Dec-21
3	ONE PEARL BANK	PEARL BANK	Apartment	...	840	06 to 10	Dec-21
...	...	...	...	...	...	...	...
104897	EUPHONY GARDENS	JALAN MATA AYER	Condominium	...	1044	01 to 05	Jan-17
104898	SYMPHONY SUITES	YISHUN CLOSE	Condominium	...	797	11 to 15	Jan-17
104899	SYMPHONY SUITES	YISHUN CLOSE	Condominium	...	689	11 to 15	Jan-17
104900	SELETAR PARK RESIDENCE	SELETAR ROAD	Condominium	...	872	01 to 05	Jan-17

104901 rows × 12 columns

Preprocessing with Pandas

- Operations on series
  - Vectorized string operations

```
project = prop['project']
project_lower = project.str.lower()
print(project_lower)
```

```
0                    parc clematis
1                  one pearl bank
2                  one pearl bank
3                  one pearl bank
4                       the crest
                    ...
104896                  floraville
104897              euphony gardens
104898              symphony suites
104899              symphony suites
104900    seletar park residence
Name: project, Length: 104901, dtype: object
```

Preprocessing with Pandas

- Operations on series
  - Vectorized string operations

```
print(project.str.replace(' ', '--'))
```

```
0                       PARC--CLEMATIS
1                    ONE--PEARL--BANK
2                    ONE--PEARL--BANK
3                    ONE--PEARL--BANK
4                          THE--CREST
                        ...
104896                     FLORAVILLE
104897               EUPHONY--GARDENS
104898               SYMPHONY--SUITES
104899               SYMPHONY--SUITES
104900    SELETAR--PARK--RESIDENCE
Name: project, Length: 104901, dtype: object
```

Preprocessing with Pandas

- Operations on series
  - Vectorized string operations

```
print(project.str.count('A'))
```

```
0          2
1          2
2          2
3          2
4          0
          ..
104896     1
104897     1
104898     0
104899     0
104900     2
Name: project, Length: 104901, dtype: int64
```

Preprocessing with Pandas

- Operations on series
  - Vectorized string operations

Example 2: Notice that the level of each apartment/condo is given as a string "XX to YY". Create two columns **`level_from`** and **`level_to`**, that are the level numbers XX and YY, as integers.

Preprocessing with Pandas

- Operations on series
  - Vectorized string operations

```
prop['level_from'] = prop['level'].str[:2].astype(int)
prop['level_to'] = prop['level'].str[-2:].astype(int)
prop
```

	project	street	type	district	...	level	date	level_from	level_to
0	PARC CLEMATIS	JALAN LEMPENG	Apartment	5	...	11 to 15	Dec-21	11	15
1	ONE PEARL BANK	PEARL BANK	Apartment	3	...	16 to 20	Dec-21	16	20
2	ONE PEARL BANK	PEARL BANK	Apartment	3	...	21 to 25	Dec-21	21	25
3	ONE PEARL BANK	PEARL BANK	Apartment	3	...	06 to 10	Dec-21	6	10
...	...	...	...	...	...	...	...	...	...
104897	EUPHONY GARDENS	JALAN MATA AYER	Condominium	27	...	01 to 05	Jan-17	1	5
104898	SYMPHONY SUITES	YISHUN CLOSE	Condominium	27	...	11 to 15	Jan-17	11	15
104899	SYMPHONY SUITES	YISHUN CLOSE	Condominium	27	...	11 to 15	Jan-17	11	15
104900	SELETAR PARK RESIDENCE	SELETAR ROAD	Condominium	28	...	01 to 05	Jan-17	1	5

104901 rows × 14 columns

Guidelines of Data Visualization

choose an
effective visual
with the
SWD CHART GUIDE

At *storytelling with data,* we encounter a ton of different graphs. Through our work, we've both learned strategies for effective application and identified common pitfalls (including some things to avoid!). In this guide, we share the good and the bad of commonly used charts and graphs for data communications.

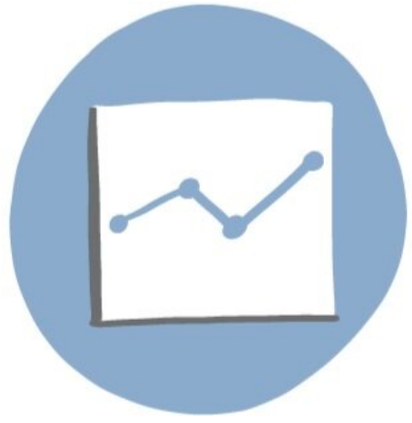
— Storytelling with Data Blog

Guidelines of Data Visualization

- Line graph

A typical line graph will have **continuous** data along both the vertical (y-axis) and horizontal (x-axis) dimensions. The y-axis usually shows the value of whatever variable we are measuring; the x-axis is most often used to show when we measured it, either chronologically or based on some independent variable.

storytelling WITH data®

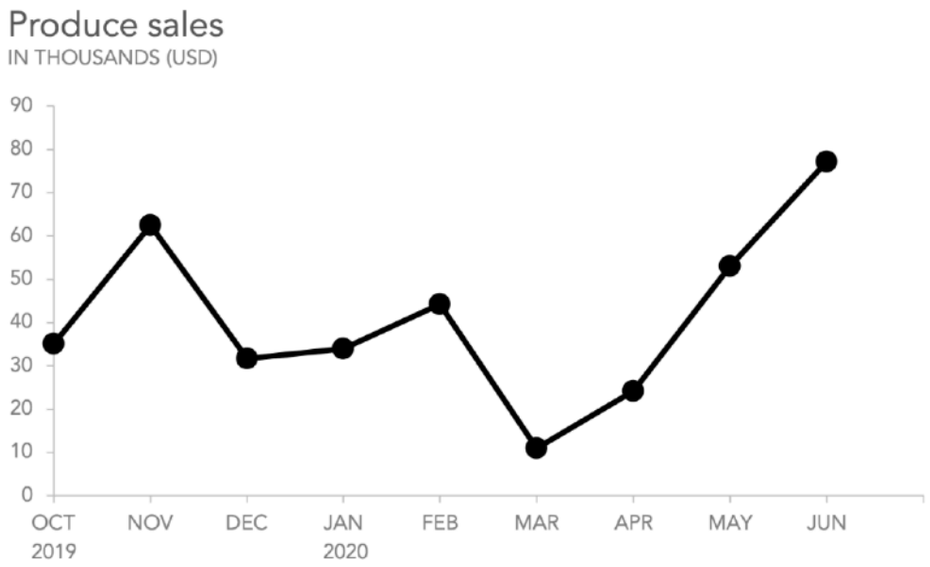


Guidelines of Data Visualization

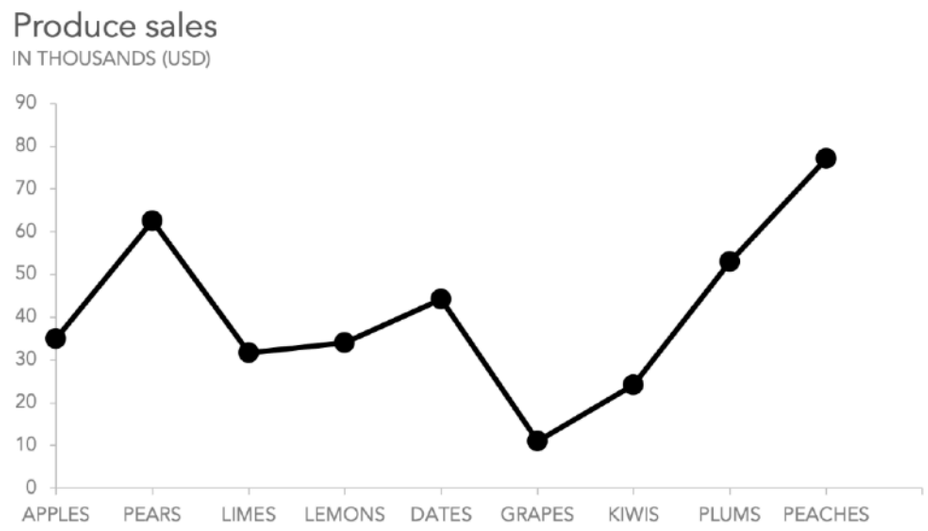
- Line graph
  - Effective cases
    - ✓ Comparing lots of data all at once
    - ✓ Showing changes and trends over time
    - ✓ Displaying forecast data and uncertainty
    - ✓ Highlighting anomalies within and across data series

Guidelines of Data Visualization

- Line graph
  - Ineffective cases
    - ✓ Displaying quantities of things
    - ✓ Working with **categorical** data



DO: Show a line of sales by month.



DON'T: Show a line of sales by category.

Guidelines of Data Visualization

- Line graph
  - Ineffective cases
    - ✓ Displaying quantities of things
    - ✓ Working with **categorical** data
    - ✓ Making part-to-whole comparison
    - ✓ Showing sparse datasets

Guidelines of Data Visualization

- Line graph

Question 1: Use proper line graph to visualize the GDP growing rates of ASEAN countries between 2000 and 2020.

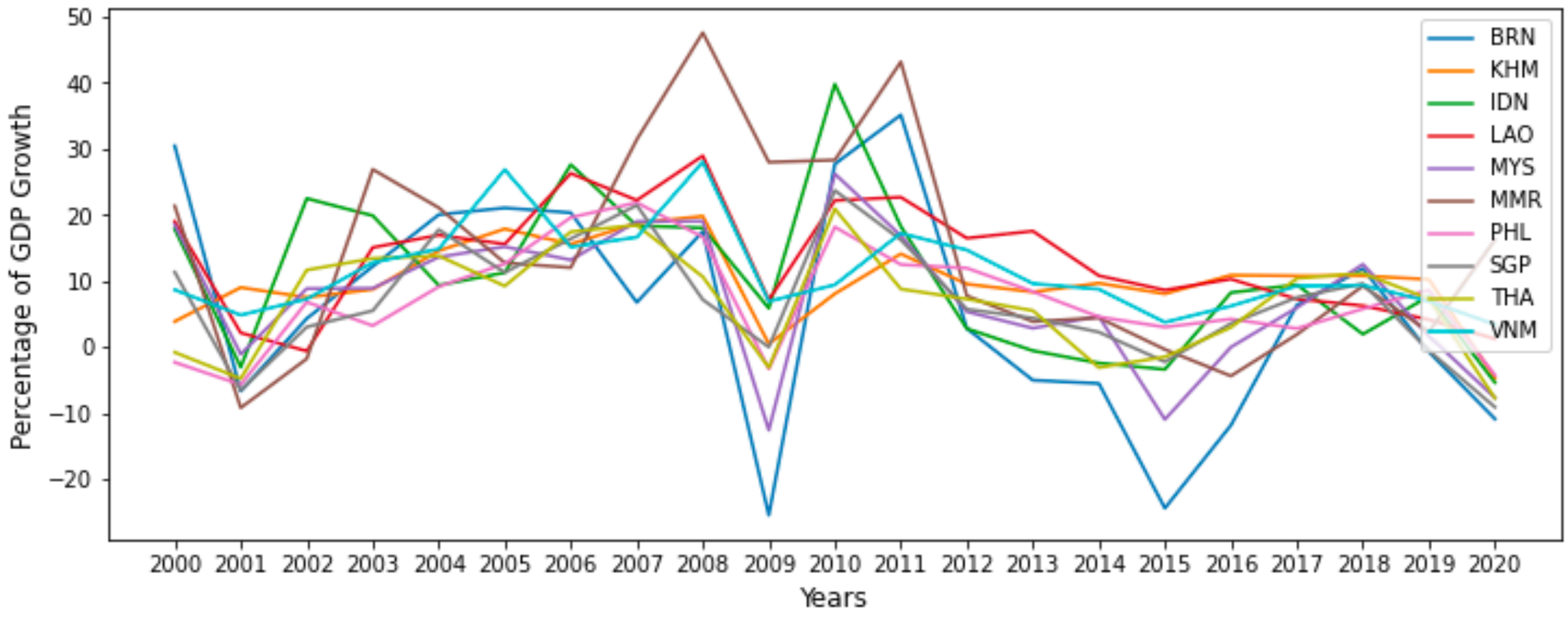
```
asean_countries = ['BRN', 'KHM', 'IDN', 'LAO', 'MYS',
                   'MMR', 'PHL', 'SGP', 'THA', 'VNM']
asean_gdp = gdp_new.loc[asean_countries, '1999':'2020']
```

Country Code	1999	2000	2001	...	2018	2019	2020
BRN	4.600000e+09	6.001153e+09	5.601091e+09	...	1.356735e+10	1.346942e+10	1.200583e+10
KHM	3.517242e+09	3.654032e+09	3.984001e+09	...	2.457175e+10	2.708939e+10	2.580856e+10
IDN	1.400000e+11	1.650000e+11	1.600000e+11	...	1.040000e+12	1.120000e+12	1.060000e+12
...	...	...	...	...	...	...	...
SGP	8.628466e+10	9.607448e+10	8.979494e+10	...	3.760000e+11	3.740000e+11	3.400000e+11
THA	1.270000e+11	1.260000e+11	1.200000e+11	...	5.070000e+11	5.440000e+11	5.020000e+11
VNM	2.868366e+10	3.117252e+10	3.268520e+10	...	2.450000e+11	2.620000e+11	2.710000e+11

Guidelines of Data Visualization

- Line graph

Question 1: Use proper line graph to visualize the GDP growing rates of ASEAN countries between 2000 and 2020.



