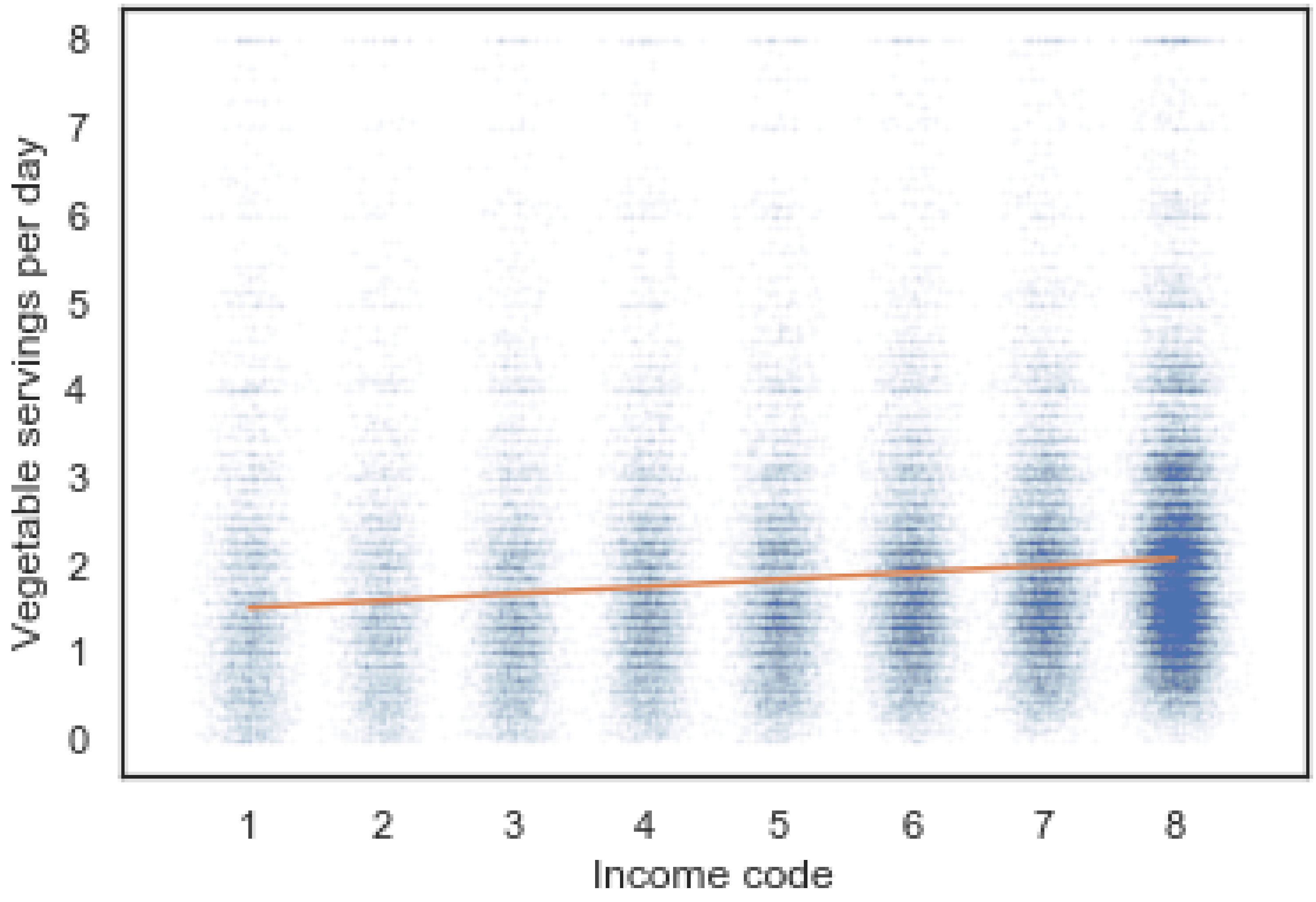
Limits of simple regression

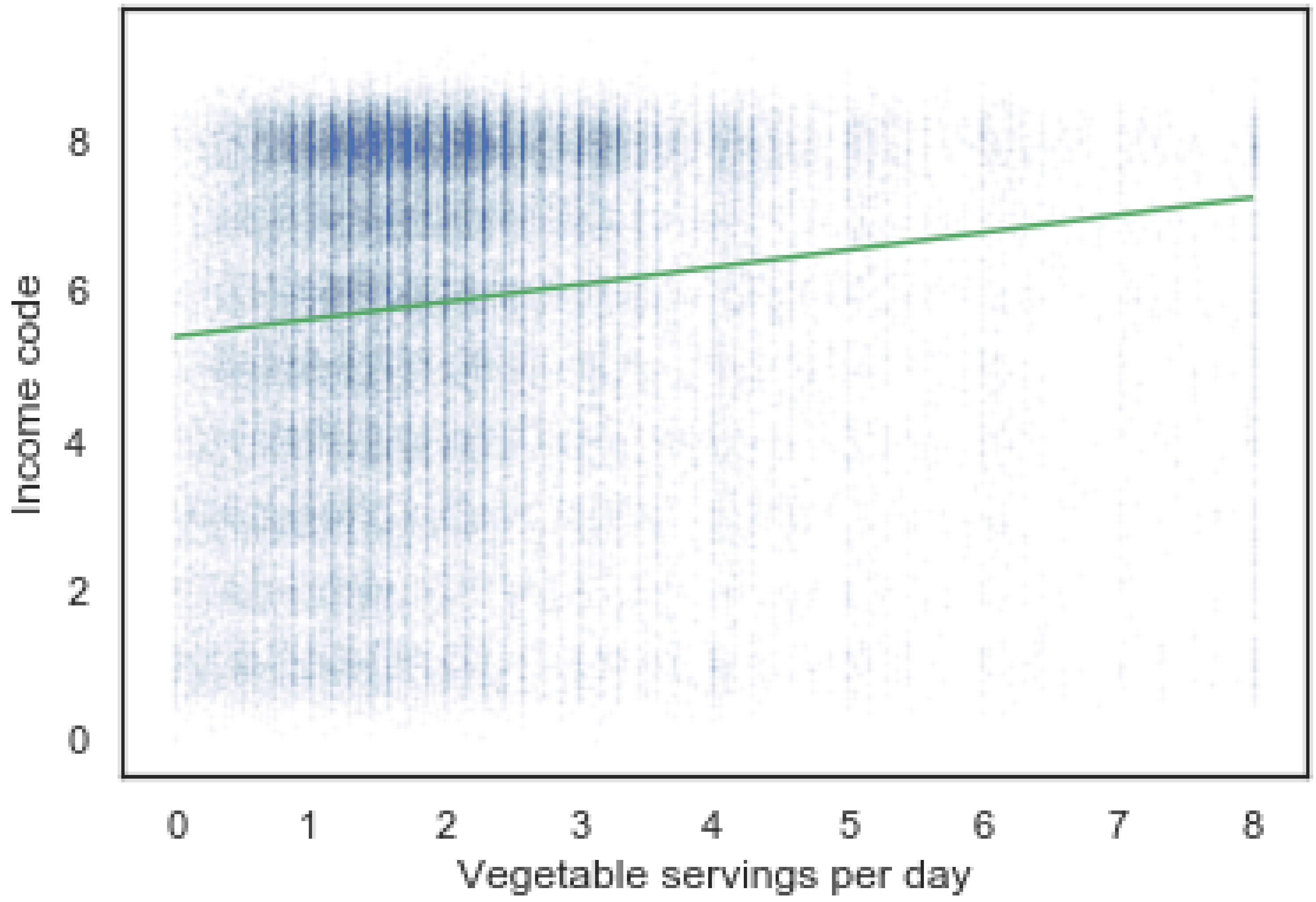
EXPLORATORY DATA ANALYSIS IN PYTHON

Allen Downey
Professor, Olin College

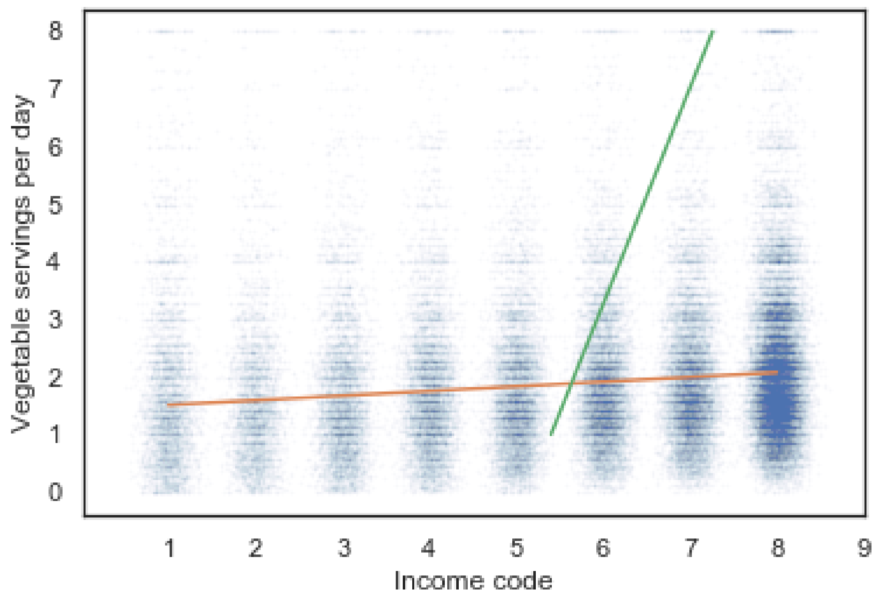
Income and vegetables

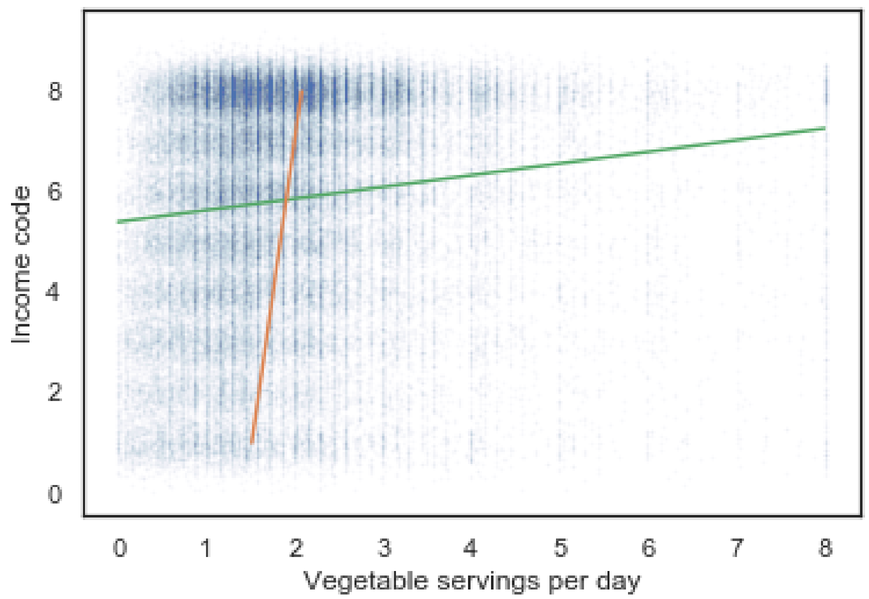


