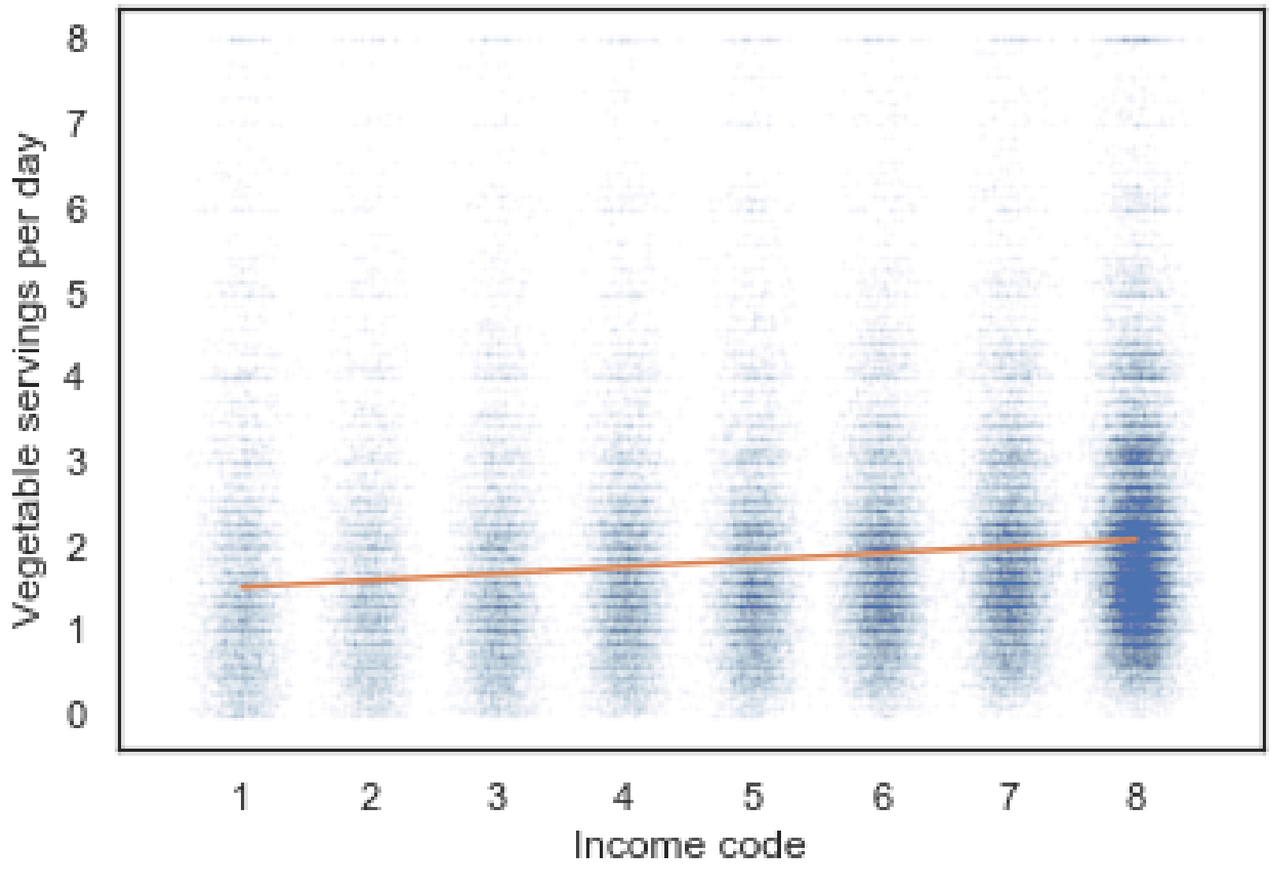
Limits of simple regression

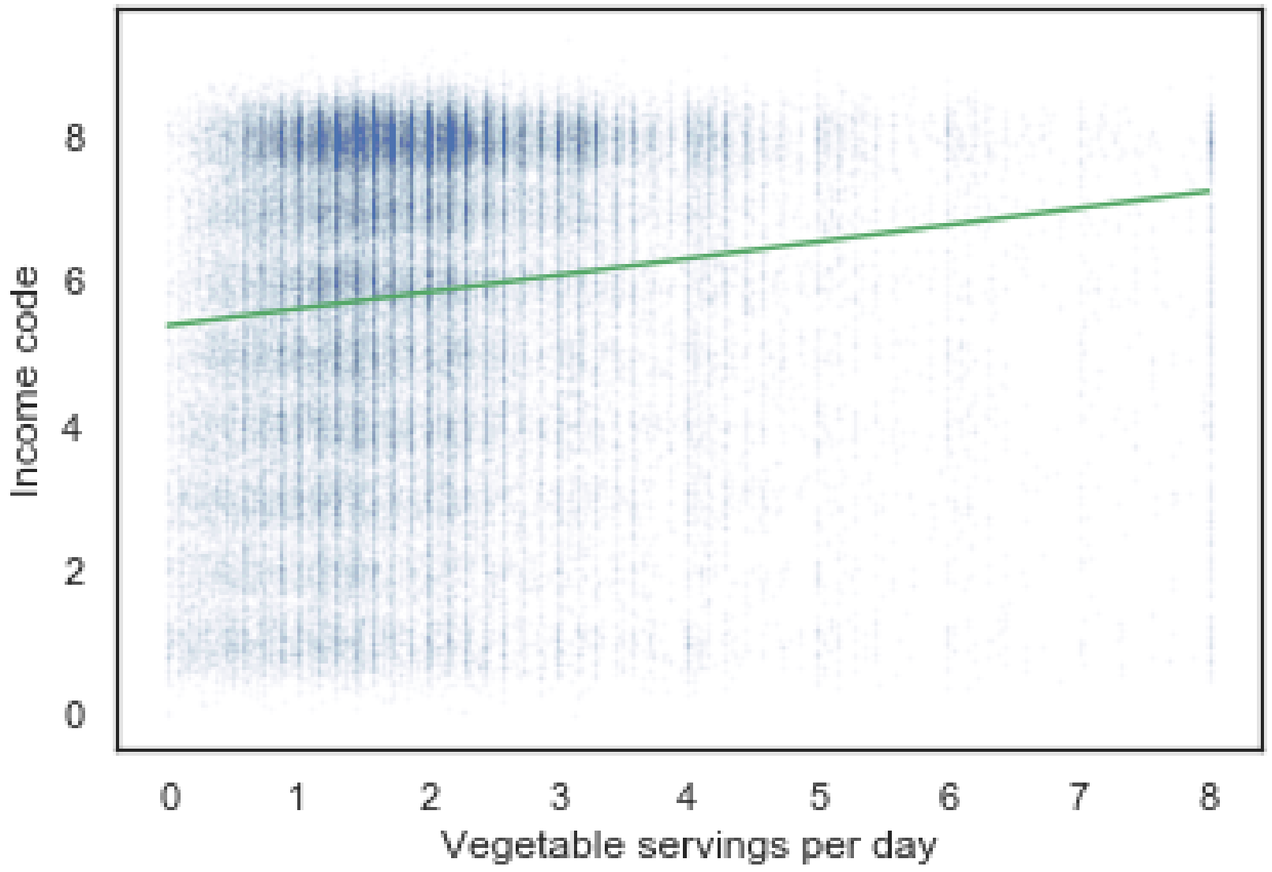
EXPLORATORY DATA ANALYSIS IN PYTHON

Allen Downey
Professor, Olin College

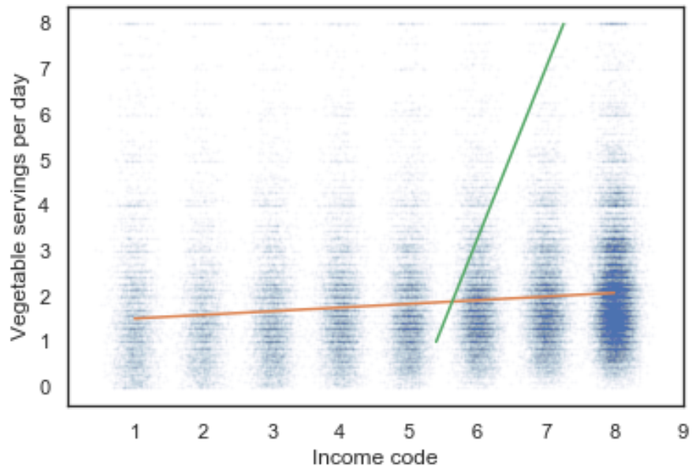
Income and vegetables

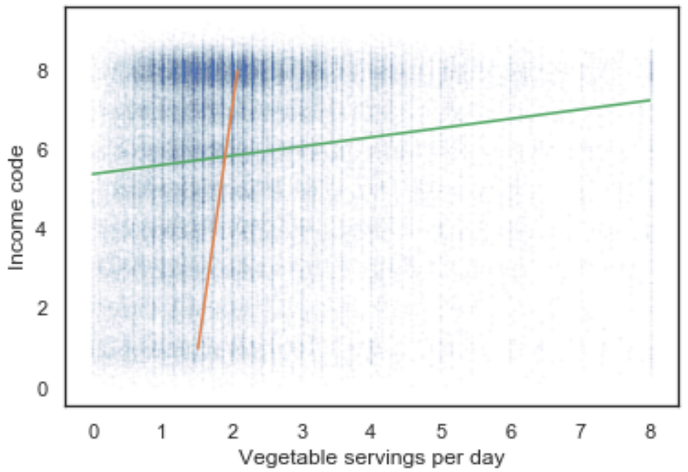


Vegetables and income

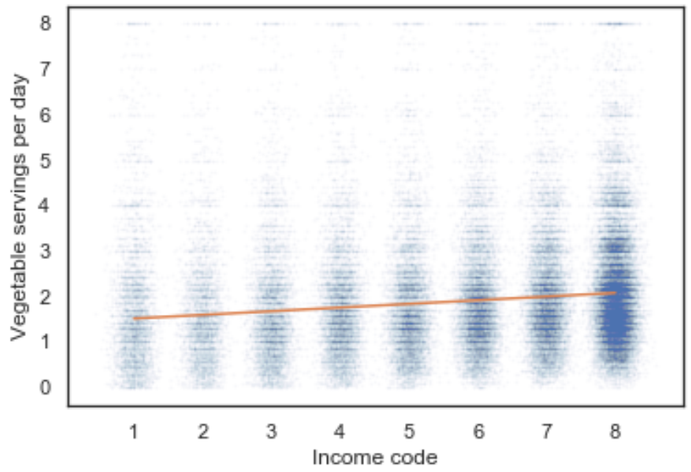


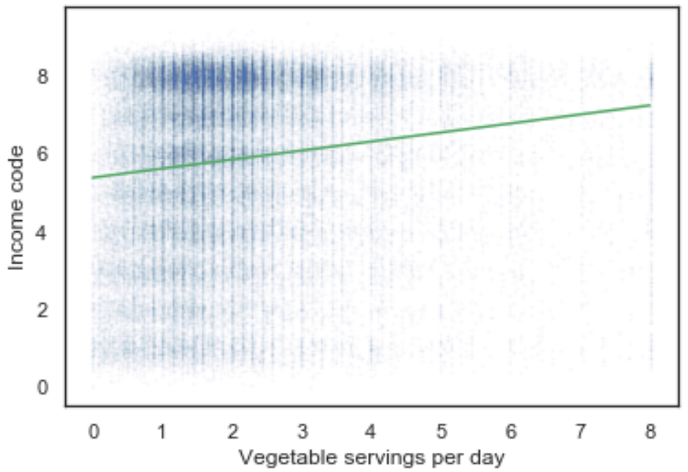
Regression is not symmetric





Regression is not causation





Multiple regression

```
import statsmodels.formula.api as smf
```

```
results = smf.ols('INCOME2 ~ _VEGESU1', data=brfss).fit()
results.params
```

```
Intercept    5.399903
_VEGESU1     0.232515
dtype: float64
```

Let's practice!

EXPLORATORY DATA ANALYSIS IN PYTHON

Multiple regression

EXPLORATORY DATA ANALYSIS IN PYTHON

Allen Downey
Professor, Olin College

Income and education

```
gss = pd.read_hdf('gss.hdf5', 'gss')
```

```
results = smf.ols('realinc ~ educ', data=gss).fit()
results.params
```

```
Intercept   -11539.147837
educ          3586.523659
dtype: float64
```

Adding age

```
results = smf.ols('realinc ~ educ + age', data=gss).fit()
results.params
```

```
Intercept   -16117.275684
educ          3655.166921
age             83.731804
dtype: float64
```

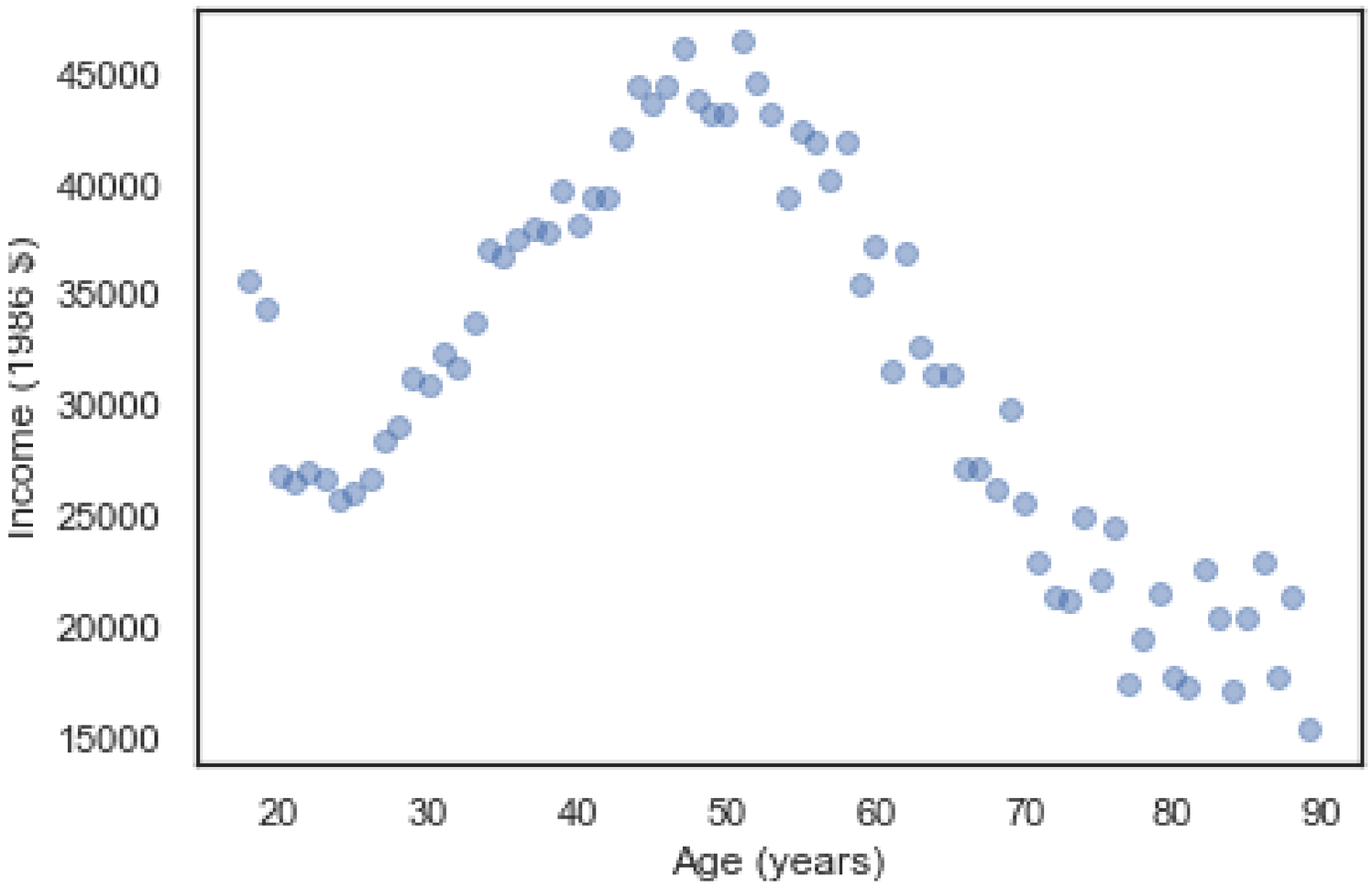
Income and age

```
grouped = gss.groupby('age')
```

```
<pandas.core.groupby.groupby.DataFrameGroupBy object
at 0x7f1264b8ce80>
```

```
mean_income_by_age = grouped['realinc'].mean()
```

```
plt.plot(mean_income_by_age, 'o', alpha=0.5)
plt.xlabel('Age (years)')
plt.ylabel('Income (1986 $)')
```


45000
40000
35000
30000
25000
20000
15000
Income (1986 $)
20
30
40
50
60
70
80
90
Age (years)

Adding a quadratic term

```
gss['age2'] = gss['age']**2
```

```
model = smf.ols('realinc ~ educ + age + age2', data=gss)
results = model.fit()
results.params
```

```
Intercept   -48058.679679
educ          3442.447178
age           1748.232631
age2           -17.437552
dtype: float64
```

Whew!

EXPLORATORY DATA ANALYSIS IN PYTHON

Visualizing regression results

EXPLORATORY DATA ANALYSIS IN PYTHON

Allen Downey
Professor, Olin College

Modeling income and age

```
gss['age2'] = gss['age']**2
gss['educ2'] = gss['educ']**2
```

```
model = smf.ols('realinc ~ educ + educ2 + age + age2', data
results = model.fit()
results.params
```

```
Intercept   -23241.884034
educ           -528.309369
educ2           159.966740
age            1696.717149
age2            -17.196984
```

Generating predictions

```
df = pd.DataFrame()
df['age'] = np.linspace(18, 85)
df['age2'] = df['age']**2
```

```
df['educ'] = 12
df['educ2'] = df['educ']**2
```

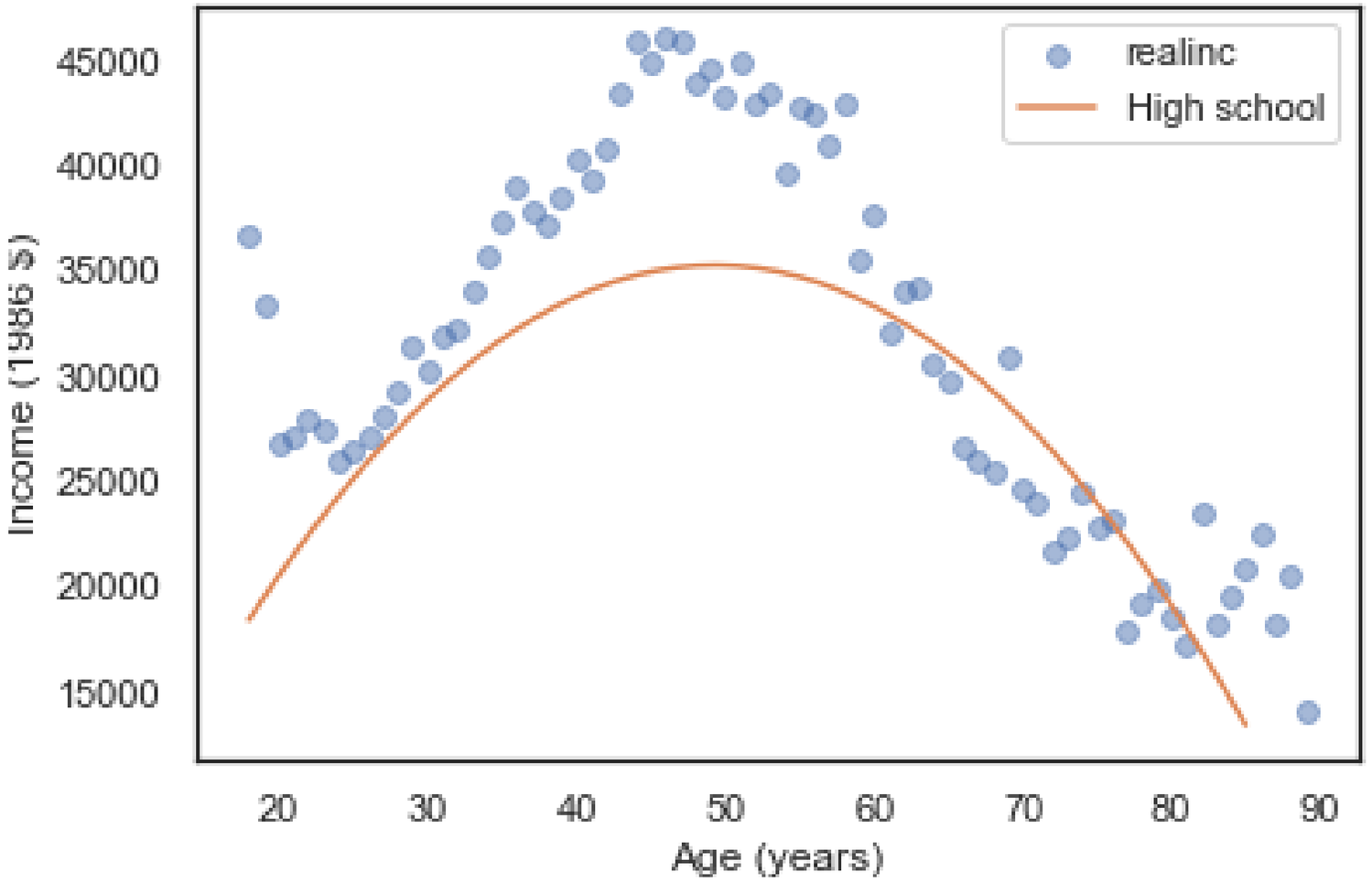
```
pred12 = results.predict(df)
```

Plotting predictions

```
plt.plot(df['age'], pred12, label='High school')
```

```
plt.plot(mean_income_by_age, 'o', alpha=0.5)
```