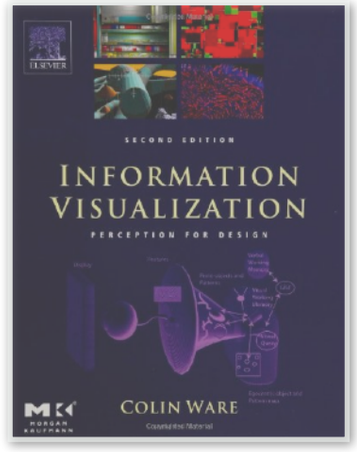
Guidelines of Data Visualization

- Line graph

Question 1: Use proper line graph to visualize the GDP growing rates of ASEAN countries between 2000 and 2020.

It is easy to spot a single hawk in a sky full of pigeons, but if the sky contains a greater variety of birds, the hawk will be more difficult to see.

— **Information Visualization: Perception for Design**

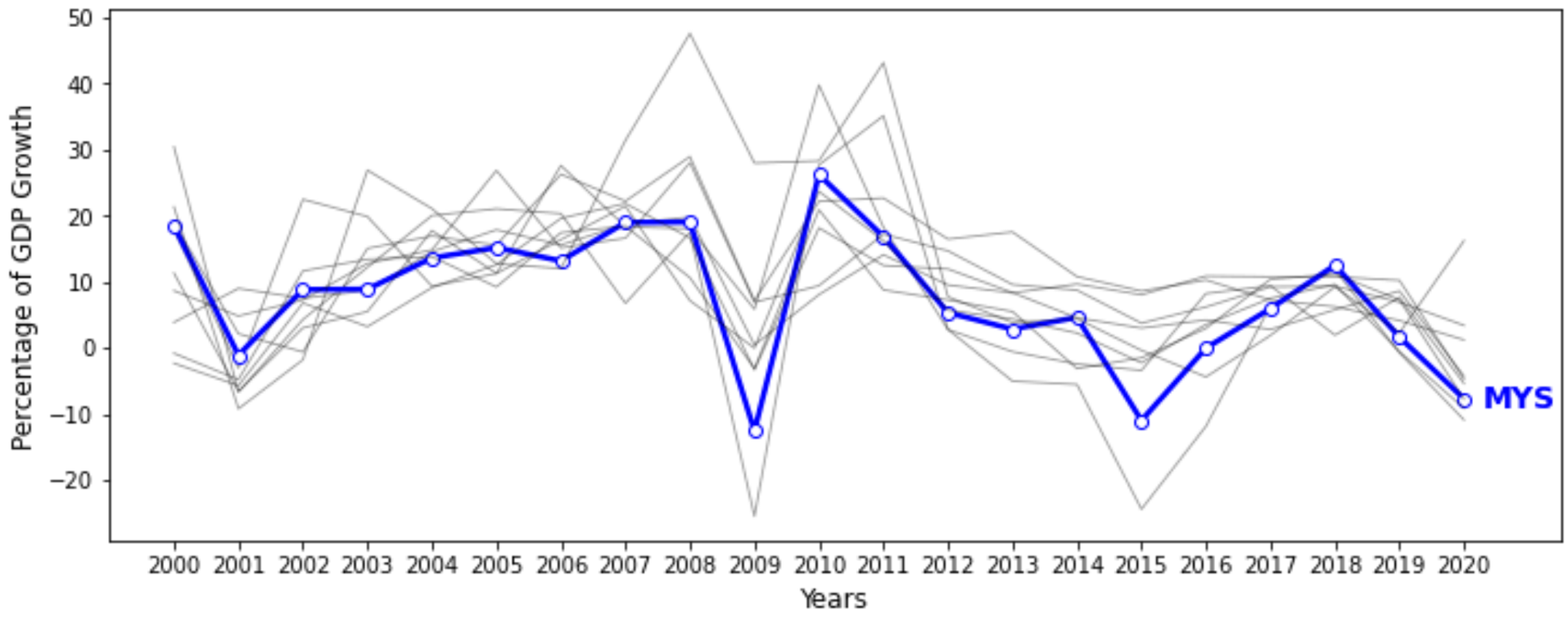


Spaghetti graph is ineffective!!

Guidelines of Data Visualization

- Line graph

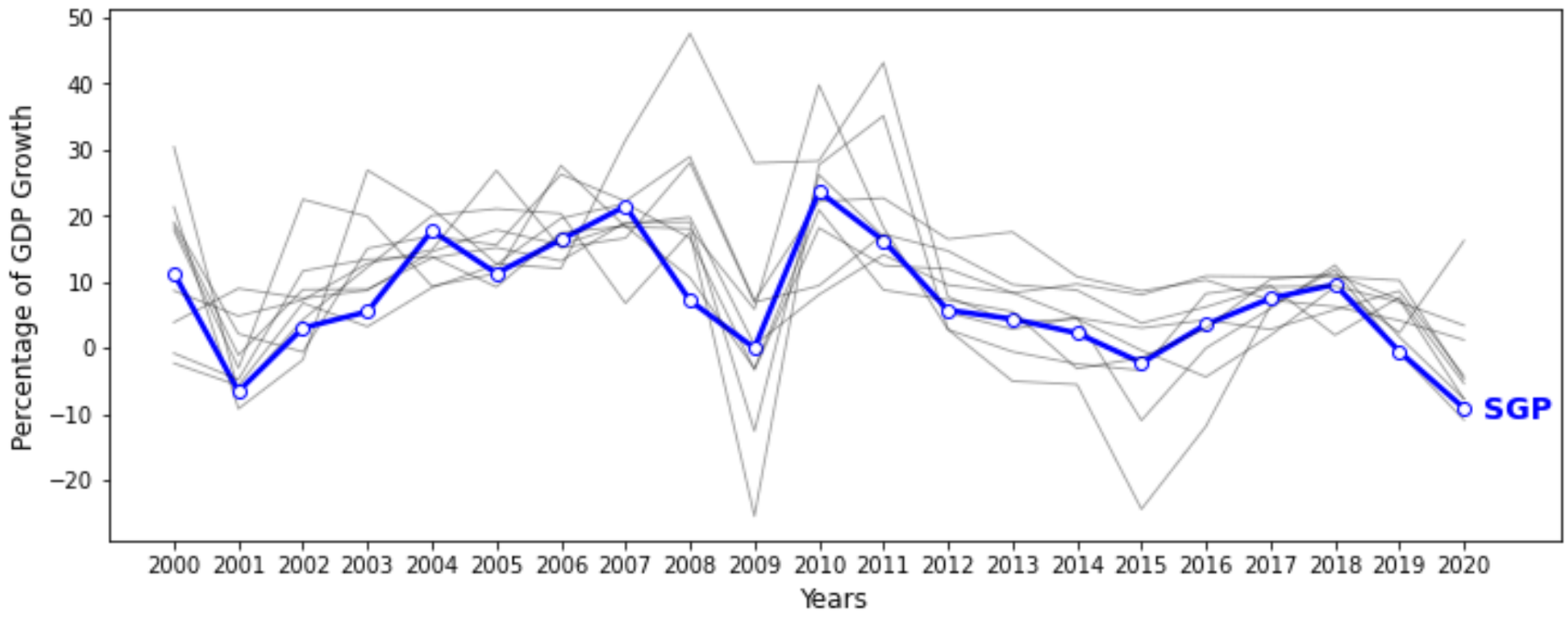
Question 1: Use proper line graph to visualize the GDP growing rates of ASEAN countries between 2000 and 2020.



Guidelines of Data Visualization

- Line graph

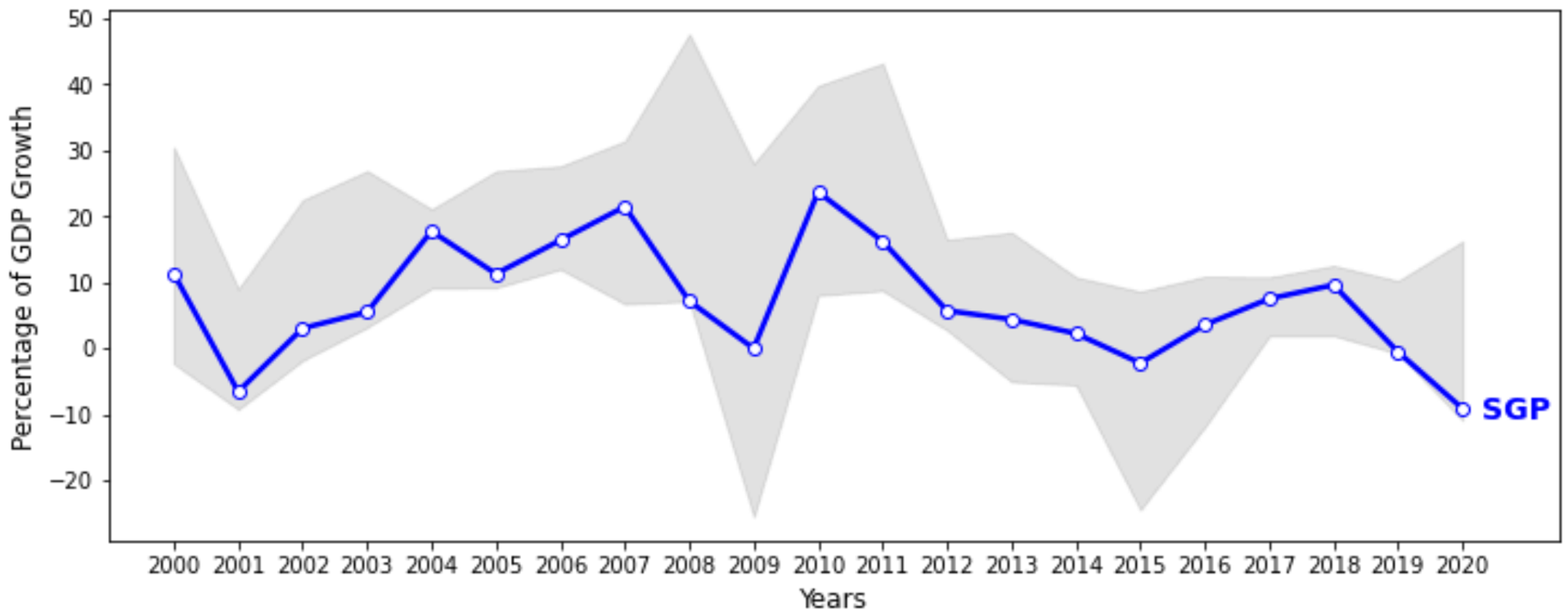
Question 1: Use proper line graph to visualize the GDP growing rates of ASEAN countries between 2000 and 2020.



Guidelines of Data Visualization

- Line graph

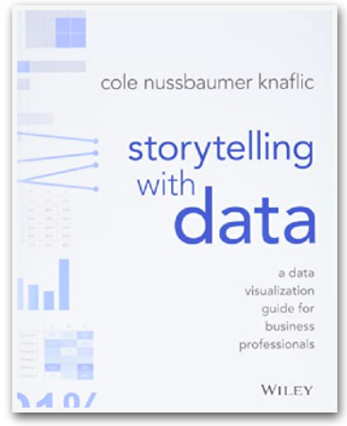
Question 1: Use proper line graph to visualize the GDP growing rates of ASEAN countries between 2000 and 2020.