Vegetables and income

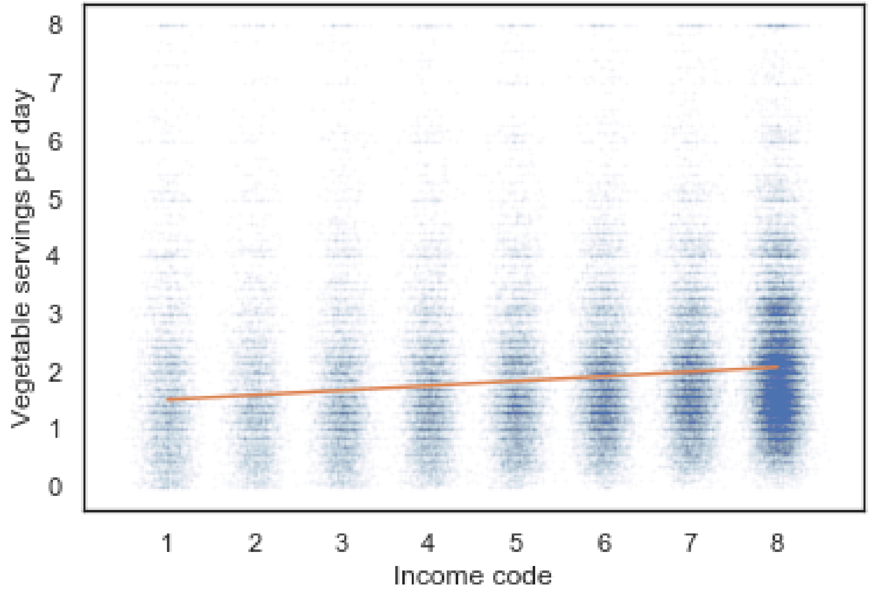


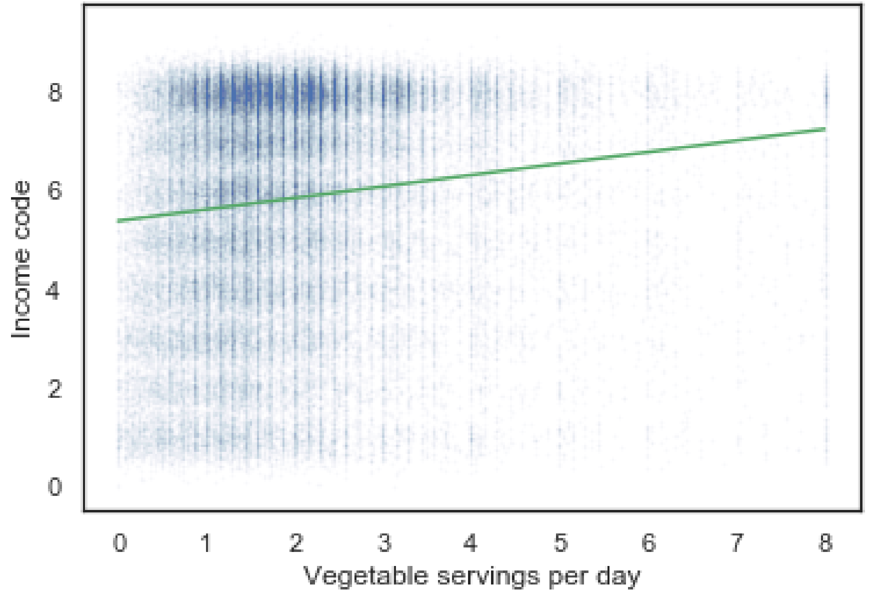
Regression is not symmetric





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
```

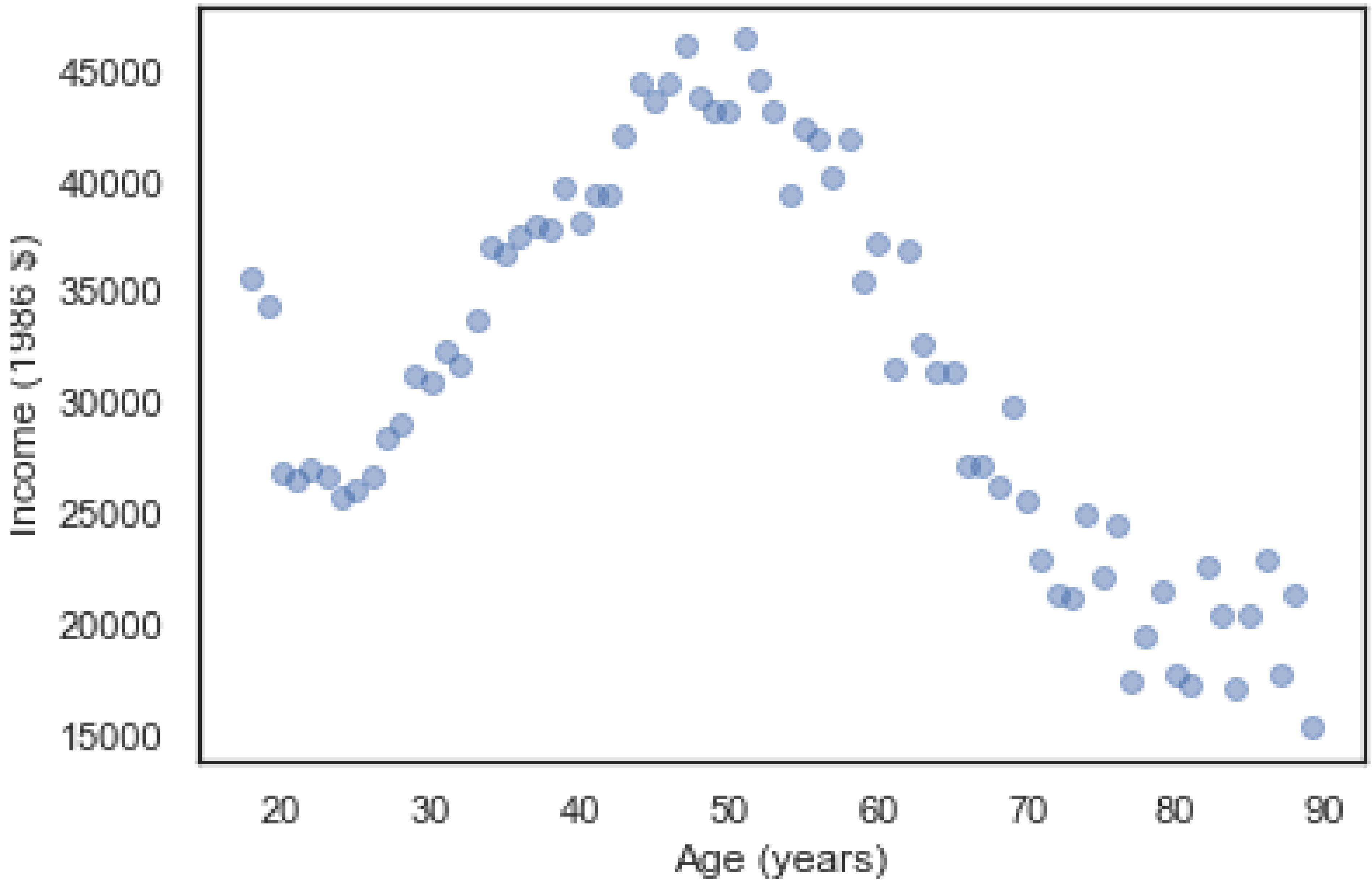
Income and age

```
