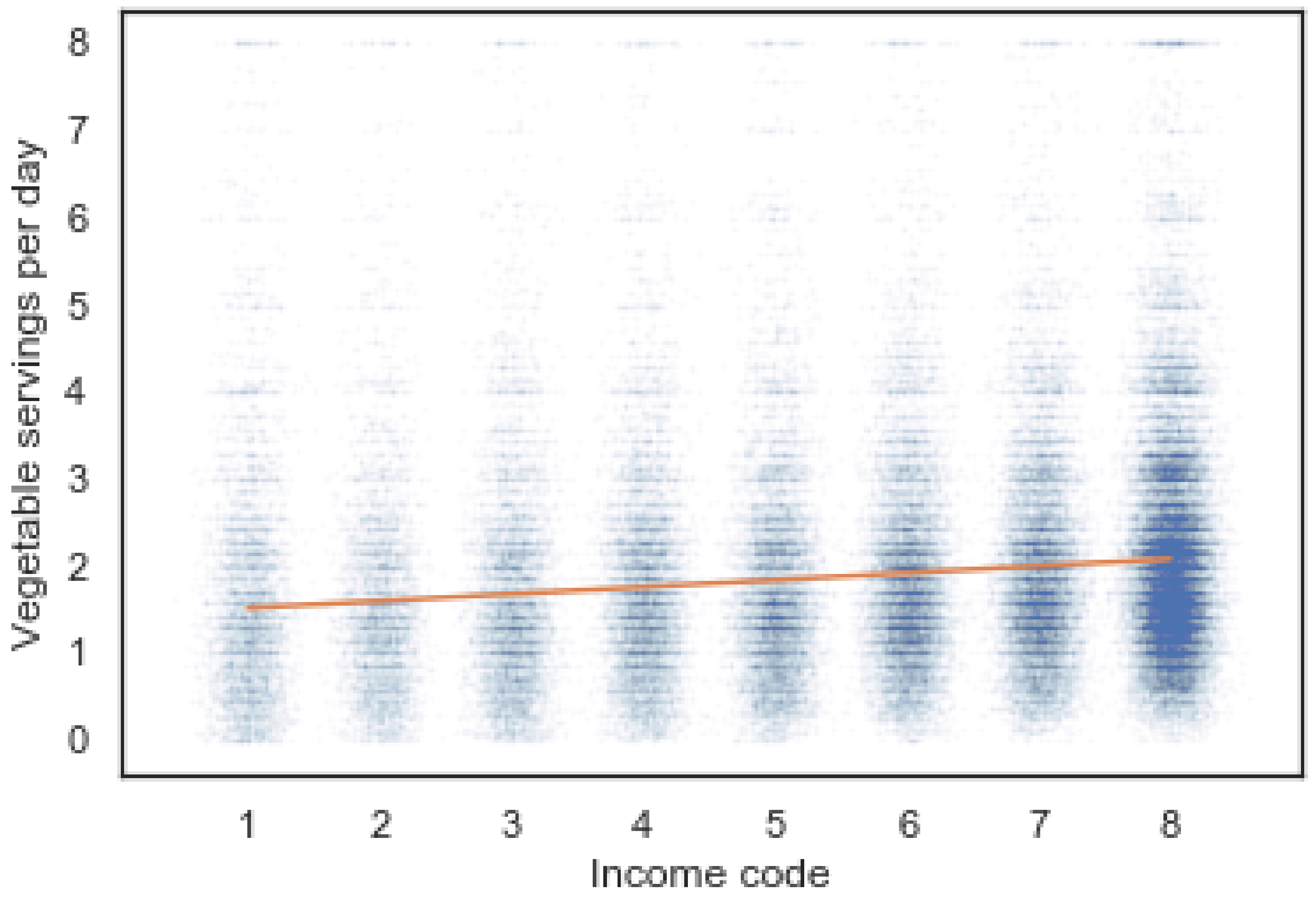
Limits of simple regression

EXPLORATORY DATA ANALYSIS IN PYTHON

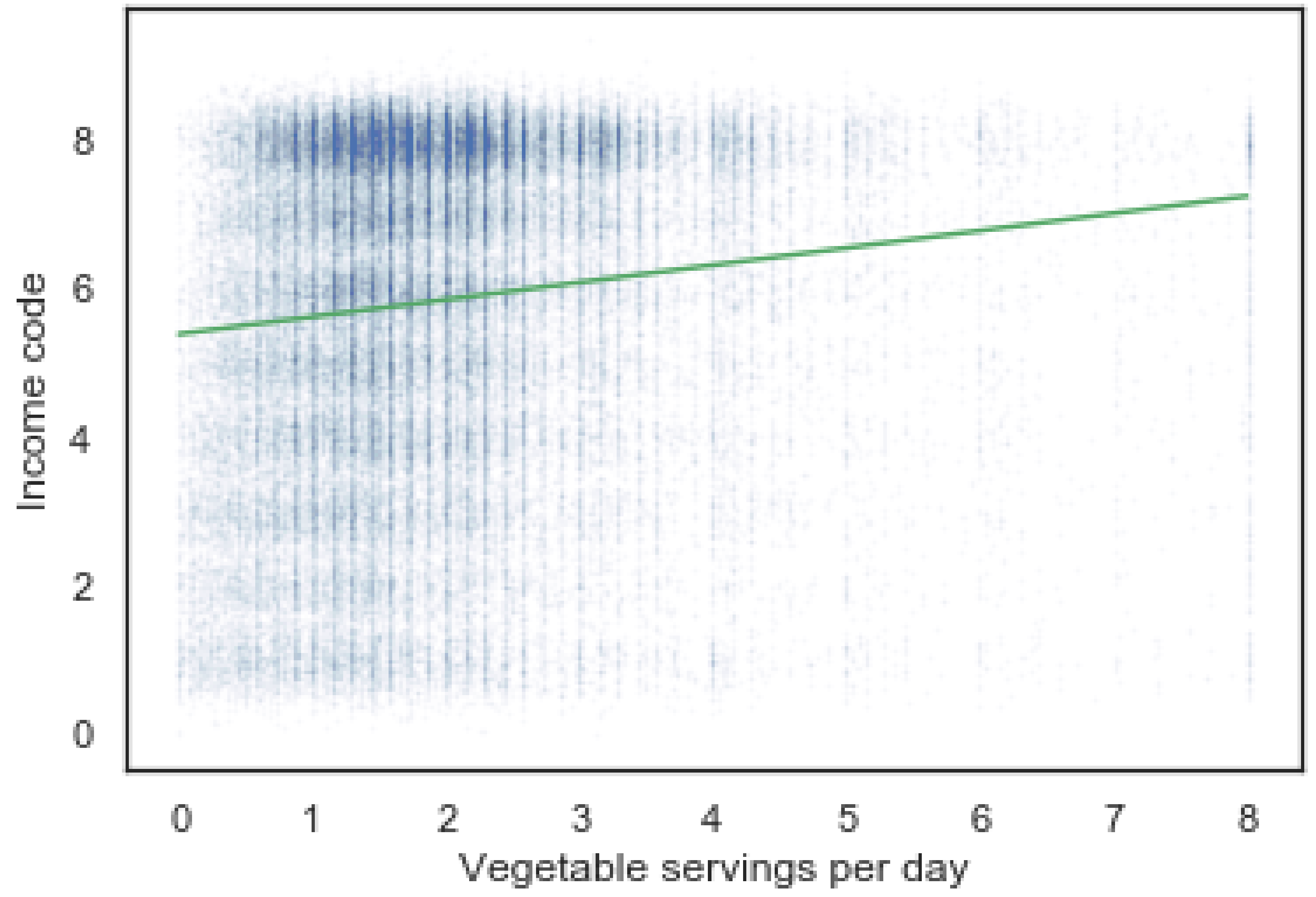
Allen Downey
Professor, Olin College

Income and vegetables



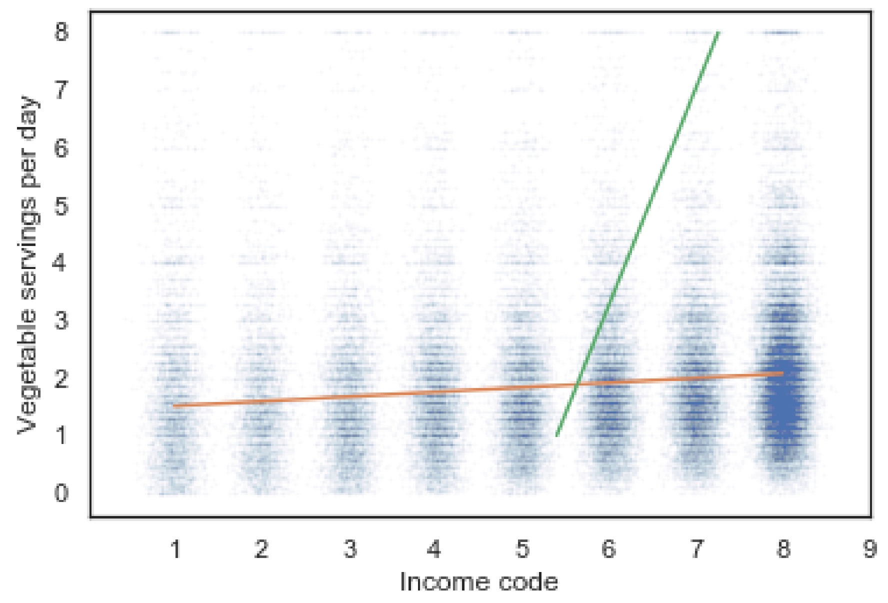
2. Income and vegetables
In a previous exercise, you made a scatter plot of vegetable consumption as a function of income, and plotted a line of best fit. Here's what it looks like (note that this version includes more data than you had for