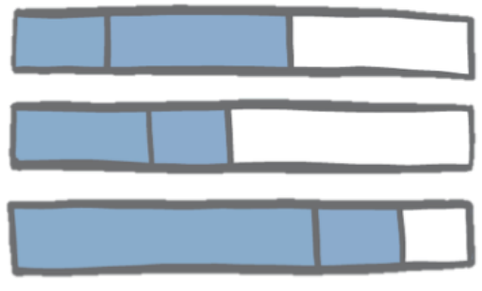


Guidelines of Data Visualization

- Bar graph

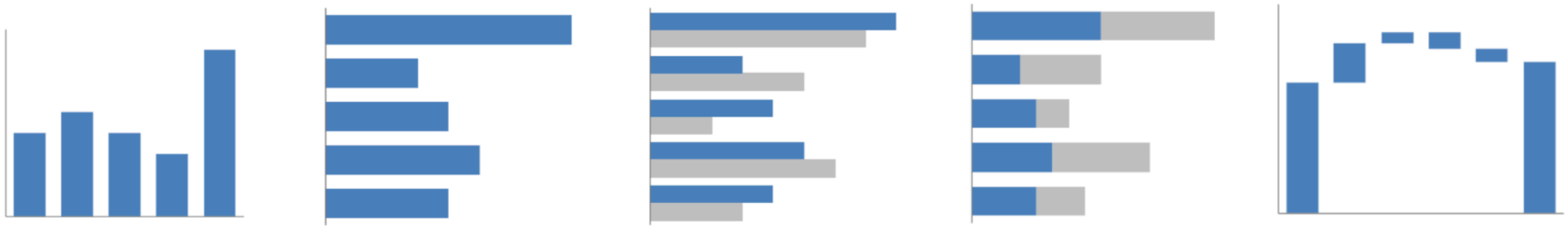
Sometimes bar charts are avoided because they are common. This is a mistake. Rather, bar charts should be leveraged because they are common, as this means less of a learning curve for your audience. Instead of using their brain power to try to understand how to read the graph, your audience spends it figuring out what information to take away from the visual.— **Storytelling with Data**





Guidelines of Data Visualization

- Bar graph



Vertical bar chart | Horizontal bar chart | Clustered bars | Stacked bars | Waterfall chart

Types of bar charts	`matplotlib` tools	Remarks
Vertical bar chart	`bar()` function	Most common bar chart
Horizontal bar chart	`barh()` function	• Implying the ranking of values • Better for long tick labels
Clustered bars	• `bar()` function, with shifted `x` values • `barh()` function, with shifted `y` values	Better for comparing data
Stacked bars	• `bar()` function, with specific `bottom` values • `barh()` function, with specific `left` values	Besides comparing the totals, also show the subcomponent pieces within each category.
Waterfall chart	`bar()` function, with specific `bottom` values	Better for visualizing the changes

Guidelines of Data Visualization

- Bar graph

Question 2: Visualize the GDPs of ASEAN countries in the year of 2019.

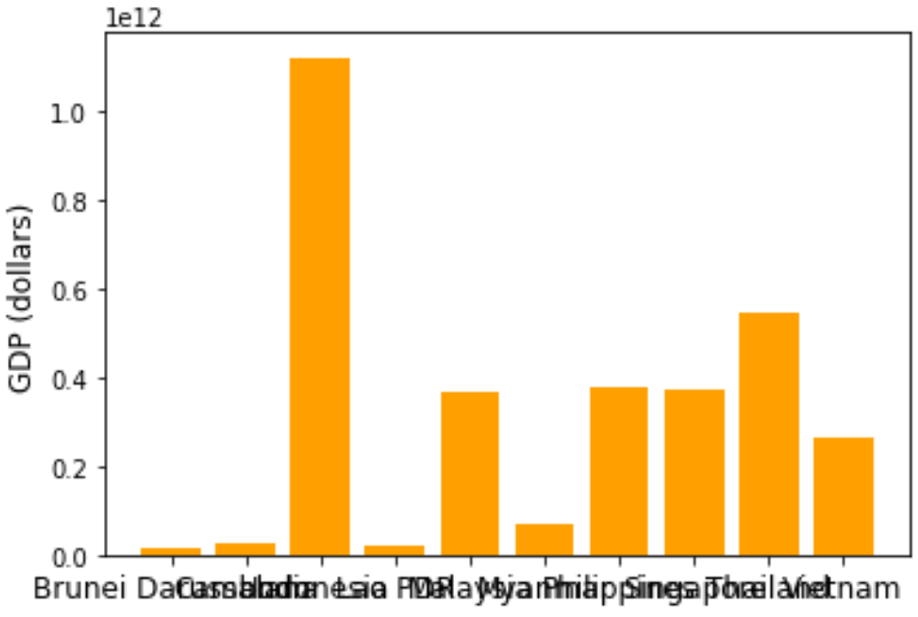
```
asean_gdp['2019']
```

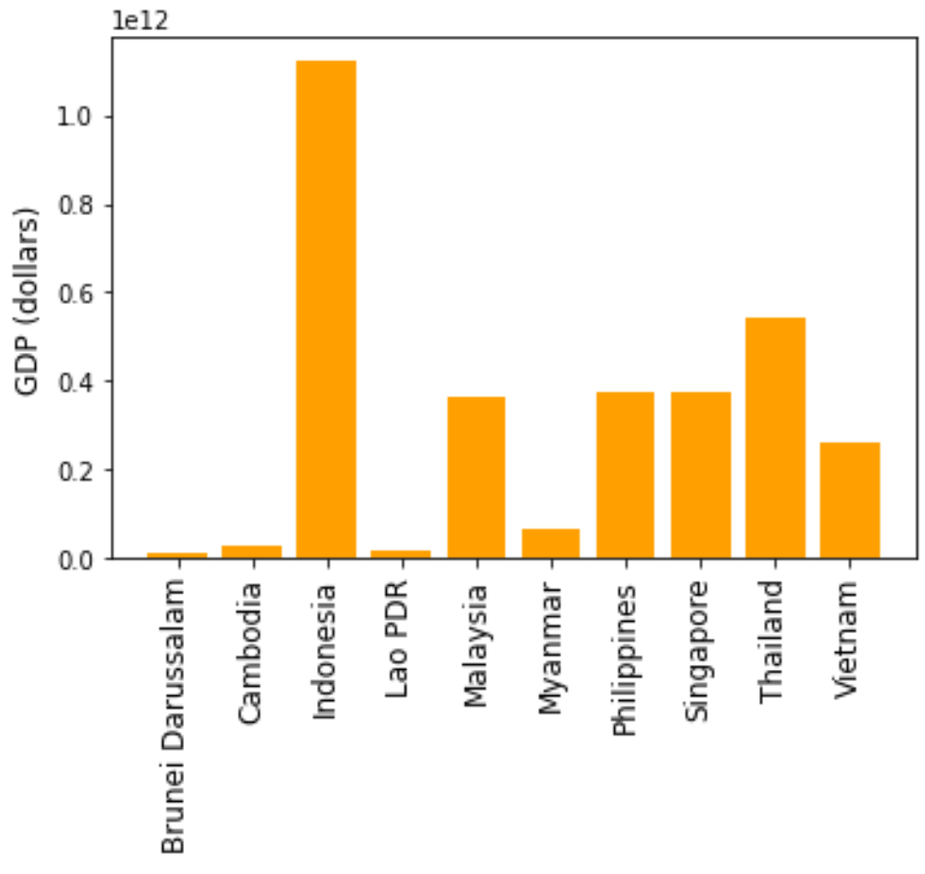
```
Country Code
BRN    1.346942e+10
KHM    2.708939e+10
IDN    1.120000e+12
LAO    1.889725e+10
MYS    3.650000e+11
MMR    6.869776e+10
PHL    3.770000e+11
SGP    3.740000e+11
THA    5.440000e+11
VNM    2.620000e+11
Name: 2019, dtype: float64
```

Guidelines of Data Visualization

- Bar graph

Question 2: Visualize the GDPs of ASEAN countries in the year of 2019.

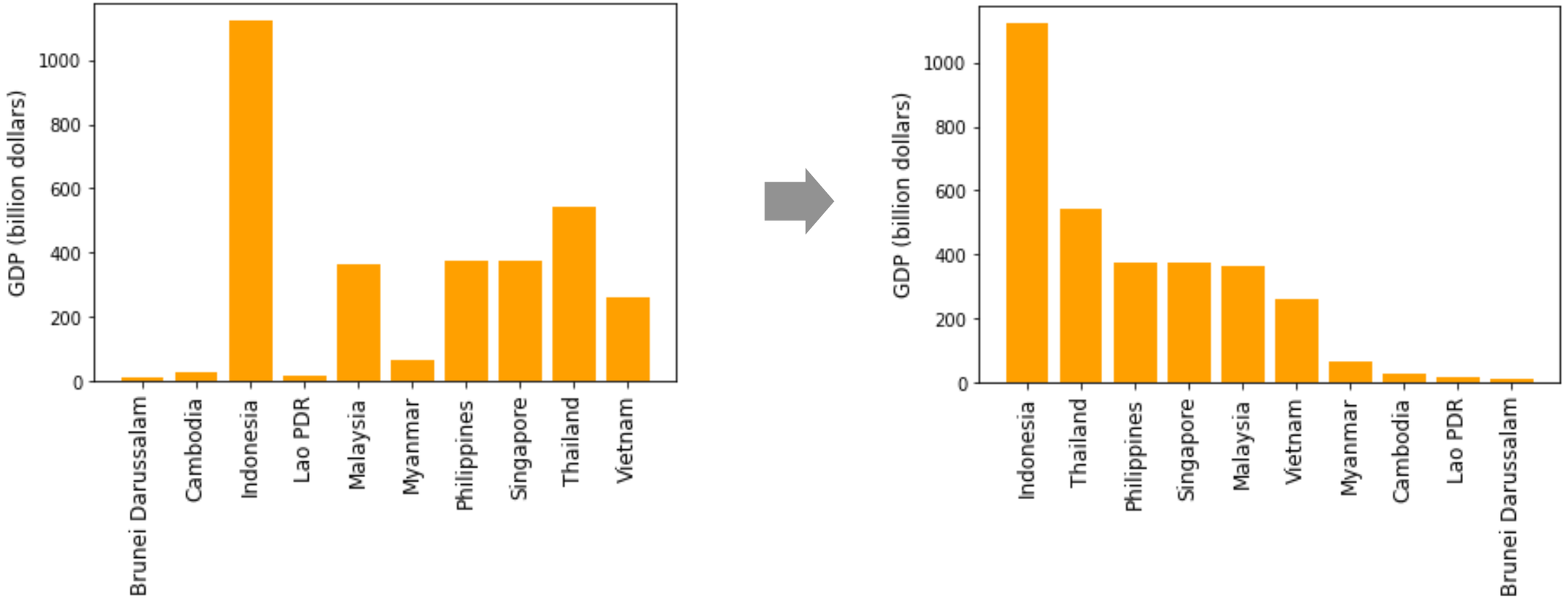




Guidelines of Data Visualization

- Bar graph

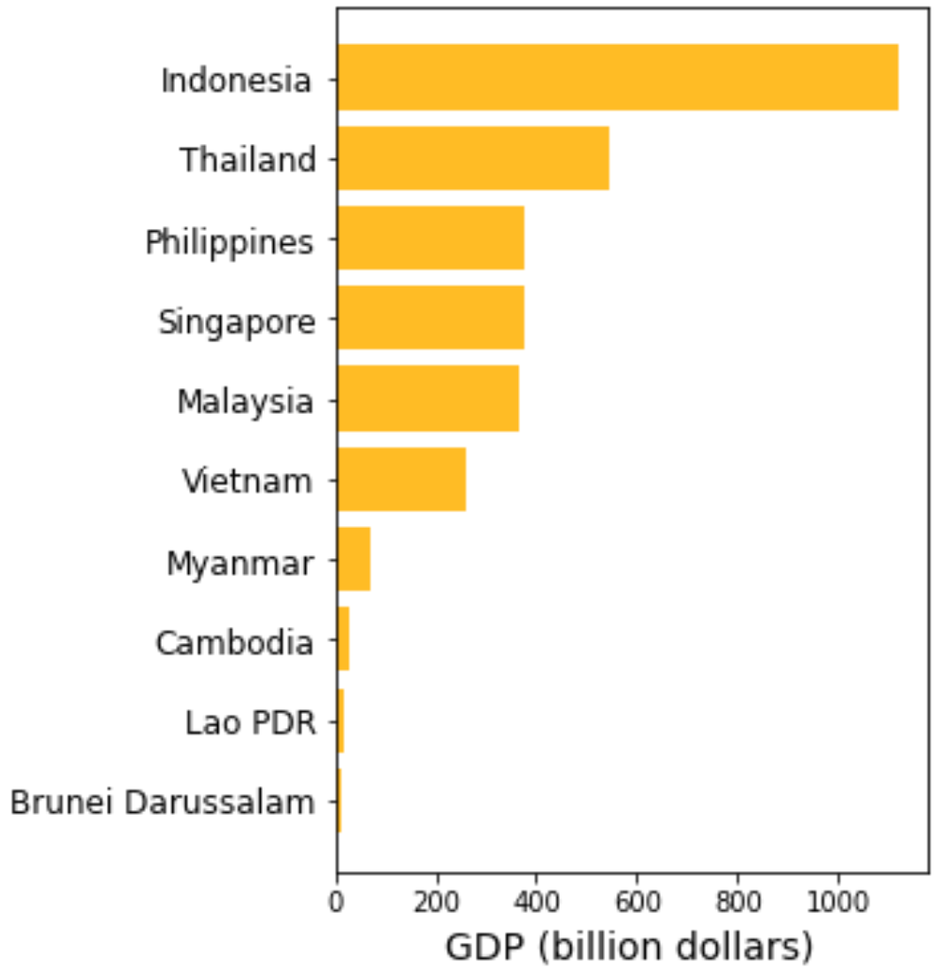
> **Question 2:** Visualize the GDPs of ASEAN countries in the year of 2019.



