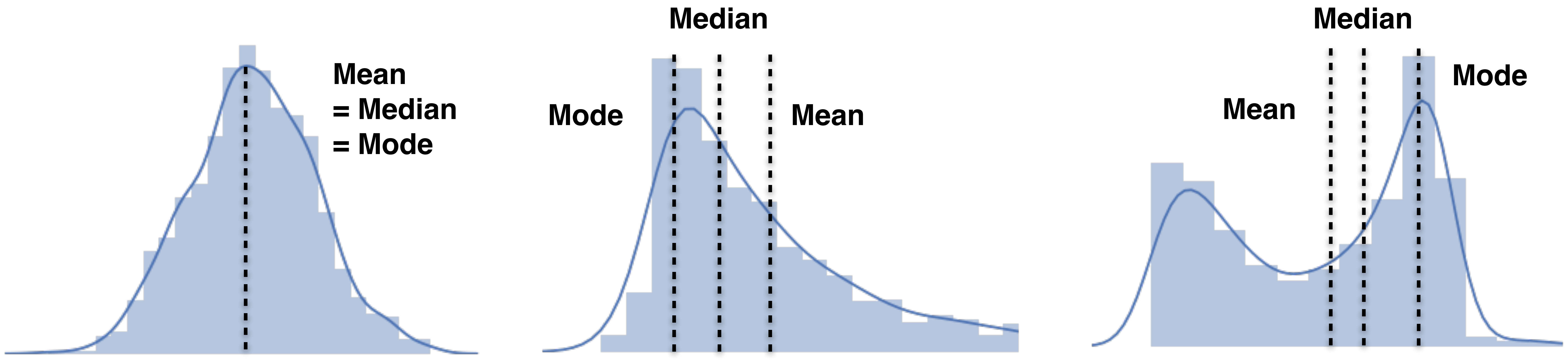
Summarize your data with descriptive stats

Be on top of your data

- Goal: Capture key quantitative characteristics
- Important angles to look at:
    - Central tendency: Which values are "typical"?
    - Dispersion: Are there outliers?
    - Overall distribution of individual variables