the exercise). The slope of the line is 0 point 07, which means that the difference between the lowest and highest income brackets is about 0 point 49 servings per day. So that's not a very big difference.

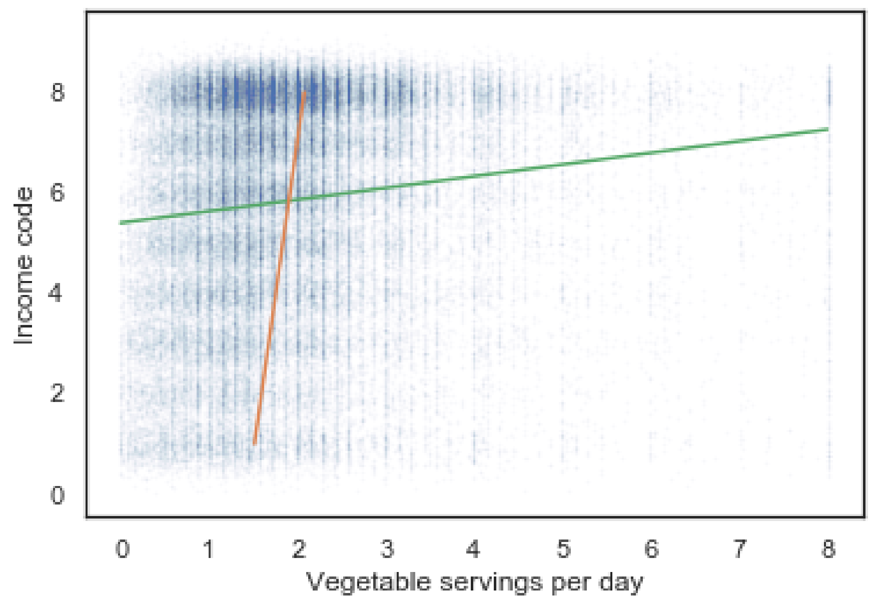
Vegetables and income



3. Vegetables and income
But it was an arbitrary choice to plot vegetables as a function of income. We could have plotted it the other way around, like this. The slope of this line is 0 point 23, which means that the difference between 0 and 8 servings per day is about 2 income codes, roughly from code 5 to code 7. If we check the codebook, income code 5 is about $30,000 per year; income code 7 is about $65,000. So if we use vegetable consumption to predict income, we see a big difference. But when we used income to predict vegetable consumption, we saw a small difference. This example shows that regression is not symmetric; the regression of A onto B is not the same as the regression of B onto A.

