IMPORTING & MANAGING FINANCIAL DATA IN PYTHON

Aggregate your data by category

Summarize numeric data by category

- So far: Summarize individual variables
- Compute descriptive statistic like mean, quantiles
- Split data into groups, then summarize groups
- Examples:
  - Largest company by exchange
  - Median market capitalization per IPO year
  - Average market capitalization per sector

Group your data by sector

```
In [1]: nasdaq.info()

RangeIndex: 3167 entries, 0 to 3166
Data columns (total 7 columns):
Stock Symbol              3167 non-null object
Company Name              3167 non-null object
Last Sale                 3165 non-null float64
Market Capitalization     3167 non-null float64
IPO Year                  1386 non-null float64
Sector                    2767 non-null object
Industry                  2767 non-null object
dtypes: float64(3), object(4)
memory usage: 173.3+ KB
```

Group your data by sector (2)

```
In [1]: nasdaq['market_cap_m'] = nasdaq['Market Capitalization'].div(1e6)

In [2]: nasdaq = nasdaq.drop('Market Capitalization', axis=1) # Drop column

In [3]: nasdaq_by_sector = nasdaq.groupby('Sector') # Create groupby object
```

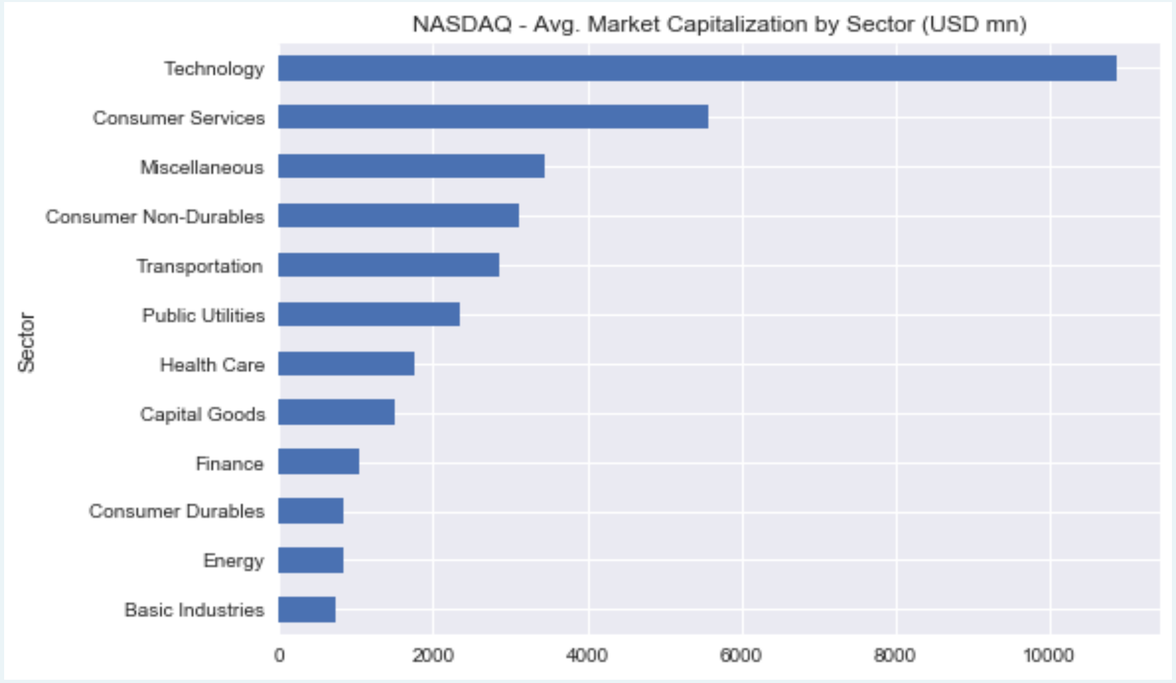
```
In [4]: for sector, data in nasdaq_by_sector:
   ...:     print(sector, data.market_cap_m.mean())

Basic Industries 724.899933858
Capital Goods 1511.23737278
Consumer Durables 839.802606627
Consumer Non-Durables 3104.05120552
…
Public Utilities 2357.86531507
Technology 10883.4342135
Transportation 2869.66000673
```