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=gss)
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
```

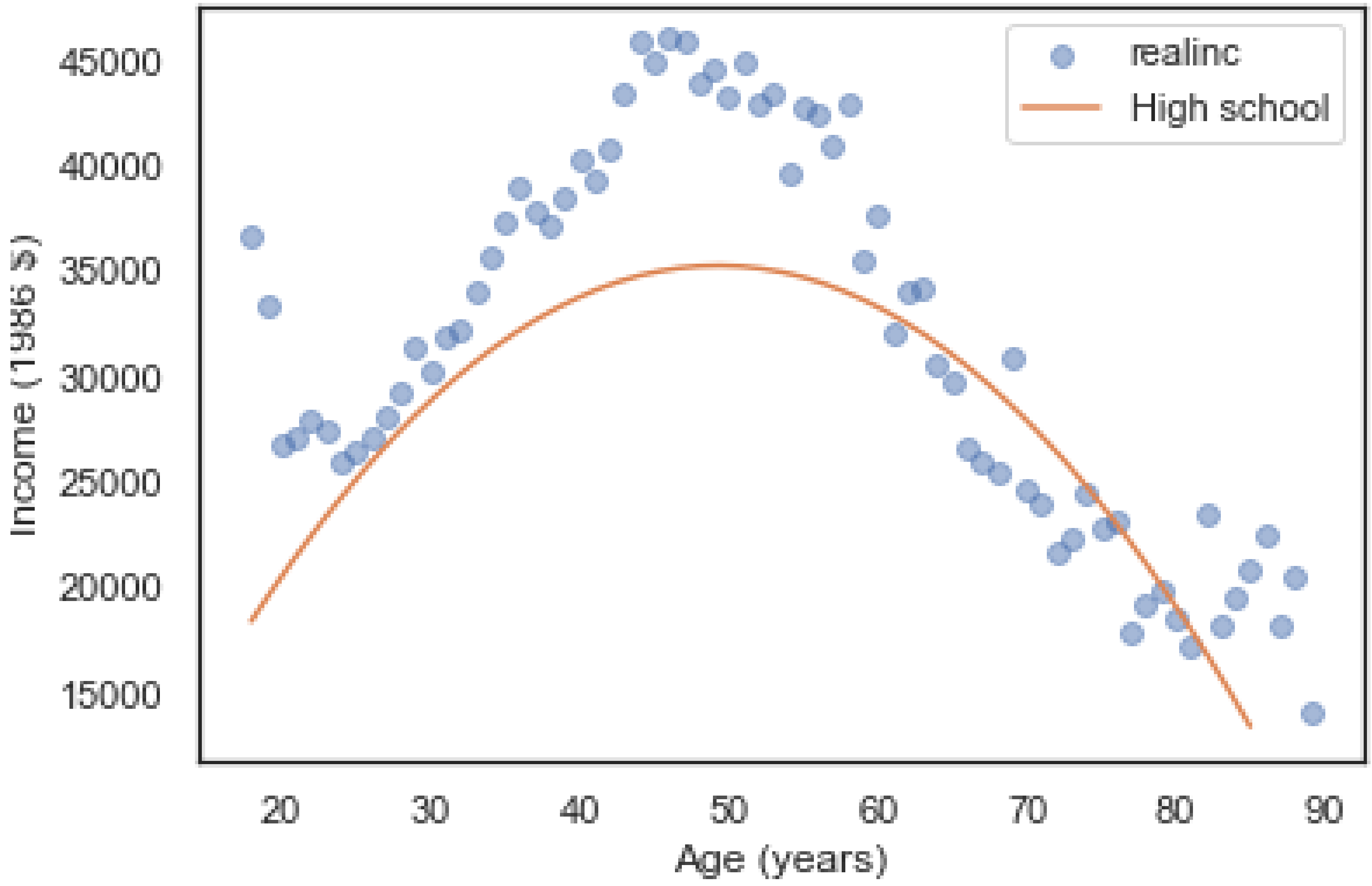
```
pred12 = results.predict(df)
```

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()
```



The blue dots show the average income in each age group.