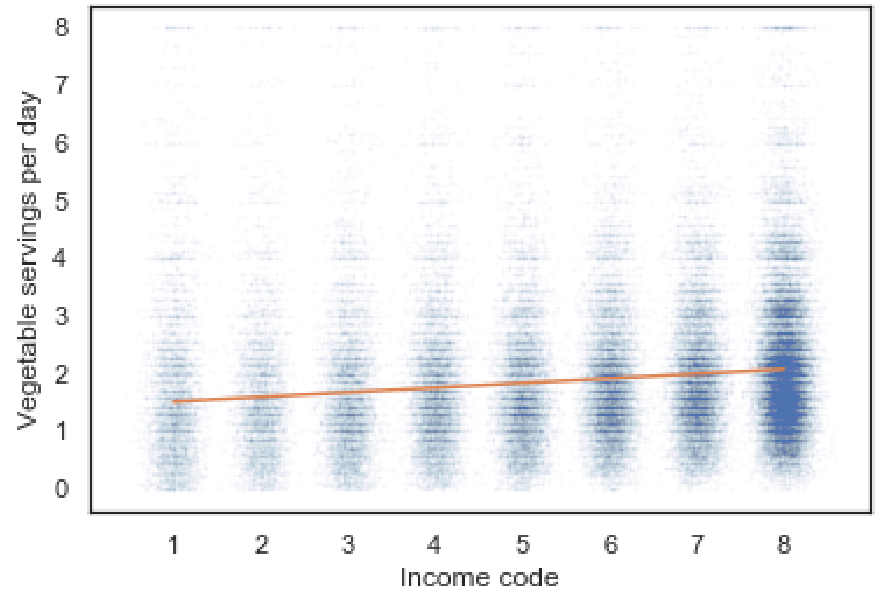
Regression is not symmetric

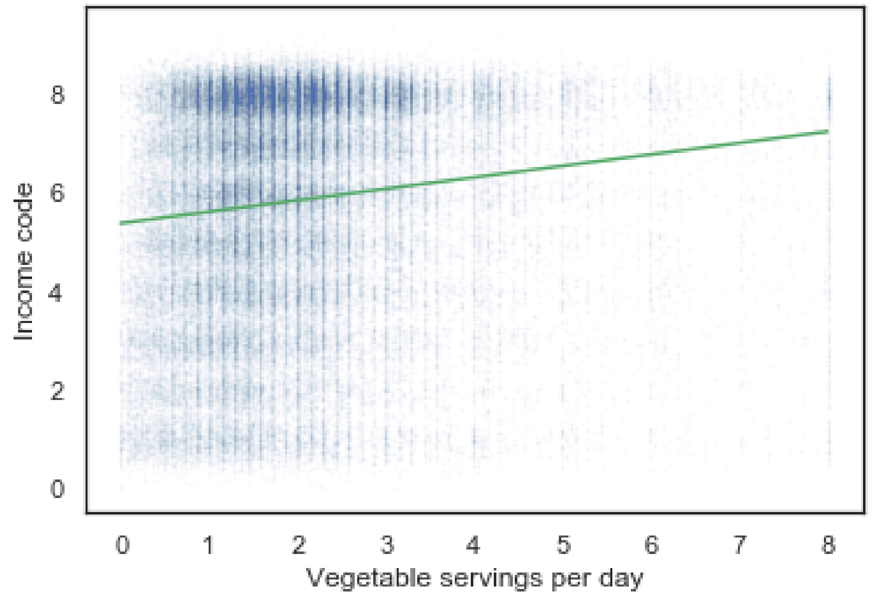




4. Regression is not symmetric
We can see that more clearly by putting the two figures side by side and plotting both regression lines on both figures. They are different because they are based on different assumptions. On the left, we treat income as a known quantity and vegetable consumption as random. On the right, vegetable consumption is known and income is random. When you run a regression model, you make decisions about how to treat the data, and those decisions affect the result you get.