Keep it simple & skip the loop

```
In [4]: mcap_by_sector  = nasdaq_by_sector.market_cap_m.mean()

In [5]: mcap_by_sector
Out[5]:
Sector
Basic Industries             724.899934
Capital Goods               1511.237373
Consumer Durables            839.802607
Consumer Non-Durables       3104.051206
Consumer Services           5582.344175
Energy                       826.607608
Finance                     1044.090205
Health Care                 1758.709197
Miscellaneous               3445.655935
Public Utilities            2357.865315
Technology                 10883.434214
Transportation              2869.660007
Name: Market Capitalization, dtype: float64
```

Visualize category summaries

```
In [5]: title = 'NASDAQ = Avg. Market Cap by Sector'

In [8]: mcap_by_sector.plot(kind='barh', title=title)

In [9]: plt.xlabel('USD mn')
```



Aggregate summary for all numeric columns

```
In [4]: nasdaq_by_sector.mean()
Out[7]:
                       Last Sale     IPO Year  market_cap_m
Sector
Basic Industries       21.597679  2000.766667    724.899934
Capital Goods          26.188681  2001.324675   1511.237373
Consumer Durables      24.363391  2003.222222    839.802607
Consumer Non-Durables  25.749565  2000.609756   3104.051206
Consumer Services      34.917318  2004.104575   5582.344175
Energy                 15.496834  2008.034483    826.607608
Finance                29.644242  2010.321101   1044.090205
Health Care            19.462531  2009.240409   1758.709197
Miscellaneous          46.094369  2004.333333   3445.655935
Public Utilities       18.643705  2006.040000   2357.865315
Technology             31.100317  2002.653285  10883.434214
Transportation         28.096758  1997.809524   2869.660007
```

IMPORTING & MANAGING FINANCIAL DATA IN PYTHON

Let’s practice!

IMPORTING & MANAGING FINANCIAL DATA IN PYTHON

More ways to aggregate your data

Many ways to aggregate

- Last segment: Group by one variable & aggregate
- More detailed ways to summarize your data:
  - Group by two or more variables
  - Apply multiple aggregations
- Examples
  - Median market cap by sector and IPO year
  - Mean & standard deviation of stock price by year

Several aggregations by category

```
In [1]: nasdaq['market_cap_m'] = nasdaq['Market Capitalization'].div(1e6)

In [2]: by_sector = nasdaq.groupby('Sector')
```

```
In [3]: by_sector.market_cap_m.agg(['size', 'mean']).sort_values('size')
Out[3]:
                       size           mean
Sector
Transportation           52   2869.660007
Energy                   66    826.607608
Public Utilities         66   2357.865315
Basic Industries         78    724.899934
Consumer Durables        88    839.802607
…
Consumer Services       348   5582.344175
Technology              433  10883.434214
Finance                 627   1044.090205
Health Care             645   1758.709197
```

Several aggregations plus new labels

```
In [4]: by_sector.market_cap_m.agg({'#Obs': 'size', 'Average': 'mean'})
Out[4]:
                        #Obs        Average
Sector
Basic Industries         78     724.899934
Capital Goods           172    1511.237373
Consumer Durables        88     839.802607
Consumer Non-Durables   103    3104.051206
Consumer Services       348    5582.344175
Energy                   66     826.607608
Finance                 627    1044.090205
Health Care             645    1758.709197
Miscellaneous            89    3445.655935
Public Utilities         66    2357.865315
Technology              433   10883.434214
Transportation           52    2869.660007
```