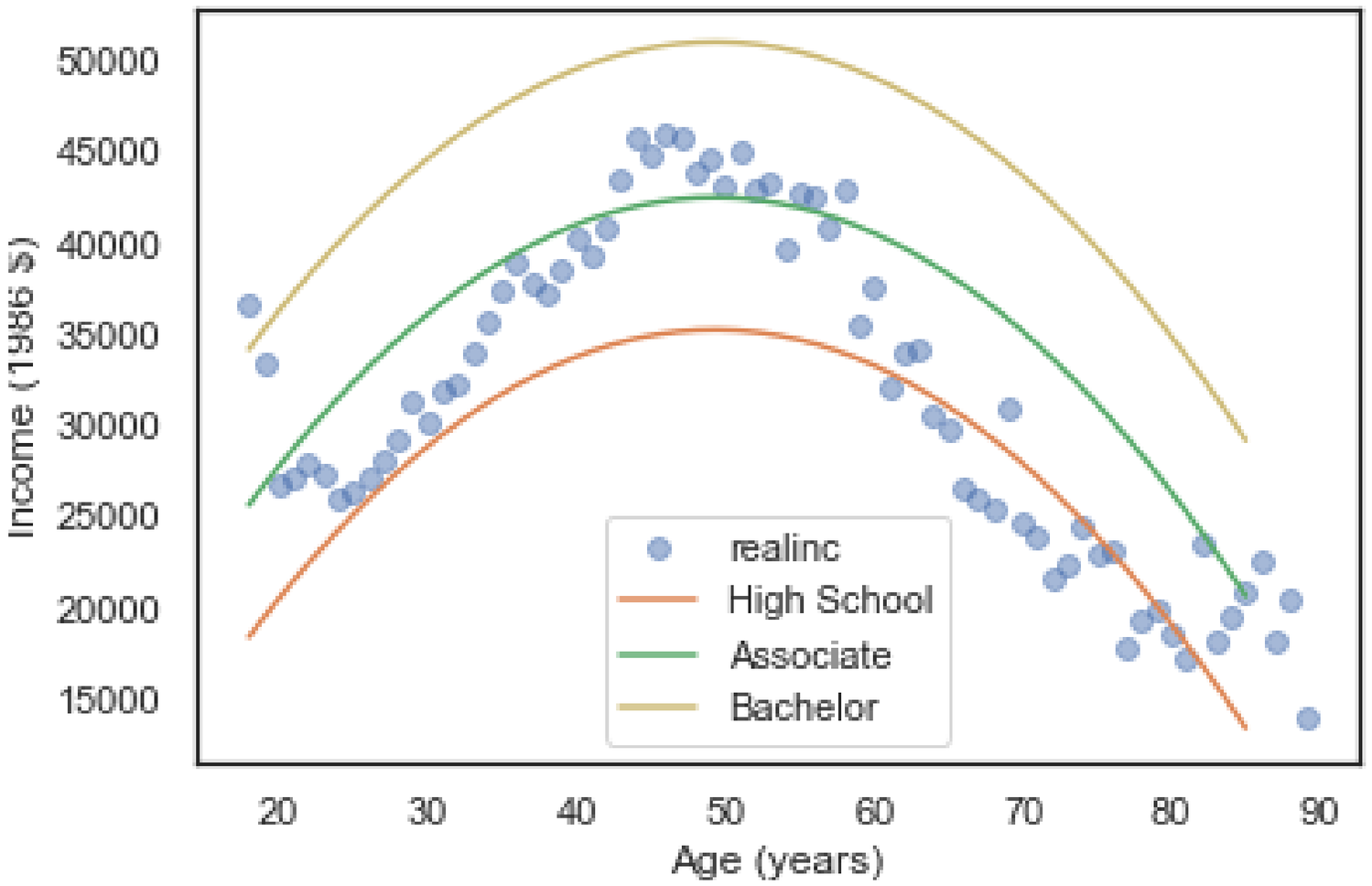
```
plt.xlabel('Age (years)')
plt.ylabel('Income (1986 $)')
plt.legend()
```


realinc
High school
45000
40000
35000
30000
25000
20000
15000
Income (1986 $)
20
30
40
50
60
70
80
90
Age (years)

Levels of education

```
df['educ'] = 14
df['educ2'] = df['educ']**2
pred14 = results.predict(df)
plt.plot(df['age'], pred14, label='Associate')
```

```
df['educ'] = 16
df['educ2'] = df['educ']**2
pred16 = results.predict(df)
plt.plot(df['age'], pred16, label='Bachelor'
```


50000
45000
40000
35000
30000
25000
20000
15000
Income (1986 $)
20
30
40
50
60
70
80
90
Age (years)
realinc
High School
Associate
Bachelor

Let's practice!

EXPLORATORY DATA ANALYSIS IN PYTHON

Logistic regression

EXPLORATORY DATA ANALYSIS IN PYTHON

Allen Downey
Professor, Olin College

Categorical variables

- Numerical variables: income, age, years of education.
- Categorical variables: sex, race.

Sex and income

```
formula = 'realinc ~ educ + educ2 + age + age2 + C(sex)'
results = smf.ols(formula, data=gss).fit()
results.params
```

```
Intercept      -22369.453641
C(sex)[T.2]     -4156.113865
educ             -310.247419
educ2             150.514091
age              1703.047502
age2              -17.238711
```

Boolean variable

```
gss['gunlaw'].value_counts()
```

```
1.0    30918
2.0     9632
```

```
gss['gunlaw'].replace([2], [0], inplace=True)
```

```
gss['gunlaw'].value_counts()
```

```
1.0    30918
0.0     9632
```

Logistic regression

```
formula = 'gunlaw ~ age + age2 + educ + educ2 + C(sex)'
results = smf.logit(formula, data=gss).fit()
```

```
results.params
```

```
Intercept      1.653862
C(sex)[T.2]    0.757249
age           -0.018849
age2           0.000189
educ          -0.124373
educ2          0.006653
```

Generating predictions

```
df = pd.DataFrame()
df['age'] = np.linspace(18, 89)
df['educ'] = 12
```

```
df['age2'] = df['age']**2
df['educ2'] = df['educ']**2
```

```
df['sex'] = 1
pred1 = results.predict(df)
```

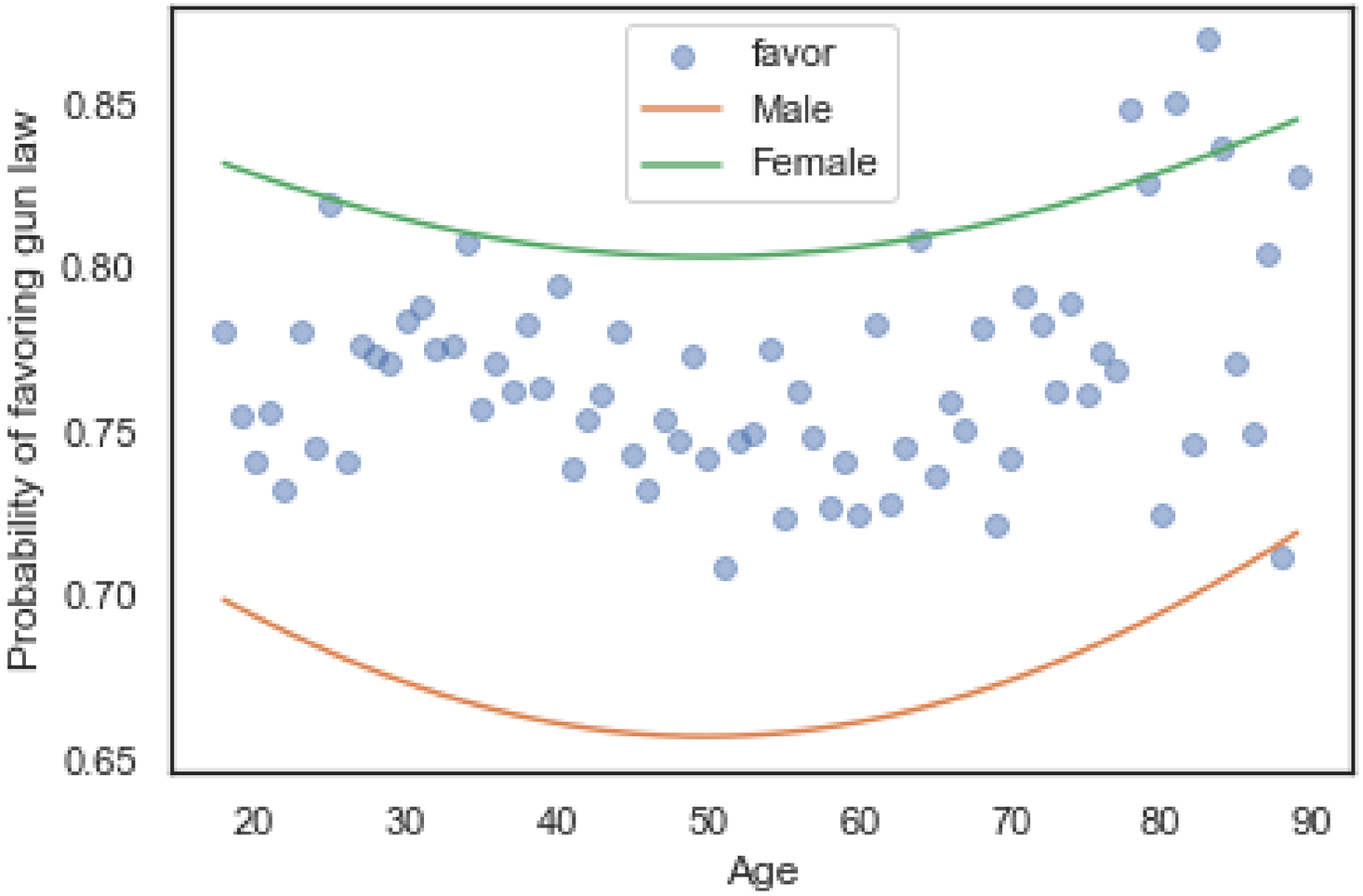
```
df['sex'] = 2
pred2 = results.predict(df)
```

Visualizing results

```
grouped = gss.groupby('age')
favor_by_age = grouped['gunlaw'].mean()
plt.plot(favor_by_age, 'o', alpha=0.5)
```

```
plt.plot(df['age'], pred1, label='Male')
plt.plot(df['age'], pred2, label='Female')
```

```
plt.xlabel('Age')
plt.ylabel('Probability of favoring gun law')
plt.legend()
```


favor
Male
Female
Probability of favoring gun law
0.85
0.80
0.75
0.70
0.65
20
30
40
50
60
70
80
90
Age

Let's practice!

EXPLORATORY DATA ANALYSIS IN PYTHON

Next steps

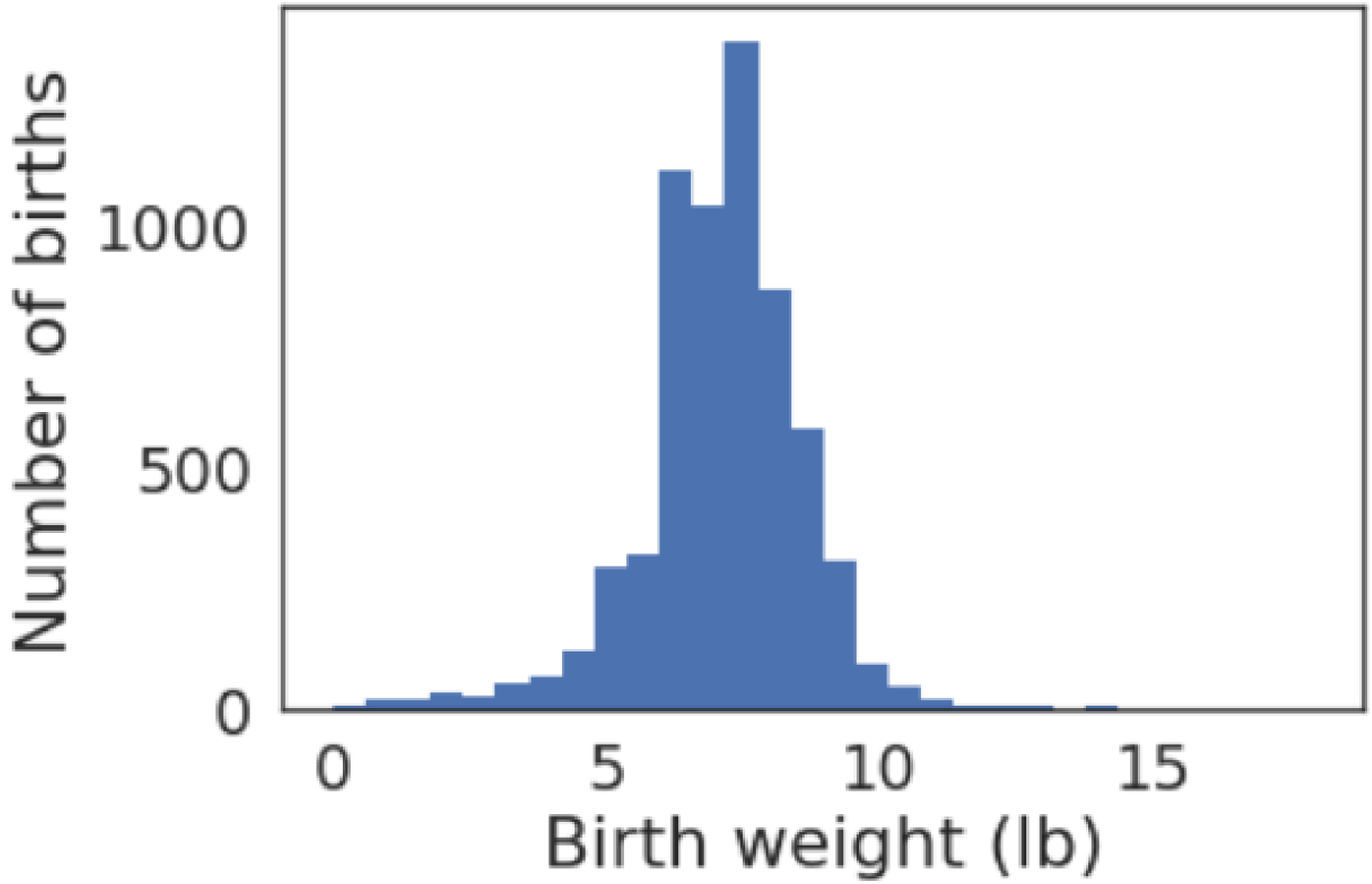
EXPLORATORY DATA ANALYSIS IN PYTHON

Allen Downey
Professor, Olin College

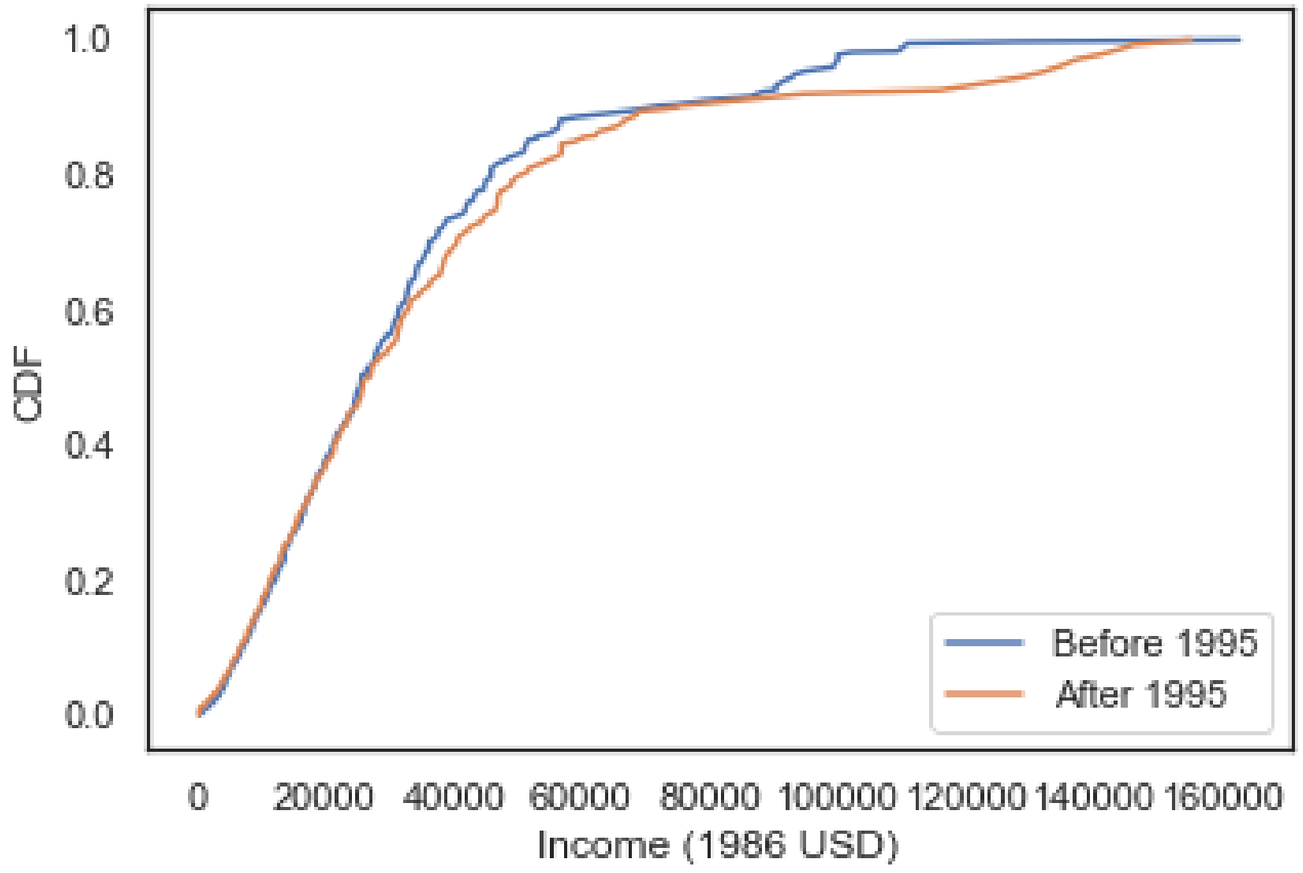
Exploratory Data Analysis

- Import, clean, and validate
- Visualize distributions
- Explore relationships between variables
- Explore multivariate relationships