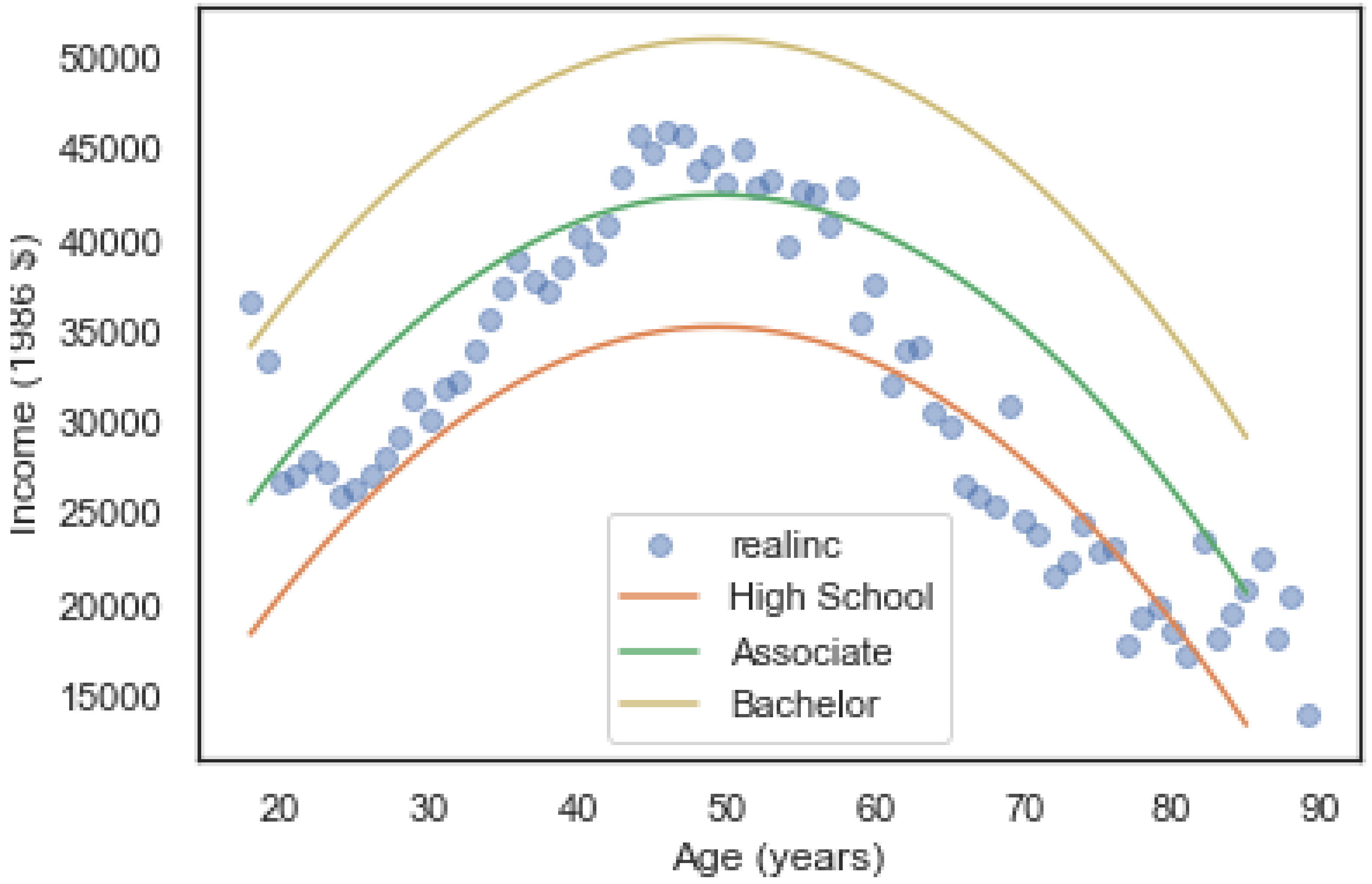
The orange line shows the predictions generated by the model, holding education constant.

This plot shows the shape of the model, a downward-facing parabola.

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'
```



Levels of education

- 14 years, which is the nominal time to earn an Associate's degree

- 16 years, which is the nominal time to earn a Bachelor's degree.

Interpreting the results

The lines show mean income, as predicted by the model, as a function of age, for three levels of education. This visualization helps validate the model since we can compare the predictions with the data. And it helps us interpret the model since we can see the separate contributions of age and education

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)
```

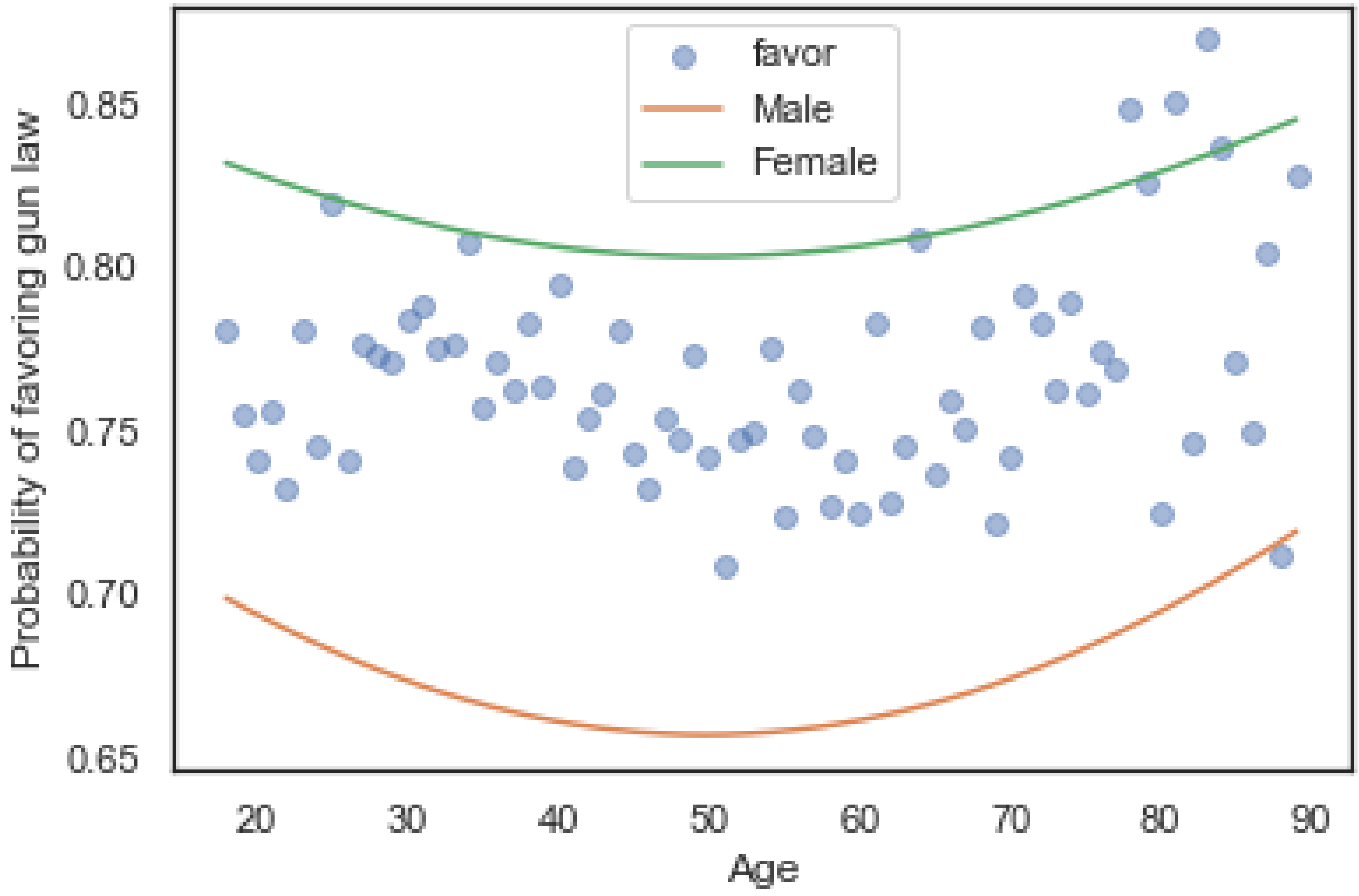
```
df['sex'] = 2
pred2 = results.predict(df)
```