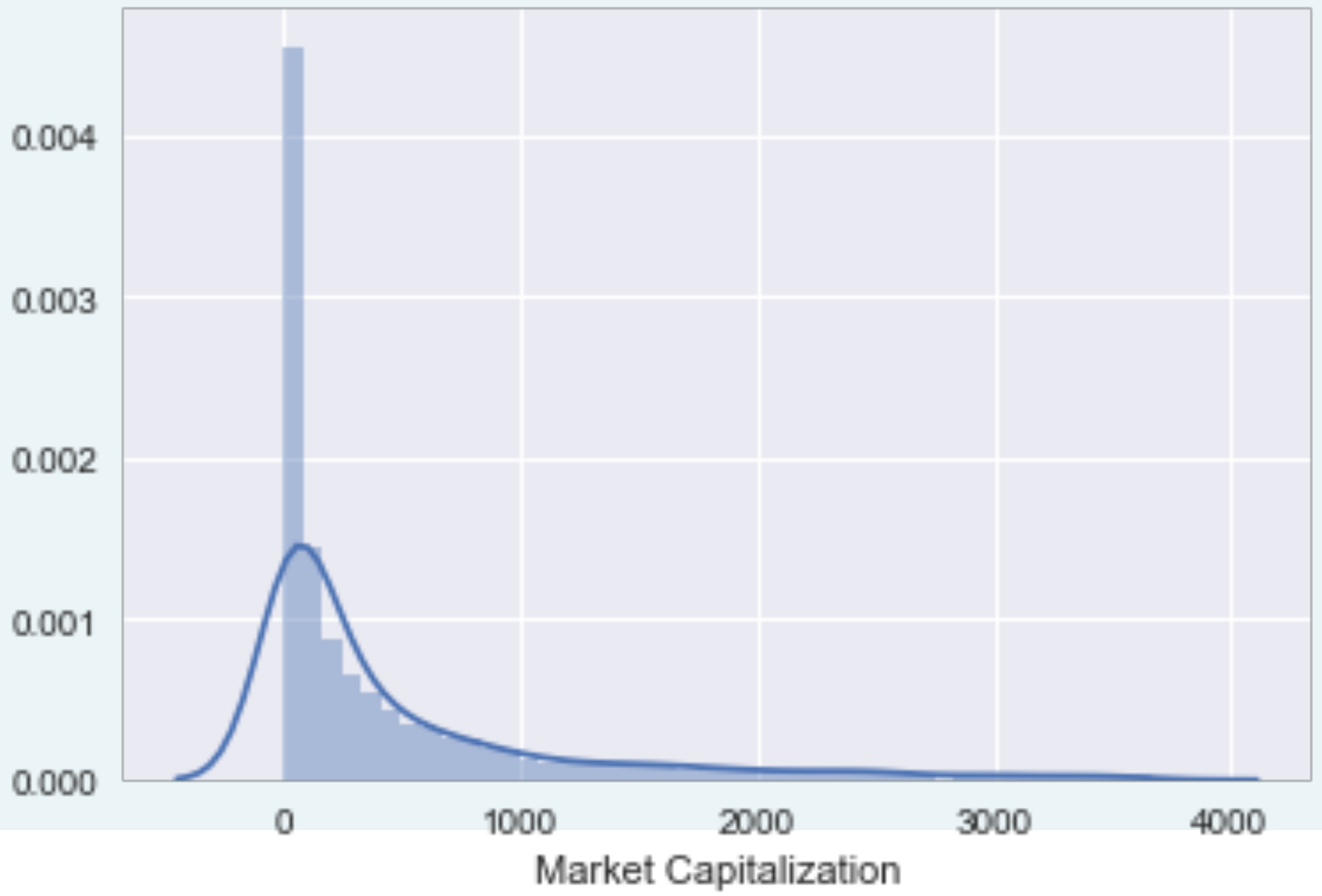
Central tendency

- **Mean** (average): $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$
- **Median**: 50% of values smaller/larger
- **Mode**: most frequent value



Calculate summary statistics

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

In [2]: market_cap = nasdaq['Market Capitalization'].div(10**6)

In [3]: market_cap.mean()
Out[3]: 3180.7126214953805

In [4]: market_cap.median()
Out[4]: 225.9684285

In [5]: market_cap.mode()
Out[5]:
0    0.0
dtype: float64
```



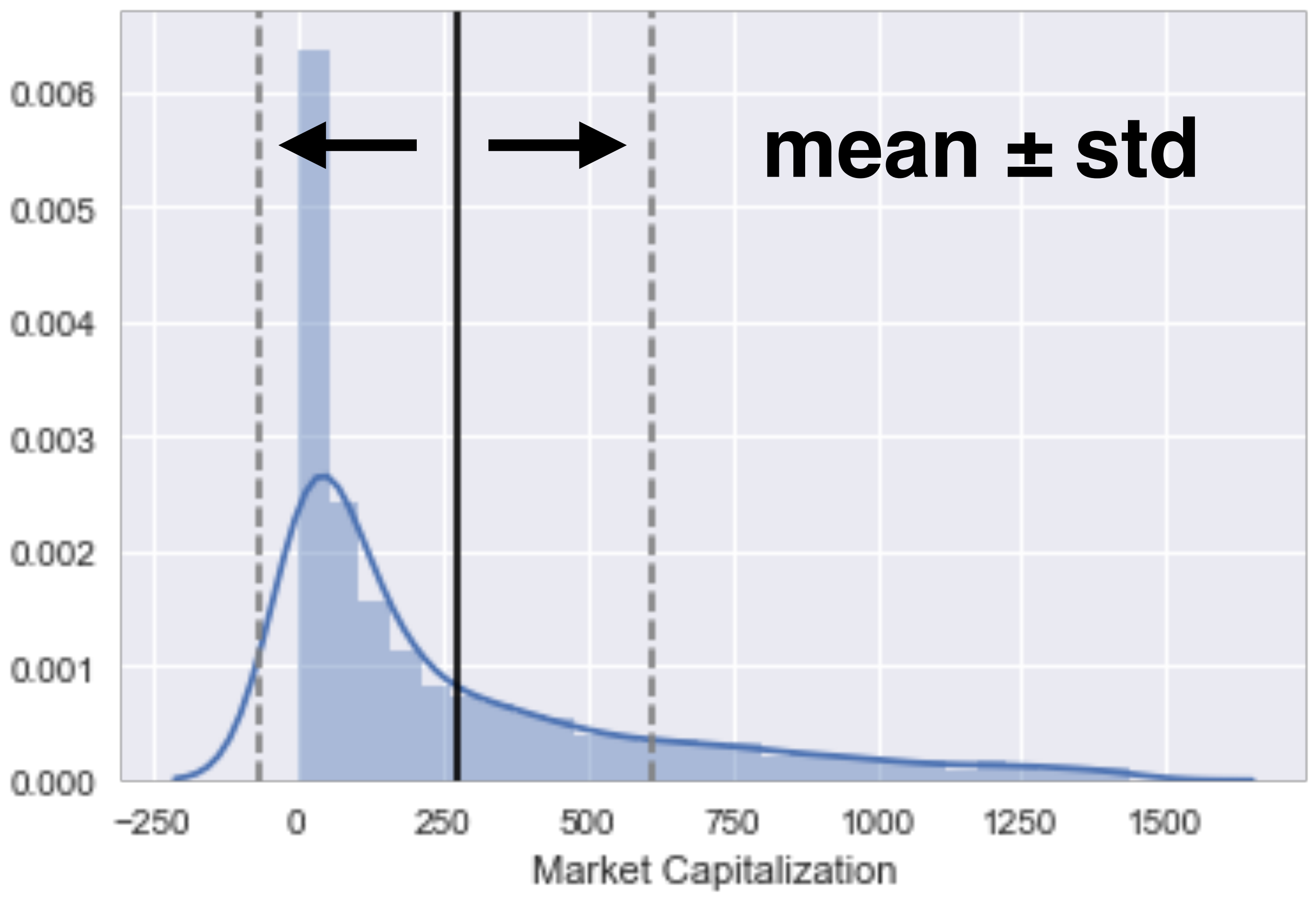
Dispersion

- **Variance:** Sum all squared differences from mean and divide by n-1

$$\text{var} = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

- **Standard deviation:** Square root of variance

$$s = \sqrt{\text{var}}$$



Calculate variance & standard deviation