Different statistics by column

```
In [5]: by_sector.agg({'market_cap_m': 'size', 'IPO Year':'median'})
Out[5]:
                       market_cap_m  IPO Year
Sector
Basic Industries                 78    1972.0
Capital Goods                   172    1972.0
Consumer Durables                88    1983.0
Consumer Non-Durables           103    1972.0
Consumer Services               348    1981.0
Energy                           66    1983.0
Finance                         627    1981.0
Health Care                     645    1981.0
Miscellaneous                    89    1987.0
Public Utilities                 66    1981.0
Technology                      433    1972.0
Transportation                   52    1986.0
```

Aggregate by two categories

```
In [7]: by_sector_year = nasdaq.groupby(['Sector', 'IPO Year'])

In [8]: by_sector_year.market_cap_m.mean()
Out[8]:
Sector              IPO Year
Basic Industries    1972.0       877.240005
                    1973.0      1445.697371
                    1986.0      1396.817381
                    1988.0        24.847526
                    …
Transportation      1986.0      1176.179710
                    1991.0      6646.778622
                    1992.0        56.074572
                    1993.0      3474.796885
                    …
                    2009.0       552.445919
                    2011.0      3711.638317
                    2013.0       125.740421
```

Select from MultiIndex()

```
In [9]: mcap_sector_year = by_sector_year.market_cap_m.mean()

In [10]: mcap_sect_year.loc['Basic Industries']
Out[10]:
IPO Year
1972.0     877.240005
1973.0    1445.697371
1986.0    1396.817381
1988.0      24.847526
1990.0     434.808483
1991.0       9.338401
…
2011.0      35.498729
2012.0     381.796074
2013.0      22.661533
2015.0     260.075564
2016.0      81.288336
Name: market_cap_m, dtype: float64
```

`.loc[col]`: select multiple row labels from first level

Select from MultiIndex() (2)

```
In [11]: mcap_sect_year.loc[['Basic Industries', 'Transportation']]
Out[11]:
Sector              IPO Year
Basic Industries    1972.0       877.240005
                    1973.0      1445.697371
                    1986.0      1396.817381
                    1988.0        24.847526
                    …
Transportation      1986.0      1176.179710
                    1991.0      6646.778622
                    1992.0        56.074572
                    1993.0      3474.796885
                    …
                    2009.0       552.445919
                    2011.0      3711.638317
                    2013.0       125.740421
```

IMPORTING & MANAGING FINANCIAL DATA IN PYTHON

Let’s practice!

IMPORTING & MANAGING FINANCIAL DATA IN PYTHON

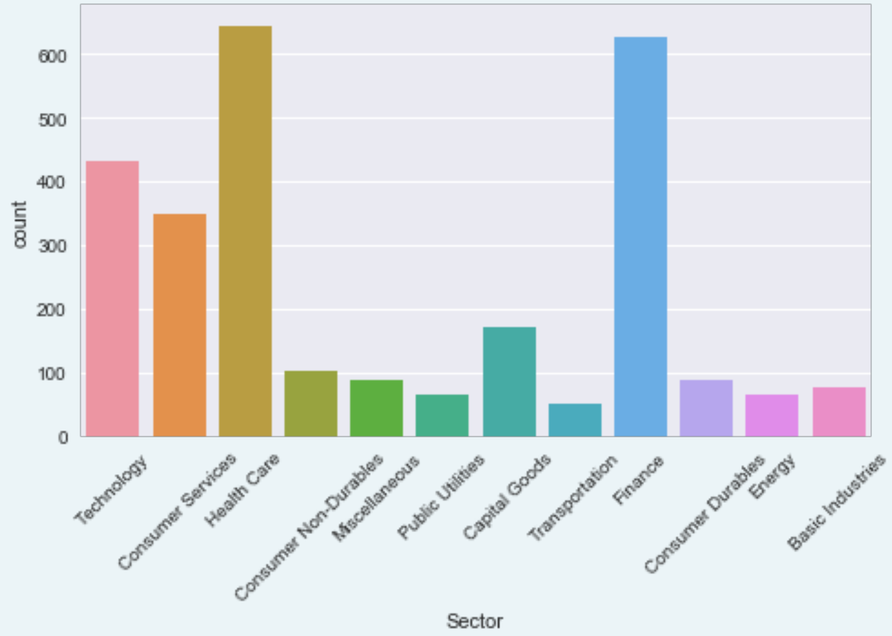
Summary statistics by category with seaborn