Guidelines of Data Visualization

- Bar graph

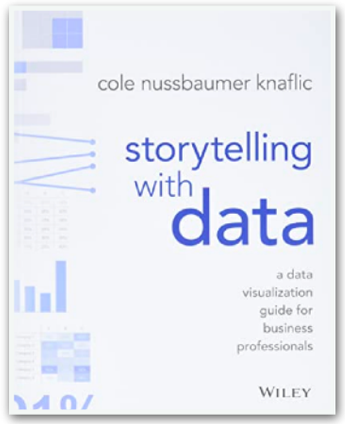
Question 2: Visualize the GDPs of ASEAN countries in the year of 2019.



Guidelines of Data Visualization

- Scatterplot

> Scatterplots can be useful for showing the relationship between two things, because they allow you to encode data simultaneously on a horizontal x-axis and vertical y-axis to see whether and what relationship exists.— **Storytelling with Data**



> An extension of a scatterplot, a bubble chart is commonly used to visualize relationships between three or more numeric variables. Each bubble in a chart represents a single data point. The values for each bubble are encoded by 1) its horizontal position on the x-axis, 2) its vertical position on the y-axis, and 3) the size of the bubble. Sometimes, the color of the bubble or its movement in animation can represent more dimensions.
>
> storytelling WITH data®

Guidelines of Data Visualization

- Scatterplot

Example 3: The dataset "life_expectancy" provides the life expectancy, public health data, and other related information. Visualize the influence of GDP per capita, the total health expenditure, and the percentage of private health expenditure on life expectancy. You may need to use the following variables:

- **`expectancy`**: the life expectancy of each country.
- **`gdp`**: the GDP of each country.
- **`total_health`**: the total health expenditure as a percentage of GDP.
- **`private_health`**: the percentage of health expenditures funded from domestic private sources. Domestic private sources include funds from households, corporations and non-profit organizations.

Guidelines of Data Visualization

- Scatterplot

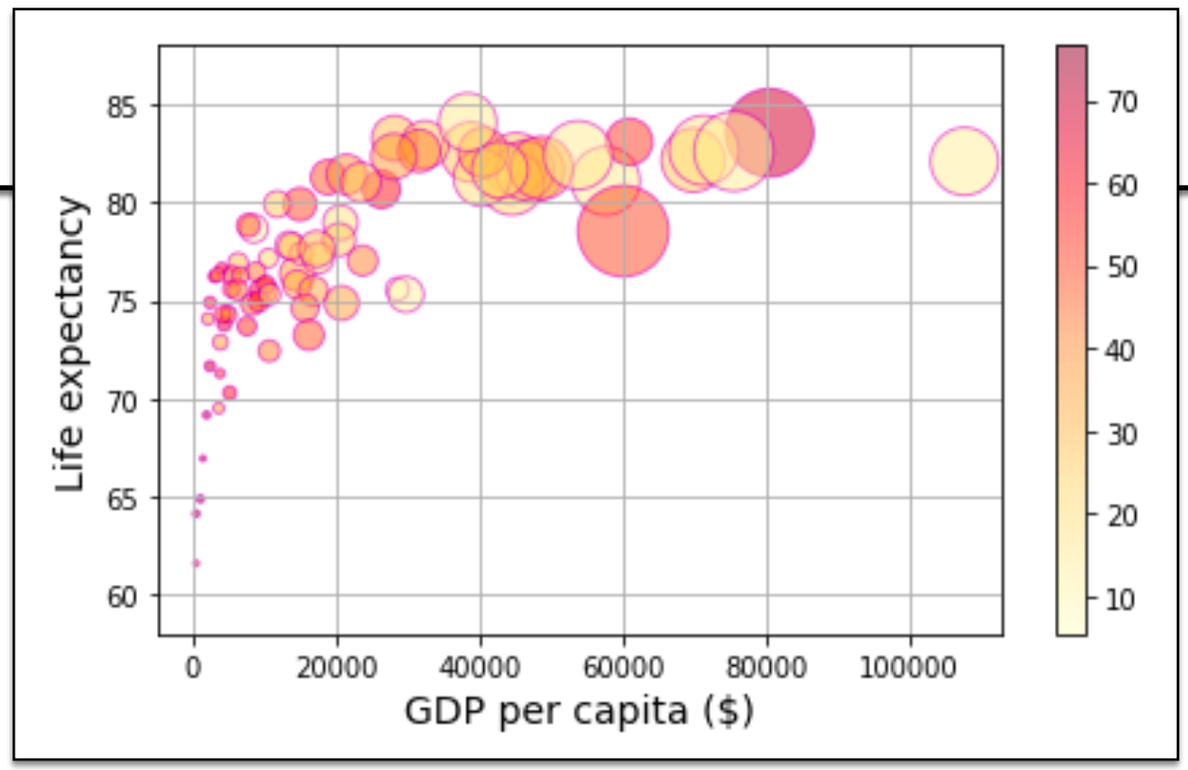
```
life = pd.read_csv('life_expectancy.csv')
life
```

```
gdp = life['gdp']
expectancy = life['expectancy']
total_exp = life['total_health']*life['gdp']/100
private_health = life['private_health']
```

Guidelines of Data Visualization

- Scatterplot

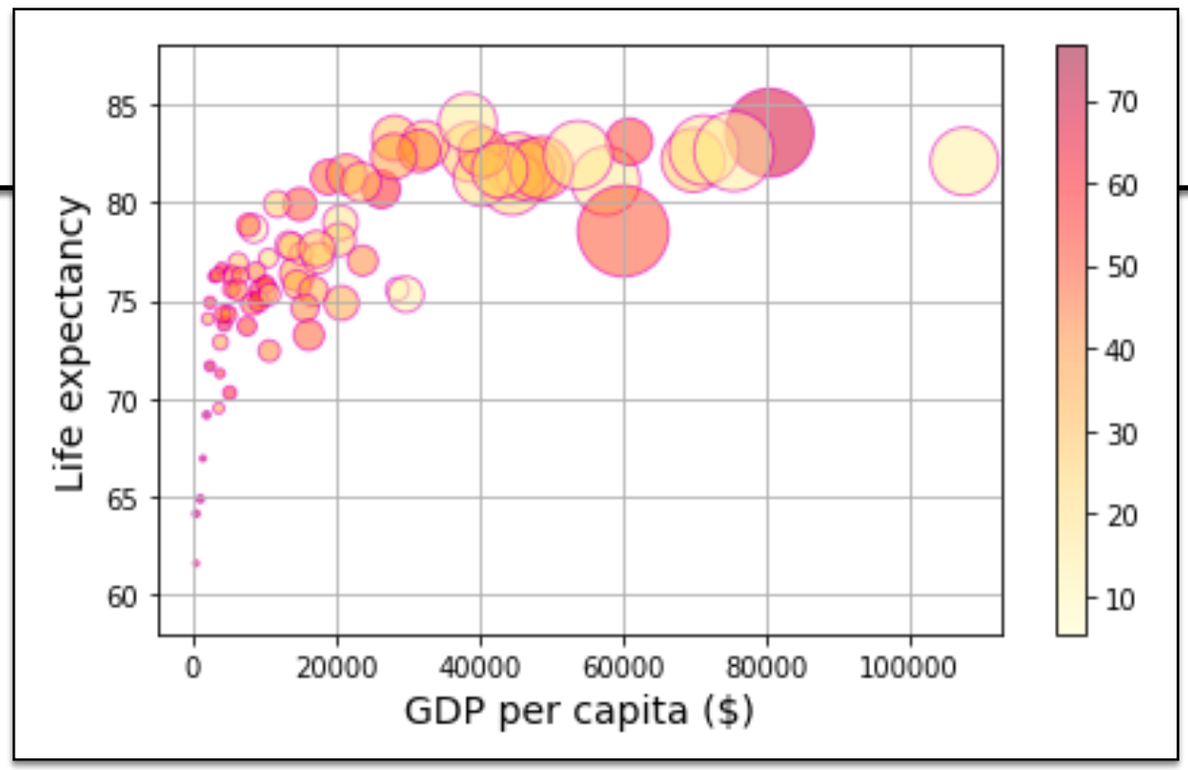
```
plt.figure(figsize=(7, 4))
plt.scatter(gdp, expectancy, s=total_exp/9,
            c=private_health, edgecolor='m',
            cmap='YlOrRd', alpha=0.5)
plt.colorbar()
plt.xlabel('GDP per capita ($)', fontsize=14)
plt.ylabel('Life expectancy', fontsize=14)
plt.ylim([58, 88])
plt.grid()
plt.show()
```



Guidelines of Data Visualization

- Scatterplot

```
plt.figure(figsize=(7, 4))
plt.scatter(gdp, expectancy, s=total_exp/9,
            c=private_health, edgecolor='m',
            cmap='YlOrRd', alpha=0.5)
plt.colorbar()
plt.xlabel('GDP per capita ($)', fontsize=14)
plt.ylabel('Life expectancy', fontsize=14)
plt.ylim([58, 88])
plt.grid()
plt.show()
```



✓ Data on the x-axis and y-axis

Guidelines of Data Visualization

- Scatterplot

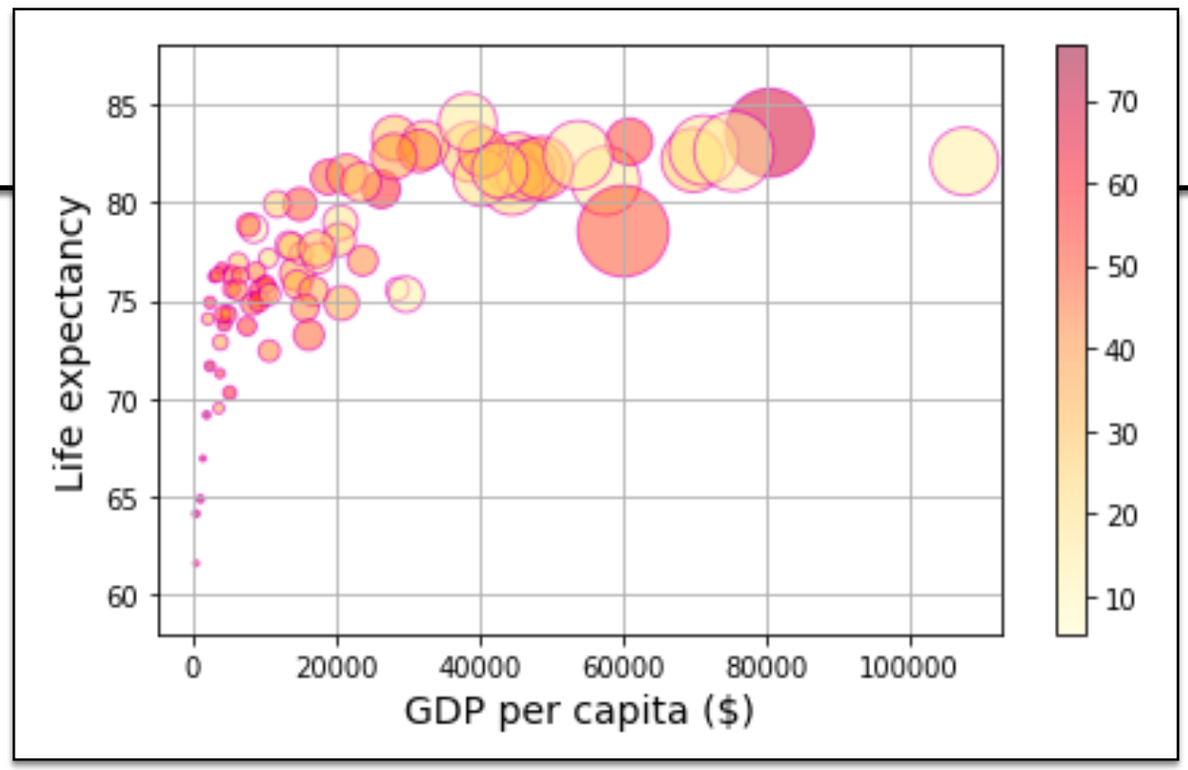
```
plt.figure(figsize=(7, 4))
plt.scatter(gdp, expectancy, s=total_exp/9,
            c=private_health, edgecolor='m',
            cmap='YlOrRd', alpha=0.5)
plt.colorbar()
plt.xlabel('GDP per capita ($)', fontsize=14)
plt.ylabel('Life expectancy', fontsize=14)
plt.ylim([58, 88])
plt.grid()
plt.show()
```