Import, clean, and validate



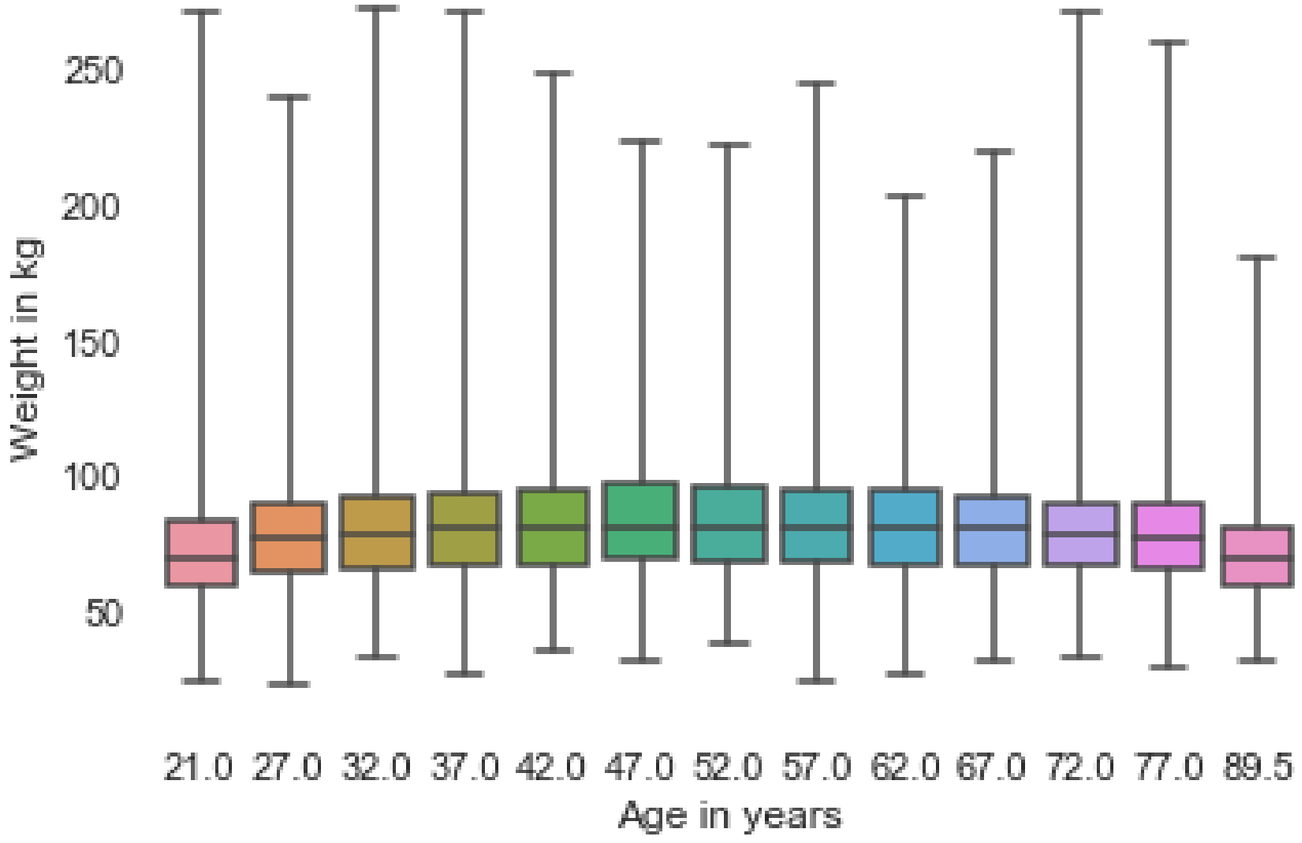
Visualize distributions



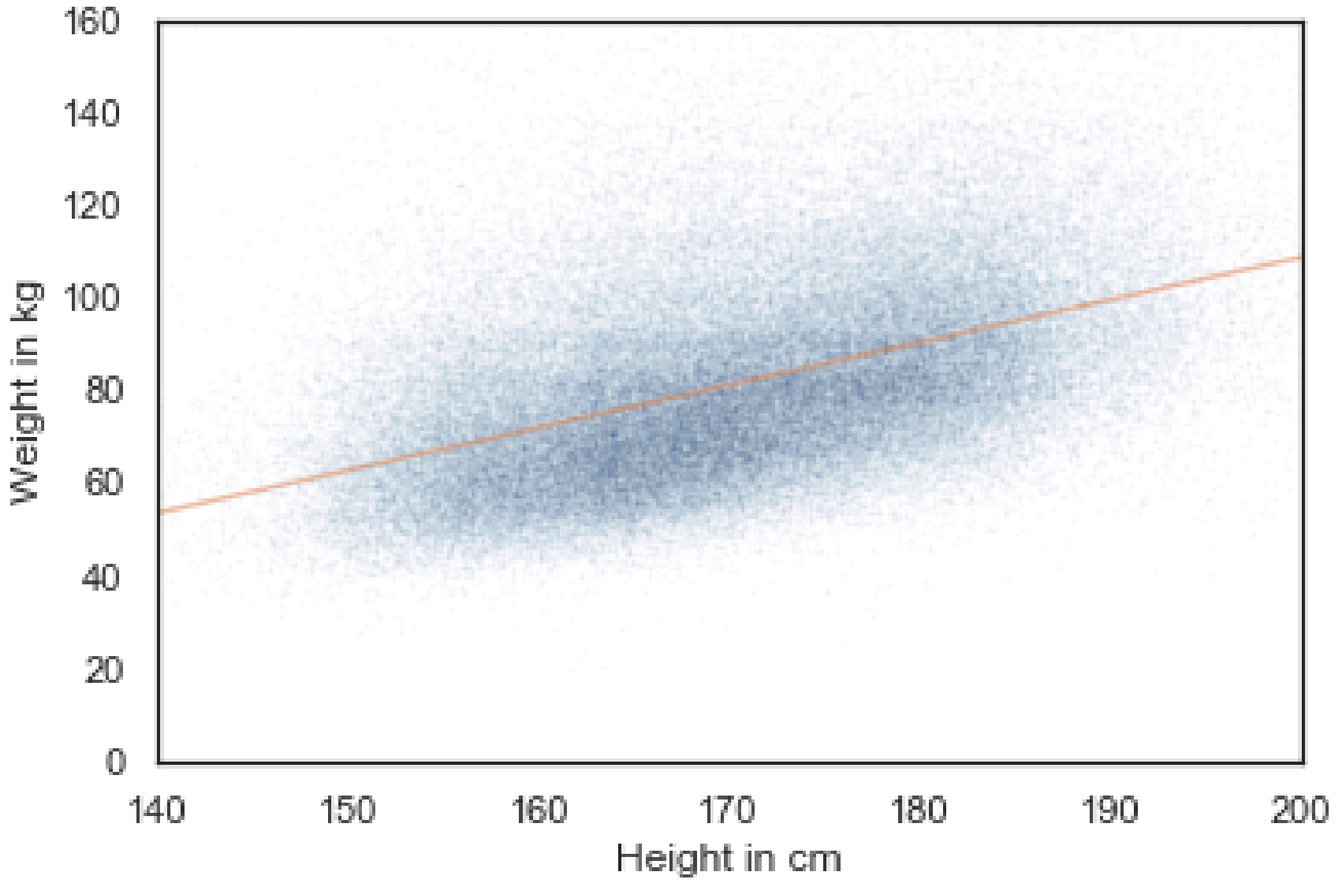
CDF, PMF, and KDE

- Use CDFs for exploration.
- Use PMFs if there are a small number of unique values.
- Use KDE if there are a lot of values.

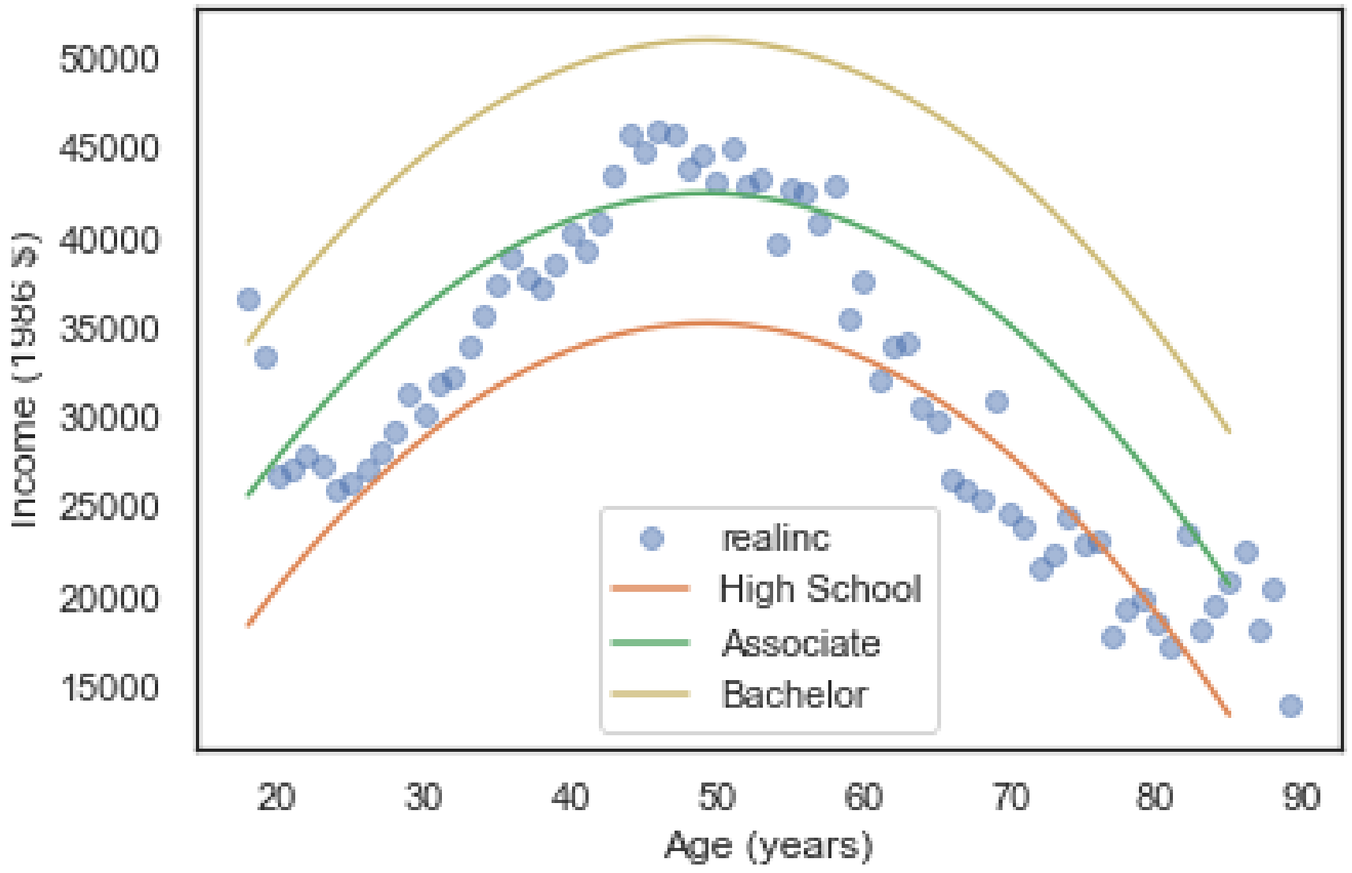
Visualizing relationships



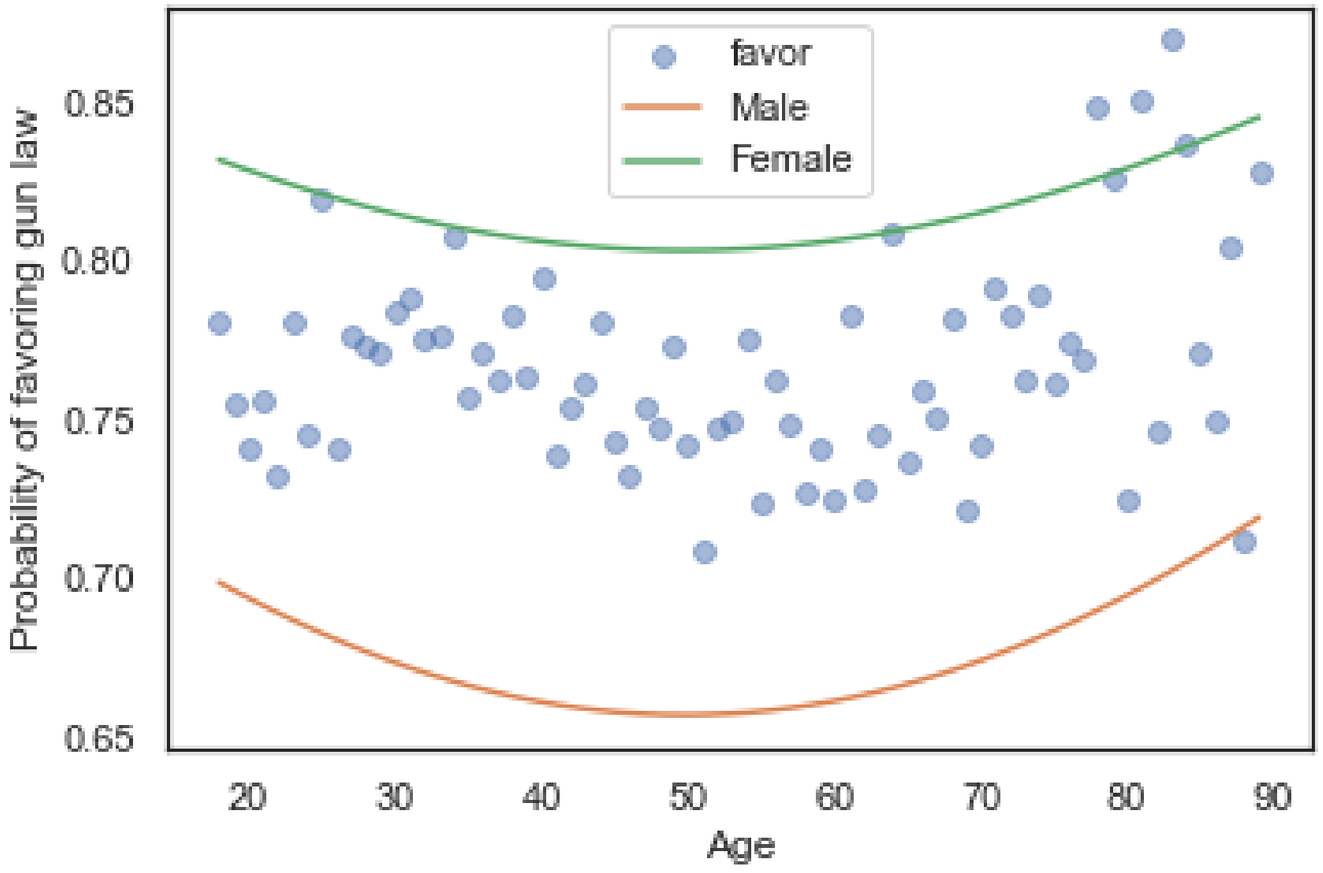
Quantifying correlation



Multiple regression



Logistic regression



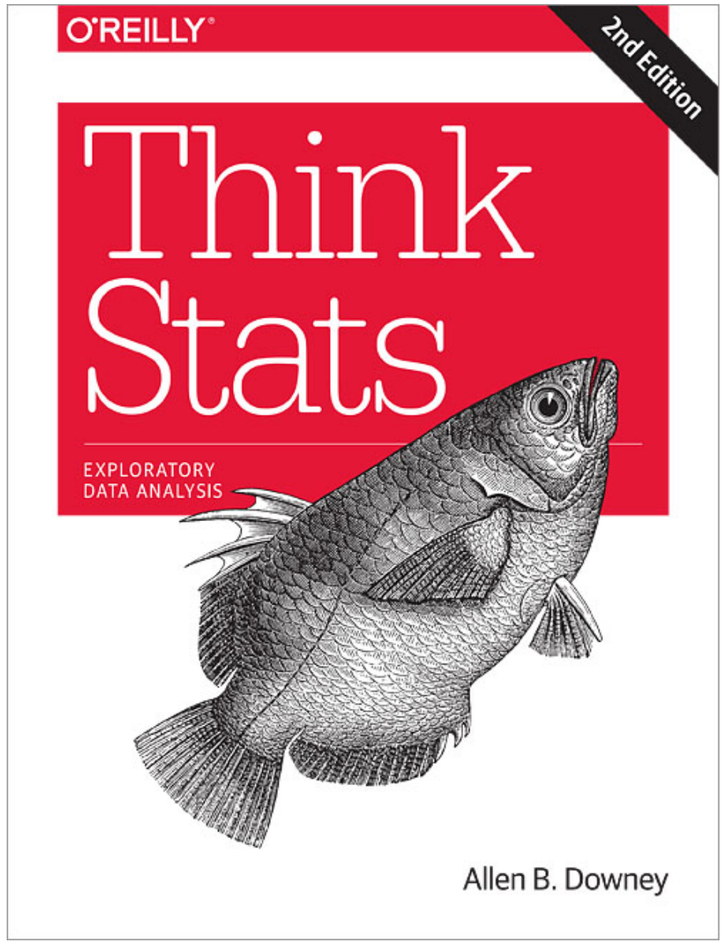
Where to next?

- Statistical Thinking in Python
- pandas Foundations
- Improving Your Data Visualizations in Python
- Introduction to Linear Modeling in Python

Think Stats

This course is based on *Think Stats*

Published by O'Reilly and available free from thinkstats2.com



Thank you!

EXPLORATORY DATA ANALYSIS IN PYTHON

Exploring relationships

EXPLORATORY DATA ANALYSIS IN PYTHON

Allen Downey
Professor, Olin College

Height and weight

Centers for Disease Control and Prevention
CDC 24/7: Saving Lives, Protecting People™

SEARCH

CDC A-Z INDEX

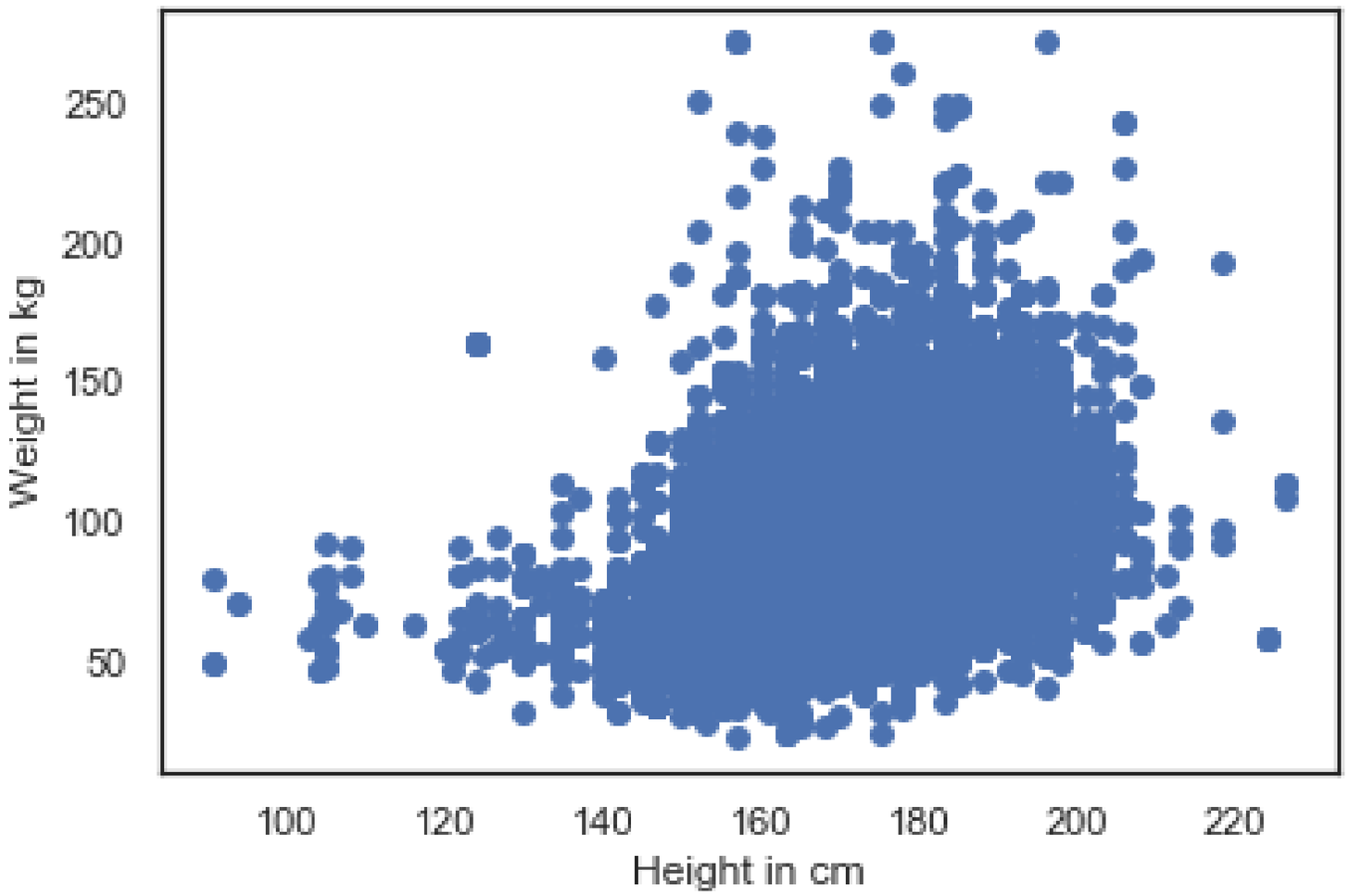
Behavioral Risk Factor Surveillance System

BRFSS™

The Behavioral Risk Factor Surveillance System (BRFSS) is the nation's premier system of health-related telephone surveys that collect state data about U.S. residents regarding their health-related risk behaviors, chronic health conditions, and use of preventive services. Established in 1984 with 15 states, BRFSS now collects data in all 50 states as well as the District of Columbia and three U.S. territories. BRFSS completes more than 400,000 adult interviews each year, making it the largest continuously conducted health survey system in the world. See More.

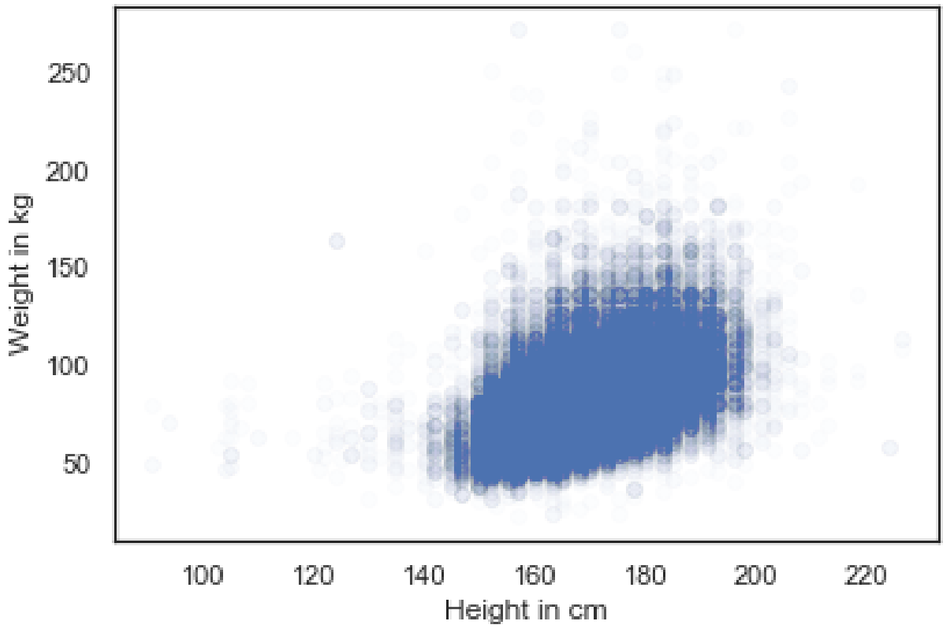
Scatter plot

```
brfss = pd.read_hdf('brfss.hdf5', 'brfss')
height = brfss['HTM4']
weight = brfss['WTKG3']

plt.plot(height, weight, 'o')

plt.xlabel('Height in cm')
plt.ylabel('Weight in kg')
plt.show()
```


250
200
150
100
50
Weight in kg
100
120
140
160
180
200
220
Height in cm

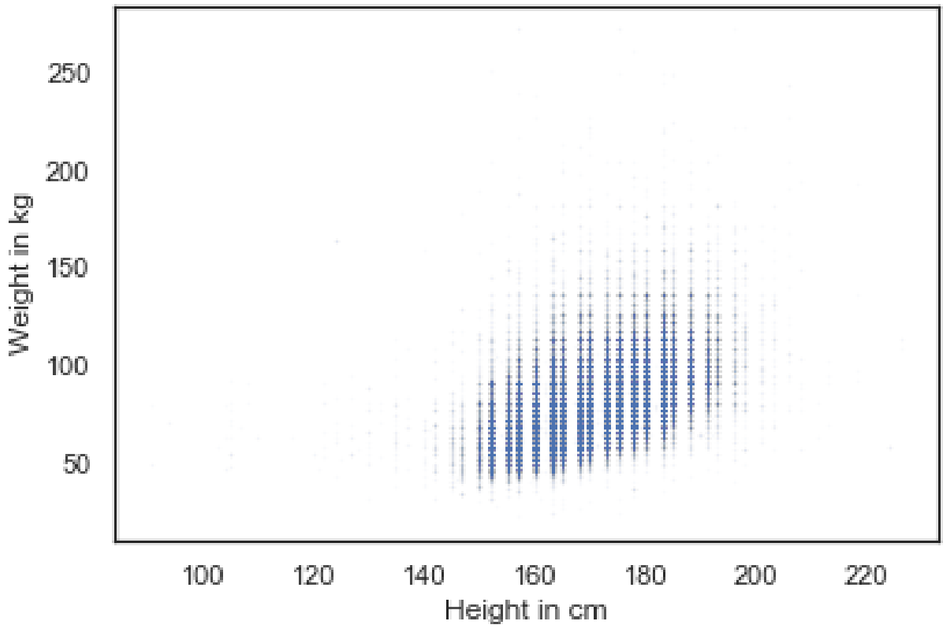
Transparency

```
plt.plot(height, weight, 'o', alpha=0.02)
plt.show()
```