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

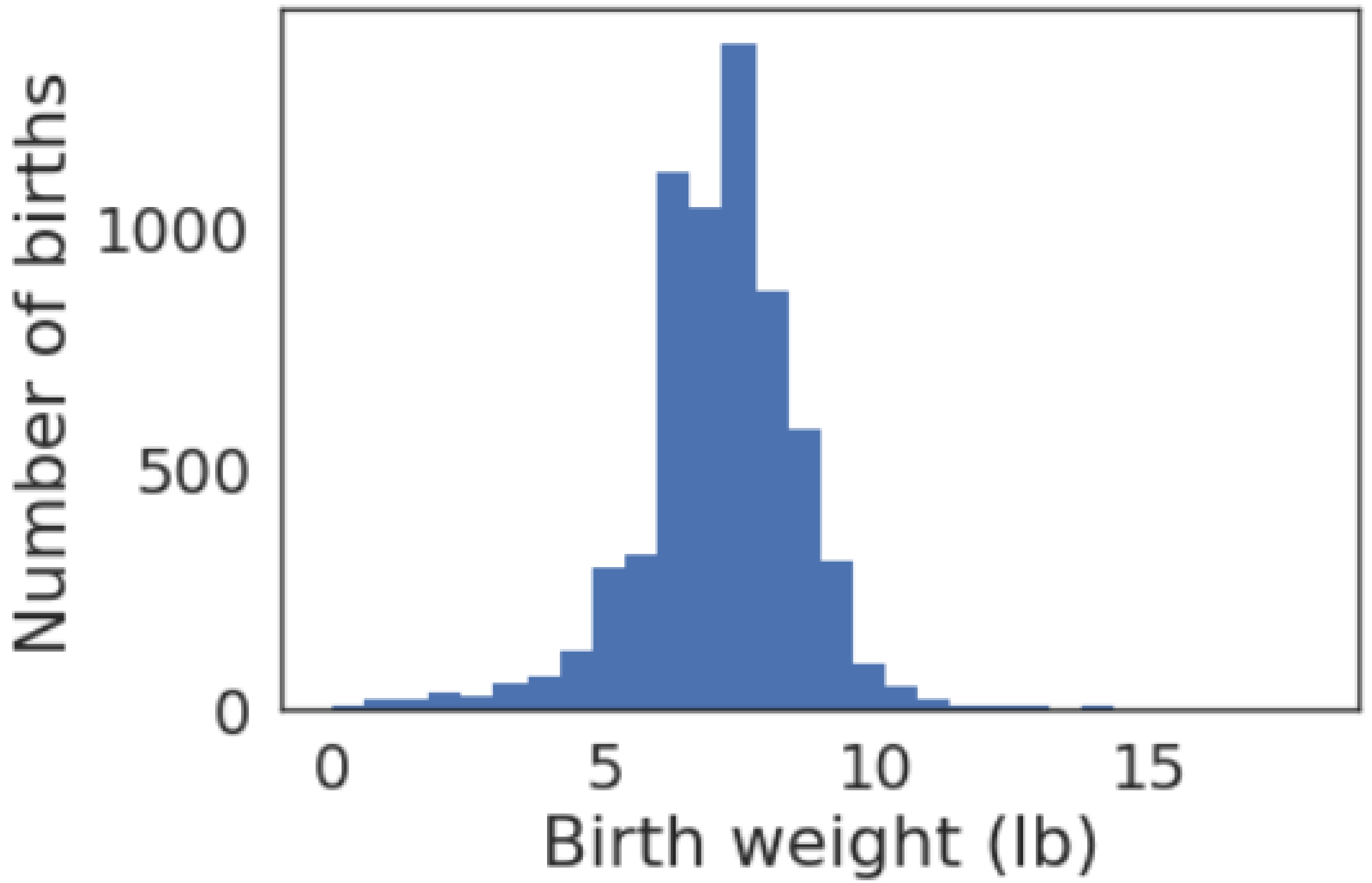
EXPLORATORY DATA ANALYSIS IN PYTHON

Allen Downey
Professor, Olin College

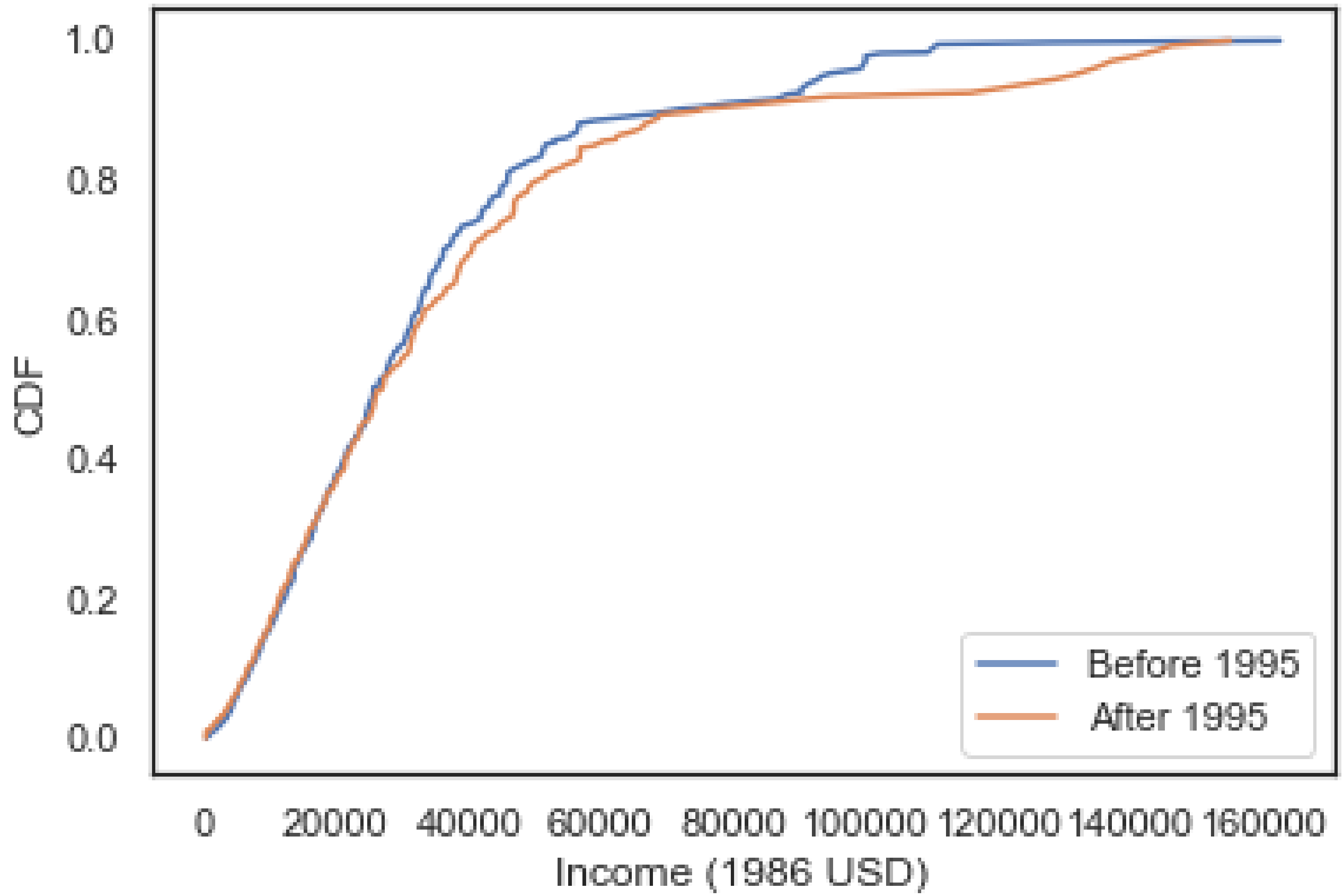
Exploratory Data Analysis