✓ Data on the x-axis and y-axis
✓ Bubble sizes

Guidelines of Data Visualization

- Scatterplot

```
plt.figure(figsize=(7, 4))
plt.scatter(gdp, expectancy, s=total_exp/9,
            c=private_health, edgecolor='m',
            cmap='YlOrRd', alpha=0.5)
plt.colorbar()
plt.xlabel('GDP per capita ($)', fontsize=14)
plt.ylabel('Life expectancy', fontsize=14)
plt.ylim([58, 88])
plt.grid()
plt.show()
```



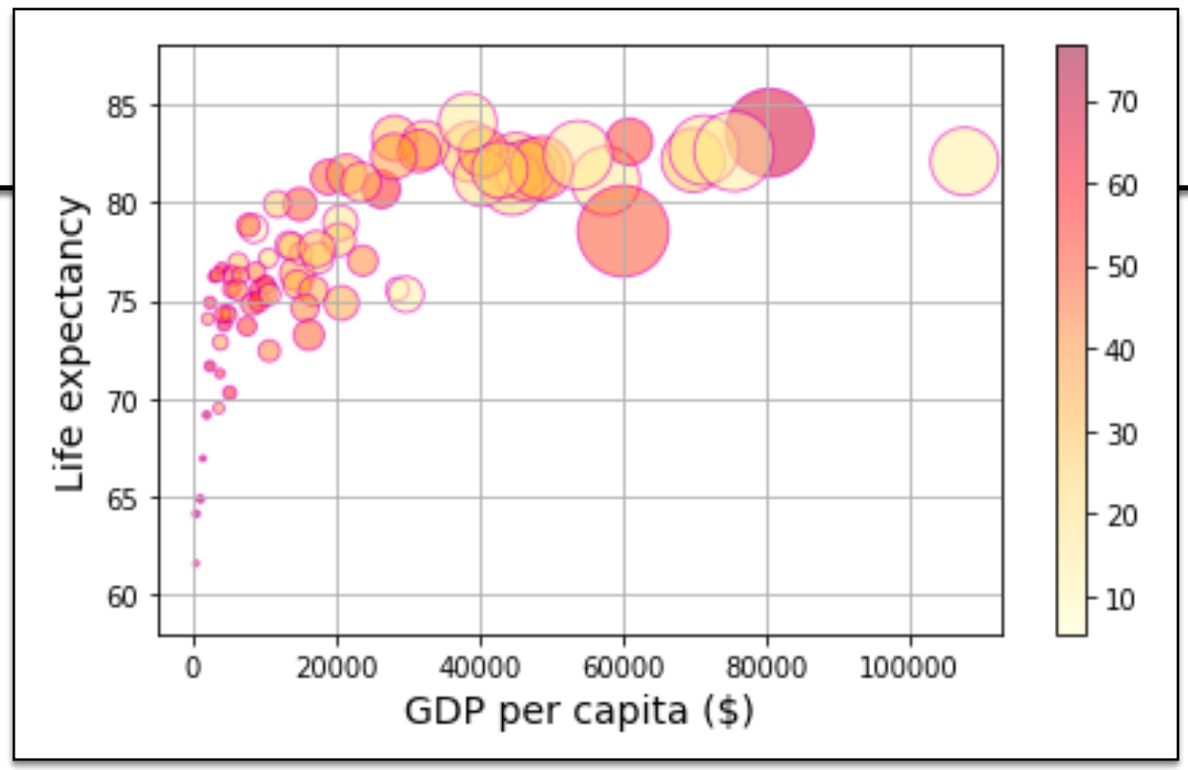
- ✓ Data on the x-axis and y-axis
- ✓ Bubble sizes
- ✓ Bubble colors

Guidelines of Data Visualization

- Scatterplot

```
plt.figure(figsize=(7, 4))
plt.scatter(gdp, expectancy, s=total_exp/9,
            c=private_health, edgecolor='m',
            cmap='YlOrRd', alpha=0.5)
plt.colorbar()
plt.xlabel('GDP per capita ($)', fontsize=14)
plt.ylabel('Life expectancy', fontsize=14)
plt.ylim([58, 88])
plt.grid()
plt.show()
```

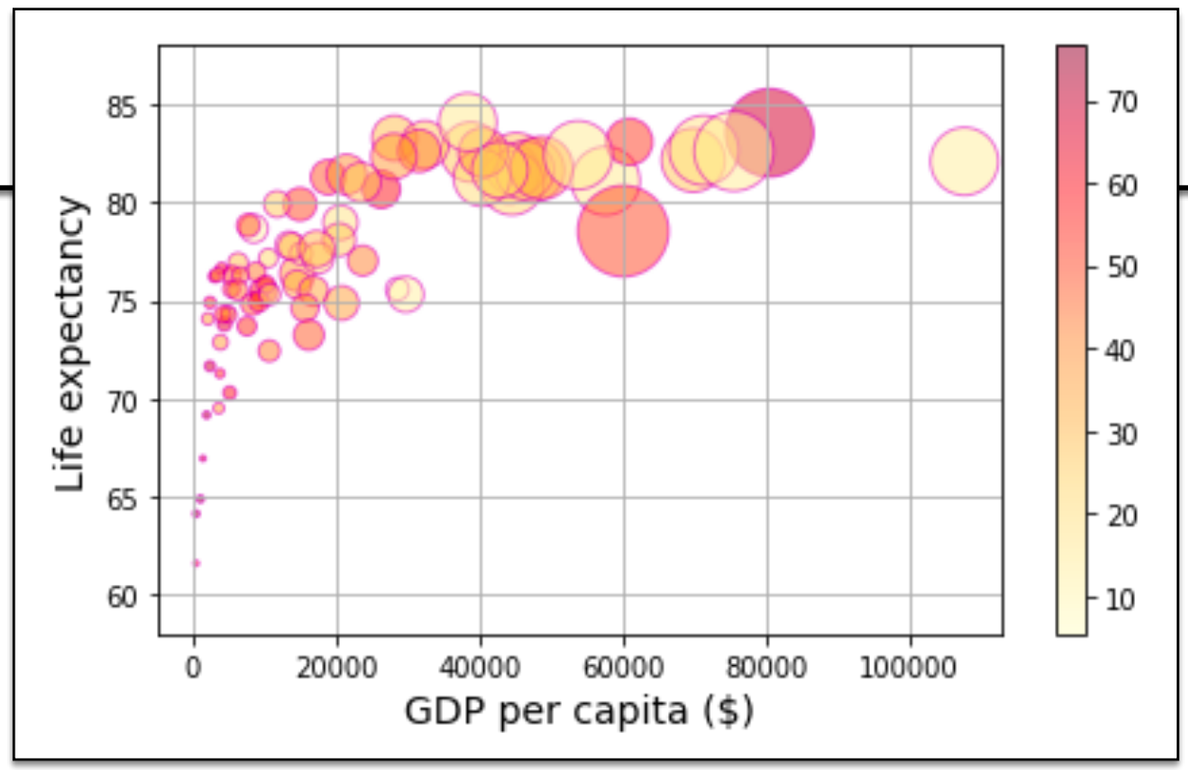
- ✓ Data on the x-axis and y-axis
- ✓ Bubble sizes
- ✓ Bubble colors
- ✓ Edge color of bubbles



Guidelines of Data Visualization

- Scatterplot

```
plt.figure(figsize=(7, 4))
plt.scatter(gdp, expectancy, s=total_exp/9,
            c=private_health, edgecolor='m',
            cmap='YlOrRd', alpha=0.5)
plt.colorbar()
plt.xlabel('GDP per capita ($)', fontsize=14)
plt.ylabel('Life expectancy', fontsize=14)
plt.ylim([58, 88])
plt.grid()
plt.show()
```

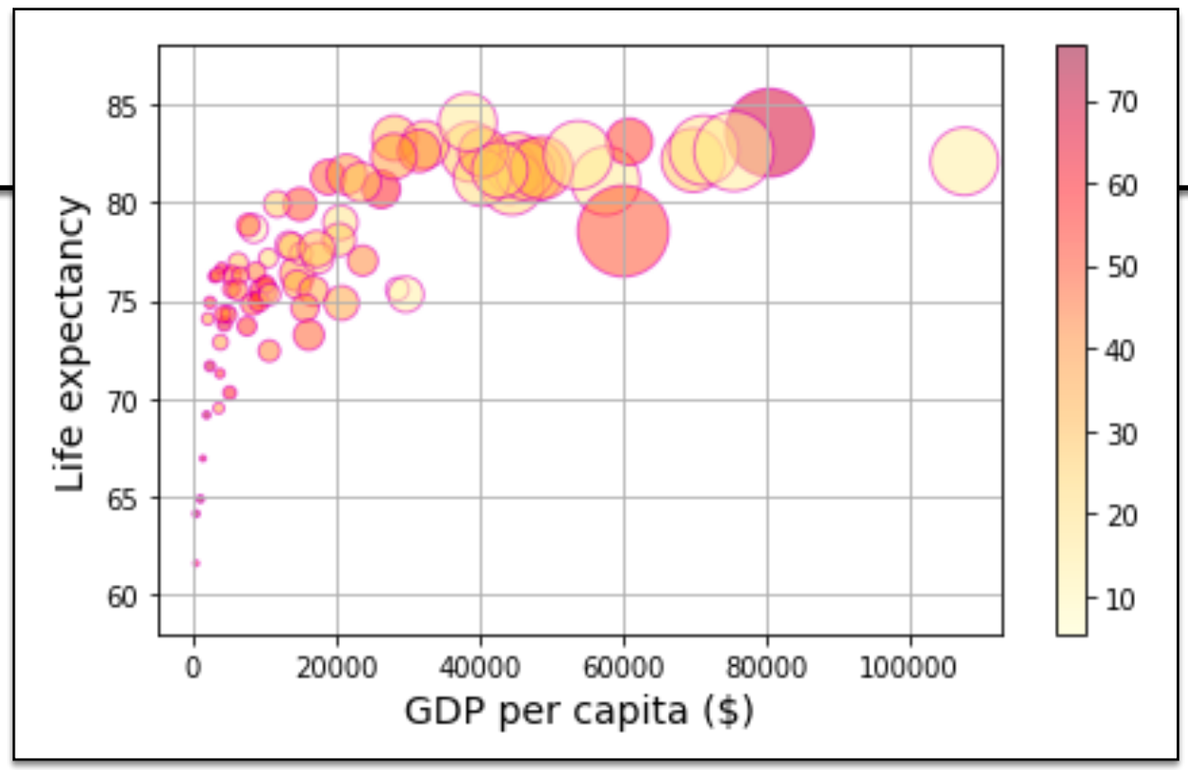


- ✓ Data on the x-axis and y-axis
- ✓ Bubble sizes
- ✓ Bubble colors
- ✓ Edge color of bubbles
- ✓ Color map

Guidelines of Data Visualization

- Scatterplot

```
plt.figure(figsize=(7, 4))
plt.scatter(gdp, expectancy, s=total_exp/9,
            c=private_health, edgecolor='m',
            cmap='YlOrRd', alpha=0.5)
plt.colorbar()
plt.xlabel('GDP per capita ($)', fontsize=14)
plt.ylabel('Life expectancy', fontsize=14)
plt.ylim([58, 88])
plt.grid()
plt.show()
```