Regression is not causation





5. Regression is not causation
This example is meant to demonstrate another point, which is that regression doesn't tell you much about causation. If you think people with lower income can't afford vegetables, you might look at the figure on the left and conclude that it doesn't make much difference. If you think better diet increases income, the figure on the right might make you think it does. But in general, regression can't tell you what causes what. In this example, A might cause B, or B might cause A, or there might be other factors that cause both A and B. Regression alone can't tell you which way it goes.

Multiple regression

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

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

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

6. Multiple regression
However, we have tools for teasing apart relationships among multiple variables; one of the most important is multiple regression. SciPy doesn't do multiple regression, so we have to switch to a new library, StatsModels. Here's the import statement. And here's how we use it. `ols` stands for "ordinary least squares", another name for regression. The first argument is a formula string that specifies that we want to regress income as a function of vegetable consumption. The second argument is the BRFSS DataFrame. The names in the formula correspond to columns in the DataFrame. The result from ols() represents the model; we have to run dot fit() to get the results. The results object contains a lot of information, but the first thing we'll look at is params, which contains the estimated slope and intercept. And we get the same results we got from SciPy, so that's good!

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
```

2. Income and education
First, we load the GSS data. Then we run a regression of real income as a function of years of education. The first argument of ols() is a formula that specifies the variables in the regression. On the left, realinc is the variable we are trying to predict; on the right, educ is the variable we are using to inform the predictions. And here are the results. The estimated slope is 3586, which means that each additional year of education is associated with an increase of almost $3600 of income. But income also depends on age, so it would be good to include that in the model, too.

Adding age

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

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

3. Adding age
Here's how. On the right side of the formula, you can list as many variables as you like, in this case, we have educ and age. The plus sign indicates that we expect the contributions of the two variables to be additive, which is a common assumption for models like this. Here are the results. The estimated slope for education is 3655, a little more than in the previous model. The estimated slope for age is only about $80 per year, which is surprisingly small.

Income and age

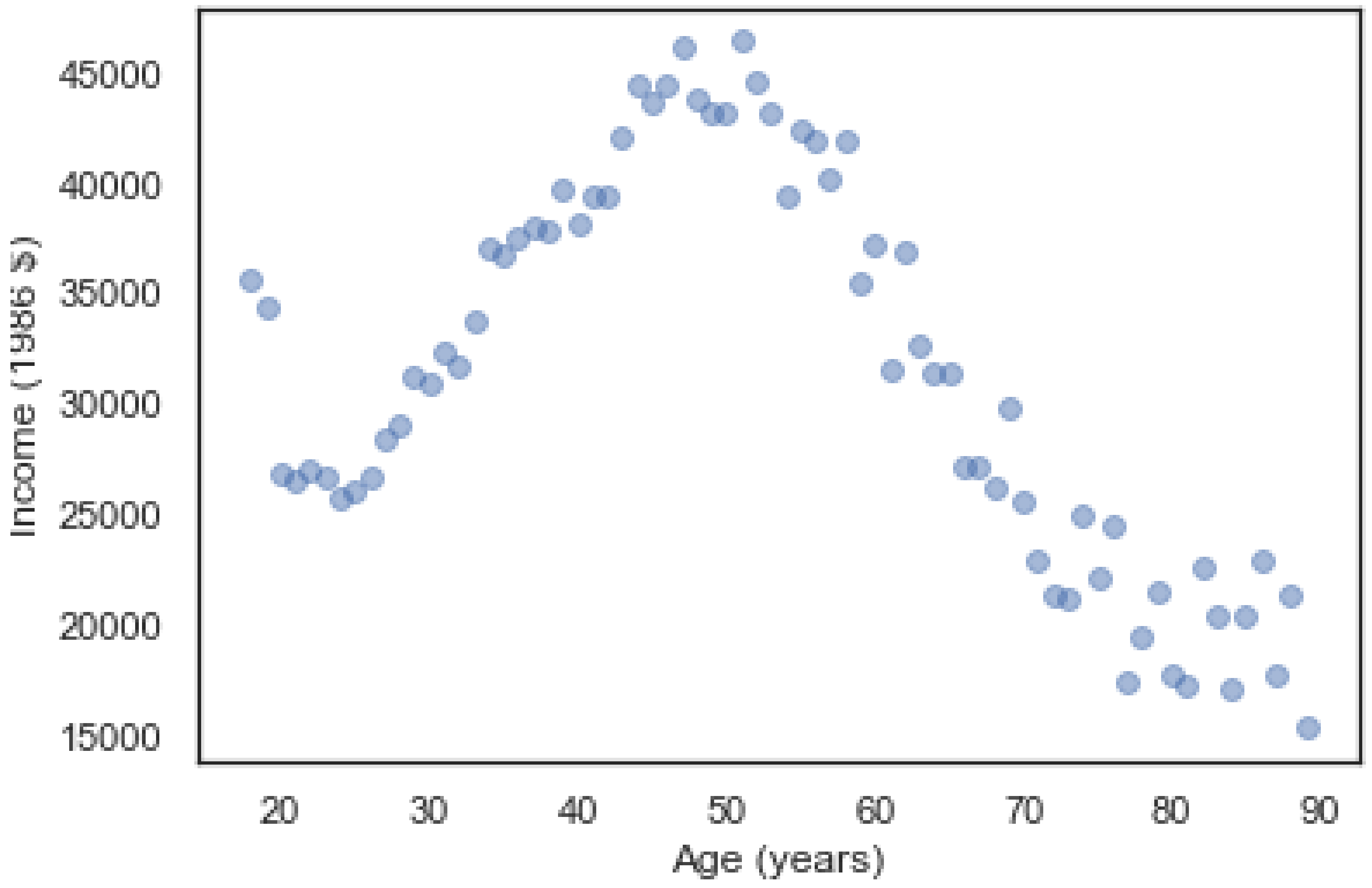
```
grouped = gss.groupby('age')
```