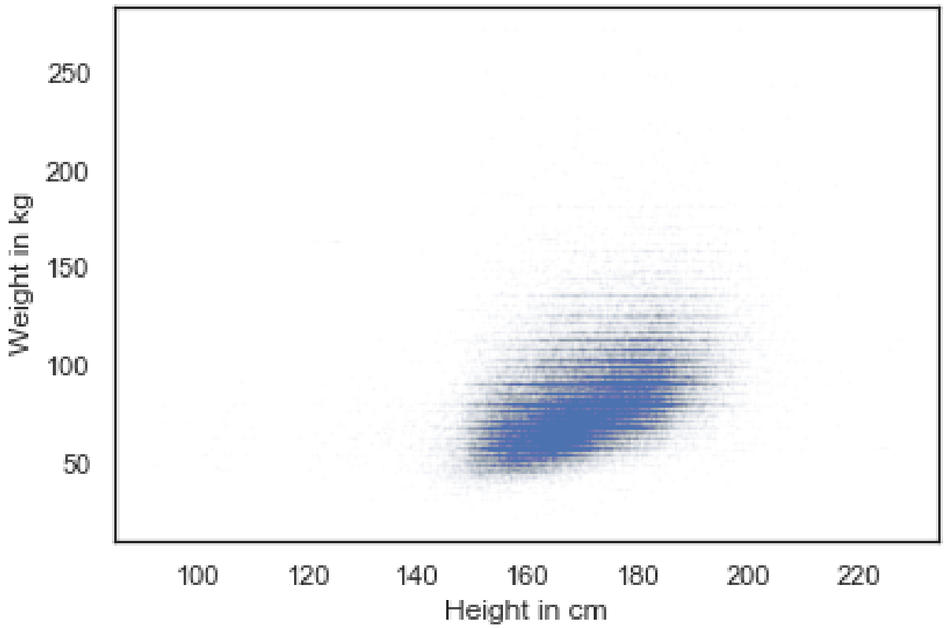
Marker size

```
plt.plot(height, weight, 'o', markersize=1, alpha=0.02)
plt.show()
```



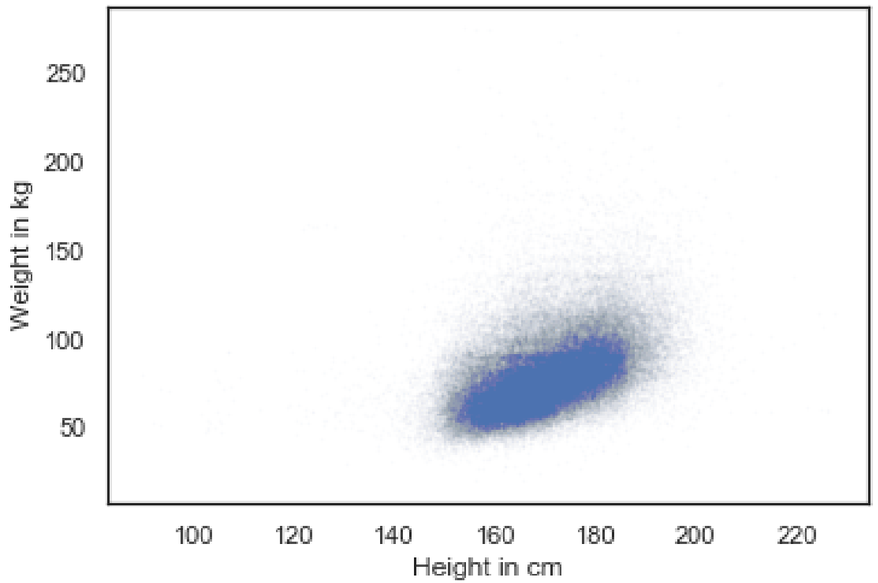
Jittering

```
height_jitter = height + np.random.normal(0, 2, size=len(brfss))
plt.plot(height_jitter, weight, 'o', markersize=1, alpha=0.02)
plt.show()
```



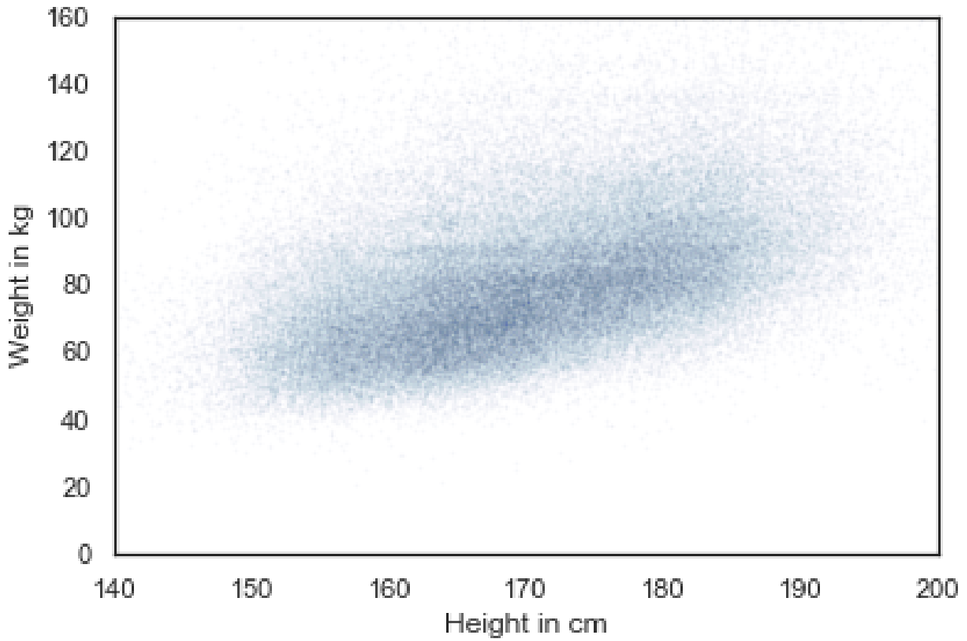
More jittering

```
height_jitter = height + np.random.normal(0, 2, size=len(brfss))
weight_jitter = weight + np.random.normal(0, 2, size=len(brfss))
plt.plot(height_jitter, weight_jitter, 'o', markersize=1, alpha=0.01
plt.show()
```

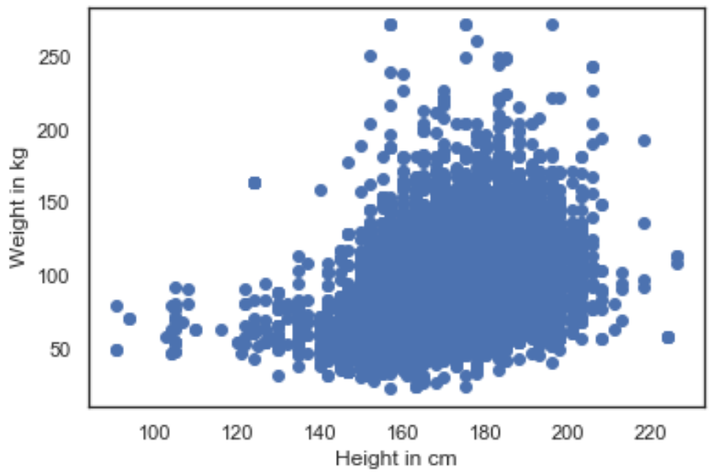


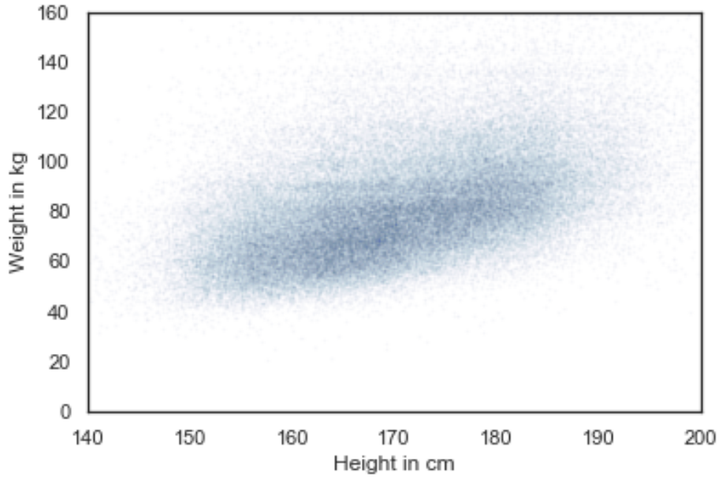
Zoom

```
plt.plot(height_jitter, weight_jitter, 'o', markersize=1, alpha=0.02
plt.axis([140, 200, 0, 160])
plt.show()
```



Before and after





Let's explore!

EXPLORATORY DATA ANALYSIS IN PYTHON

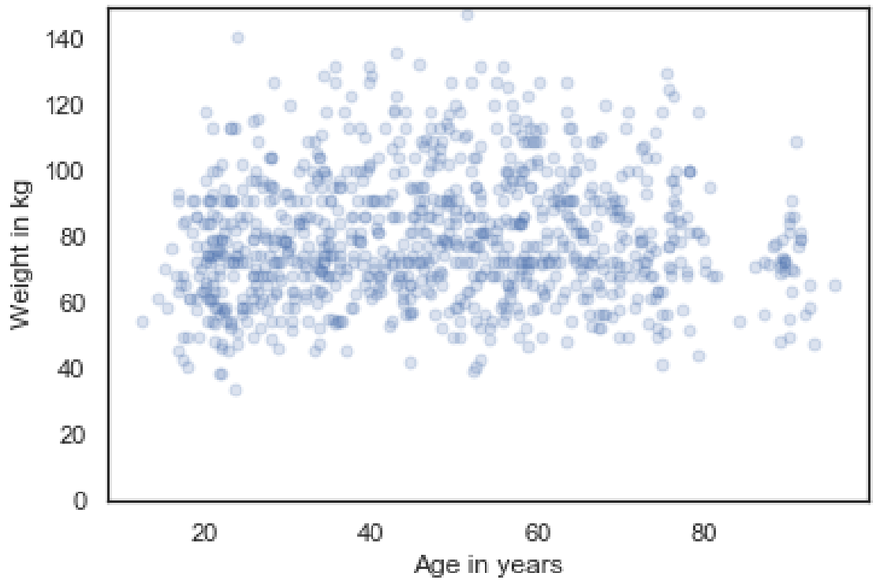
Visualizing relationships

EXPLORATORY DATA ANALYSIS IN PYTHON

Allen Downey
Professor, Olin College

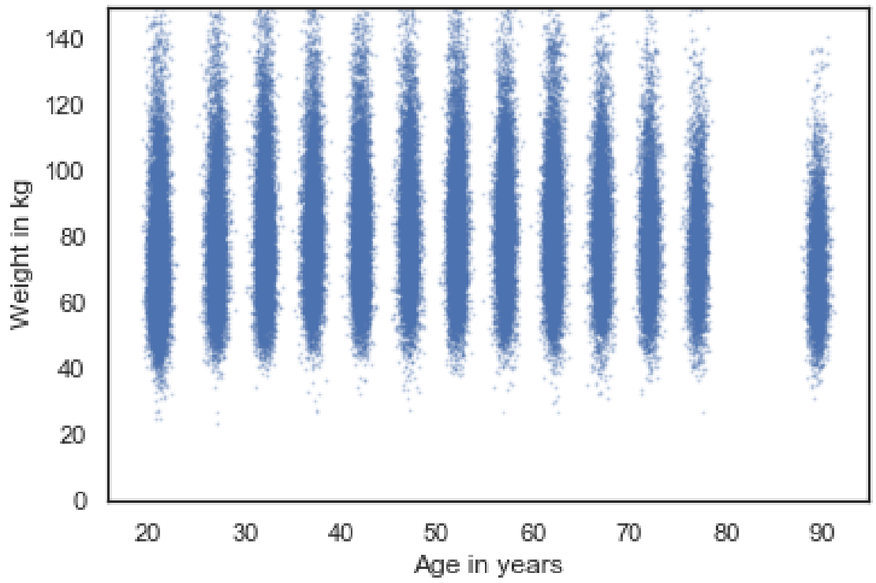
Weight and age

```
age = brfss['AGE'] + np.random.normal(0, 2.5, size=len(brfss))
weight = brfss['WTKG3']
plt.plot(age, weight, 'o', markersize=5, alpha=0.2)
plt.show()
```



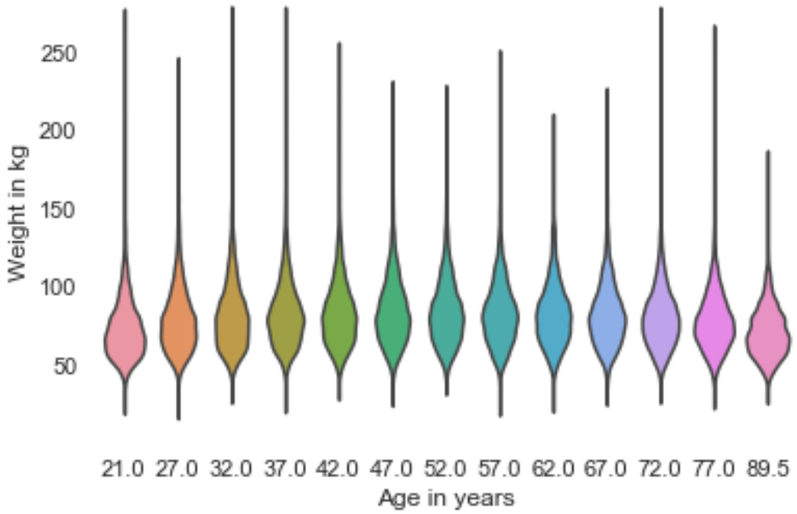
More data

```
age = brfss['AGE'] + np.random.normal(0, 0.5, size=len(brfss))
weight = brfss['WTKG3'] + np.random.normal(0, 2, size=len(brfss))
plt.plot(age, weight, 'o', markersize=1, alpha=0.2)
plt.show()
```



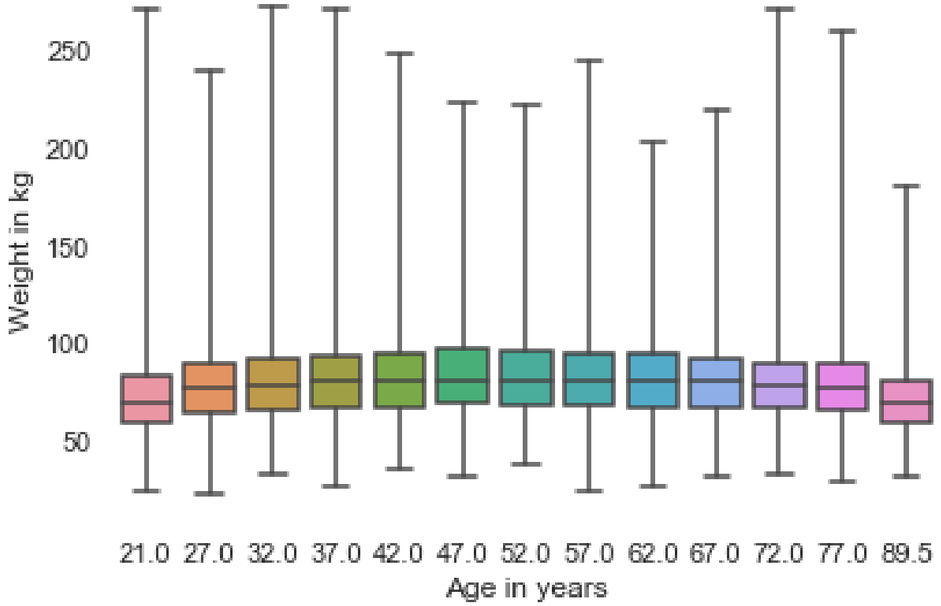
Violin plot

```
data = brfss.dropna(subset=['AGE', 'WTKG3'])

sns.violinplot(x='AGE', y='WTKG3', data=data, inner=None)
plt.show()
```



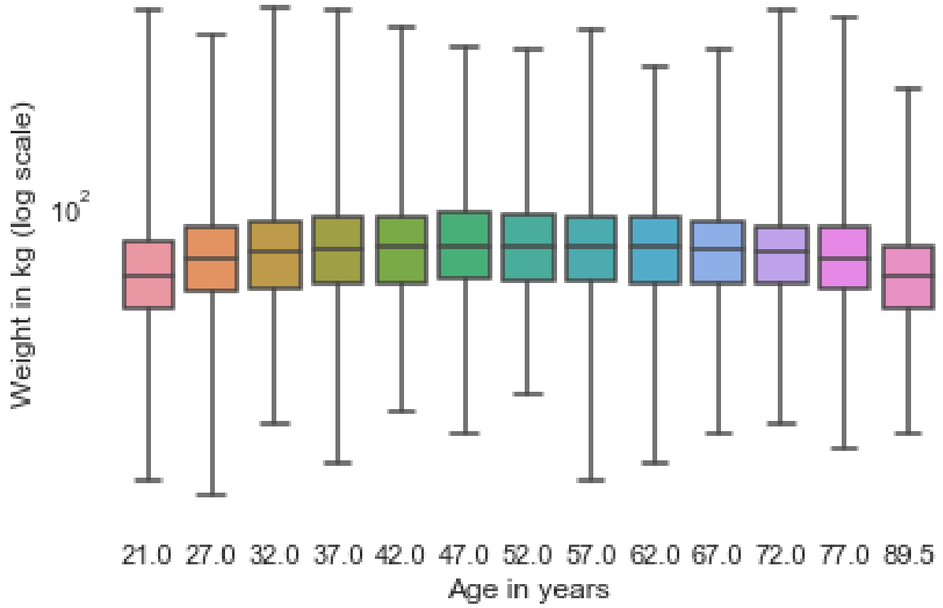
Box plot

```
sns.boxplot(x='AGE', y='WTKG3', data=data, whis=10)
plt.show()
```



Log scale

```
sns.boxplot(x='AGE', y='WTKG3', data=data, whis=10)
plt.yscale('log')
plt.show()
```



Let's practice!

