

# Summarize your data with descriptive stats

IMPORTING AND MANAGING FINANCIAL DATA IN PYTHON



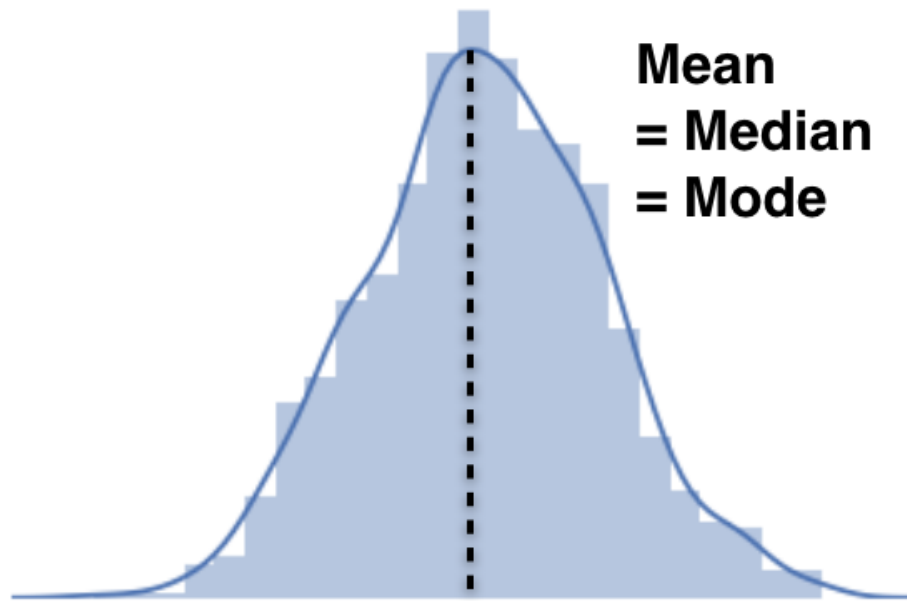
**Stefan Jansen**  
Instructor

# 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

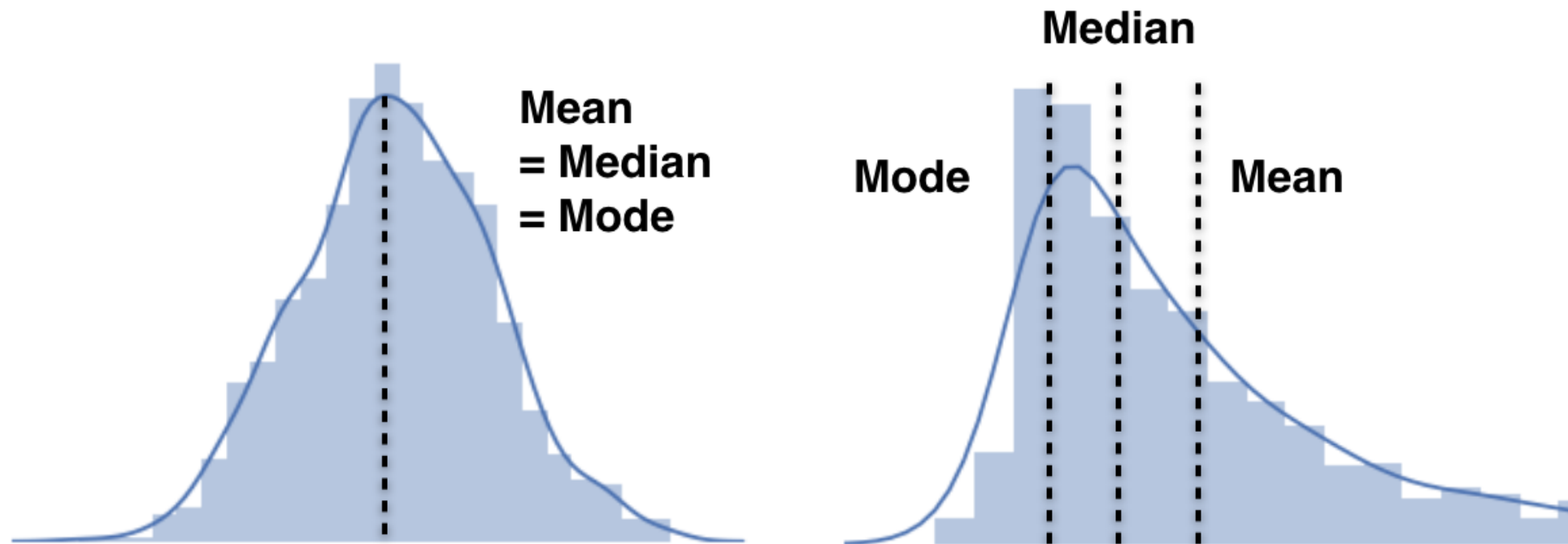
# Central tendency

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



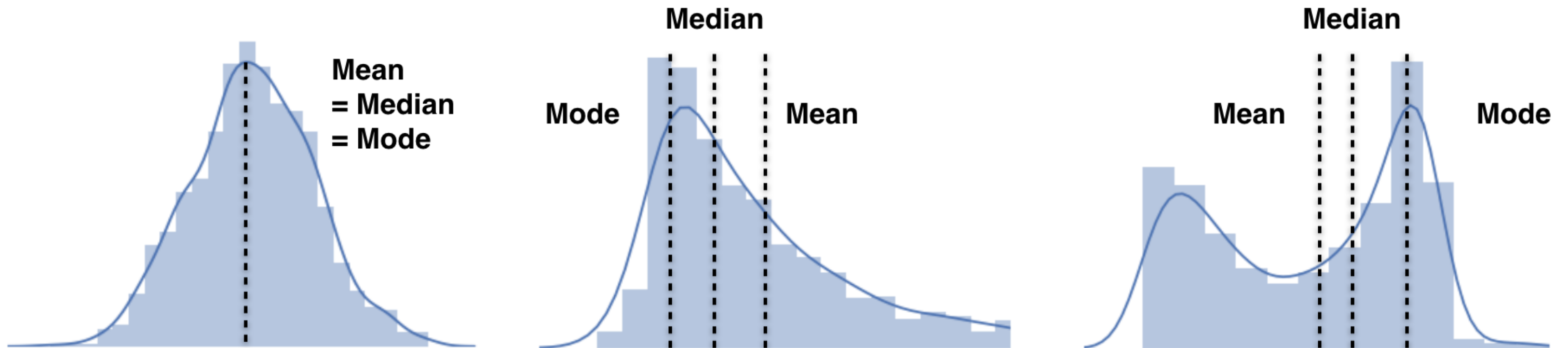