```
In [6]: market_cap.var()
Out[6]: 648773812.8182

In [7]: np.sqrt(variance)
Out[7]: 25471.0387

In [8]: market_cap.std()
Out[8]: 25471.0387
```

Let’s practice!

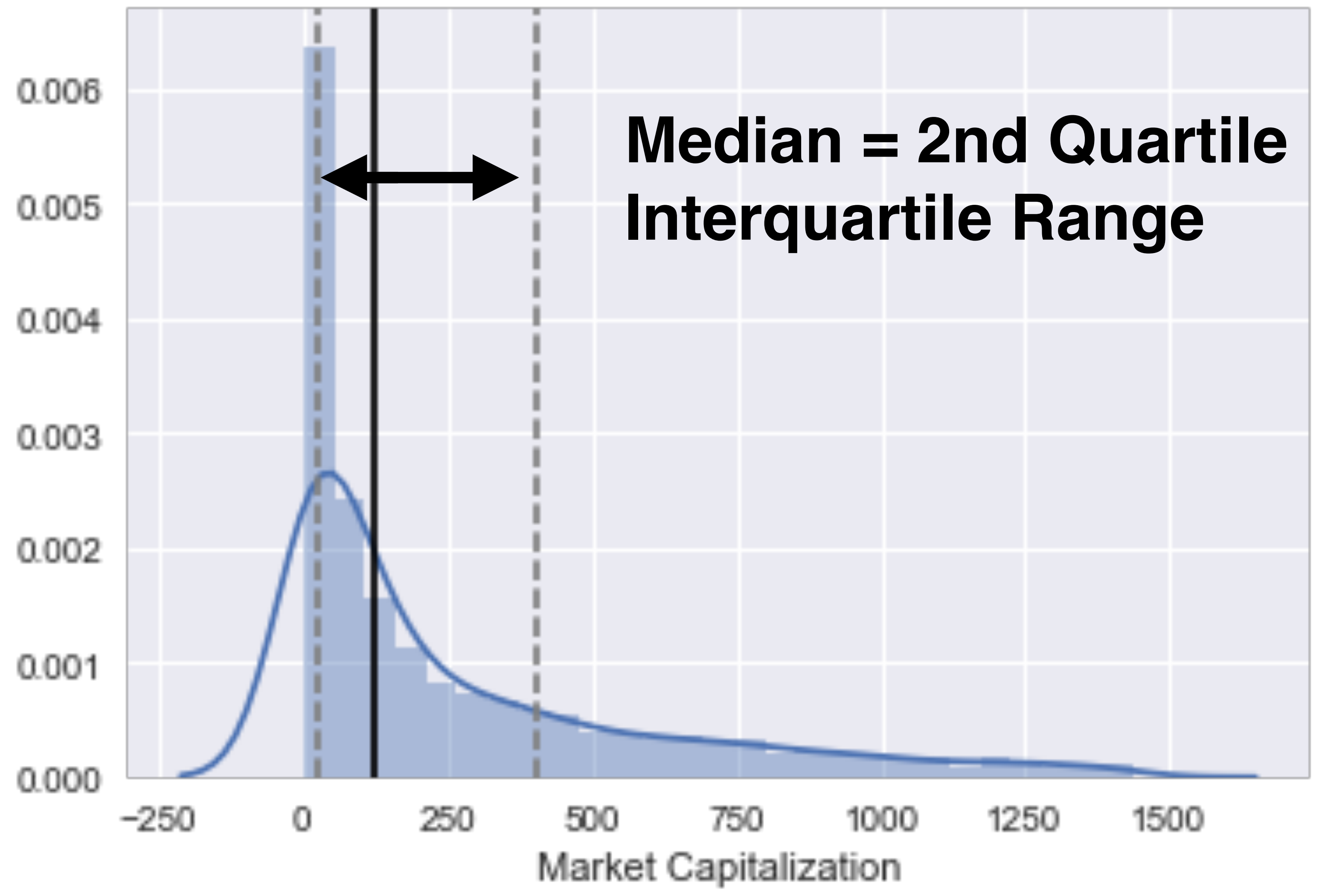
Describe the distribution of your data with quantiles

Describe data distributions

- First glance: Central tendency and standard deviation
- How to get a more granular view of the distribution?
- Calculate and plot quantiles

More on dispersion: Quantiles

- **Quantiles**: Groups with equal share of observations
  - Quartiles: 4 groups, 25% of data each
  - Deciles: 10 groups, 10% of data each
  - Interquartile range: 3rd quartile - 1st quartile



Quantiles with pandas

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

In [2]: market_cap = nasdaq['Market Capitalization’].div(10**6)

In [3]: median = market_cap.quantile(.5)

In [4]: median == market_cap.median()
Out[4]: True

In [5]: quantiles = market_cap.quantile([.25, .75])
0.25      43.375930
0.75     969.905207

In [6]: quantiles[.75] - quantiles[.25] # Interquartile Range
Out[6]: 926.5292771575
```