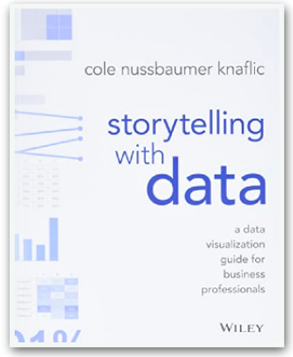


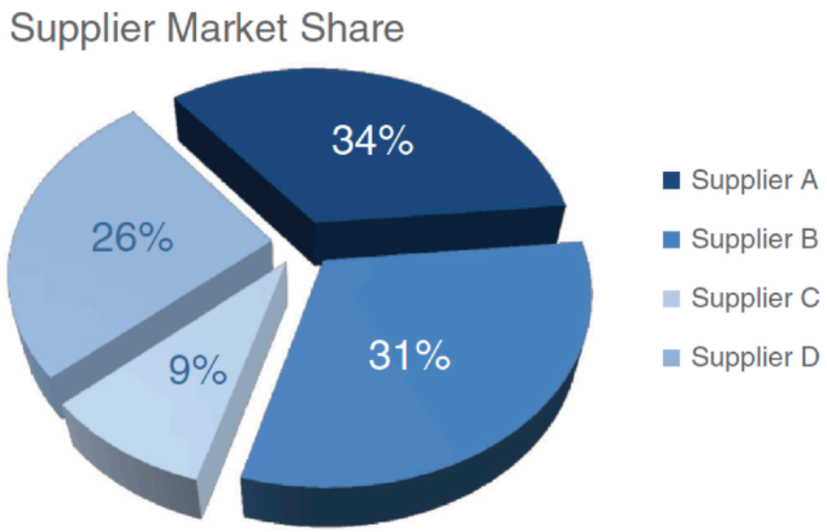
- ✓ Data on the x-axis and y-axis
- ✓ Bubble sizes
- ✓ Bubble colors
- ✓ Edge color of bubbles
- ✓ Color map
- ✓ Transparency

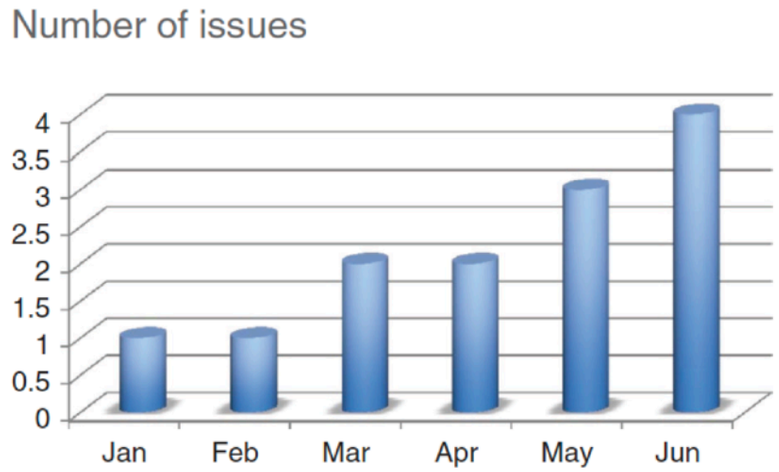
Guidelines of Data Visualization

- What to avoid
  - 3D visuals

> One of the golden rules of data visualization goes like this: never use 3D.— **Storytelling with Data**

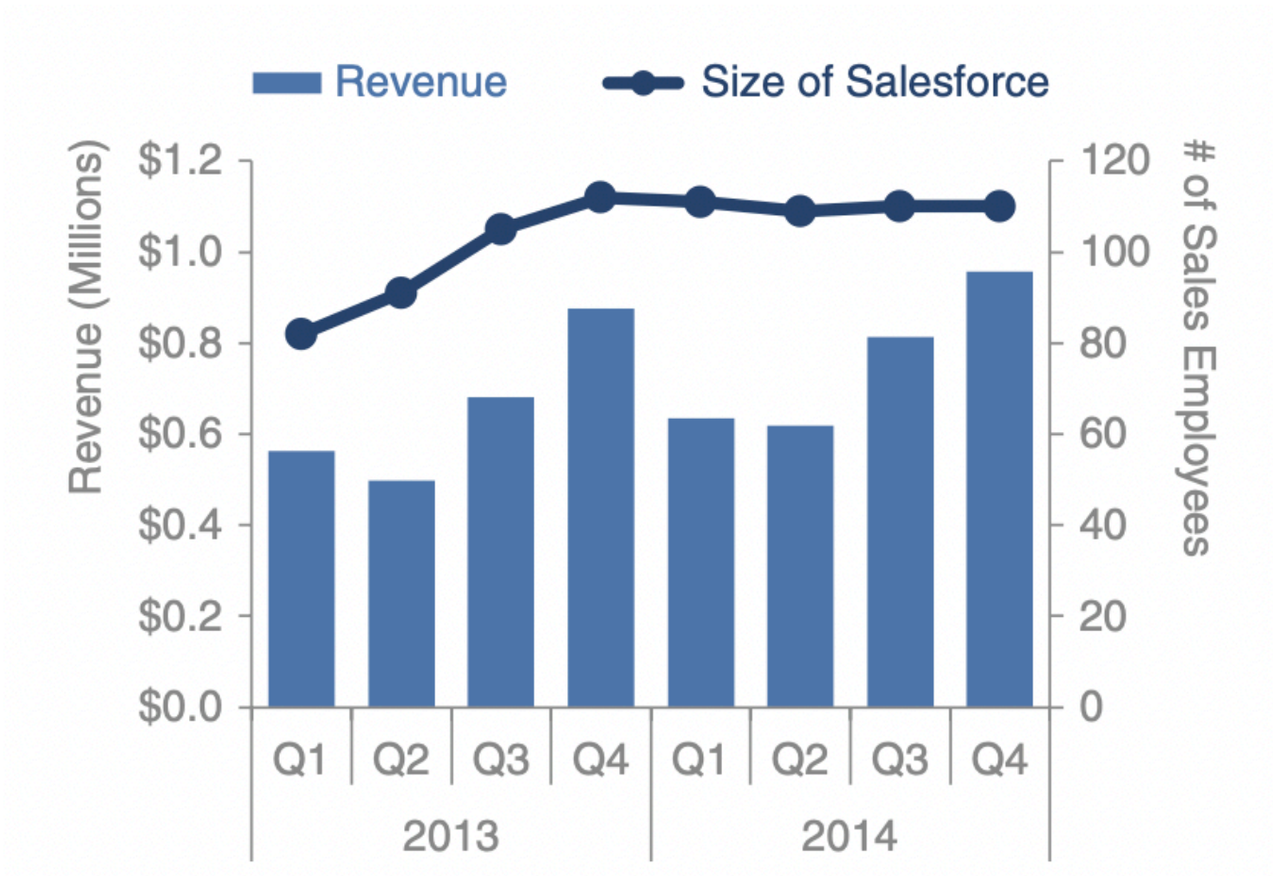






Guidelines of Data Visualization

- What to avoid
  - Secondary y-axis



Guidelines of Data Visualization

- What to avoid
  - Tables in presentations

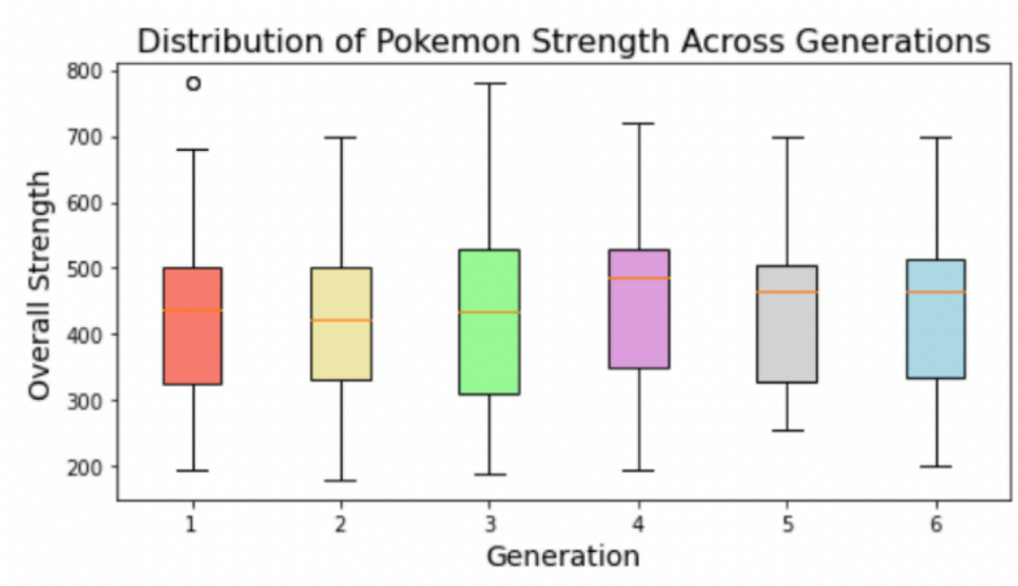
> Using a table in a live presentation is rarely a good idea. As your audience reads it, you lose their ears and attention to make your point verbally.— **Storytelling with Data**

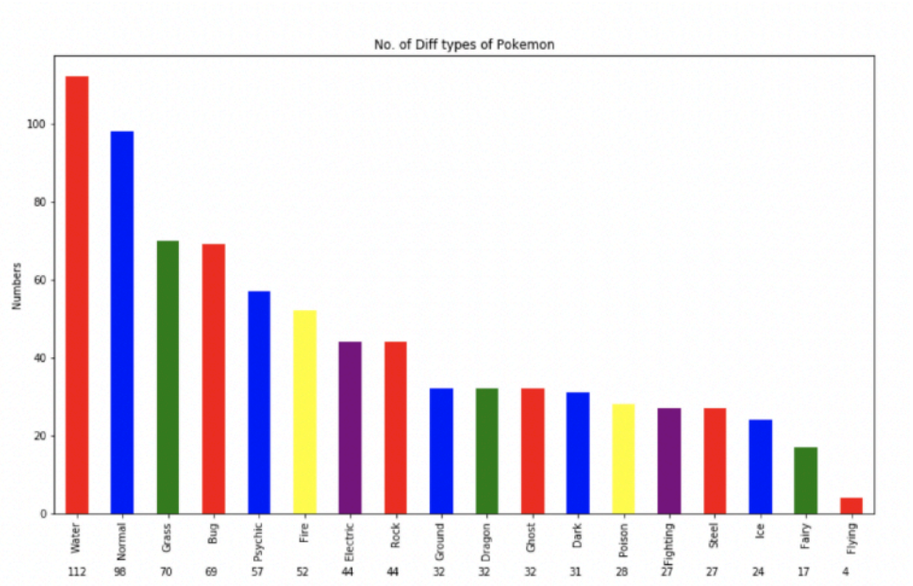
GEO and Desc	Consulting	Research	Products	Components	Systems	Support	TOTAL
Grand Total	**4,634,068.91**	**3,883,621.28**	**5,843,307.44**	**2,890,064.81**	**2,531,168.87**	**831,178.79**	**20,395,527.67**
UK	1,812,064.34	1,659,130.10	1,677,140.18	1,734,038.89	1,593,727.52	287,646.49	8,763,747.52
Germany	385,195.50	145,927.68	737,681.62	520,211.67	129	70,969.04	1,860,114.51
France	53,743.00	360,629.35	296,676.27	57,801.30	261,631.24	41,861.04	1,072,342.20
Italy	9,730.00	62,204.26	287,966.62	28,900.65	0	37,208.19	426,009.72
Spain	30,246.48	935,909.79	160,940.14	86,701.94	0	29,158.87	1,242,957.22
Portugal	0.00	124,531.50	115,279.71	31,790.71	0.00	11,678.77	283,280.69
Sw itzerland	1,194,630.10	209,724.09	302,458.78	31,790.71	278,337.21	53,392.28	2,070,333.17
Nertherlands	317,484.50	85,999.93	320,678.71	34,680.78	0.00	44,457.28	803,301.20
Western Europe Other	119,394.74	284,601.35	1,281,434.33	28,900.65	397,085.90	199,308.40	2,310,725.37
Russia	94,053.01	6,532.05	211,934.18	14,450.32	0.00	8,548.41	335,517.97
Poland	325,191.36	8,431.18	237,751.17	21,675.49	0.00	7,490.30	600,539.50
Eastern Europe Other	292,335.88	62,453.86	213,365.73	18,785.42	258.00	39,459.72	626,658.61

Guidelines of Data Visualization

- What to avoid
  - Change color without good reasons

> One question regularly raised in my workshops is around novelty. Does it make sense to change up the colors or graph types so the audience doesn't get bored? My answer is a resounding No! The story you are telling should be what keeps your audience's attention, not the design elements of your graphs.— **Storytelling with Data**