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

Import, clean, and validate



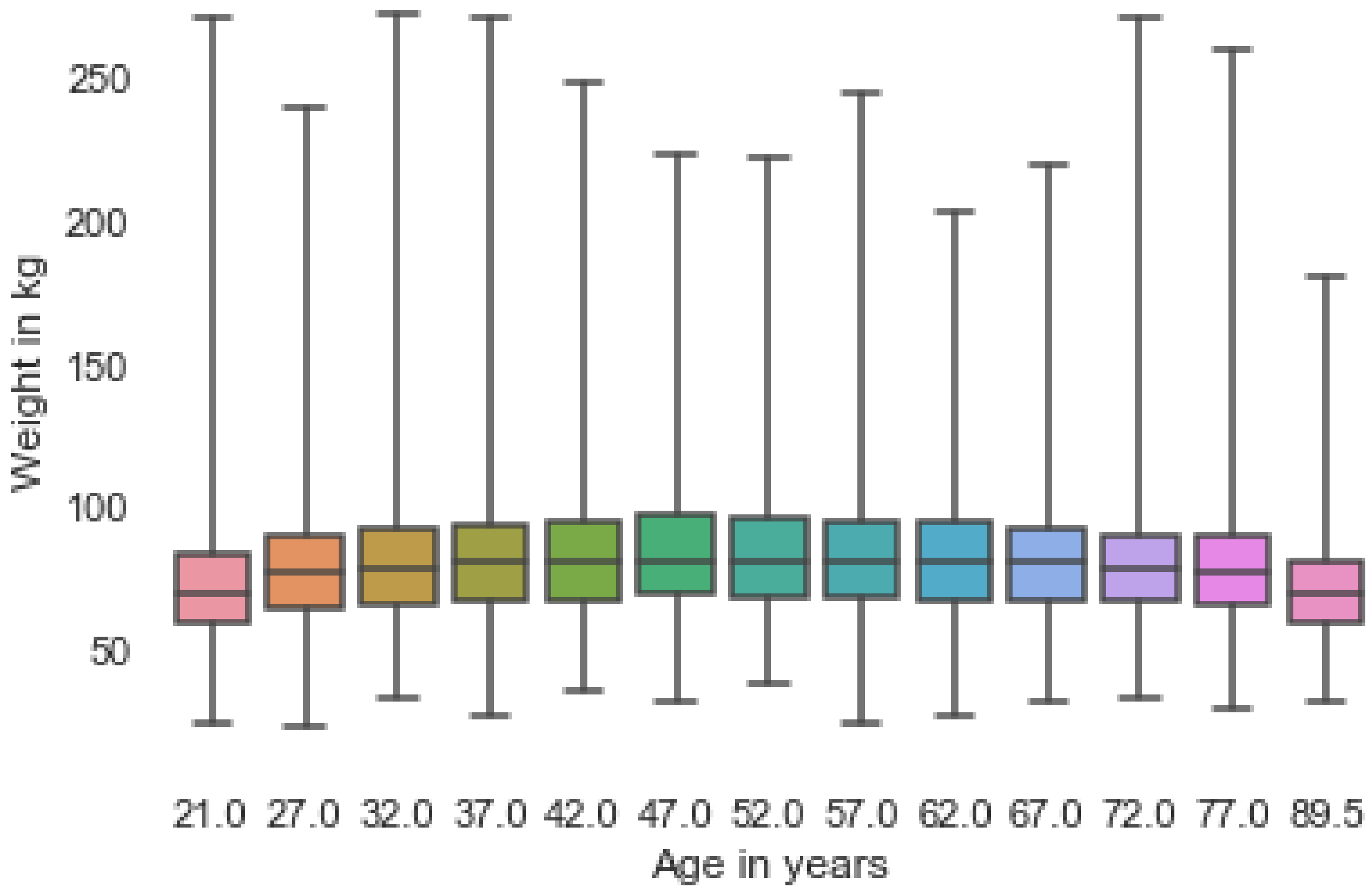
Visualize distributions



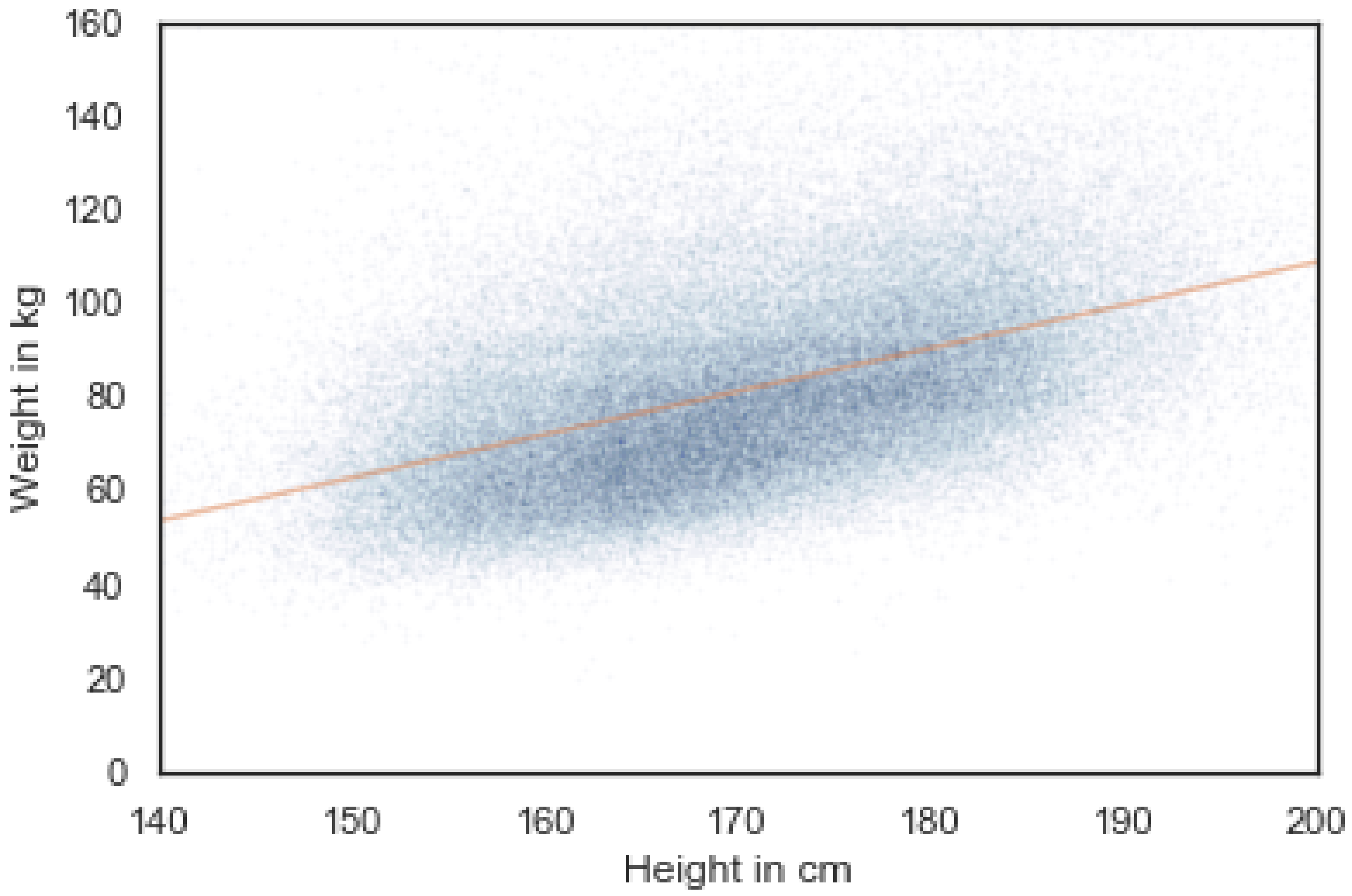
CDF, PMF, and KDE

- 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