# Central tendency

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



# 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

```
nasdaq = pd.read_excel('listings.xlsx', sheetname='nasdaq', na_values='n/a')  
market_cap = nasdaq['Market Capitalization'].div(10**6)
```

```
market_cap.mean()
```

```
3180.7126214953805
```

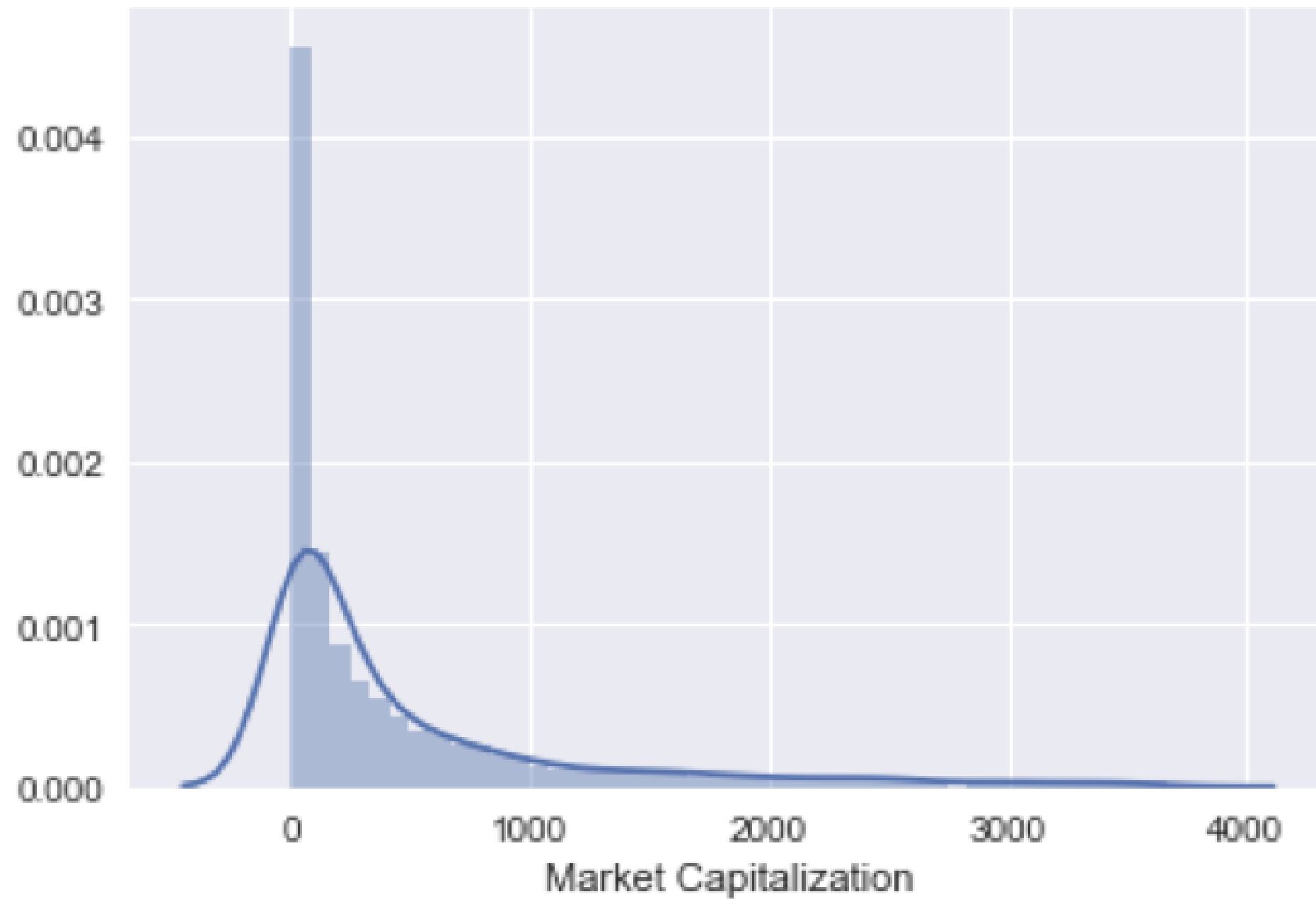
```
market_cap.median()
```

```
225.9684285
```

```
market_cap.mode()
```

```
0.0
```