Guidelines of Data Visualization

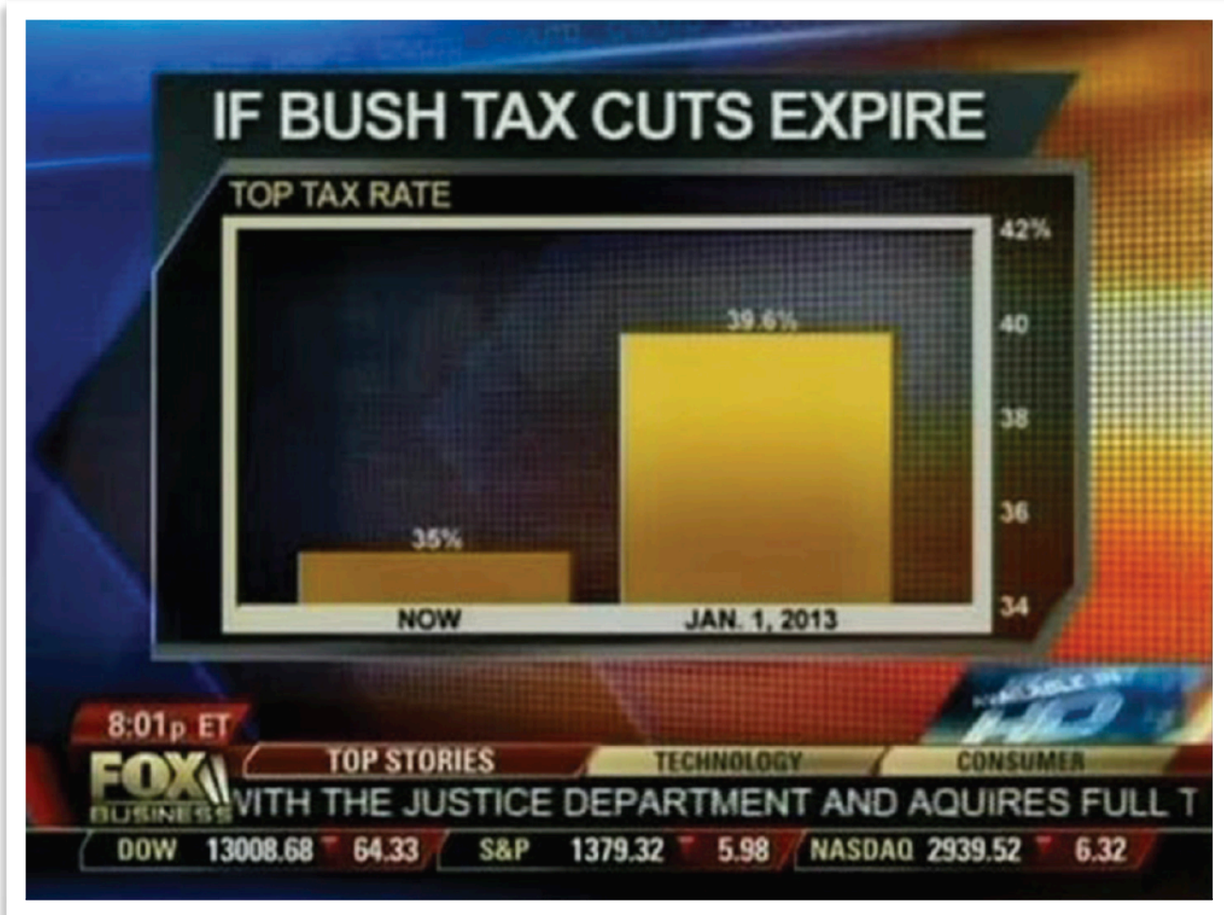
- What to avoid
  - Change color without good reasons

> One question regularly raised in my workshops is around novelty. Does it make sense to change up the colors or graph types so the audience doesn't get bored? My answer is a resounding No! The story you are telling should be what keeps your audience's attention, not the design elements of your graphs.— **Storytelling with Data**

> A change in colors signals just that—a change. So leverage this when you want your audience to feel change for some reason, but never simply for the sake of novelty.
> — **Storytelling with Data**

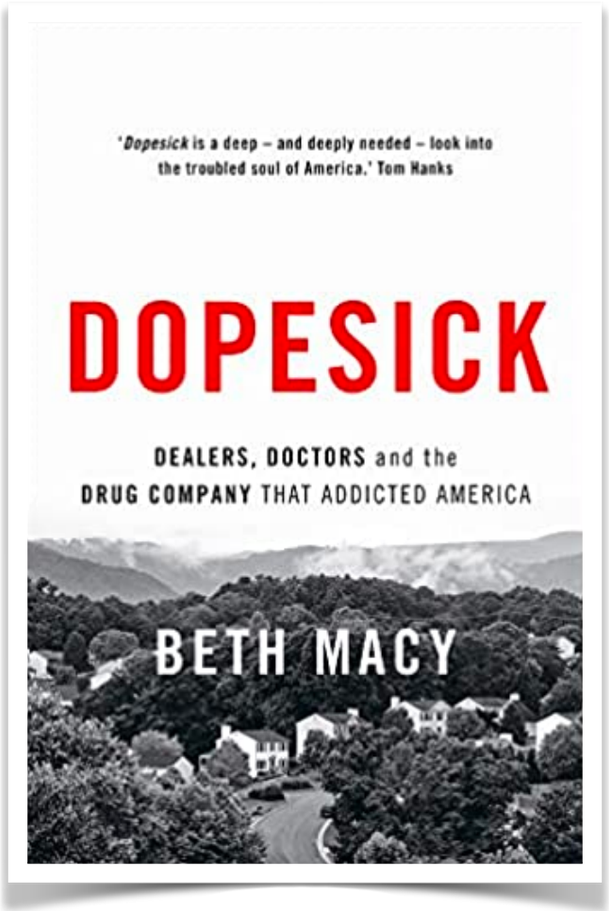
Guidelines of Data Visualization

- What to avoid
  - Unethical use of data visualization



Guidelines of Data Visualization

- What to avoid
  - Unethical use of data visualization



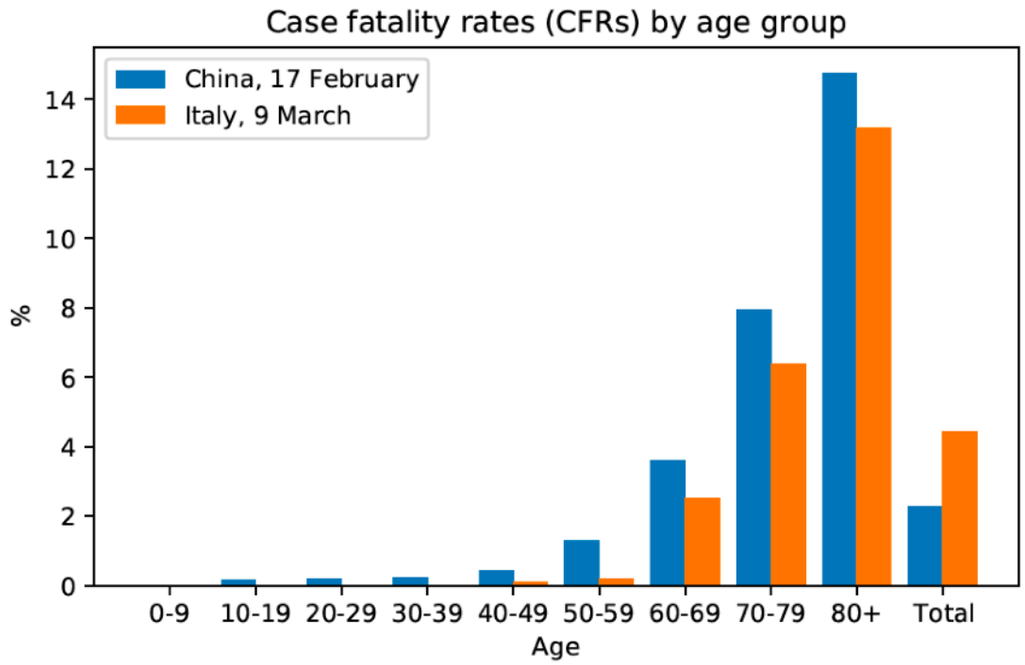
Guidelines of Data Visualization

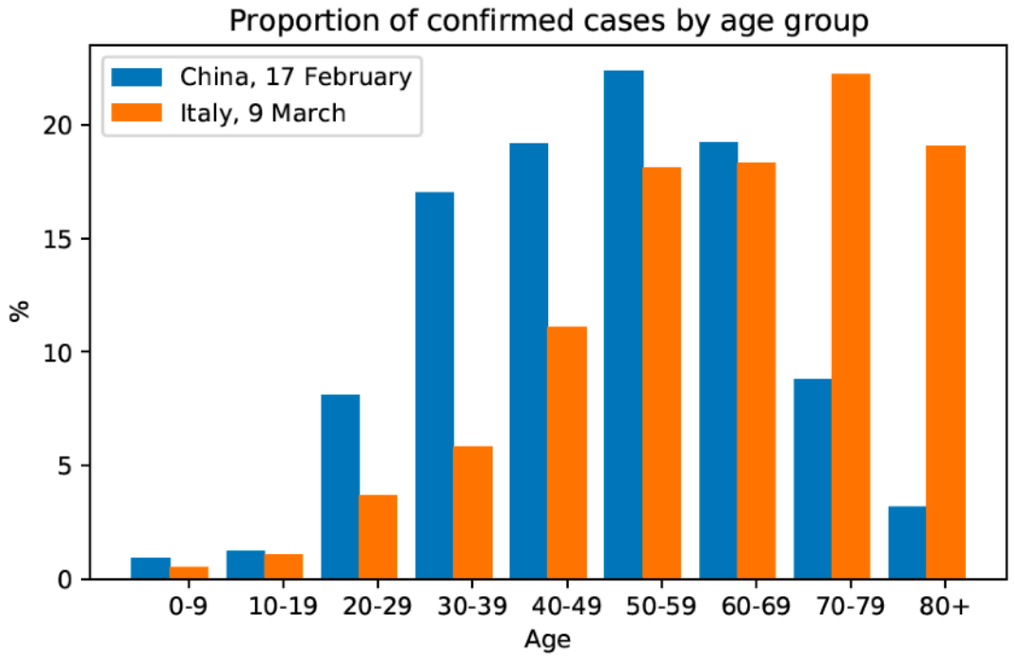
- What to avoid
  - Unethical use of data visualization

Guidelines of Data Visualization

- Interpretation of data visuals

Question 3: The bar chart below shows the fatality rates of Covid-19 in Italy and China in 2020, respectively. It is observed that the fatality rate for each age group in China is higher than Italy, but the overall fatality rate in Italy is higher than China. Can you give an explanation to the seemingly paradoxical data.





Guidelines of Data Visualization

- Interpretation of data visuals

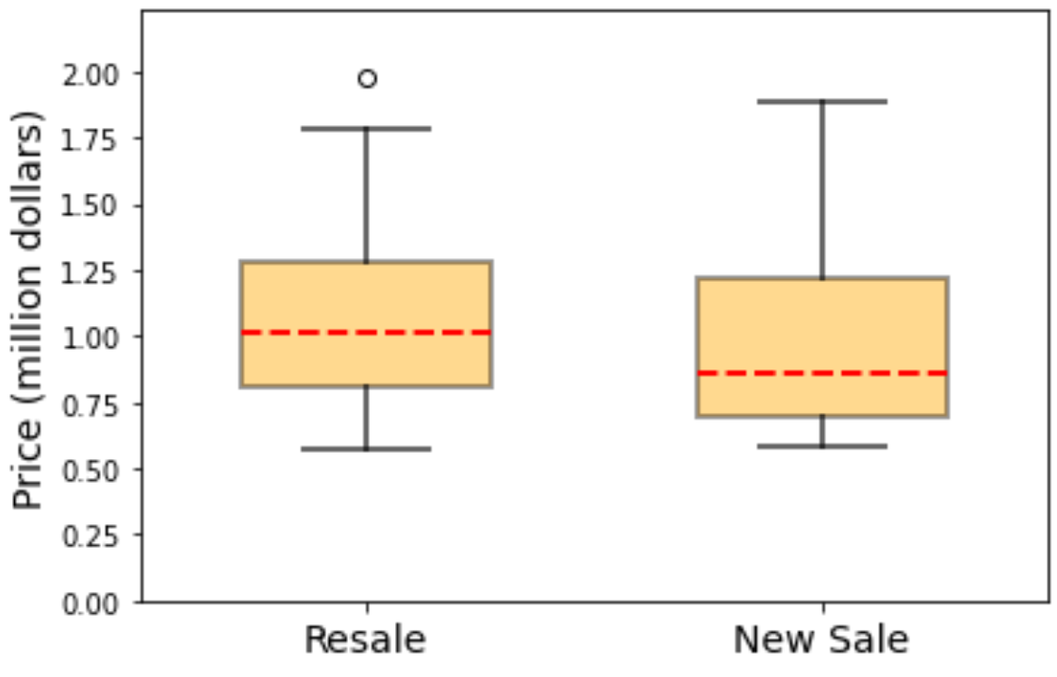
Question 4: The file "condo_2018.csv" provides a dataset of condominium transactions in the year of 2018. All properties involved in these transactions are located in district 28 and have a lease period of 99 years. The figures below show that 1) resale condos are more expensive than new sale condos; and 2) the commencing year is negatively correlated with the price, implying that newly built condos are cheaper than older ones. Can you use data visualization to explain these observations.

	project	street	segment	commence	sale	price	area	level
0	PARC BOTANNIA	FERNVALE STREET	OCR	2016.0	New Sale	1672280.0	1281	16 to 20
1	PARC BOTANNIA	FERNVALE STREET	OCR	2016.0	New Sale	1622810.0	1453	01 to 05
2	PARC BOTANNIA	FERNVALE STREET	OCR	2016.0	New Sale	915680.0	667	06 to 10
...	...	...	...	...	...	...	...	...
316	SUNRISE GARDENS	SUNRISE AVENUE	OCR	1995.0	Resale	1088000.0	1464	01 to 05
317	PARC BOTANNIA	FERNVALE STREET	OCR	2016.0	New Sale	857000.0	667	06 to 10
318	THE GREENWICH	SELETAR ROAD	OCR	2009.0	Resale	635000.0	603	01 to 05

Guidelines of Data Visualization

- Interpretation of data visuals

Question 4: The file "condo_2018.csv" provides a dataset of condominium transactions in the year of 2018. All properties involved in these transactions are located in district 28 and have a lease period of 99 years. The figures below show that 1) resale condos are more expensive than new sale condos; and 2) the commence year is negatively correlated with the price, implying that newly built condos are cheaper than older ones. Can you use data visualization to explain these observations.