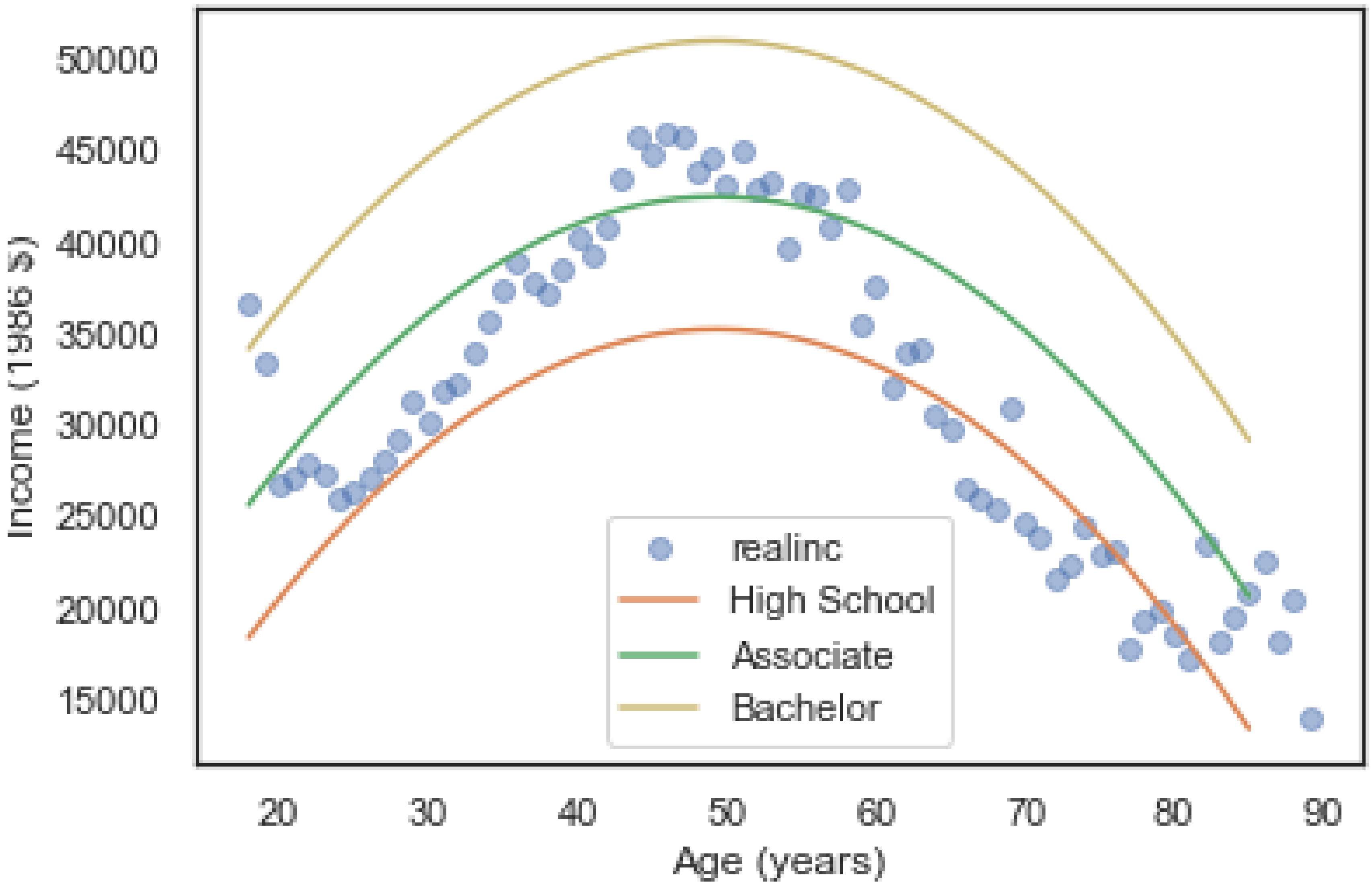


We used the coefficient of correlation to quantify the strength of a relationship.

We also used simple regression to find the line of best fit, like the one here that shows weight as a function of height.