Categorical plots with `seaborn`

- Specialized ways to plot combinations of categorical and numerical variables
- Visualize estimates of summary statistics per category
- Understand how categories impact numerical variables
- Compare using key metrics of distributional characteristics
- Example: Mean Market Cap per Sector or IPO Year with indication of dispersion

The basics: countplot

```
In [1]: nasdaq = pd.read_excel('listings.xlsx', sheetname='nasdaq',
                               na_values='n/a')

In [2]: sns.countplot(x='Sector', data=nasdaq)

In [3]: plt.xticks(rotation=45)
```



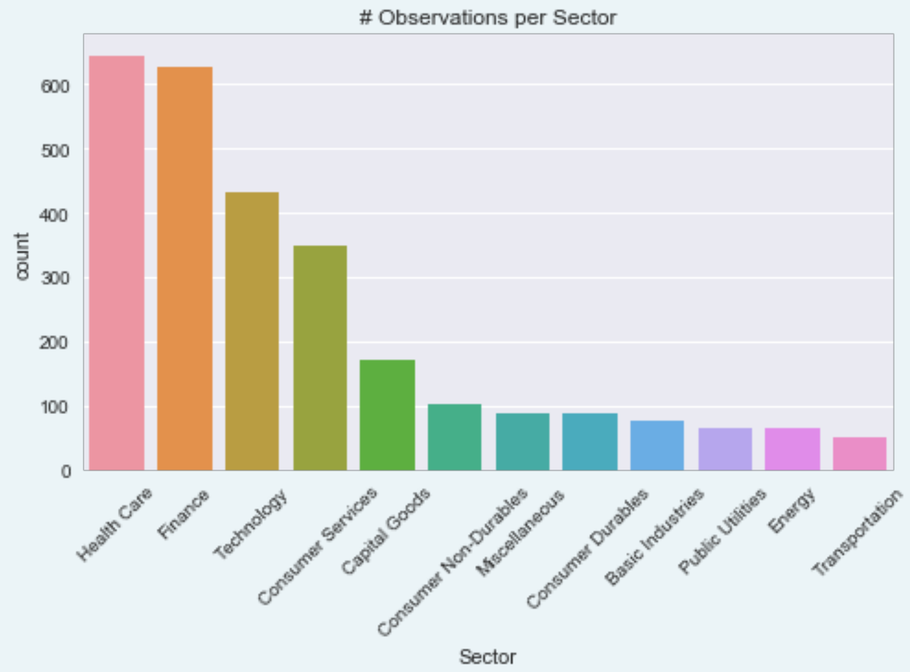
countplot, sorted

```
In [4]: sector_size = nasdaq.groupby('Sector').size()

In [5]: order = sector_size.sort_values(ascending=False)
```

```
In [6]: order.head()
Out[6]:
Sector
Health Care           645
Finance               627
Technology            433
Consumer Services     348
Capital Goods         172
dtype: int64

In [7]: order = order.index.tolist()
['Health Care', ‘Finance’,…, 'Energy', 'Transportation']
```