EXPLORATORY DATA ANALYSIS IN PYTHON

Correlation

EXPLORATORY DATA ANALYSIS IN PYTHON

Allen Downey
Professor, Olin College

Correlation coefficient

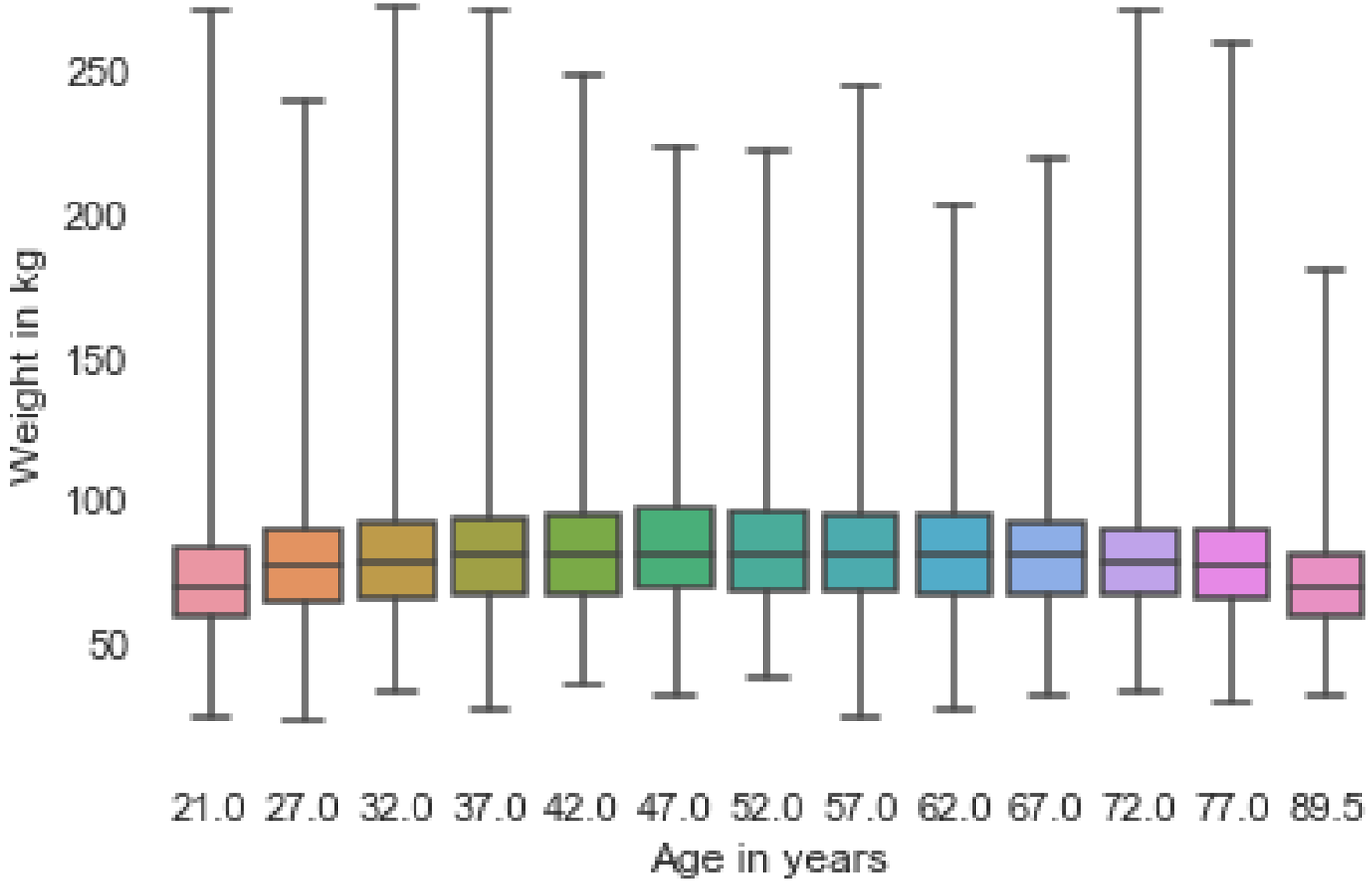
```
columns = ['HTM4', 'WTKG3', 'AGE']
subset = brfss[columns]
```

```
subset.corr()
```

Correlation matrix

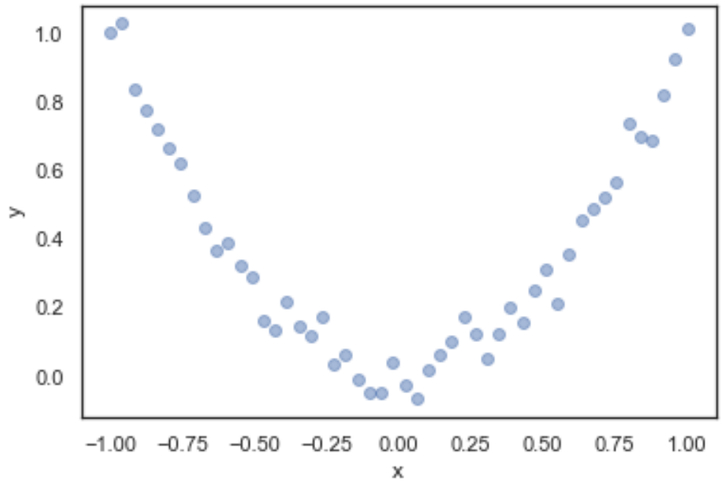
```
          HTM4          WTKG3          AGE
HTM4      1.000000      0.474203     -0.093684
WTKG3     0.474203      1.000000      0.021641
AGE      -0.093684      0.021641      1.000000
```

- Height with itself: 1
- Height and weight: 0.47
- Height and age: -0.09
- Weight and age: 0.02


250
200
150
100
50
Weight in kg
21.0
27.0
32.0
37.0
42.0
47.0
52.0
57.0
62.0
67.0
72.0
77.0
89.5
Age in years

```
xs = np.linspace(-1, 1)
ys = xs**2
ys += normal(0, 0.05, len(xs))
np.corrcoef(xs, ys)
```

```
array([[ 1.        , -0.01111647],
       [-0.01111647,  1.        ]])
```

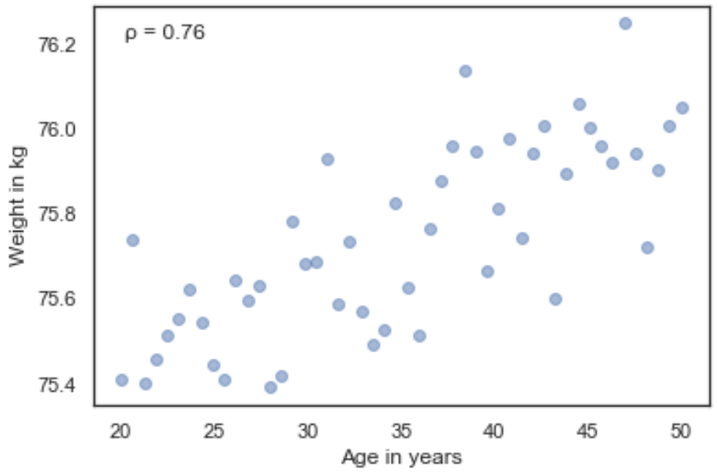


You keep using that word

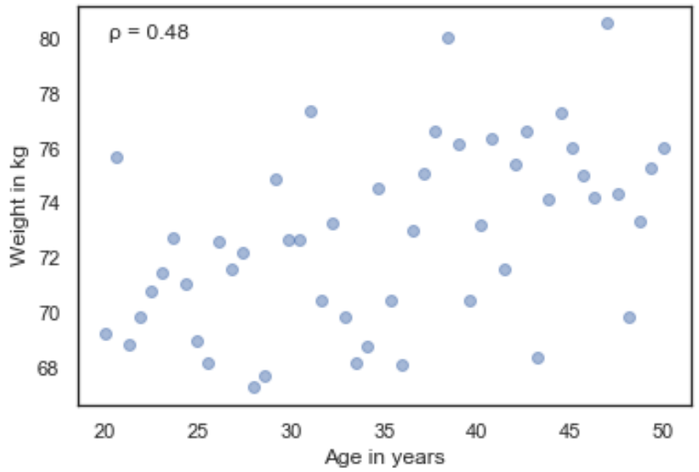
I do not think it means what you think it means

Strength of relationship

Hypothetical #1



Hypothetical #2



Let's practice!

EXPLORATORY DATA ANALYSIS IN PYTHON

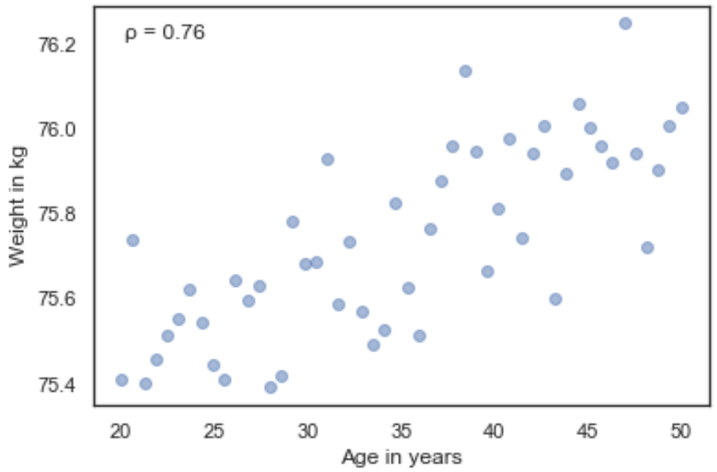
Simple regression

EXPLORATORY DATA ANALYSIS IN PYTHON

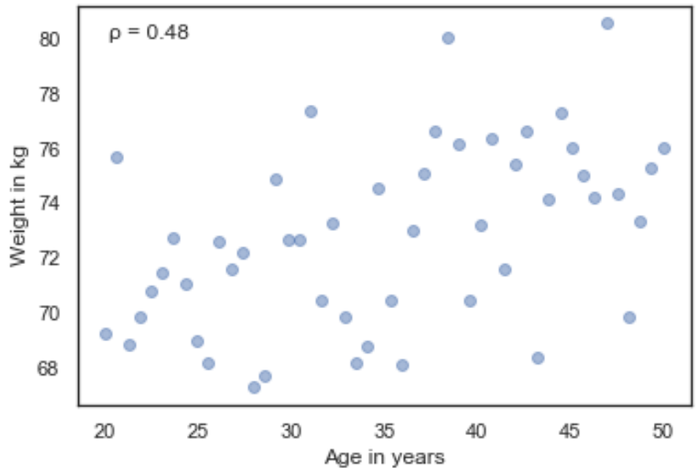
Allen Downey
Professor, Olin College

Strength of relationship

Hypothetical #1



Hypothetical #2



Strength of effect

```
from scipy.stats import linregress


# Hypothetical 1
res = linregress(xs, ys)
```

```
LinregressResult(slope=0.01882103490324438 6,
            intercept=75.08049023710964,
                rvalue=0.7579660563439402,
                pvalue=1.8470158725246148e-10,
                stderr=0.00233784926056081 8)
```

Strength of effect

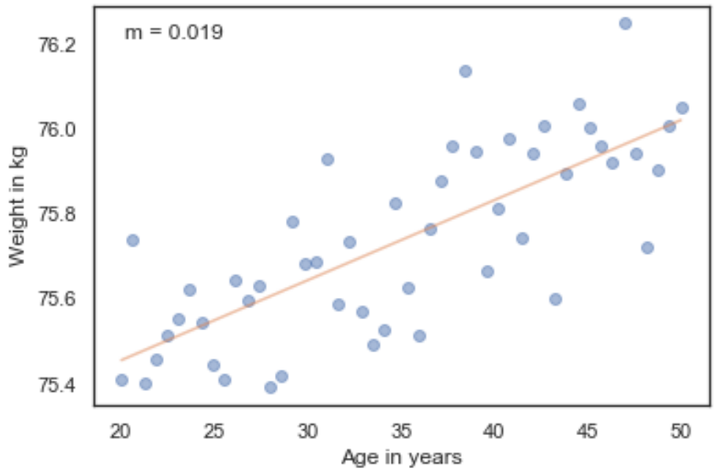
```
# Hypothetical 2
res = linregress(xs, ys)
```

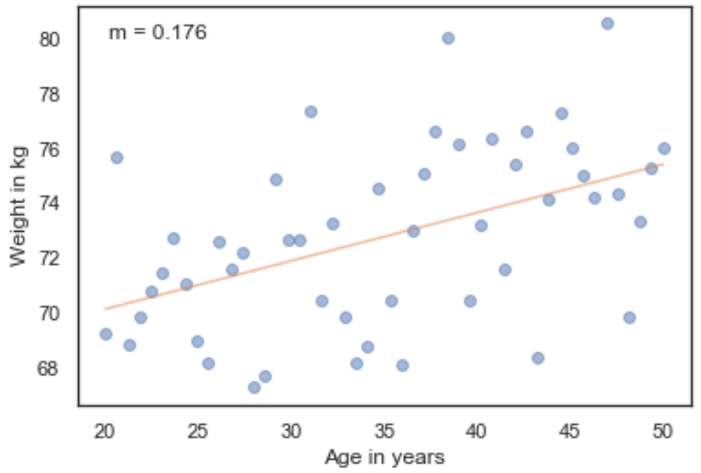
```
LinregressResult(slope=0.1764206980648885,
                 intercept=66.60980474219305,
                     rvalue=0.4782776976576317,
                     pvalue=0.00044306002837762 41,
                     stderr=0.0467569852112163 1)
```

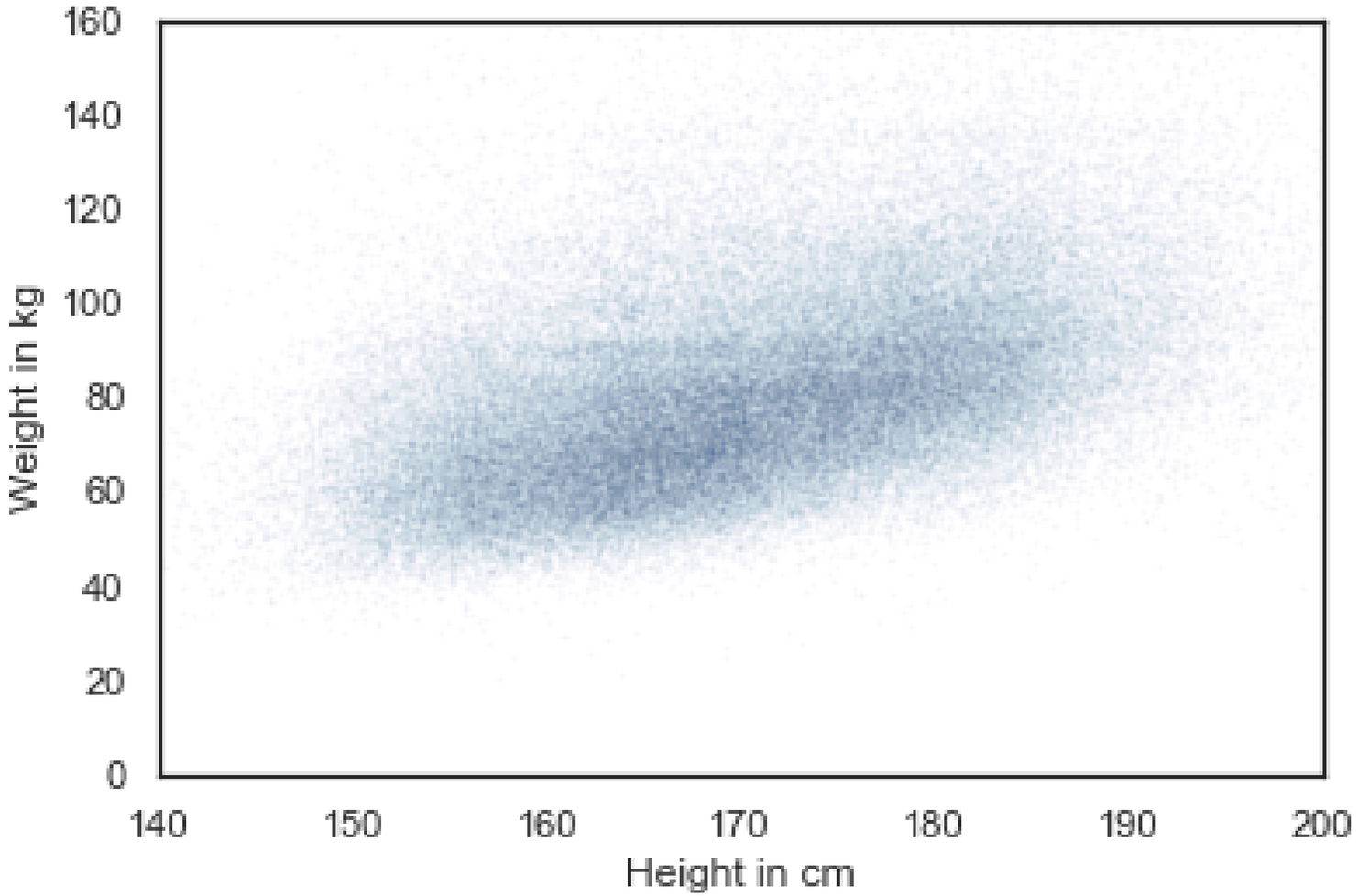
Regression lines

```
fx = np.array([xs.min(), xs.max()])
fy = res.intercept + res.slope * fx
plt.plot(fx, fy, '-')
```

```
fx = ...
fy = ...
plt.plot(fx, fy, '-')
```






160
140
120
100
80
60
40
20
0
Weight in kg
140
150
160
170
180
190
200
Height in cm

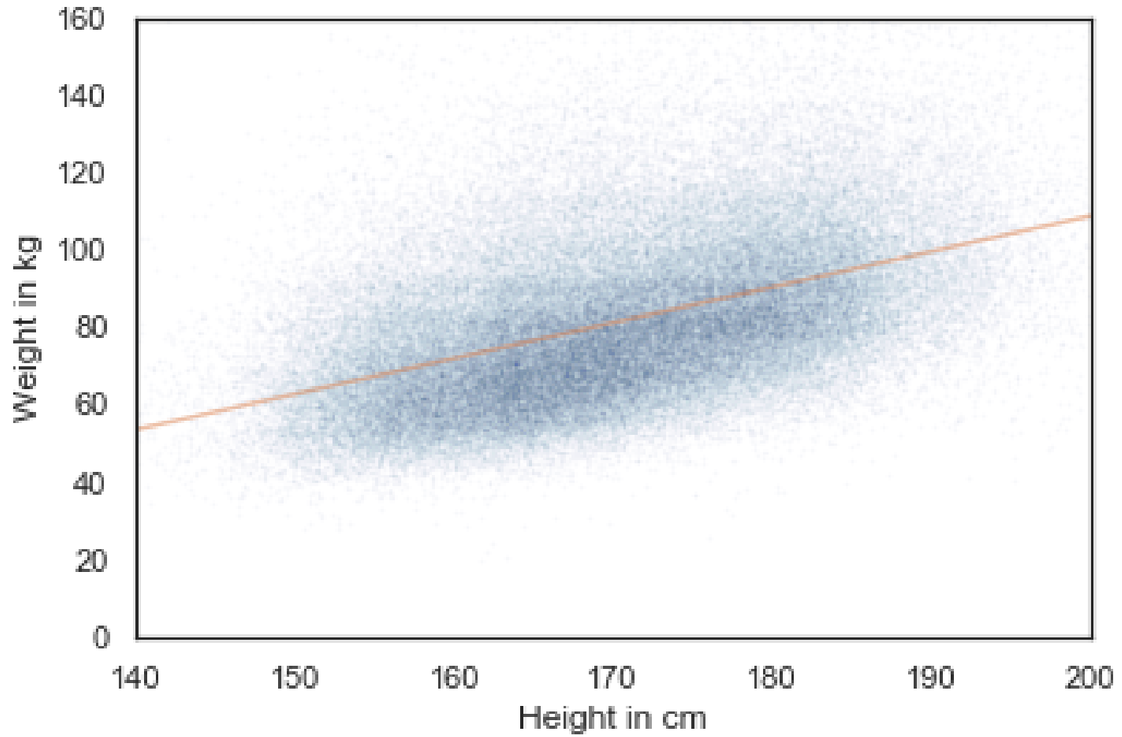
Regression line

```
subset = brfss.dropna(subset=['WTKG3', 'HTM4'])
```

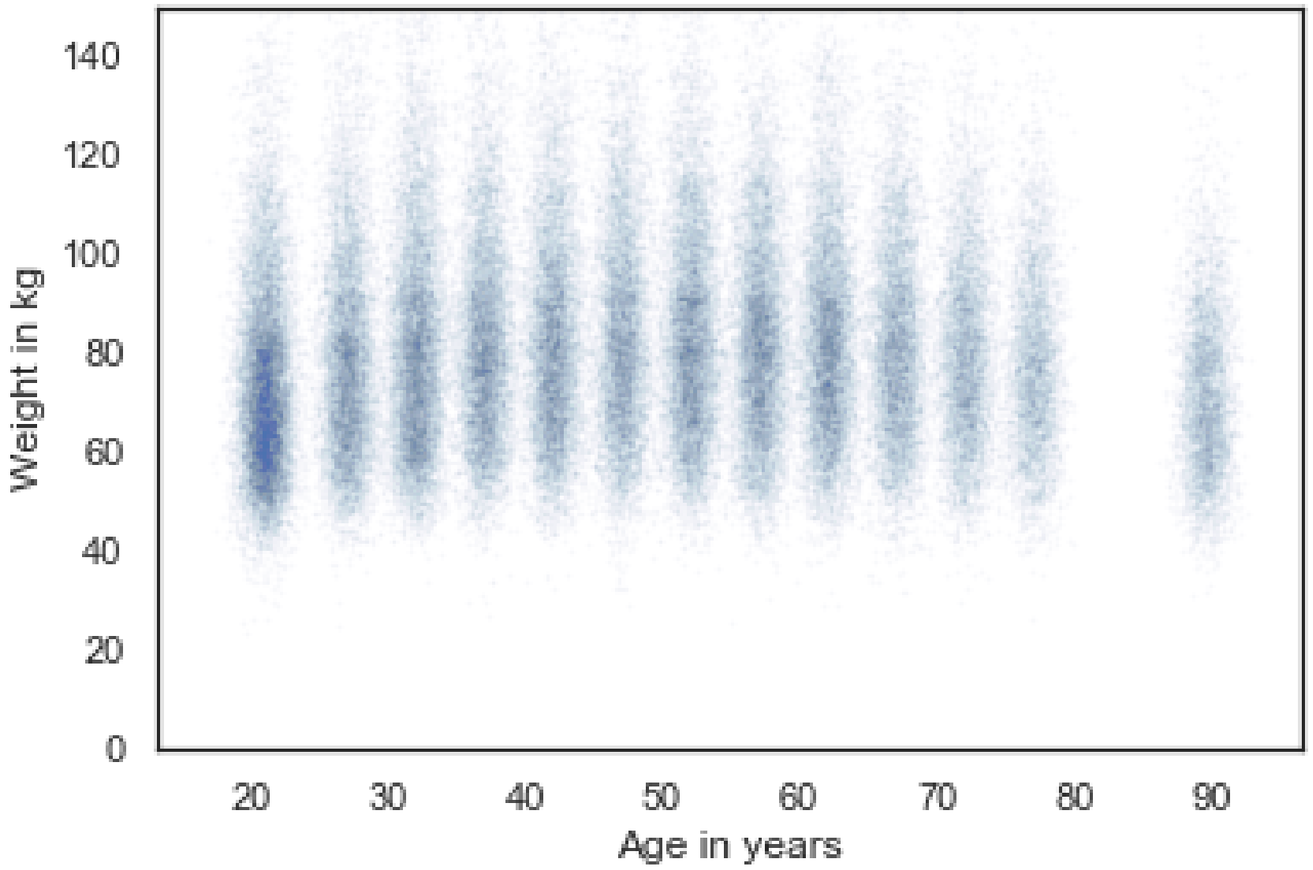
```
xs = subset['HTM4']
ys = subset['WTKG3']
res = linregress(xs, ys)
```

```
LinregressResult(slope=0.9192115381848297,
           intercept=-75.12704250330233,
                rvalue=0.4742030879024584,
                pvalue=0.0,
                stderr=0.005632863769802998)
```

```
fx = np.array([xs.min(), xs.max()])
fy = res.intercept + res.slope * fx
plt.plot(fx, fy, '-')
```