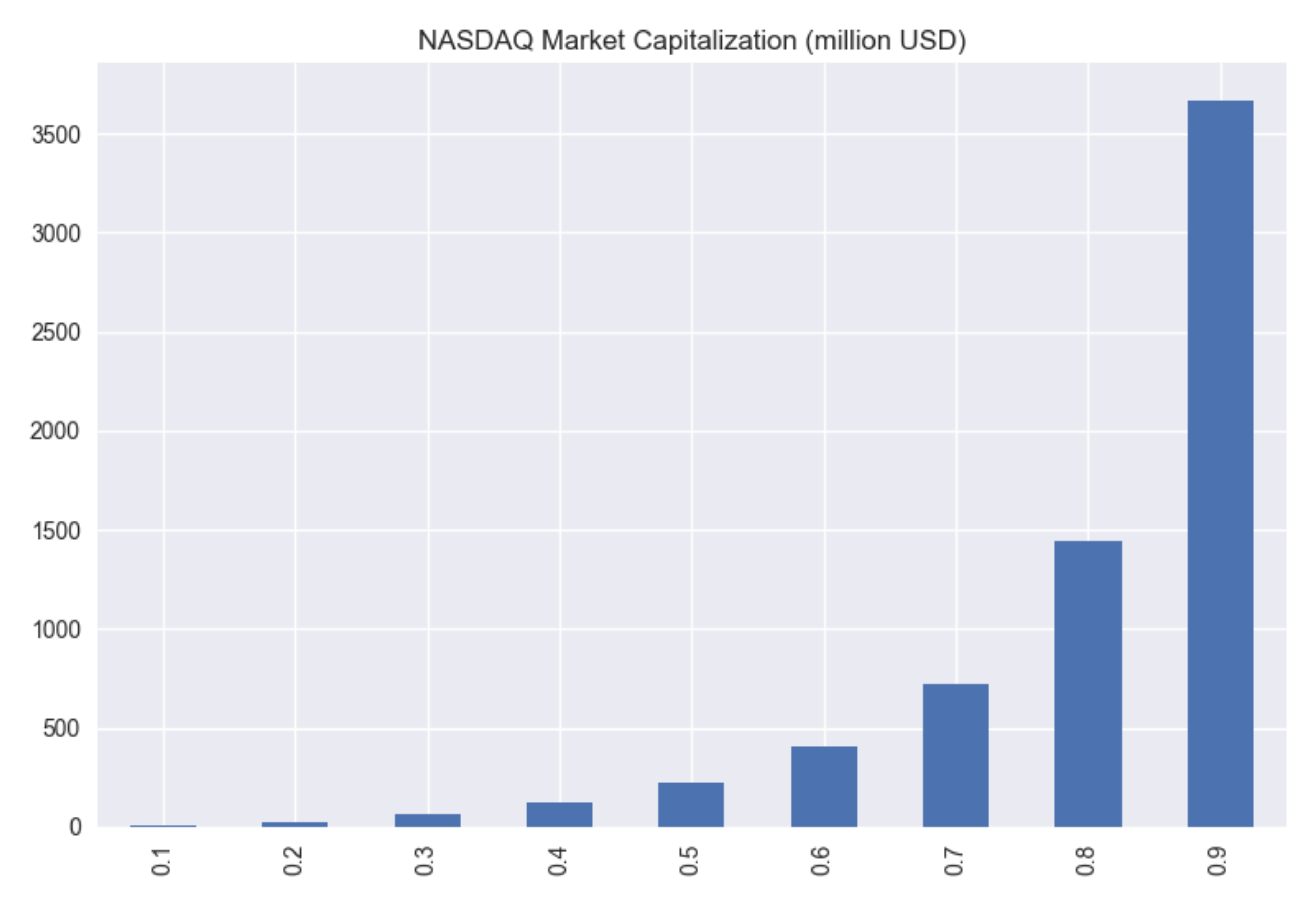
Selecting from pd.Series()

Quantiles with pandas & numpy

```
In [1]: deciles = np.arange(start=.1, stop=.91, step=.1)

In [2]: deciles
Out[2]: array([ 0.1,  0.2,  0.3,  0.4,  ...,  0.7,  0.8,  0.9])

In [3]: market_cap.quantile(deciles)
Out[3]:
0.1       4.884565
0.2      26.993382
0.3      65.714547
0.4     124.320644
0.5     225.968428
0.6     402.469678
0.7     723.163197
0.8    1441.071134
0.9    3671.499558
Name: Market Capitalization, dtype: float64
```

Visualize quantiles with bar chart

```
In [3]: title = 'NASDAQ Market Capitalization (million USD)'

In [4]: market_cap.quantile(deciles).plot(kind='bar', title=title)

In [5]: plt.tight_layout(); plt.show();
```



All statistics in one go

```
In [3]: market_cap.describe()
count        3167.000000
mean         3180.712621
std         25471.038707
min             0.000000
25%            43.375930
50%           225.968428
75%           969.905207
max        740024.467000
Name: Market Capitalization
```