```
<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 $)')
```

4. Income and age
To see what's going on, let's look more closely at the relationship between income and age. I'll use groupby(), which is a Pandas feature we haven't seen before, to divide the DataFrame into age groups. The result is a GroupBy object that contains one group for each value of age. The GroupBy object behaves like a DataFrame in many ways. You can use brackets to select a column, like realinc in this example, and then invoke a method like mean(). The result is a Pandas series that contains the mean income for each age group, which we can plot like this.



5. Mean income over age
Here's the result. Average income increases from age 20 to age 50, then starts to fall. And that explains why the estimated slope is so small, because the relationship is non-linear. Remember that correlation and simple regression can't measure non-linear relationships. But multiple regression can!

Adding a quadratic term

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

```python
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
```

6. Adding a quadratic term
To describe a non-linear relationship, one option is to add a new variable that is a non-linear combination of other variables. As an example, I'll create a new variable called age2 that equals age squared. Now we can run a regression with both age and age2 on the right side. And here are the results. The slope associated with age is substantial, about $1700 per year. The slope associated with age2 is about -17, which is harder to interpret.

Whew!

EXPLORATORY DATA ANALYSIS IN PYTHON

Visualizing regression results

EXPLORATORY DATA ANALYSIS IN PYTHON

1. Visualizing regression results
In the previous lesson we ran a multiple regression model to characterize the relationship between income and age. Because the model is non-linear, the parameters are hard to interpret. In this lesson we'll see a way to interpret them visually, and to validate them against data.

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
```

2. Modeling income and age
Here's the model from the previous exercise. First, we created new variables for educ squared and age squared. Then we ran the regression model with educ, educ2, age, and age2. And here are the results. The parameters are hard to interpret. Fortunately, we don't have to -- sometimes the best way to understand a model is by looking at its predictions rather than its parameters.