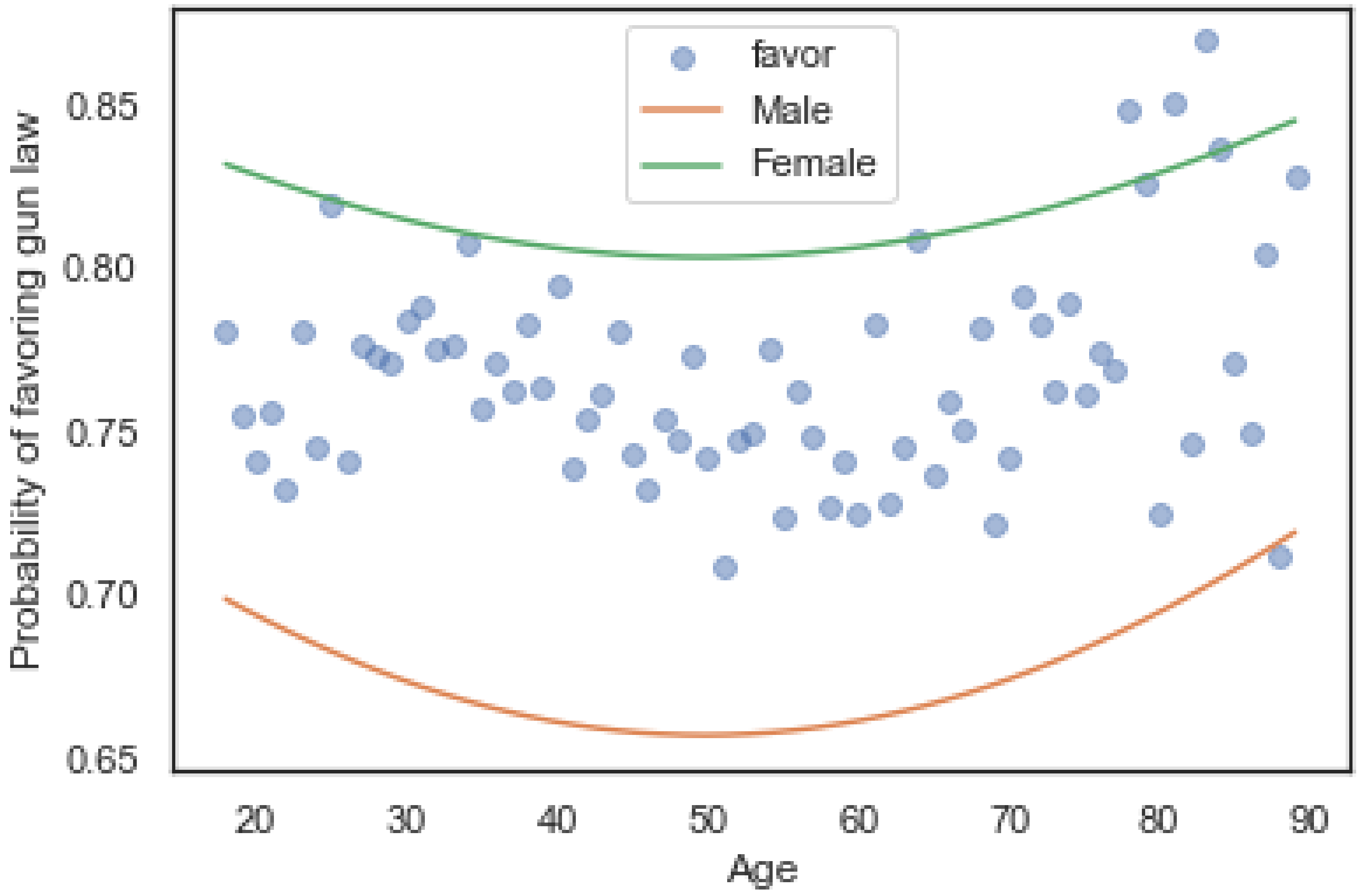
But remember that both of these methods only capture linear relationships; if the relationship is non-linear, they can be misleading. Always look at a visualization, like this scatter plot, before computing correlation or simple regression.

Multiple regression



we used multiple regression to add control variables and to describe non-linear relationships. For example, this plot shows the non-linear relationship between income and age, controlling for level of education.

Logistic regression



we used logistic regression to explain and predict binary variables. For example, this figure shows the relationship between support for gun control, as a function of age, for male and female respondents in the GSS.

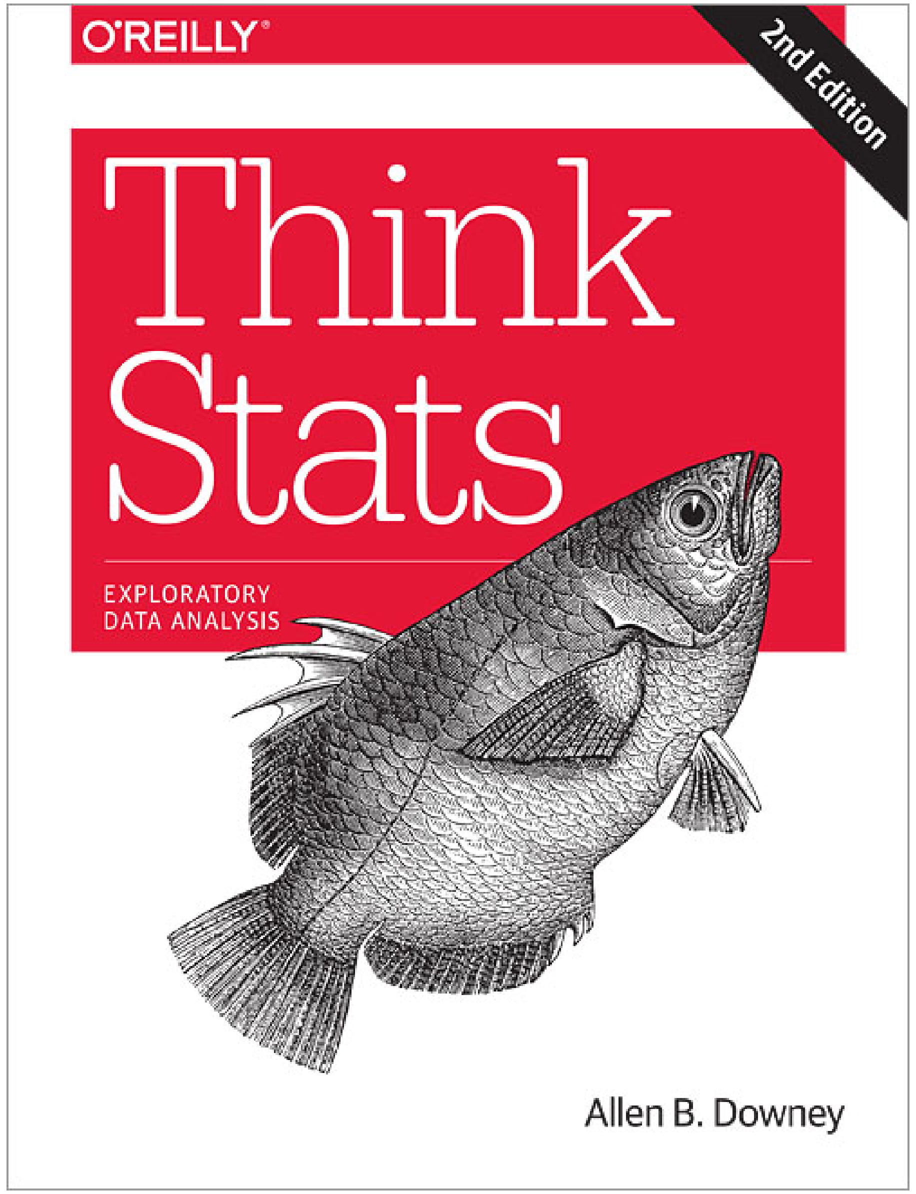
Where to next?

- 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