Linear relationships

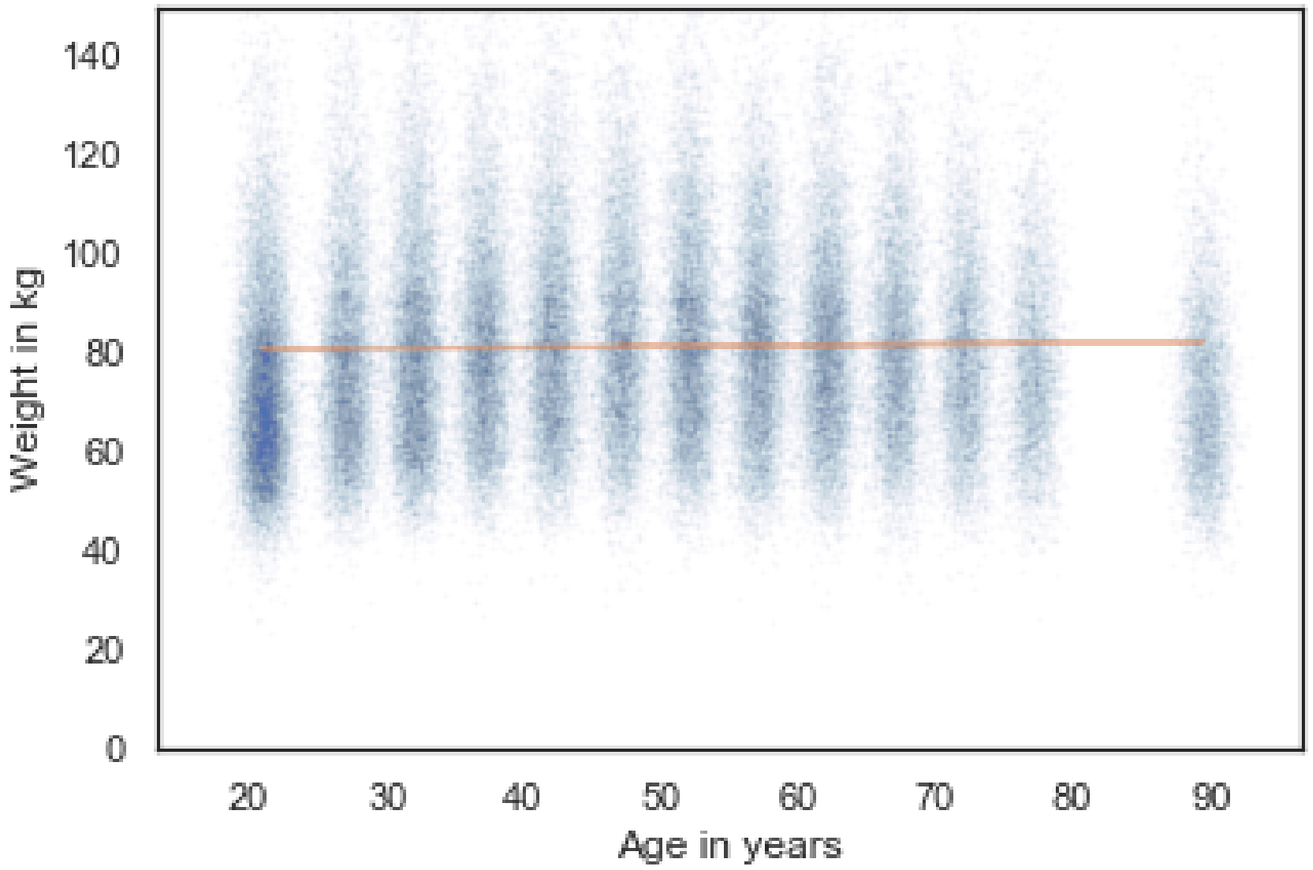


Nonlinear relationships

```
subset = brfss.dropna(subset=['WTKG3', 'AGE'])
xs = subset['AGE']
ys = subset['WTKG3']

res = linregress(xs, ys)
```

```
LinregressResult(slope=0.02398115956696872 4,
           intercept=80.07977583683224,
               rvalue=0.021641432889064068,
               pvalue=4.374327493007566e-11,
               stderr=0.003638139410742186)
```

Not a good fit



Let's practice!

EXPLORATORY DATA ANALYSIS IN PYTHON

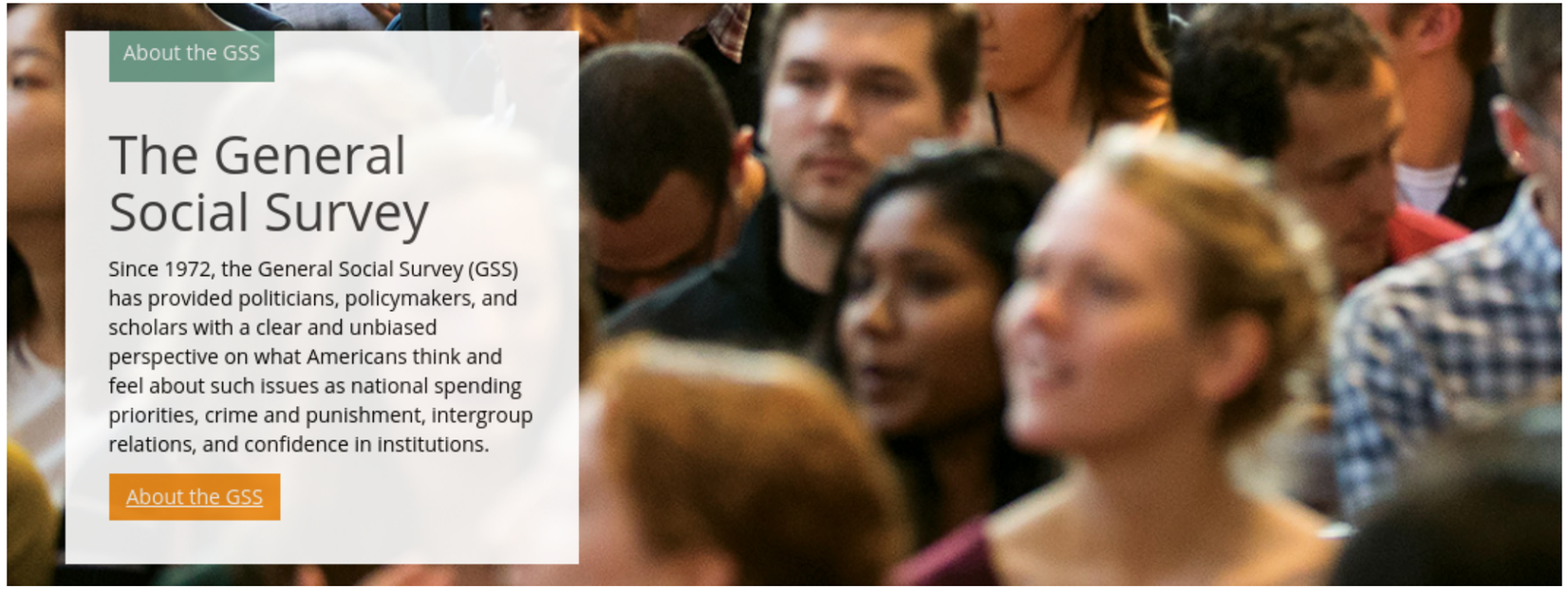
Probability mass functions

EXPLORATORY DATA ANALYSIS IN PYTHON

Allen Downey
Professor, Olin College

GSS

- Annual sample of U.S. population.
- Asks about demographics, social and political beliefs.
- Widely used by policy makers and researchers.

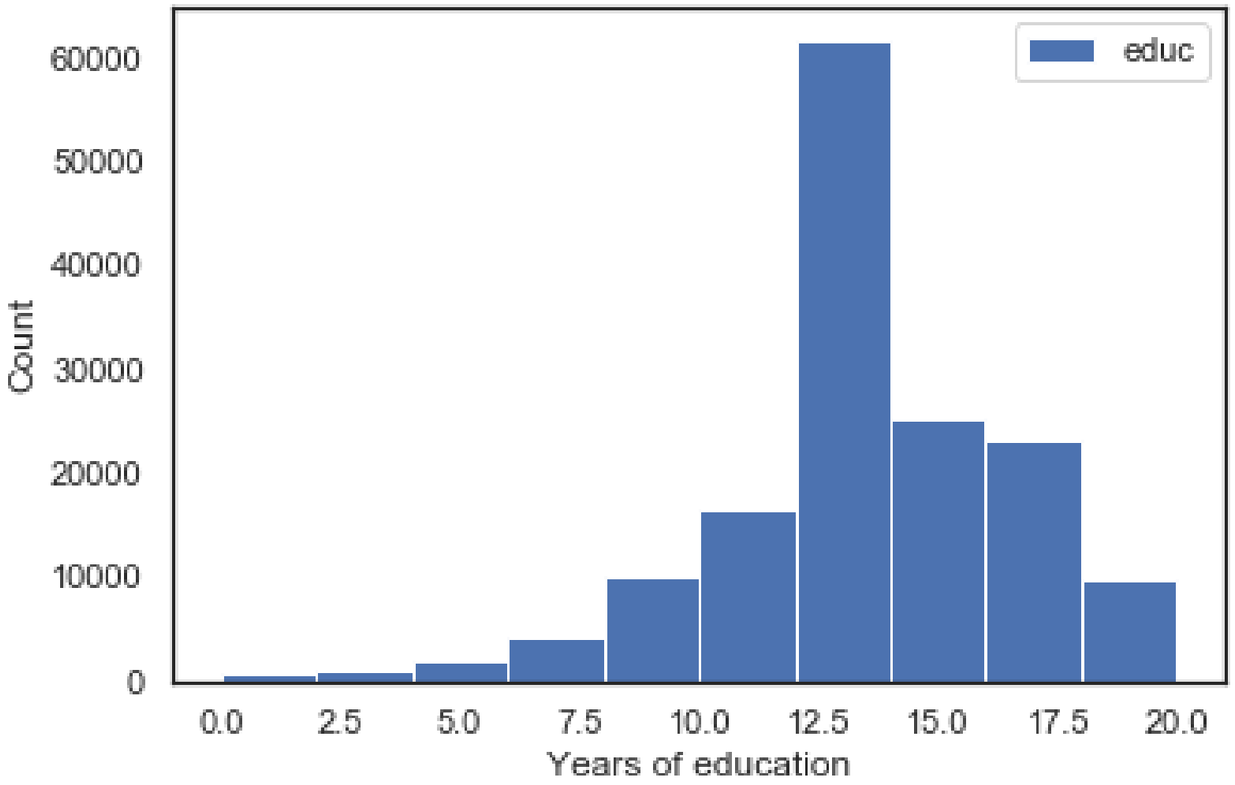


Read the data

```
gss = pd.read_hdf('gss.hdf5', 'gss')

gss.head()
```

	year	sex	age	cohort	race	educ	realinc	wtssall
0	1972	1	26.0	1946.0	1	18.0	13537.0	0.8893
1	1972	2	38.0	1934.0	1	12.0	18951.0	0.4446
2	1972	1	57.0	1915.0	1	12.0	30458.0	1.3339
3	1972	2	61.0	1911.0	1	14.0	37226.0	0.8893
4	1972	1	59.0	1913.0	1	12.0	30458.0	0.8893

```
educ = gss['educ']

plt.hist(educ.dropna(), label='educ')
plt.show()
```



PMF

```
pmf_educ = Pmf(educ, normalize=False)

pmf_educ.head()
```

```
0.0    566
1.0    118
2.0    292
3.0    686
4.0    746
Name: educ, dtype: int64
```

PMF

```
pmf_educ[12]
```

```
47689
```

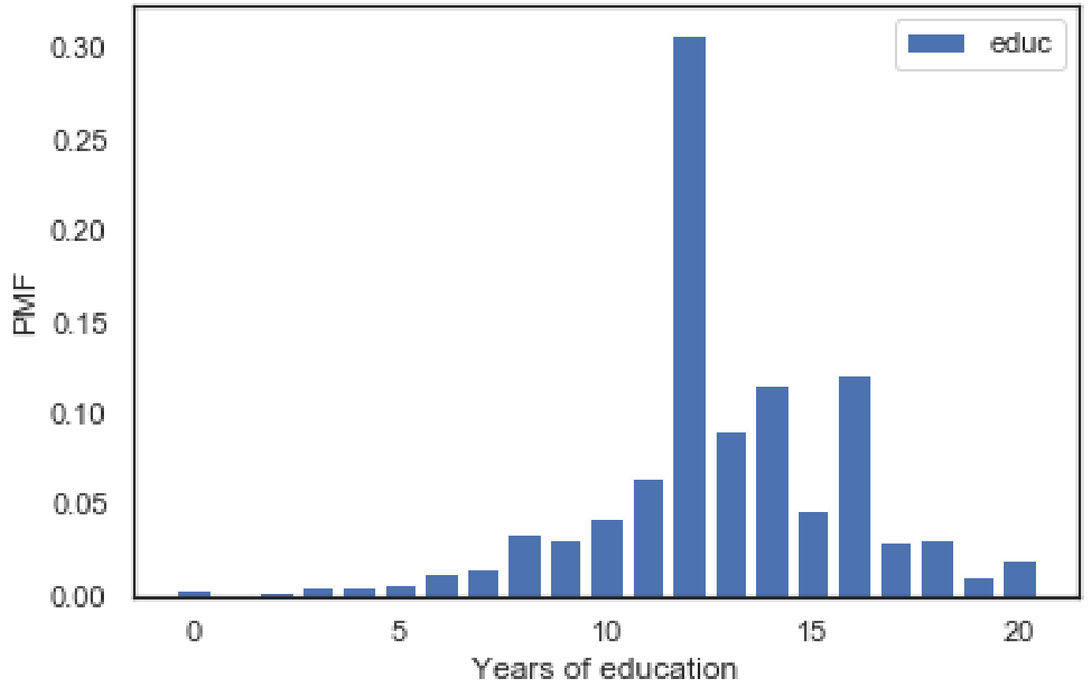
```
pmf_educ = Pmf(educ, normalize=True)

pmf_educ.head()
```

```
0.0    0.003663
1.0    0.000764
2.0    0.001890
3.0    0.004440
4.0    0.004828
Name: educ, dtype: int64
```

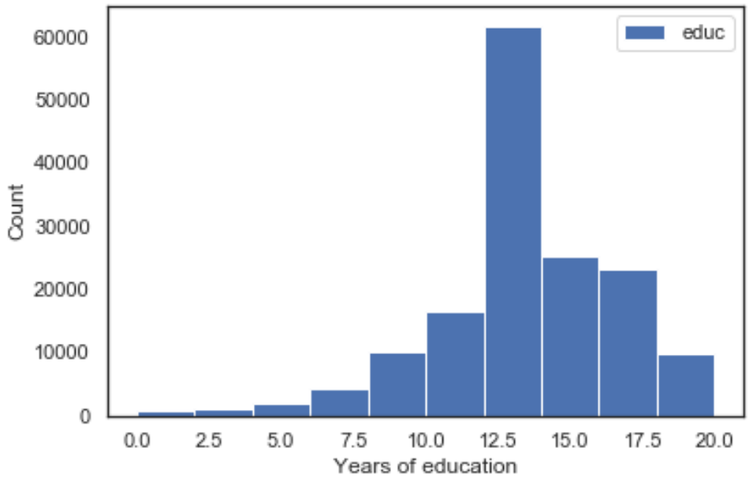
```
pmf_educ[12]
```

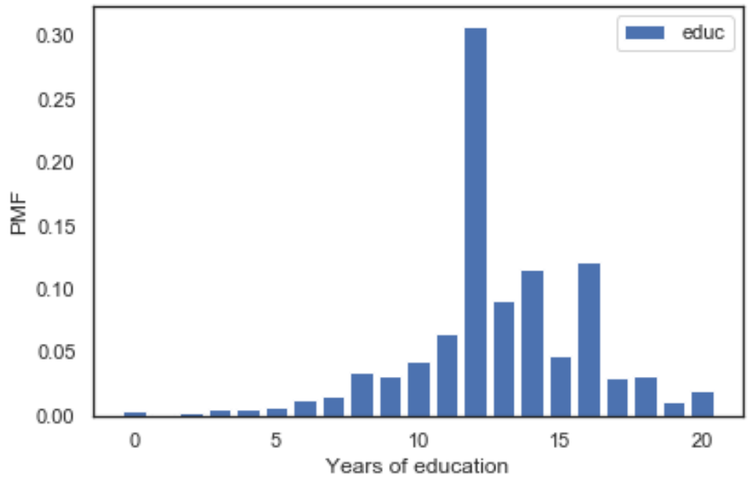
```
0.3086386994058790 7
```

```
pmf_educ.bar(label='educ')
plt.xlabel('Years of education')
plt.ylabel('PMF')
plt.show()
```



Histogram vs. PMF





Let's make some PMFs!

EXPLORATORY DATA ANALYSIS IN PYTHON

Cumulative distribution functions

EXPLORATORY DATA ANALYSIS IN PYTHON

Allen Downey
Professor, Olin College

From PMF to CDF

If you draw a random element from a distribution:

- PMF (Probability Mass Function) is the probability that you get exactly x
- CDF (Cumulative Distribution Function) is the probability that you get a value <= x

for a given value of x.

Example

PMF of {1, 2, 2, 3, 5}

PMF(1) = 1/5

PMF(2) = 2/5

PMF(3) = 1/5

PMF(5) = 1/5

CDF is the cumulative sum of the PMF.

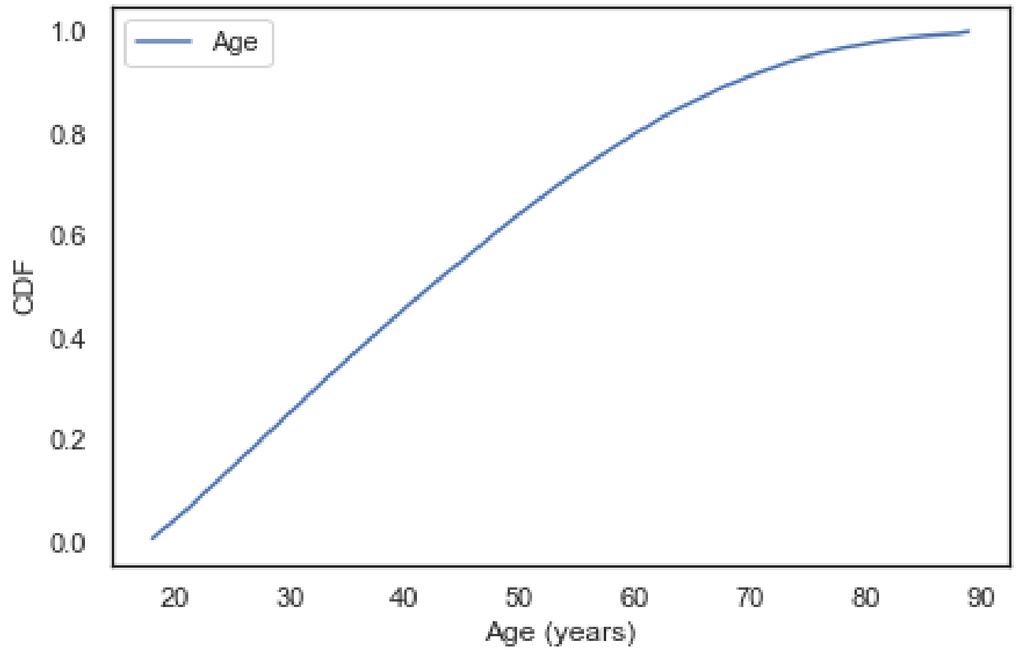
CDF(1) = 1/5

CDF(2) = 3/5

CDF(3) = 4/5

CDF(5) = 1

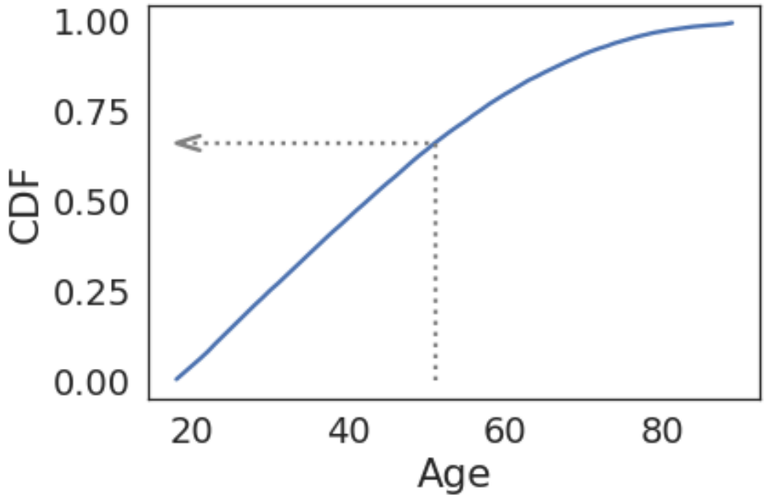
```
cdf = Cdf(gss['age'])
cdf.plot()
plt.xlabel('Age')
plt.ylabel('CDF')
plt.show()
```



Evaluating the CDF

```
q = 51
p = cdf(q)
print(p)
```

```
0.66
```



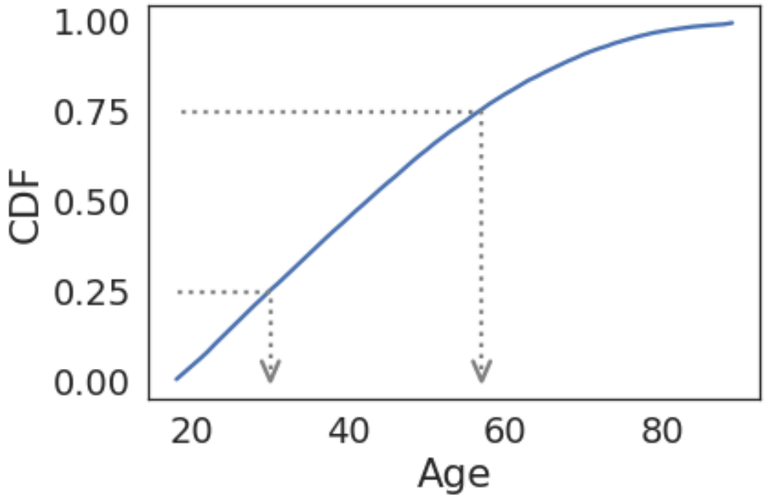
Evaluating the inverse CDF

```
p = 0.25
q = cdf.inverse(p)
print(q)
```

```
30
```

```
p = 0.75
q = cdf.inverse(p)
print(q)
```

```
57
```



Let's practice!