# Calculate summary statistics



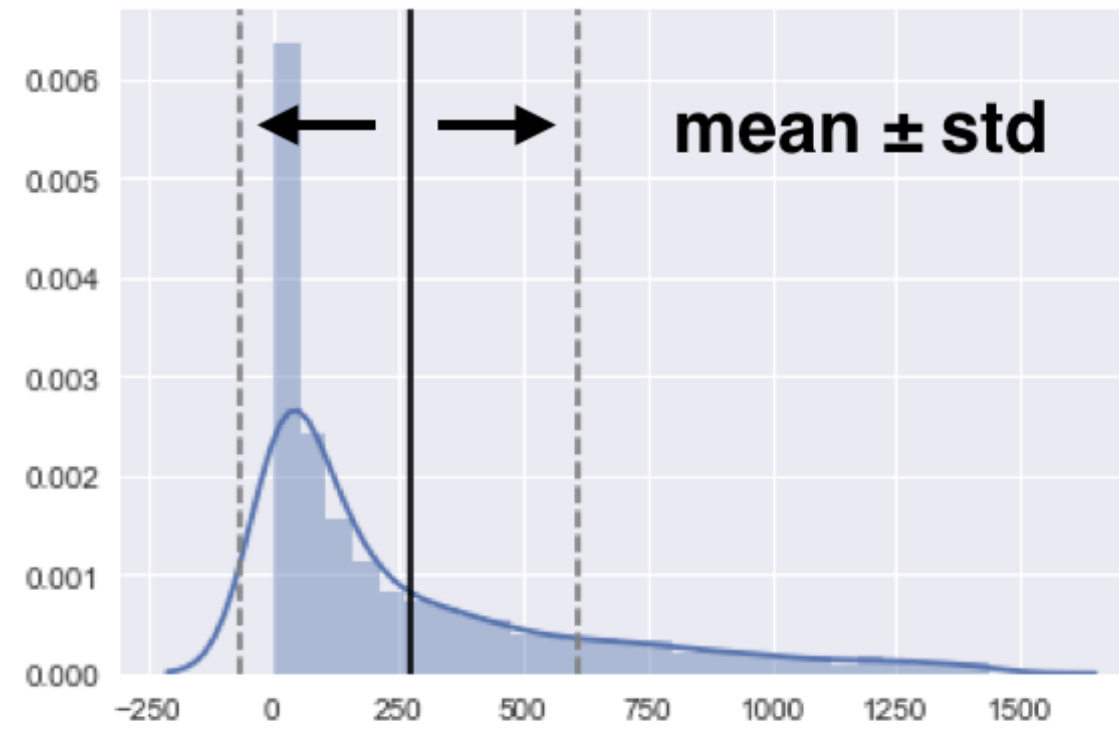
# Dispersion

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

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

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

- $sd = \sqrt{var}$





# Calculate variance and standard deviation

```
variance = market_cap.var()  
print(variance)
```

```
648773812.8182
```

```
np.sqrt(variance)
```

```
25471.0387
```

```
market_cap.std()
```