1st Quartile

Median

3rd Quartile

All statistics in one go (2)

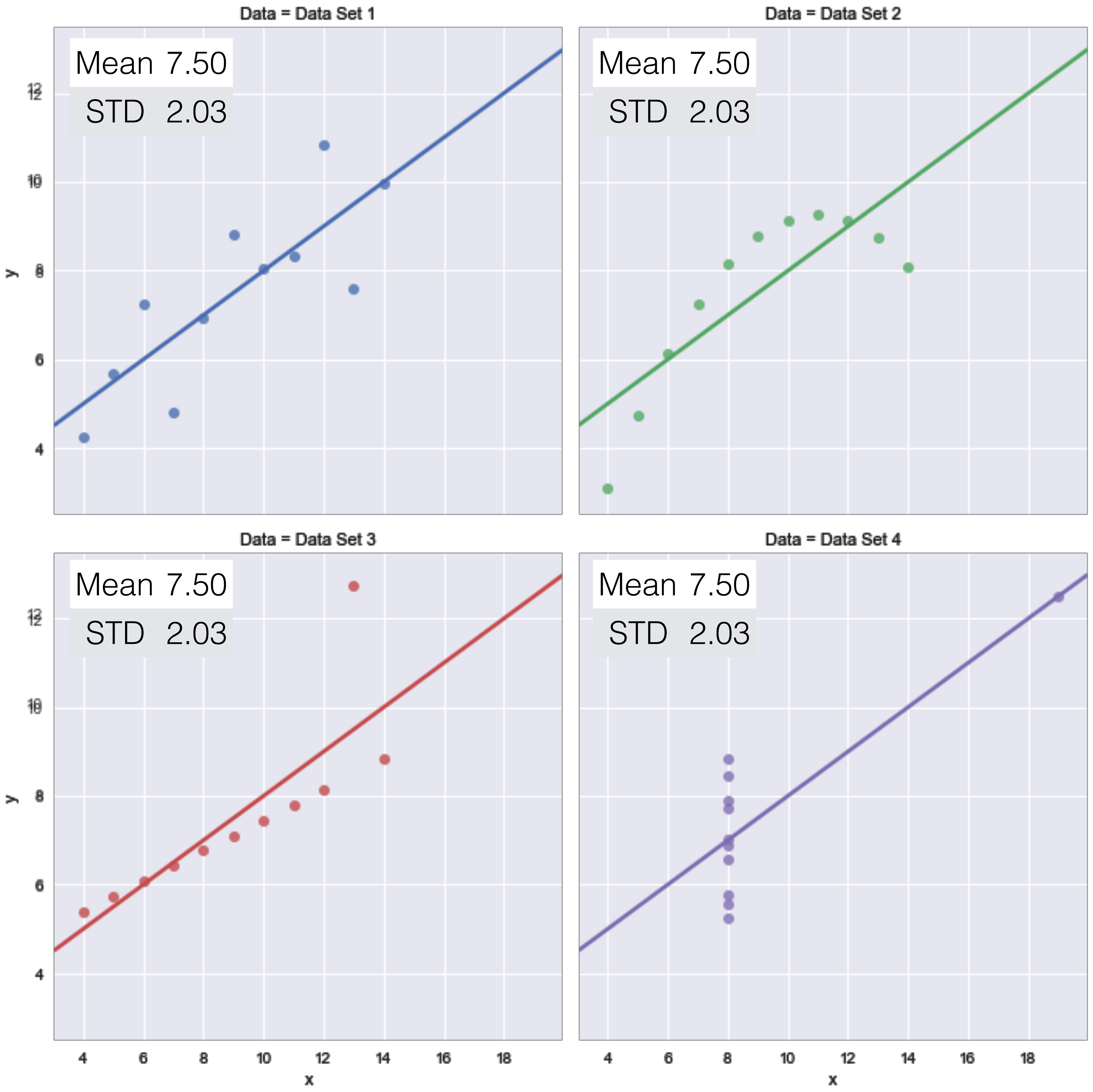
```
In [3]: market_cap.describe(percentiles=np.arange(.1, .91, .1))
Out[7]:
count        3167.000000
mean         3180.712621
std         25471.038707
min             0.000000
10%             4.884565
20%            26.993382
30%            65.714547
40%           124.320644
50%           225.968428
60%           402.469678
70%           723.163197
80%          1441.071134
90%          3671.499558
max        740024.467000
Name: Market Capitalization
```

np.arange(start, stop, step):
like range() but with decimal values & steps

Let's practice!

Visualize the distribution of your data

Always look at your data!