EXPLORATORY DATA ANALYSIS IN PYTHON

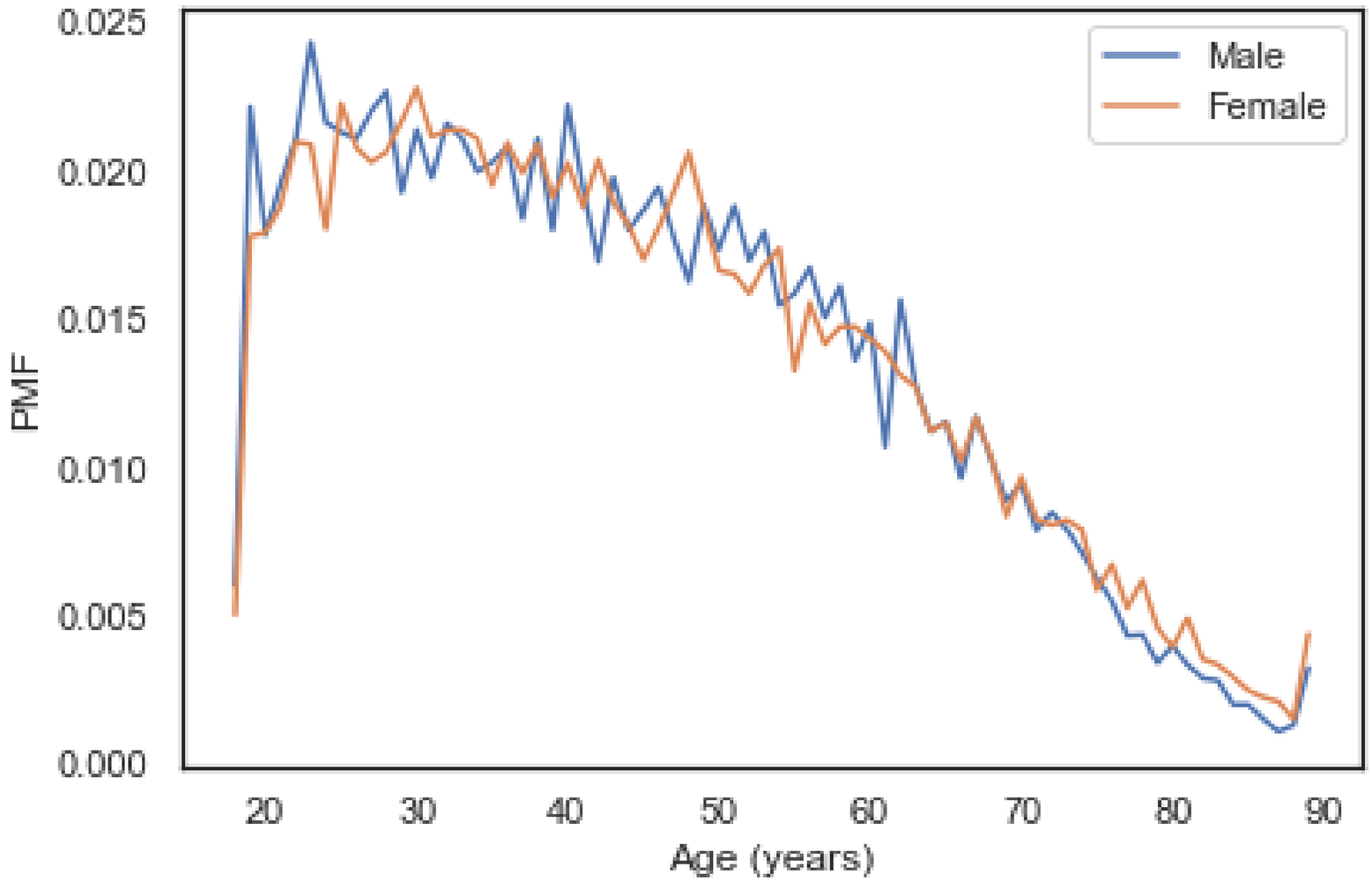
Comparing distributions

EXPLORATORY DATA ANALYSIS IN PYTHON

Allen Downey
Professor, Olin College

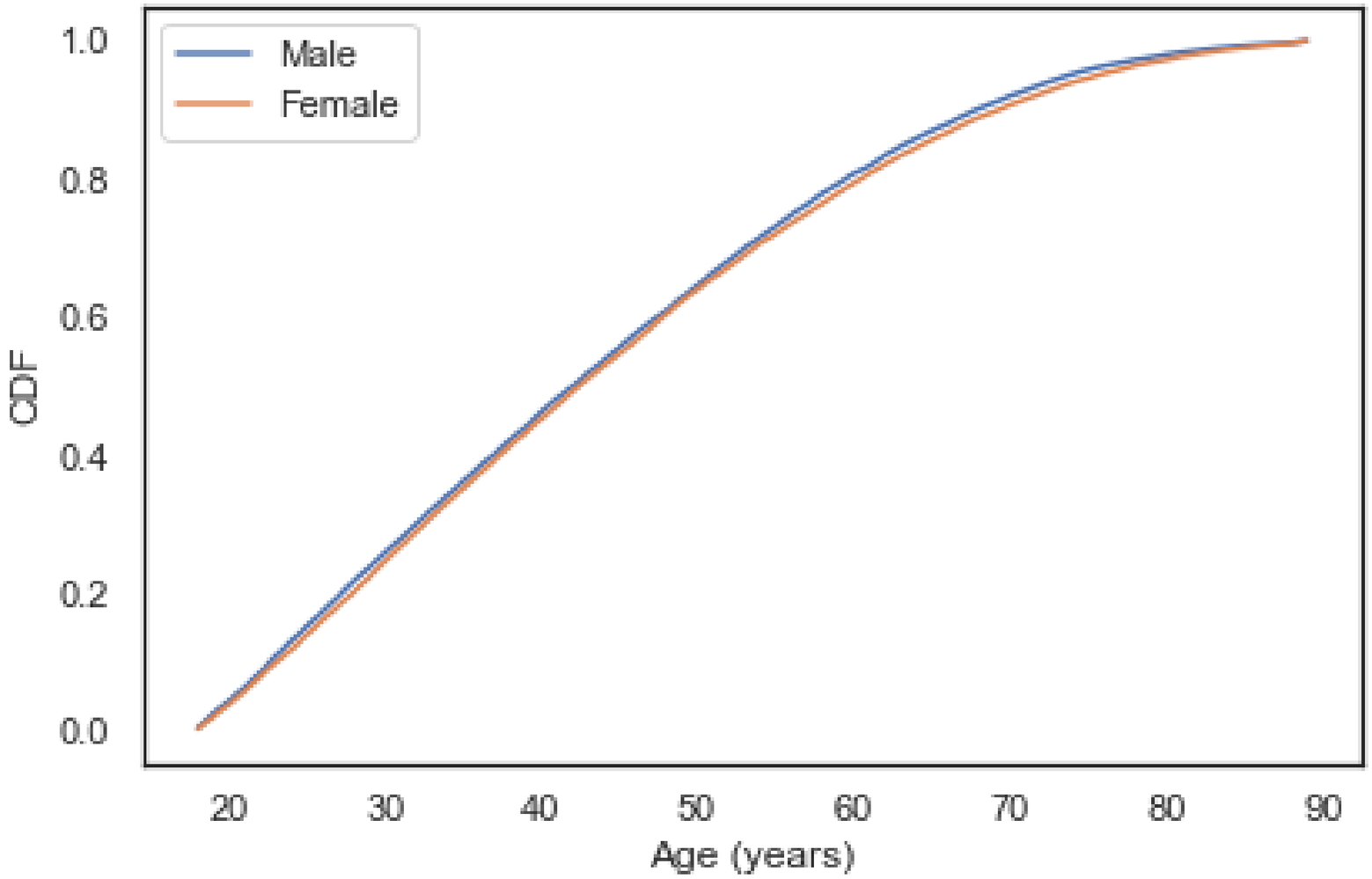
Multiple PMFs

```
male = gss['sex'] == 1

age = gss['age']

male_age = age[male]
female_age = age[~male]

Pmf(male_age).plot(label='Male')
Pmf(female_age).plot(label='Female')

plt.xlabel('Age (years)')
plt.ylabel('Count')
plt.show()
```


0.025
0.020
0.015
0.010
0.005
0.000
PMF
20
30
40
50
60
70
80
90
Age (years)
Male
Female

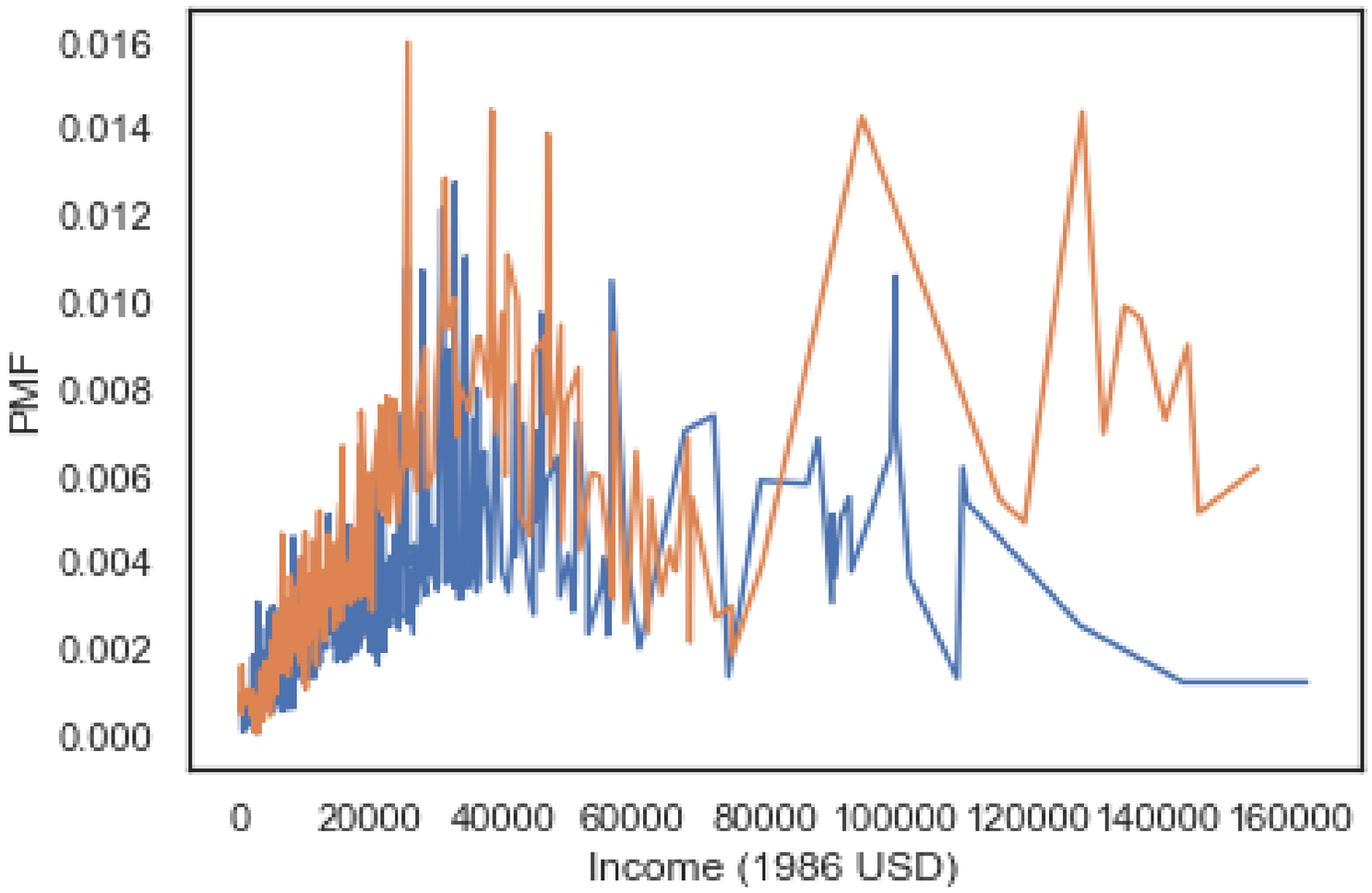
Multiple CDFs

```
Cdf(male_age).plot(label='Male')
Cdf(female_age).plot(label='Female')

plt.xlabel('Age (years)')
plt.ylabel('Count')
plt.show()
```


Male
Female
CDF
1.0
0.8
0.6
0.4
0.2
0.0
20
30
40
50
60
70
80
90
Age (years)

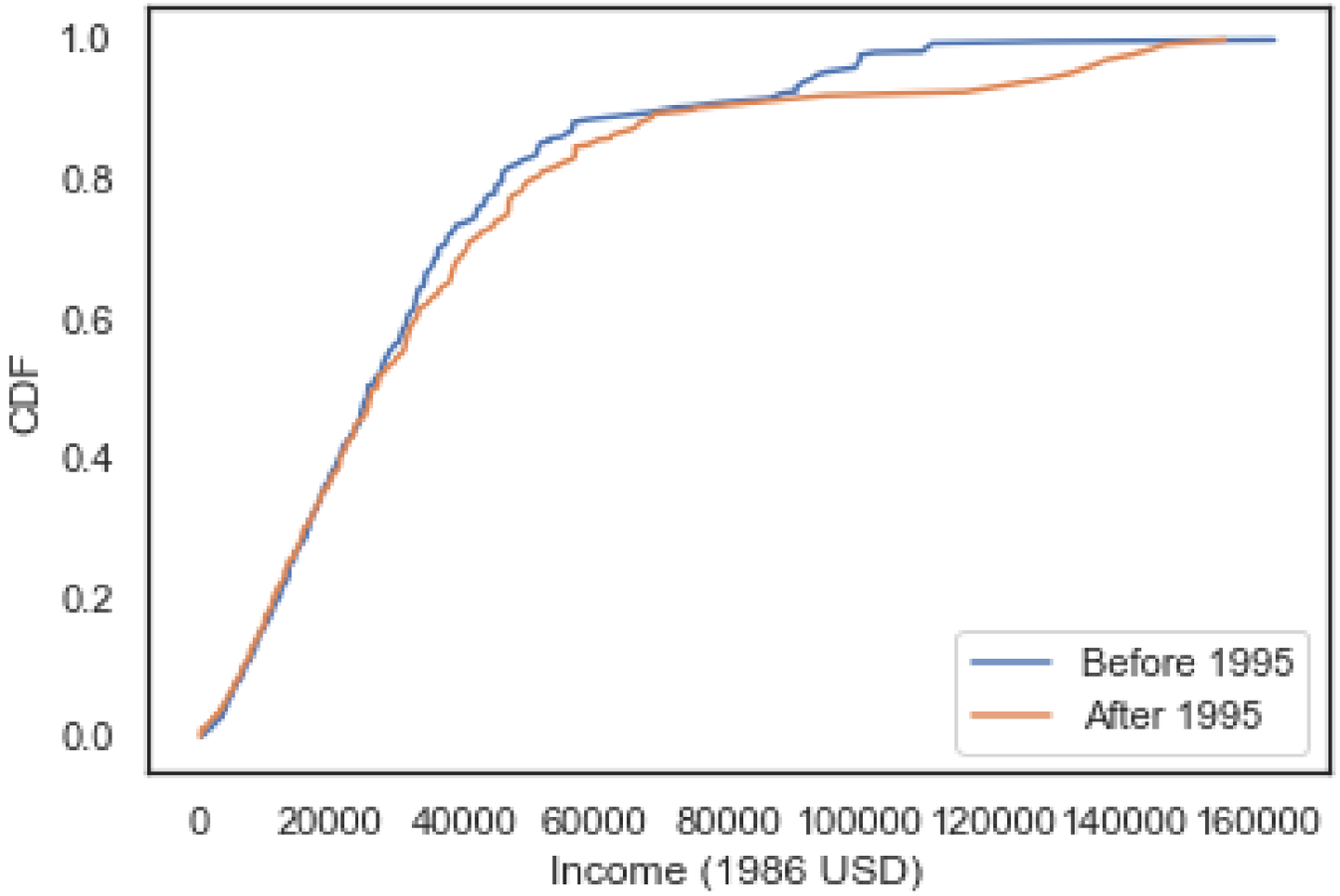
Income distribution

```
income = gss['realinc']

pre95 = gss['year'] < 1995

Pmf(income[pre95]).plot(label='Before 1995')
Pmf(income[~pre95]).plot(label='After 1995')

plt.xlabel('Income (1986 USD)')
plt.ylabel('PMF')
plt.show()
```


0.016
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
PMF
0
20000
40000
60000
80000
100000
120000
140000
160000
Income (1986 USD)

Income CDFs

```
Cdf(income[pre95]).plot(label='Before 1995')
Cdf(income[~pre95]).plot(label='After 1995')
```


1.0
0.8
0.6
0.4
0.2
0.0
CDF
0
20000
40000
60000
80000
100000
120000
140000
160000
Income (1986 USD)
Before 1995
After 1995

Let's practice!

EXPLORATORY DATA ANALYSIS IN PYTHON

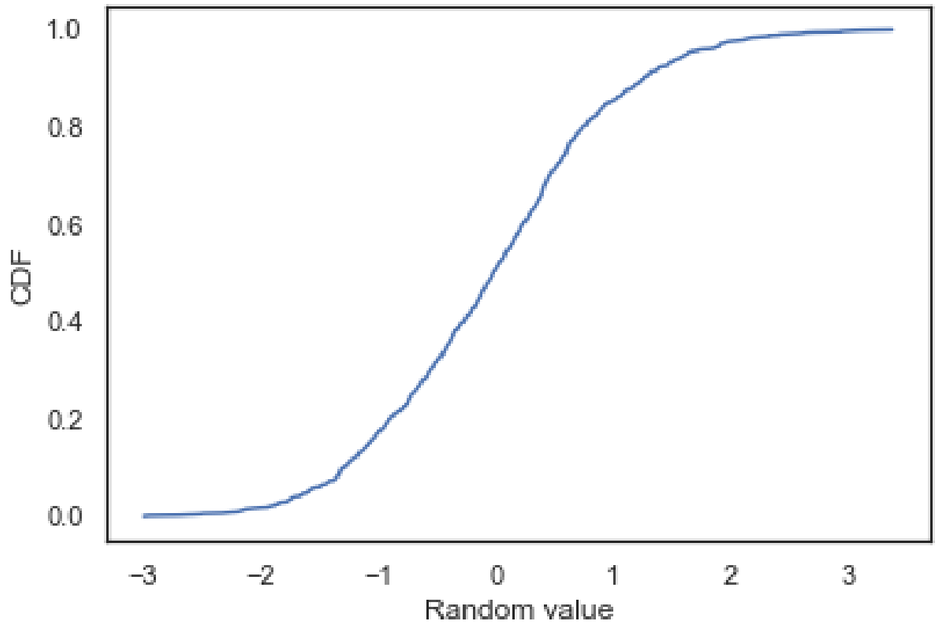
Modeling distributions

EXPLORATORY DATA ANALYSIS IN PYTHON

Allen Downey
Professor, Olin College

The normal distribution

```
sample = np.random.normal(size=1000)
Cdf(sample).plot()
```



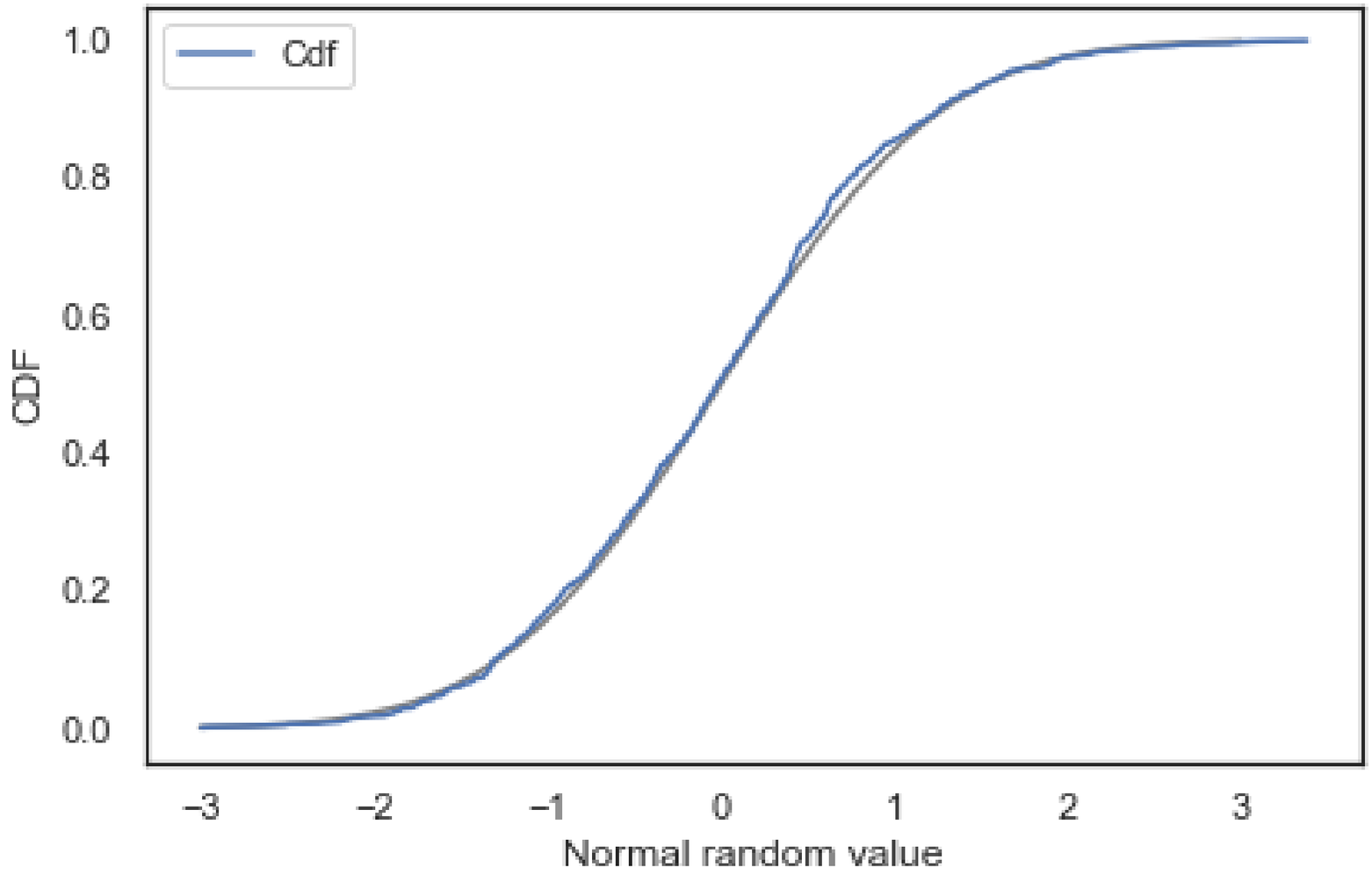
The normal CDF

```
from scipy.stats import norm
```

```
xs = np.linspace(-3, 3)
ys = norm(0, 1).cdf(xs)
```