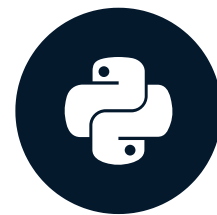
```
25471.0387
```

# Let's practice!

IMPORTING AND MANAGING FINANCIAL DATA IN PYTHON

# Describe the distribution of your data with quantiles

IMPORTING AND MANAGING FINANCIAL DATA IN PYTHON



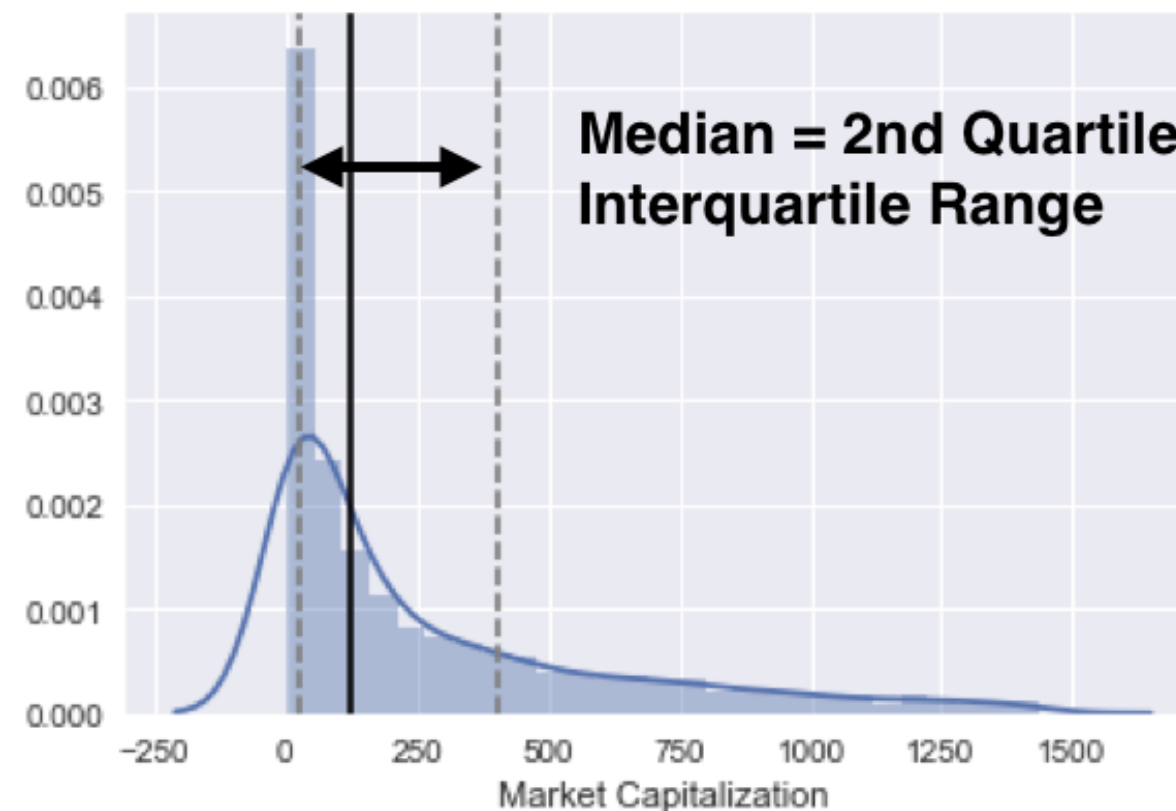
**Stefan Jansen**  
Instructor

# 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

```
market_cap = nasdaq['Market Capitalization'].div(10**6)
median = market_cap.quantile(.5)
median == market_cap.median()
```

```
True
```

```
quantiles = market_cap.quantile([.25, .75])
```

```
0.25    43.375930
0.75   969.905207
```

```
quantiles[.75] - quantiles[.25] # Interquartile Range
```

```
926.5292771575
```

# Quantiles with pandas & numpy

```
deciles = np.arange(start=.1, stop=.91, step=.1)
deciles
```

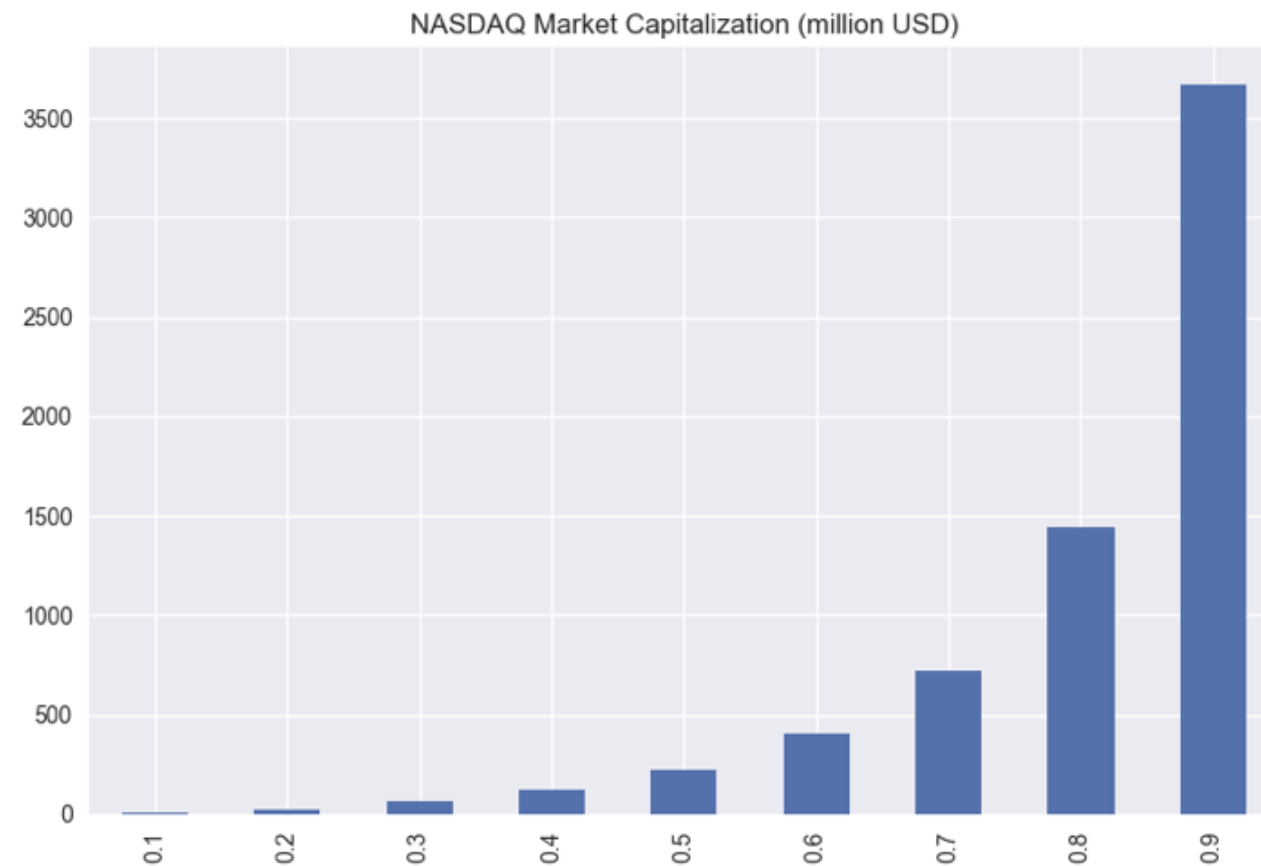
```
array([ 0.1,  0.2,  0.3,  0.4, ...,  0.7,  0.8,  0.9])
```

```
market_cap.quantile(deciles)
```

```
0.1      4.884565
0.2     26.993382
0.3     65.714547
0.4    124.320644
0.5    225.968428
0.6    402.469678
...
```