- Identical metrics can represent very different data

Introducing `seaborn` plots

- Many attractive and insightful statistical plots
- Based on `matplotlib`
- Swiss Army knife: `seaborn.distplot()`
  - Histogram
  - Kernel Density Estimation (KDE)
  - Rugplot

10 year treasury: Trend & distribution

```
In [1]: ty10 = web.DataReader('DGS10', 'fred', date(1962, 1, 1))

In [2]: ty10.info()
DatetimeIndex: 14443 entries, 1962-01-02 to 2017-05-11
Data columns (total 1 columns):
DGS10    13825 non-null float64

In [3]: ty10.describe()
Out[3]:
              DGS10
count  13825.000000
mean       6.291073
std        2.851161
min        1.370000
25%        4.190000
50%        6.040000
75%        7.850000
max       15.840000
```

Missing values:

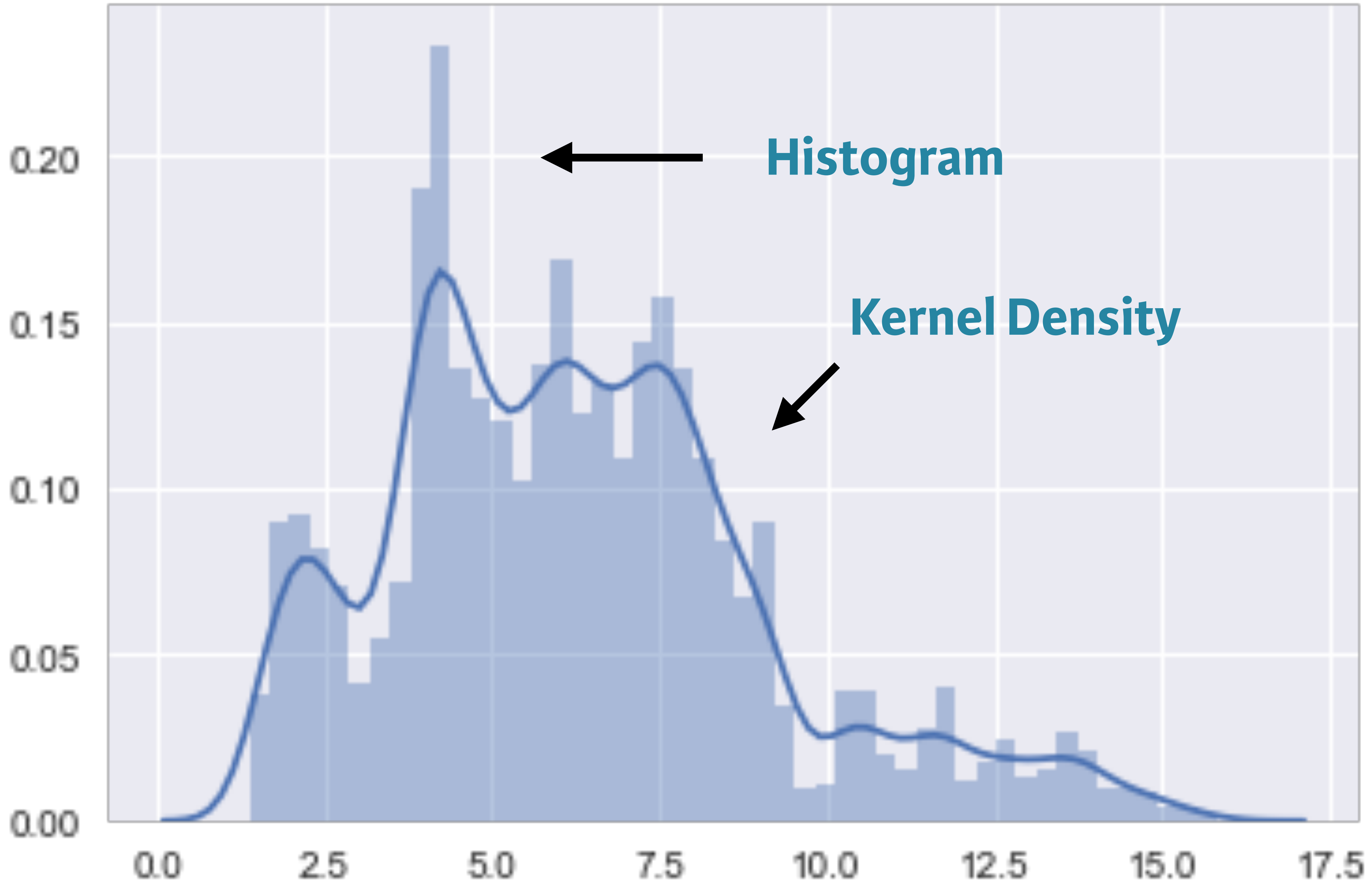
- .dropna()
- .fillna()

10 year treasury: Time series trend

```
In [4]: ty10.dropna(inplace=True) # Avoid creation of copy

In [5]: ty10.plot(title='10-year Treasury'); plt.tight_layout()
```