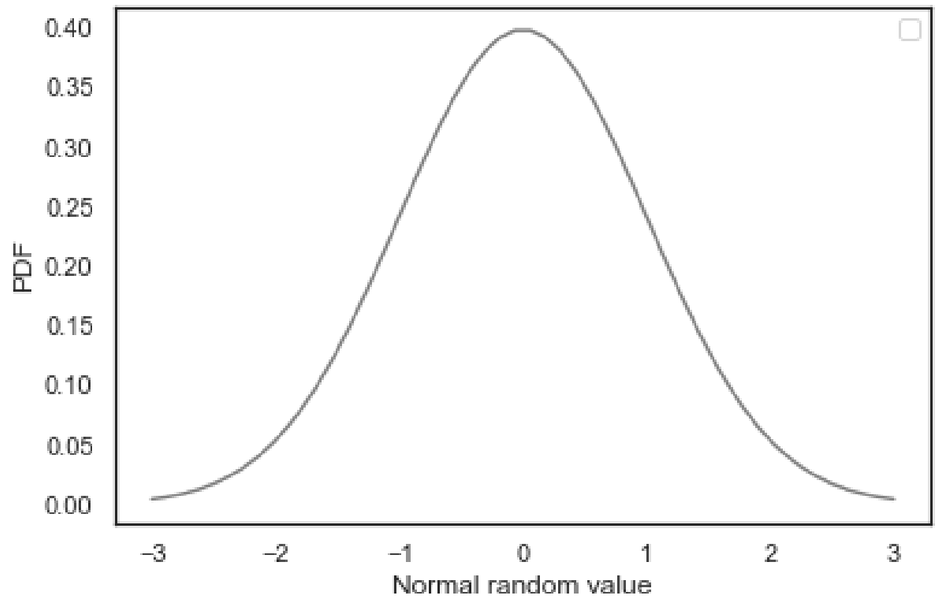
```
plt.plot(xs, ys, color='gray')
```

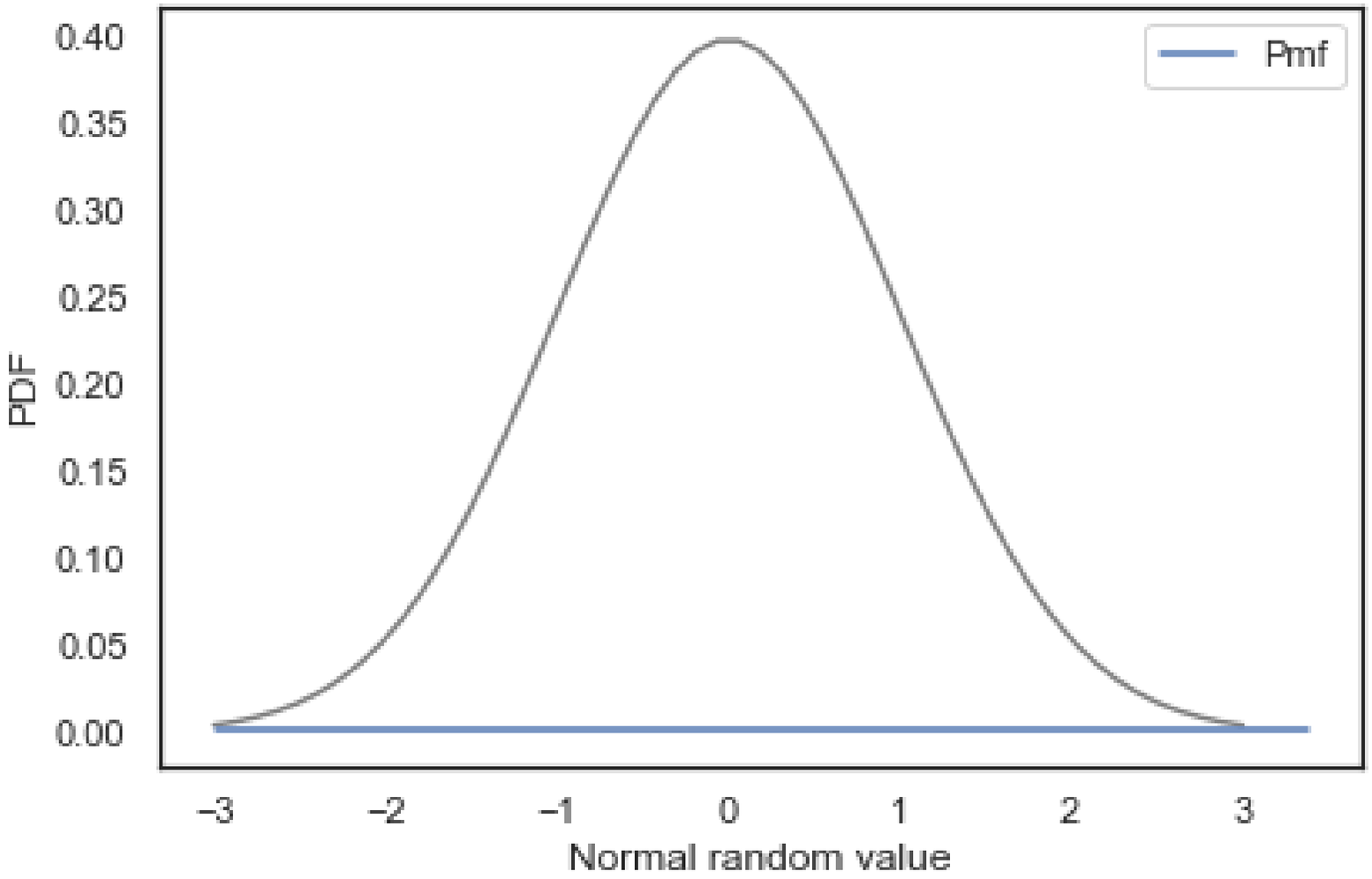
```
Cdf(sample).plot()
```


Cdf
1.0
0.8
0.6
0.4
0.2
0.0
CDF
−3
−2
−1
0
1
2
3
Normal random value

The bell curve

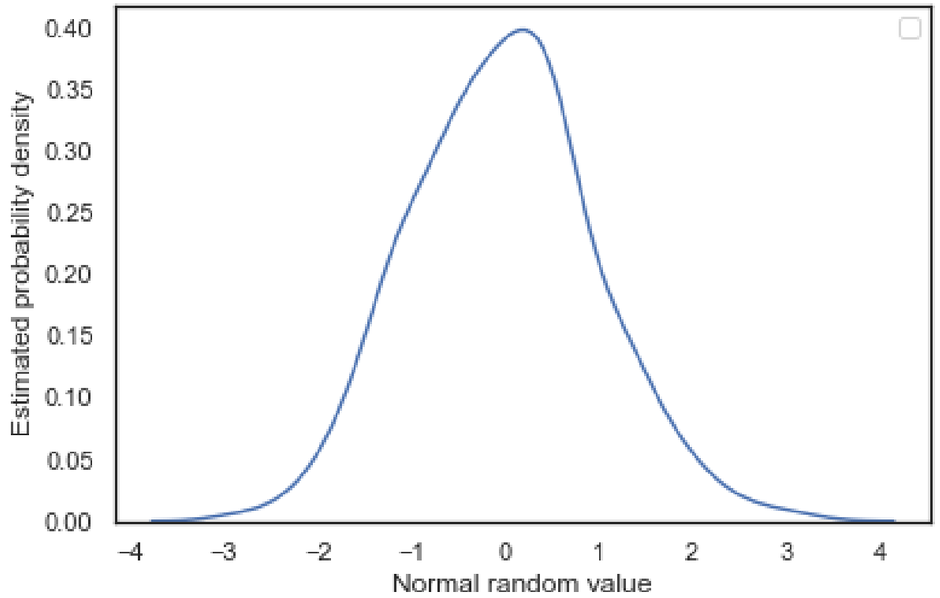
```
xs = np.linspace(-3, 3)
ys = norm(0,1).pdf(xs)
plt.plot(xs, ys, color='gray')
```




Pmf
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
PDF
−3
−2
−1
0
1
2
3
Normal random value

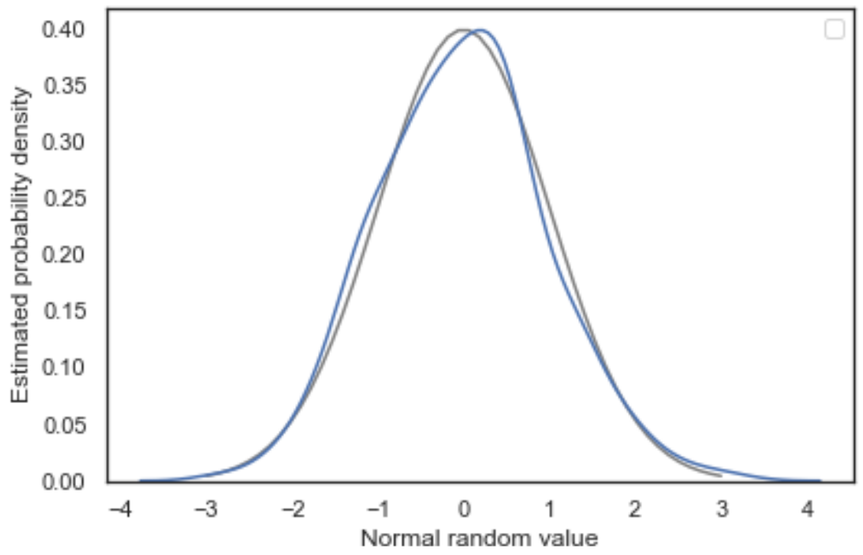
KDE plot

```
import seaborn as sns

sns.kdeplot(sample)
```



KDE and PDF

```
xs = np.linspace(-3, 3)
ys = norm.pdf(xs)
plt.plot(xs, ys, color='gray')
sns.kdeplot(sample)
```



PMF, CDF, KDE

- Use CDFs for exploration.
- Use PMFs if there are a small number of unique values.
- Use KDE if there are a lot of values.

Let's practice!

EXPLORATORY DATA ANALYSIS IN PYTHON

Dataframes and Series

EXPLORATORY DATA ANALYSIS IN PYTHON

Allen Downey
Professor, Olin College

Using data to answer questions

What is the average birth weight of babies in the United States?

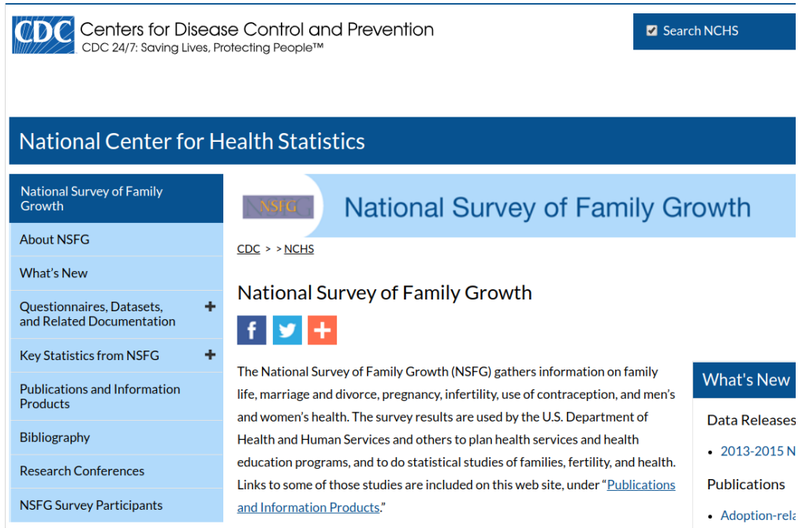
- Find appropriate data, or collect it
- Read data in your development environment
- Clean and validate

National Survey of Family Growth (NSFG)

NSFG data, from the National Center for Health Statistics

"nationally representative of women 15-44 years of age in the ... United States

"information on family life, marriage and divorce, pregnancy, infertility, use of contraception, and general and reproductive health."



Reading data

```
import pandas as pd

nsfg = pd.read_hdf('nsfg.hdf5', 'nsfg')

type(nsfg)
```

```
pandas.core.frame.DataFrame
```

Reading data

```
nsfg.head()
```

```
   caseid  outcome  birthwgt_lb1  birthwgt_oz1  prglngth  nbrnaliv  agecon  \
0   60418        1           5.0           4.0        40       1.0    2000
1   60418        1           4.0          12.0        36       1.0    2291
2   60418        1           5.0           4.0        36       1.0    3241
3   60419        6           NaN           NaN        33       NaN    3650
4   60420        1           8.0          13.0        41       1.0    2191

   agepreg  hpagelb  wgt2013_2015
0   2075.0     22.0   3554.964843
1   2358.0     25.0   3554.964843
2   3308.0     52.0   3554.964843
3      NaN      NaN   2484.535358
4   2266.0     24.0   2903.782914
```

Columns and rows

```
nsfg.shape
```

```
(9358, 10)
```

```
nsfg.columns
```

```
Index(['caseid', 'outcome', 'birthwgt_lb1', 'birthwgt_oz1', 'prglngt
       'nbrnaliv', 'agecon', 'agepreg', 'hpagelb', 'wgt2013_2015'],
      dtype='object')
```

Columns and rows

BIRTHWGT_LB1 (46-47)

Variable Type : raw

BD-3 : How much did (BABY'S NAME/this 1st baby) weigh at birth? (POUNDS)

value	label	Total
.	INAPPLICABLE	2873
0-5	UNDER 6 POUNDS	936
6	6 POUNDS	1666
7	7 POUNDS	2146
8	8 POUNDS	1168
9-95	9 POUNDS OR MORE	474
98	Refused	1
99	Don't know	94
	Total	**9358**

Each column is a Series

```
pounds = nsfg['birthwgt_lb1']

type(pounds)
```

```
pandas.core.series.Series
```

Each column is a series

```
pounds.head()
```

```
0    5.0
1    4.0
2    5.0
3    NaN
4    8.0
Name: birthwgt_lb1, dtype: float64
```

Let's start exploring!

EXPLORATORY DATA ANALYSIS IN PYTHON

Clean and Validate

EXPLORATORY DATA ANALYSIS IN PYTHON

Allen Downey
Professor, Olin College

Selecting columns

```
pounds = nsfg['birthwgt_lb1']
```

```
ounces = nsfg['birthwgt_oz1']
```

```
pounds.value_counts().sort_index()
```

```
0.0        6
1.0       34
2.0       47
3.0       67
4.0      196
5.0      586
6.0     1666
7.0     2146
8.0     1168
9.0      363
10.0      82
11.0      17
12.0       7
13.0       2
14.0       2
17.0       1
98.0       1
99.0      94
Name: birthwgt_lb1, dtype: int64
```

BIRTHWGT_LB1 (46-47)

Variable Type : raw

BD-3 : How much did (BABY'S NAME/this 1st baby) weigh at birth? (POUNDS)

value	label	Total
.	INAPPLICABLE	2873
0-5	UNDER 6 POUNDS	936
6	6 POUNDS	1666
7	7 POUNDS	2146
8	8 POUNDS	1168
9-95	9 POUNDS OR MORE	474
98	Refused	1
99	Don't know	94
	Total	**9358**

Describe

```
pounds.describe()
```

```
count    6485.000000
mean        8.055204
std        11.178893
min         0.000000
25%         6.000000
50%         7.000000
75%         8.000000
max        99.000000
Name: birthwgt_lb1, dtype: float64
```

Replace

```
pounds = pounds.replace([98, 99], np.nan)

pounds.mean()
```

```
6.703286384976526
```

```
ounces.replace([98, 99], np.nan, inplace=True)
```

Arithmetic with Series

```
birth_weight = pounds + ounces / 16.0

birth_weight.describe()
```

```
count    6355.000000
mean        7.120978
std         1.422236
min         0.000000
25%         6.375000
50%         7.187500
75%         8.000000
max        17.937500
dtype: float64
```

Let's practice!

EXPLORATORY DATA ANALYSIS IN PYTHON

Filter and Visualize

EXPLORATORY DATA ANALYSIS IN PYTHON

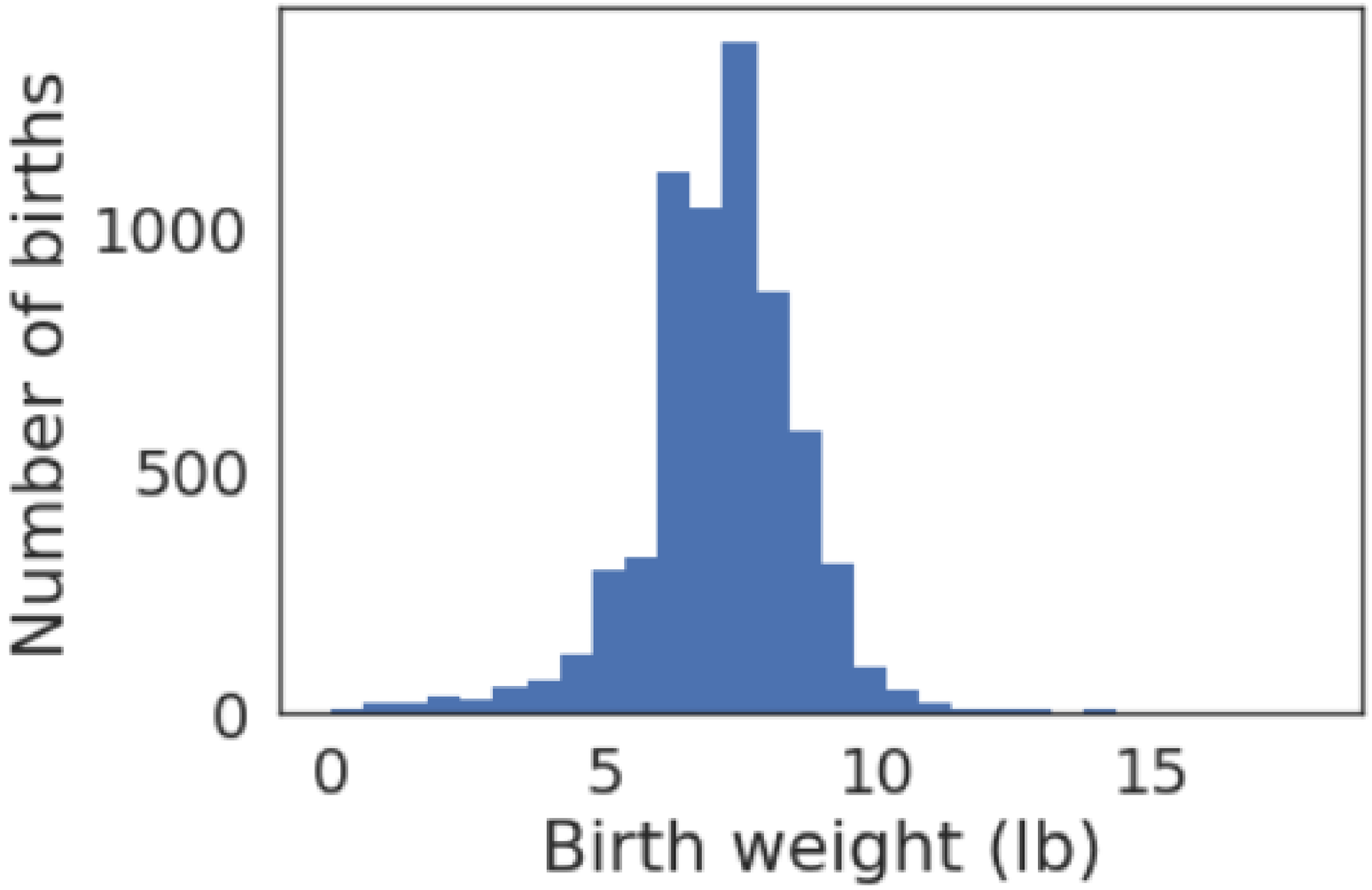
Allen Downey
Professor, Olin College

Histogram

```
import matplotlib.pyplot as plt
```

```
plt.hist(birth_weight.dropna(), bins=30)
```

```
plt.xlabel('Birth weight (lb)')
plt.ylabel('Fraction of births')
plt.show()
```


Number of births
1000
500
0
0
5
10
15
Birth weight (lb)

Boolean Series

```
preterm = nsfg['prglngth'] < 37

preterm.head()
```

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

Boolean Series

```
preterm.sum()
```

```
3742
```

```
preterm.mean()
```

```
0.39987176747168196
```

Filtering

```
preterm_weight = birth_weight[preterm]
preterm_weight.mean()
```

```
5.577598314606742
```

```
full_term_weight = birth_weight[~preterm]
full_term_weight.mean()
```

```
7.372323879231473
```

Filtering

Other logical operators:

- `&` for AND (both must be true)
- `|` for OR (either or both can be true)

Example:

```
birth_weight[A & B]    # both true
birth_weight[A | B]    # either or both true
```

Resampling

- NSFG is not representative
- Some groups are "oversampled"
- We can correct using `resample_rows_weighted()`

Finish it off!

EXPLORATORY DATA ANALYSIS IN PYTHON