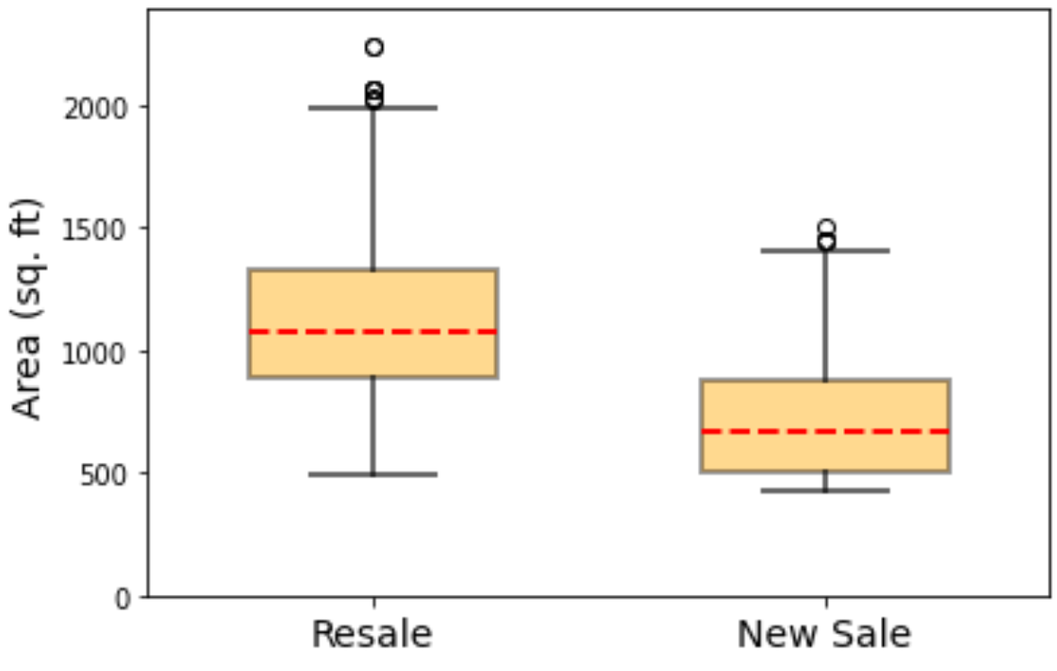


	commence	price
commence	1.000000	-0.129123
price	-0.129123	1.000000

Guidelines of Data Visualization

- Interpretation of data visuals

Question 4: The file "condo_2018.csv" provides a dataset of condominium transactions in the year of 2018. All properties involved in these transactions are located in district 28 and have a lease period of 99 years. The figures below show that 1) resale condos are more expensive than new sale condos; and 2) the commence year is negatively correlated with the price, implying that newly built condos are cheaper than older ones. Can you use data visualization to explain these observations.



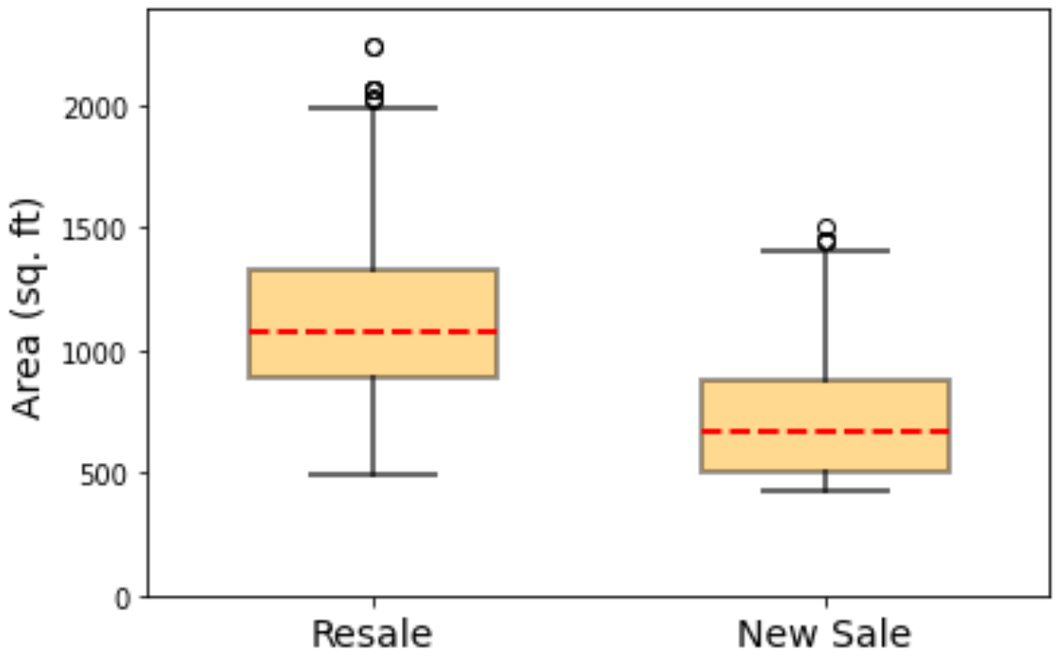
Area is a confounder!

	commence	price	area
commence	1.000000	-0.129123	-0.600679
price	-0.129123	1.000000	0.829969
area	-0.600679	0.829969	1.000000

Guidelines of Data Visualization

- Interpretation of data visuals

Question 4: The file "condo_2018.csv" provides a dataset of condominium transactions in the year of 2018. All properties involved in these transactions are located in district 28 and have a lease period of 99 years. The figures below show that 1) resale condos are more expensive than new sale condos; and 2) the commence year is negatively correlated with the price, implying that newly built condos are cheaper than older ones. Can you use data visualization to explain these observations.



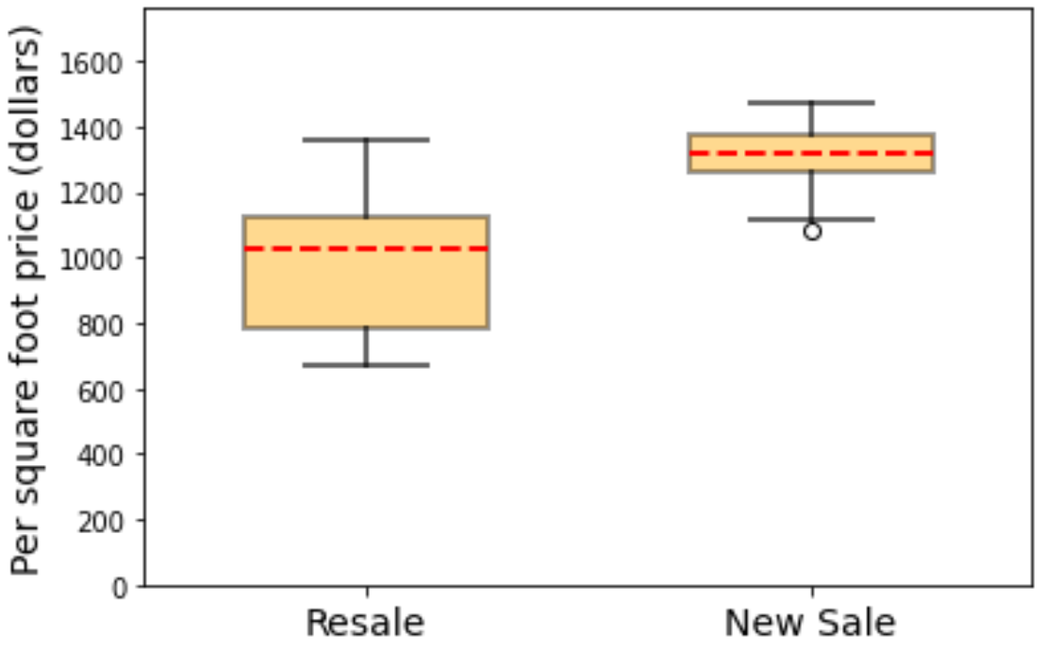
New condos are smaller on average, so they are cheaper.

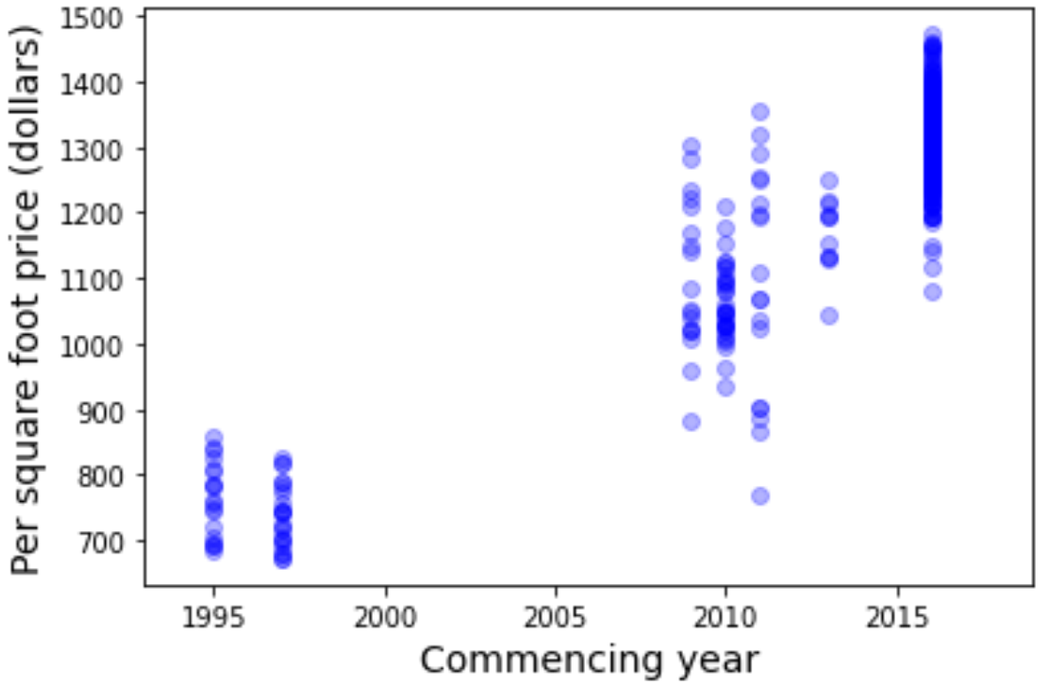
	commence	price	area
commence	1.000000	-0.129123	-0.600679
price	-0.129123	1.000000	0.829969
area	-0.600679	0.829969	1.000000

Guidelines of Data Visualization

- Interpretation of data visuals

Question 4: The file "condo_2018.csv" provides a dataset of condominium transactions in the year of 2018. All properties involved in these transactions are located in district 28 and have a lease period of 99 years. The figures below show that 1) resale condos are more expensive than new sale condos; and 2) the commence year is negatively correlated with the price, implying that newly built condos are cheaper than older ones. Can you use data visualization to explain these observations.





Guidelines of Data Visualization

- Interpretation of data visuals

Question 4: The file "condo_2018.csv" provides a dataset of condominium transactions in the year of 2018. All properties involved in these transactions are located in district 28 and have a lease period of 99 years. The figures below show that 1) resale condos are more expensive than new sale condos; and 2) the commence year is negatively correlated with the price, implying that newly built condos are cheaper than older ones. Can you use data visualization to explain these observations.