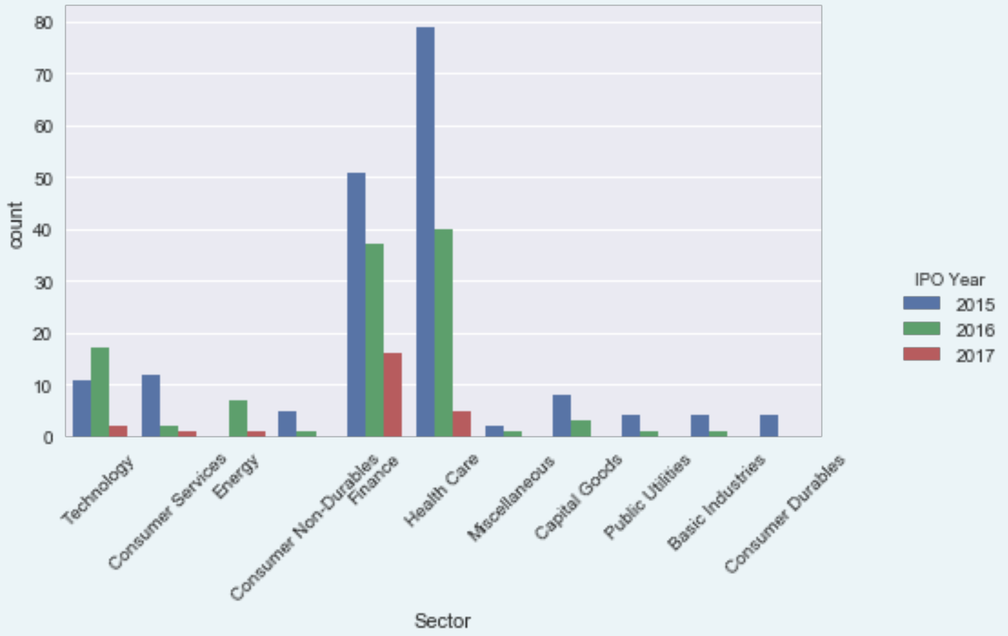
countplot, sorted (2)

```
In [8]: sns.countplot(x='Sector', data=nasdaq, order=order)

In [9]: plt.xticks(rotation=45)

In [10]: plt.title('# Observations per Sector’)
```



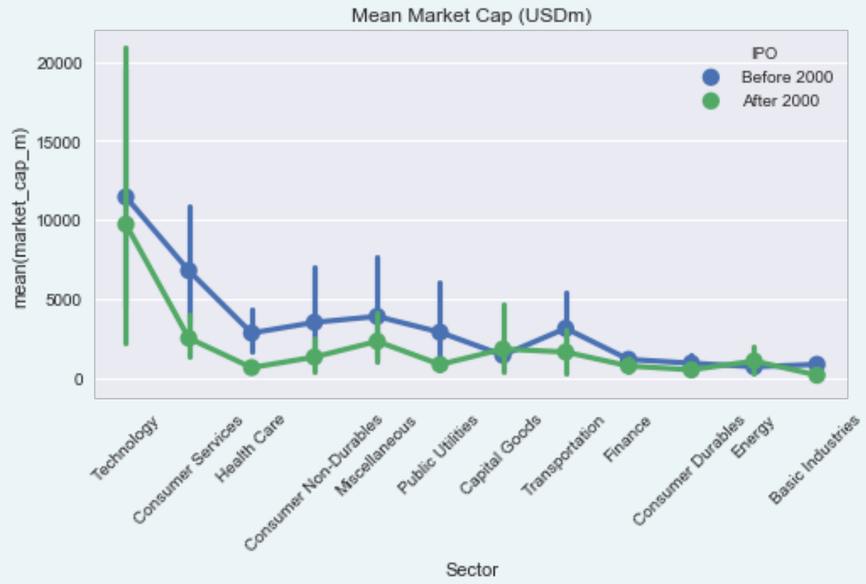
countplot, multiple categories

```
In [11]: recent_ipos = nasdaq[nasdaq['IPO Year'] > 2014]

In [12]: recent_ipos['IPO Year'] = recent_ipos['IPO Year'].astype(int)

In [13]: sns.countplot(x='Sector', hue='IPO Year', data=recent_ipos)
```



Compare stats with PointPlot

```
In [11]: nasdaq['IPO'] = nasdaq['IPO Year’].apply(lambda x:
                          'After 2000' if x > 2000 else 'Before 2000')

In [12]: sns.pointplot(x='Sector', y='market_cap_m', hue='IPO',
                                  data=nasdaq)

In [13]: plt.xticks(rotation=45); plt.title('Mean Market Cap')
```



IMPORTING & MANAGING FINANCIAL DATA IN PYTHON

Let’s practice!