# Visualize quantiles with bar chart

```
title = 'NASDAQ Market Capitalization (million USD)'\nmarket_cap.quantile(deciles).plot(kind='bar', title=title)\nplt.tight_layout(); plt.show()
```





# All statistics in one go

```
market_cap.describe()
```

```
count      3167.000000
mean       3180.712621
std        25471.038707
min         0.000000
25%        43.375930   # 1st quantile
50%        225.968428   # Median
75%        969.905207   # 3rd quantile
max       740024.467000
Name: Market Capitalization
```

# All statistics in one go

```
market_cap.describe(percentiles=np.arange(.1, .91, .1))
```

```
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
...
```

# Let's practice!

IMPORTING AND MANAGING FINANCIAL DATA IN PYTHON

# Visualize the distribution of your data

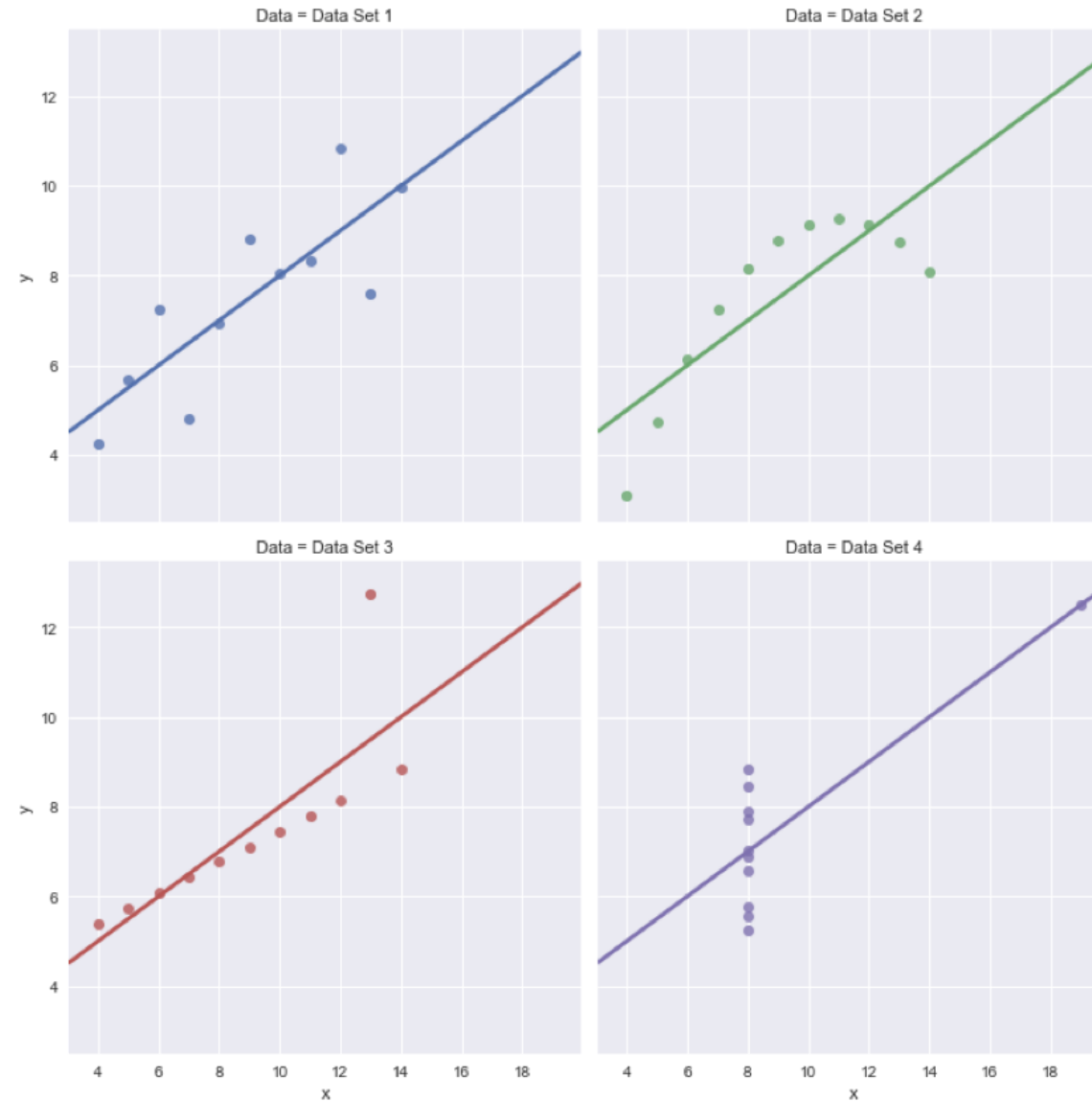
IMPORTING AND MANAGING FINANCIAL DATA IN PYTHON