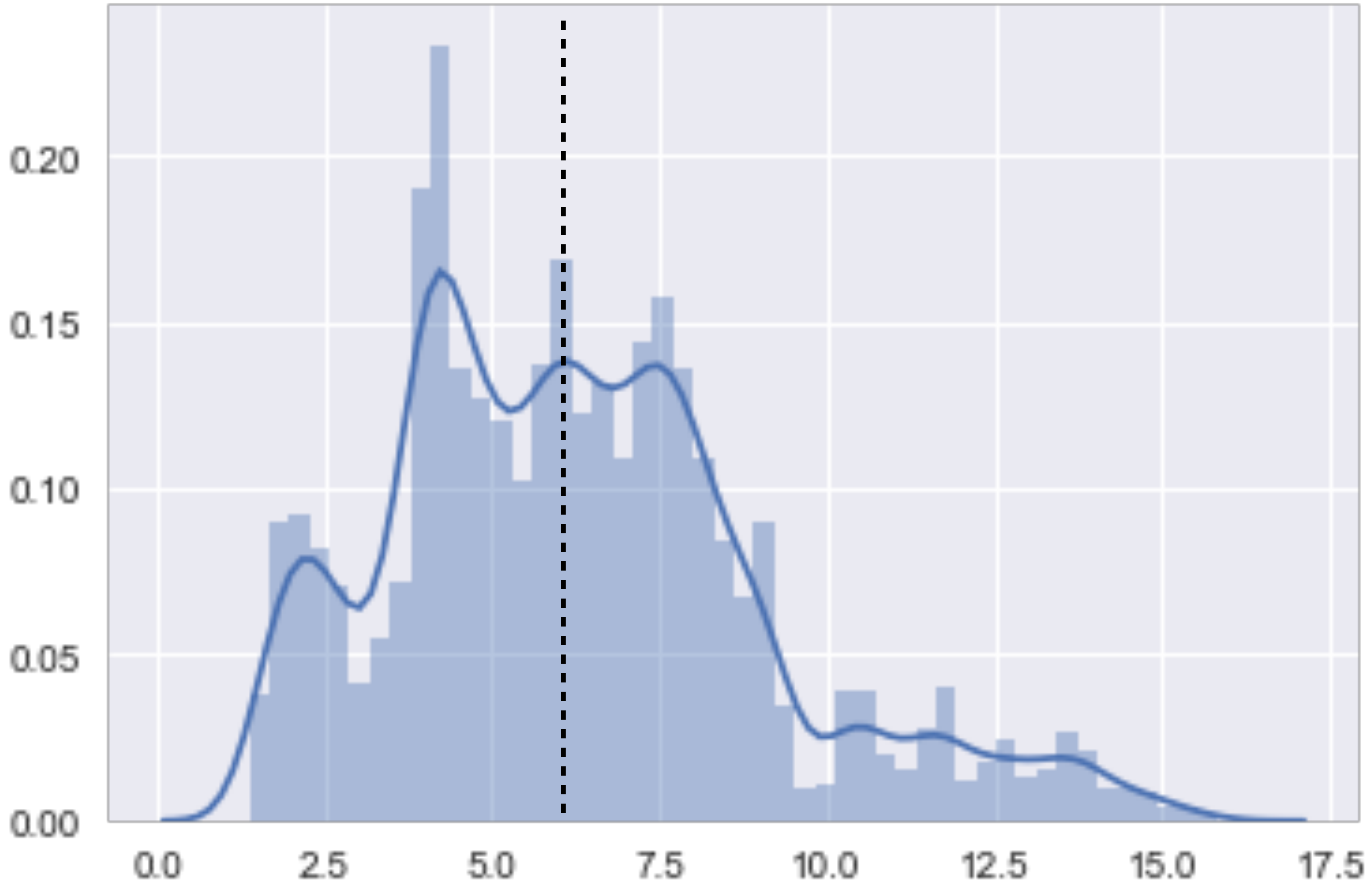
10 year treasury: Historical distribution

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

In [7]: sns.distplot(ty10);
```



10 year treasury: Trend & distribution (2)

```
In [6]: ax = sns.distplot(ty10)

In [7]: ax.axvline(ty10['DGS10'].median(), color='black', ls='--')
```



Let's practice!

Summarize categorical variables

From categorical to quantitative variables

- So far, we have analyzed quantitative variables
- Categorical variables require a different approach
- Concepts like average don't make much sense
- Instead, we'll rely on their frequency distribution

Categorical listing information

```
In [2]: amex = pd.read_excel('listings.xlsx', sheetname='amex',
                              na_values=['n/a'])

In [3]: amex.info()

 RangeIndex: 360 entries, 0 to 359
 Data columns (total 8 columns):
 Stock Symbol              360 non-null object
 Company Name              360 non-null object
 Last Sale                 346 non-null float64
 Market Capitalization     360 non-null float64
 IPO Year                  105 non-null float64
 Sector                    238 non-null object
 Industry                  238 non-null object
 dtypes: datetime64[ns](1) float64(3), object(4)
```

Columns of dtype 'object' are categorical

Categorical listing information (2)

```
In [2]: amex = amex.Sector.nunique()
Out[2]: 12

In [3]: amex.apply(lambda x: x.nunique())
Out[3]:
Stock Symbol              360
Company Name              326
Last Sale                 323
Market Capitalization     317
IPO Year                   24
Sector                     12
Industry                   68
```

apply(): call function on each column

lambda: "anonymous function", receives each column as argument x

How many observations per sector?

```
In [2]: amex.Sector.value_counts()

Out[4]:
Health Care              49  # Mode
Basic Industries         44
Energy                   28
Consumer Services        27
Capital Goods            24
Technology               20
Consumer Non-Durables    13
Finance                  12
Public Utilities         11
Miscellaneous             5
Consumer Durables         4
Transportation            1
Name: Sector, dtype: int64
```

.value_counts(): count of each unique value

How many IPOs per year?