IMPORTING & MANAGING FINANCIAL DATA IN PYTHON

Distributions by category with seaborn

Distributions by category

- Last segment: Summary statistics
- Number of observations, mean per category
- Now: Visualize distribution of a variable by levels of a categorical variable to facilitate comparison
- Example: Distribution of Market Cap by Sector or IPO Year
- More detail than summary stats

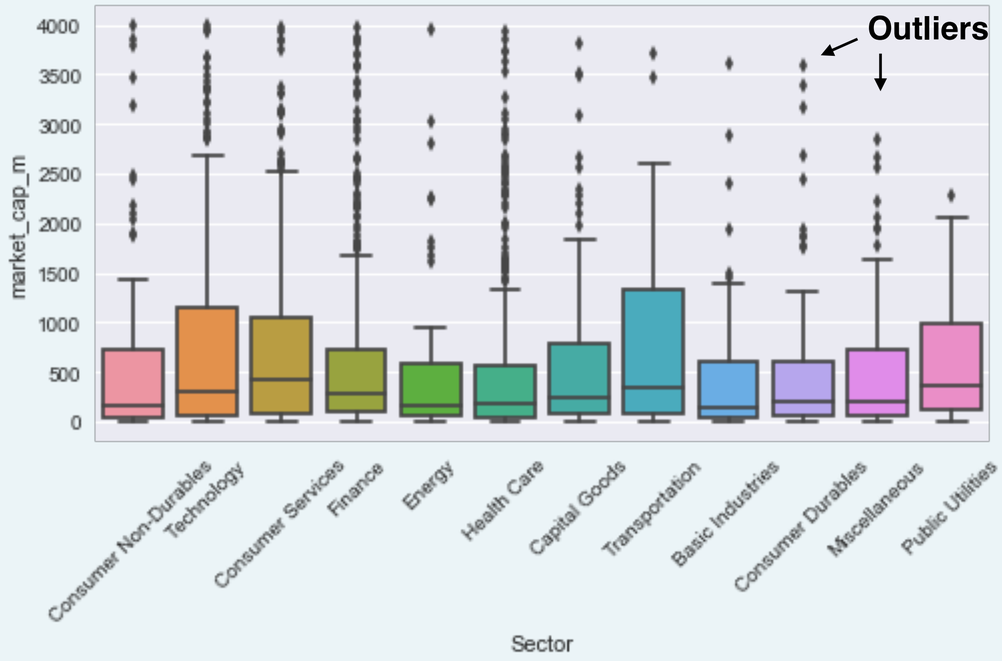
Clean data: Removing outliers

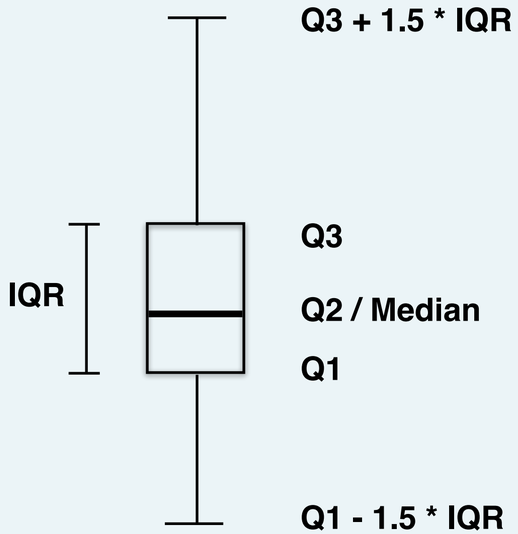
```
In [1]: nasdaq = pd.read_excel('listings.xlsx', sheetname='nasdaq',
                               na_values='n/a')

In [2]: nasdaq['market_cap_m'] = nasdaq['Market Capitalization'].div(1e6)
```

```
In [3]: nasdaq = nasdaq[nasdaq.market_cap_m > 0] # Active companies only

In [4]: outliers = nasdaq.market_cap_m.quantile(.9) # Outlier threshold

In [5]: nasdaq = nasdaq[nasdaq.market_cap_m < outliers] # Remove outliers
```