**Stefan Jansen**  
Instructor

# 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 and distribution

```
ty10 = web.DataReader('DGS10', 'fred', date(1962, 1, 1))  
ty10.info()
```

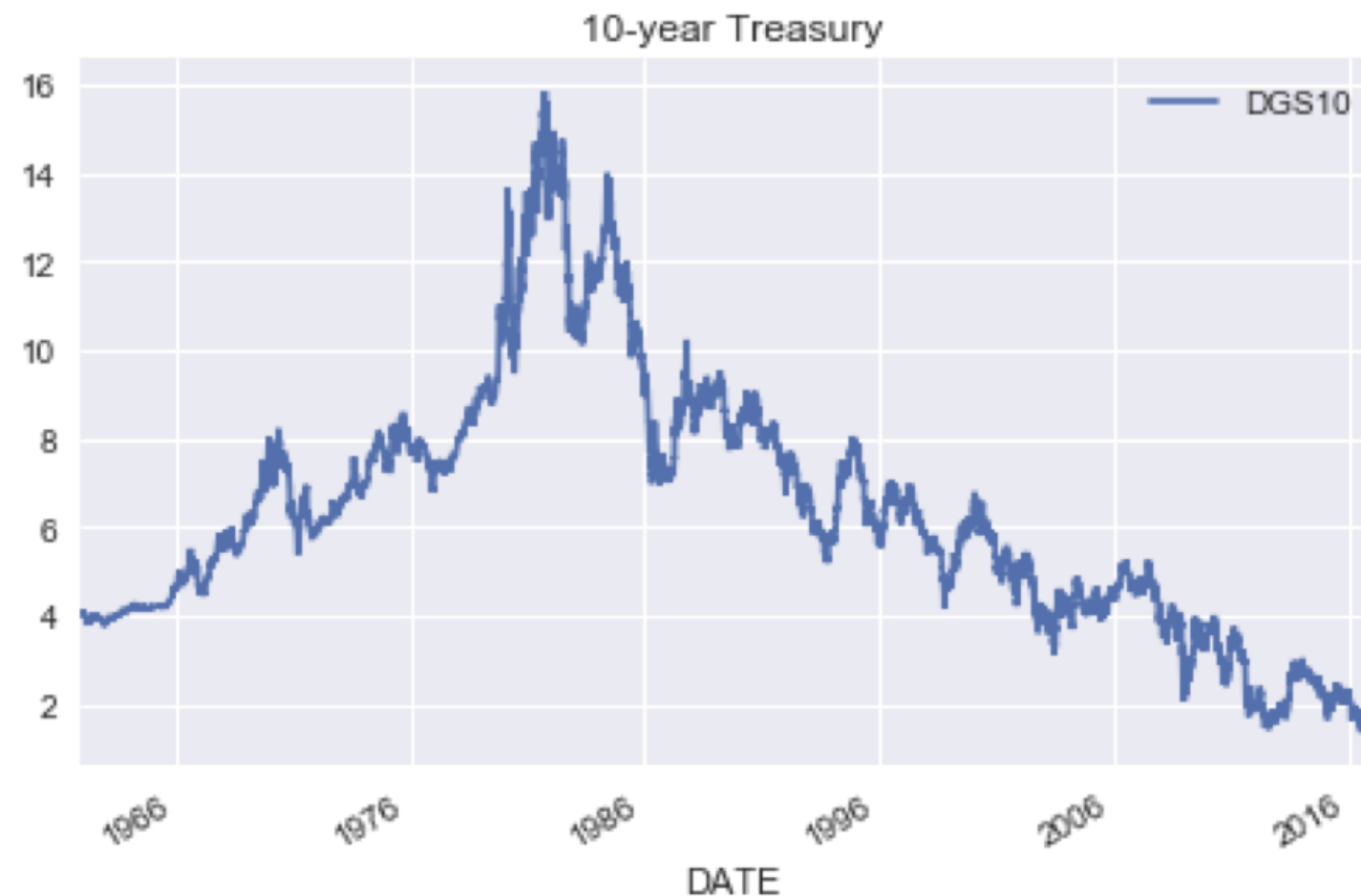
```
DatetimeIndex: 14443 entries, 1962-01-02 to 2017-05-11  
Data columns (total 1 columns):  
DGS10      13825 non-null float64
```

```
ty10.describe()
```

```
          DGS10  
mean      6.291073  
std       2.851161  
min       1.370000  
25%       4.190000  
50%       6.040000  
...
```