```
In [2]: amex['IPO Year'].value_counts()
Out[6]:
2002.0     19 # Mode
2015.0     11
1999.0      9
1993.0      7
2014.0      6
2013.0      5
2017.0      5
2003.0      5
2004.0      5
1992.0      4
2016.0      3
…
2009.0      1
1990.0      1
1991.0      1
Name: IPO Year, dtype: int64
```

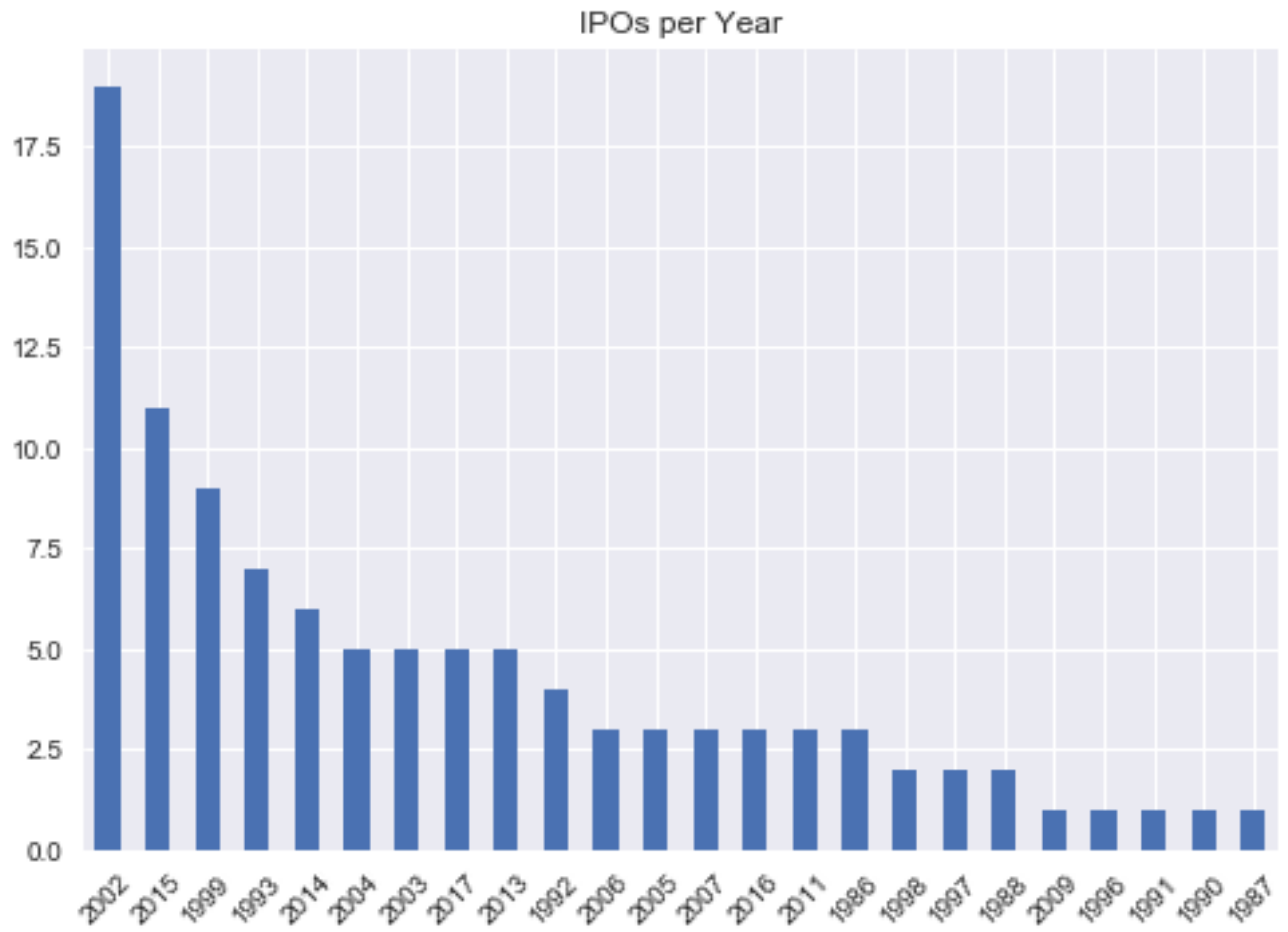
Years represented as float because of missing values

Convert IPO Year to int

```
In [7]: ipo_by_yr = amex['IPO Year'].dropna().astype(int).value_counts()

In [8]: ipo_by_yr
Out[8]:
2002    19
2015    11
1999     9
1993     7
2014     6
2004     5
2003     5
2017     5
2013     5
1992     4
2016     3
…
1987     1
Name: IPO Year, dtype: int64
```

Convert `IPO Year` to `int` (2)

```
In [9]: ipo_by_yr.plot(kind='bar', title='IPOs per Year')

In [10]:plt.xticks(rotation=45)
```



Let's practice!