Boxplot: Quartiles & outliers

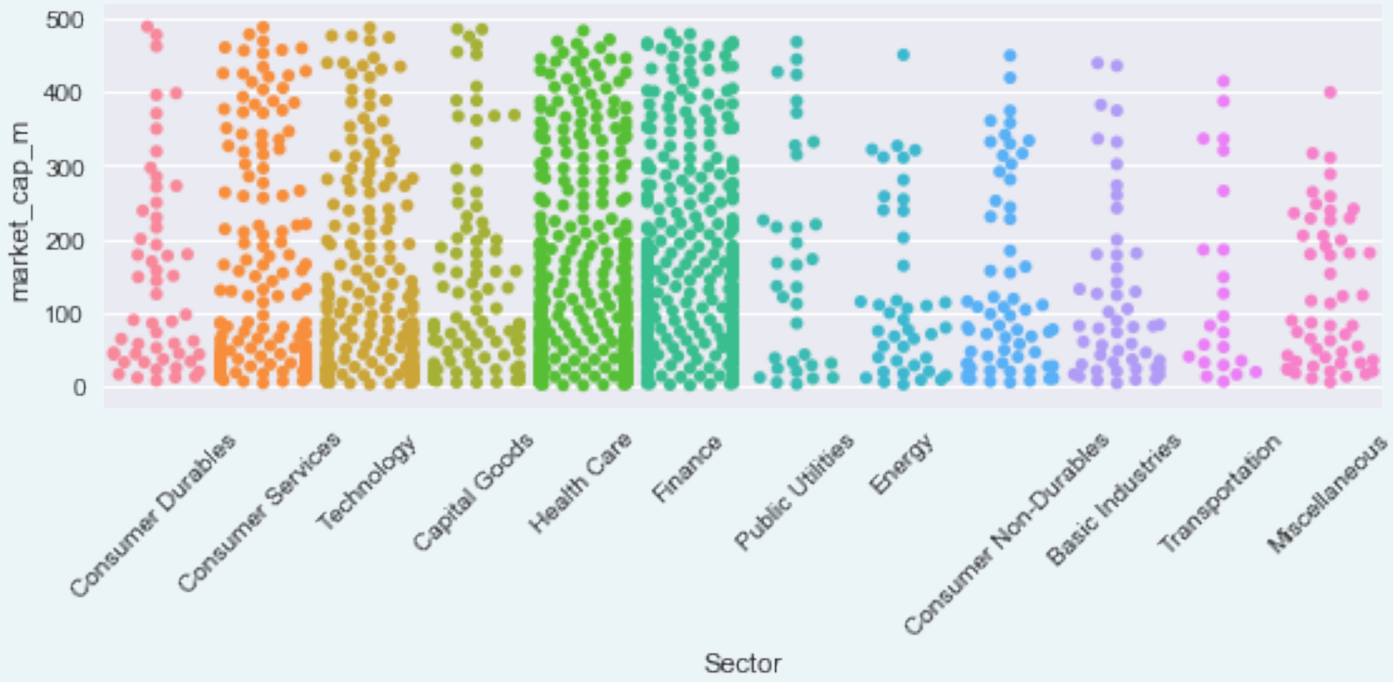
```
In [6]: import seaborn as sns

In [7]: sns.boxplot(x='Sector', y='market_cap_m', data=nasdaq)

In [8]: plt.xticks(rotation=75);
```





A variation: SwarmPlot

```
In [7]: sns.swarmplot(x='Sector', y='market_cap_m', data=nasdaq)

In [8]: plt.xticks(rotation=75);
```



All observations, no overlap

IMPORTING & MANAGING FINANCIAL DATA IN PYTHON

Let’s practice!

IMPORTING & MANAGING FINANCIAL DATA IN PYTHON

Congratulations!