# 10 year treasury: time series trend

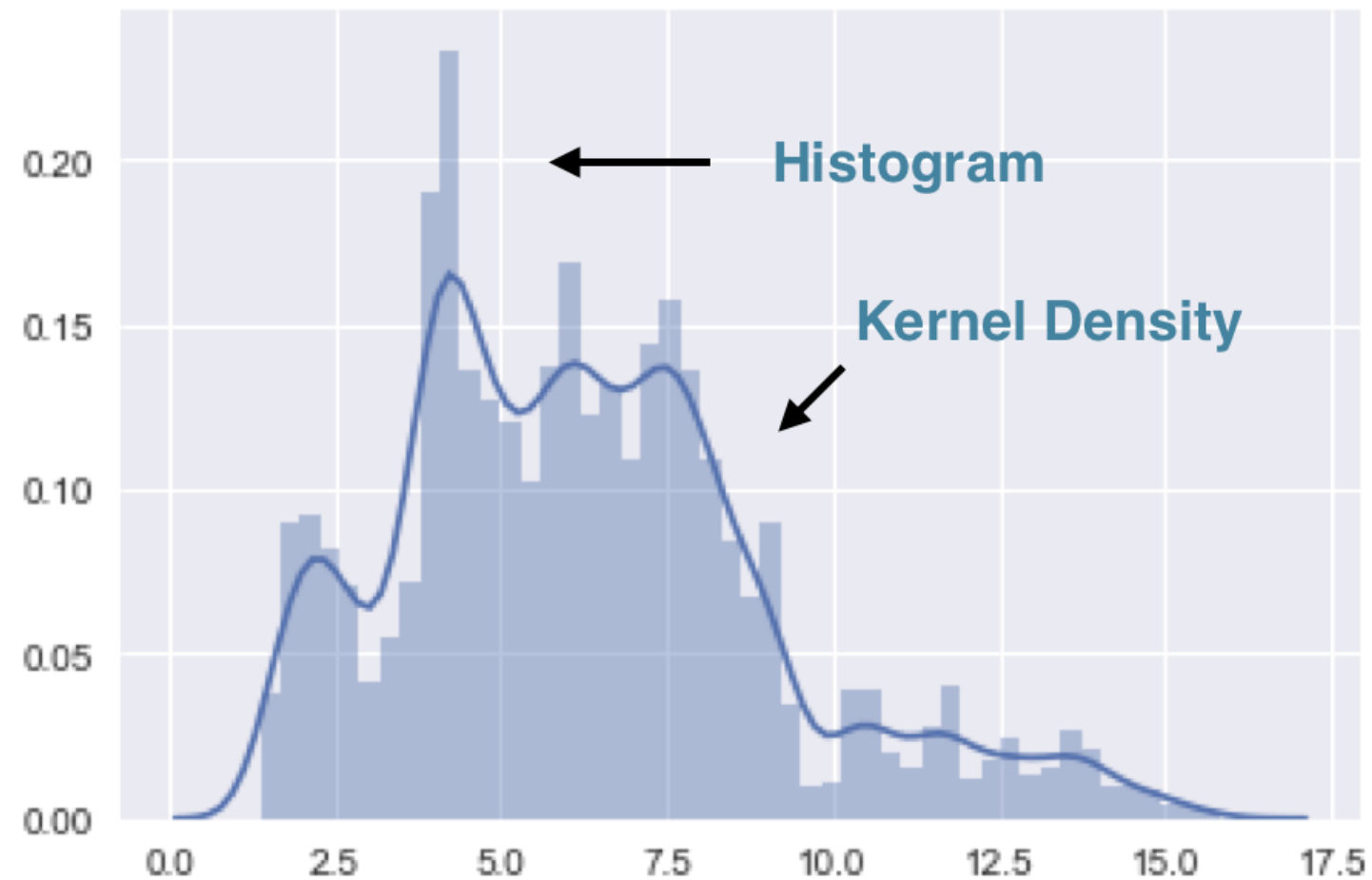
```
ty10.dropna(inplace=True) # Avoid creation of copy  
ty10.plot(title='10-year Treasury'); plt.tight_layout()
```





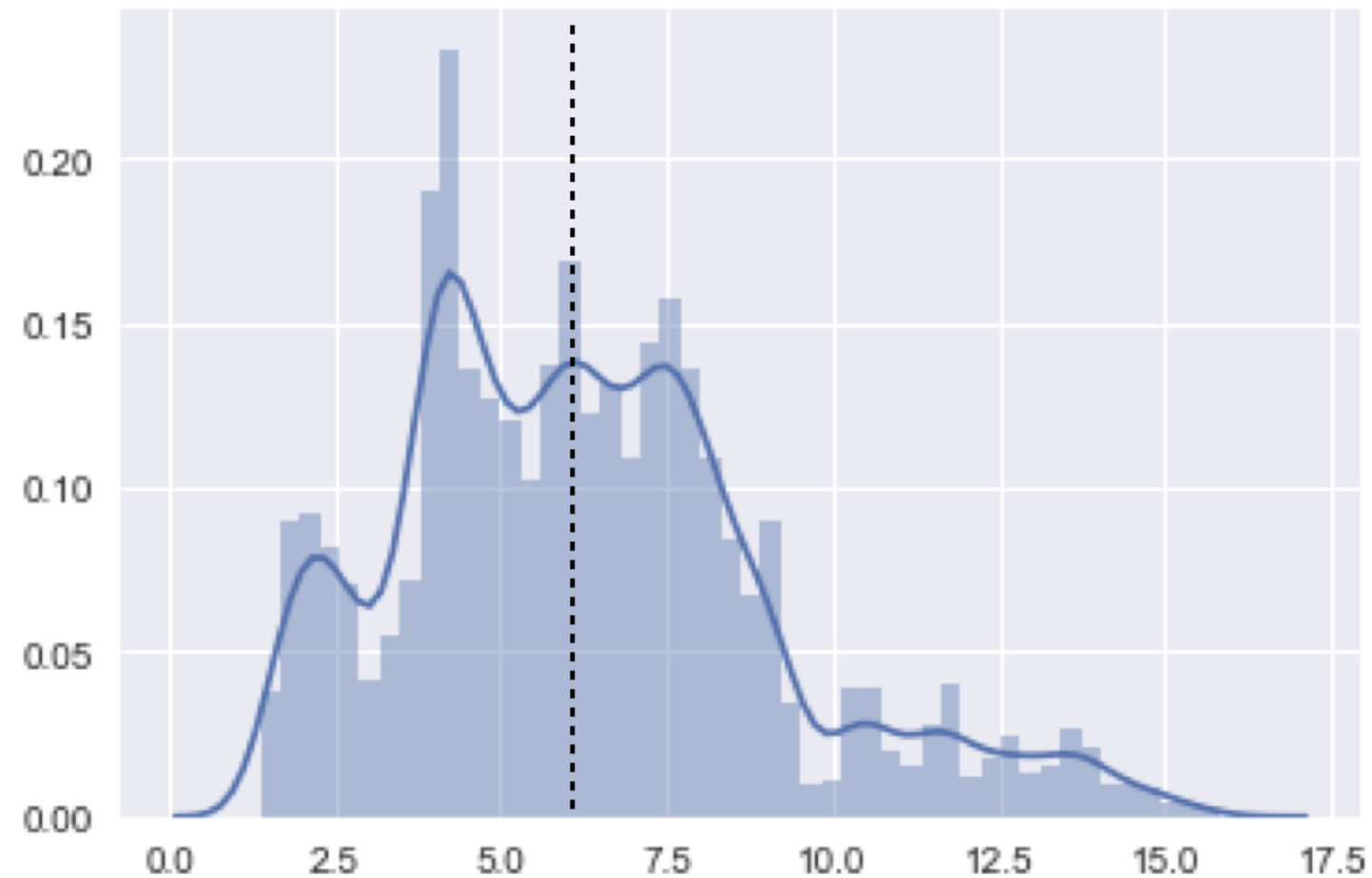
# 10 year treasury: historical distribution

```
import seaborn as sns
sns.distplot(ty10)
```



# 10 year treasury: trend and distribution

```
ax = sns.distplot(ty10)
ax.axvline(ty10['DGS10'].median(), color='black', ls='--')
```



# Let's practice!

IMPORTING AND MANAGING FINANCIAL DATA IN PYTHON

# Summarize categorical variables

IMPORTING AND MANAGING FINANCIAL DATA IN PYTHON



**Stefan Jansen**  
Instructor

# 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

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

# Categorical listing information

```
amex = amex['Sector'].nunique()
```

```
12
```

- `apply()` : call function on each column
- `lambda` : "anonymous function", receives each column as argument `x`

```
amex.Sector.apply(lambda x: x.nunique())
```

```
Stock Symbol      360  
Company Name      326  
Last Sale         323  
Market Capitalization 317  
...
```

# How many observations per sector?

```
amex['Sector'].value_counts()
```

```
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
...
```



# How many IPOs per year?

```
amex['IPO Year'].value_counts()
```

```
2002.0    19 # Mode
2015.0    11
1999.0     9
1993.0     7
2014.0     6
2013.0     5
2017.0     5
...
2009.0     1
1990.0     1
1991.0     1
Name: IPO Year, dtype: int64
```

# Convert IPO Year to int

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

```
2002    19  
2015    11  
1999     9  
1993     7  
2014     6  
2004     5  
2003     5  
2017     5  
...  
1987     1  
Name: IPO Year, dtype: int64
```

# Convert IPO Year to int

```
ipo_by_yr.plot(kind='bar', title='IPOs per Year')  
plt.xticks(rotation=45)
```



# Let's practice!

IMPORTING AND MANAGING FINANCIAL DATA IN PYTHON