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)
```

3. Generating predictions
The regression results object provides a method called predict() that uses the model to generate predictions. It takes a DataFrame as a parameter and returns a Series with a prediction for each row in the DataFrame. To use it, I'll create a new DataFrame with age running from 18 to 85, and age2 set to age squared. Next, I'll pick a level for educ, like 12 years, which is the most common value. When you assign a single value to a column in a DataFrame, Pandas makes a copy for each respondent. Then we can use results to predict the average income for each age group, holding education constant.

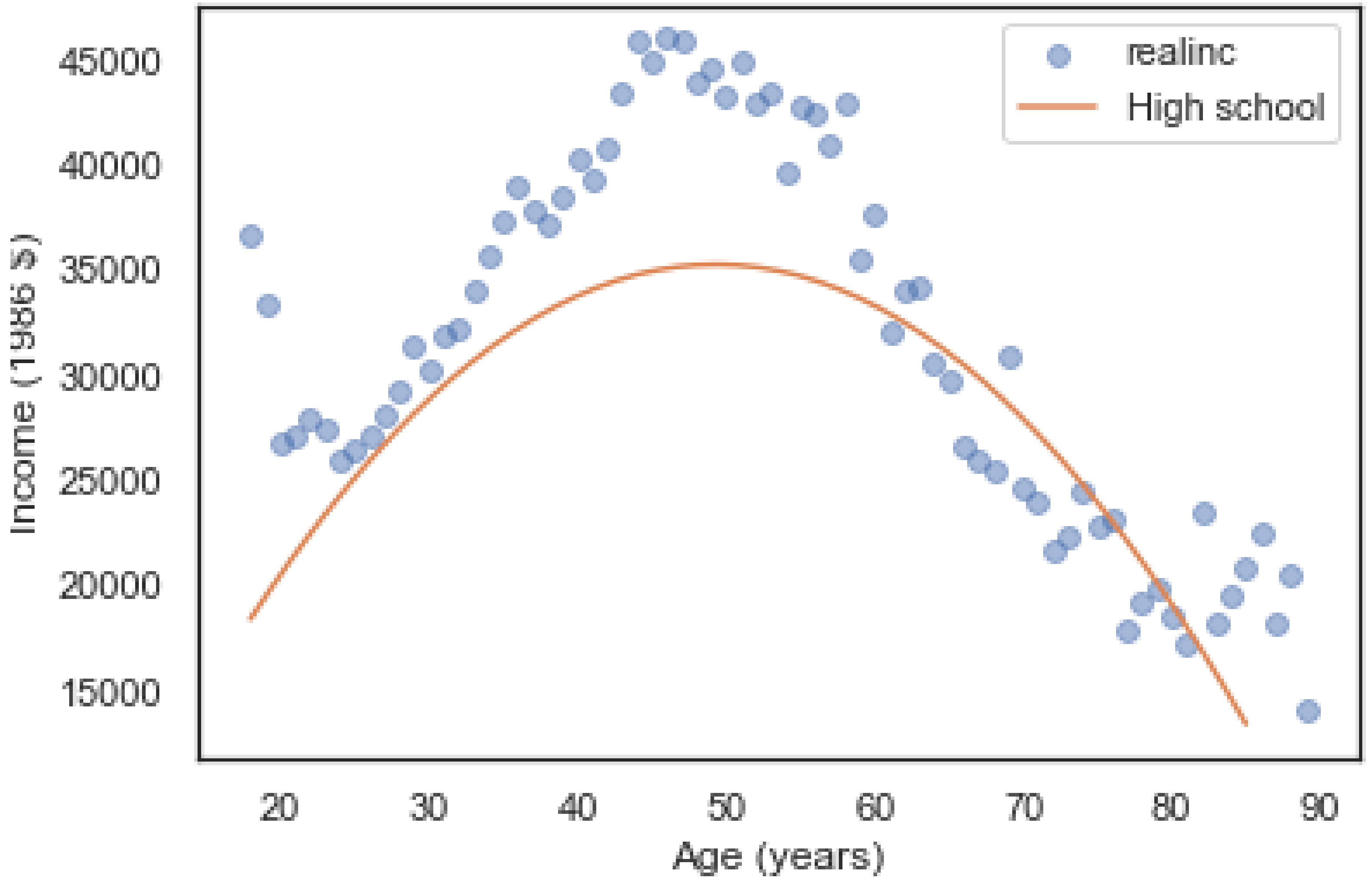
Plotting predictions

```
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()
```

4. Plotting predictions
The result from predict() is a Series with one prediction for each row. So we can plot it like this, with age on the x-axis and the predicted income for each age group on the y-axis. We can plot the data for comparison, like this; recall that we computed mean_income_by_age in the previous lesson. And we should label the axes, as always.



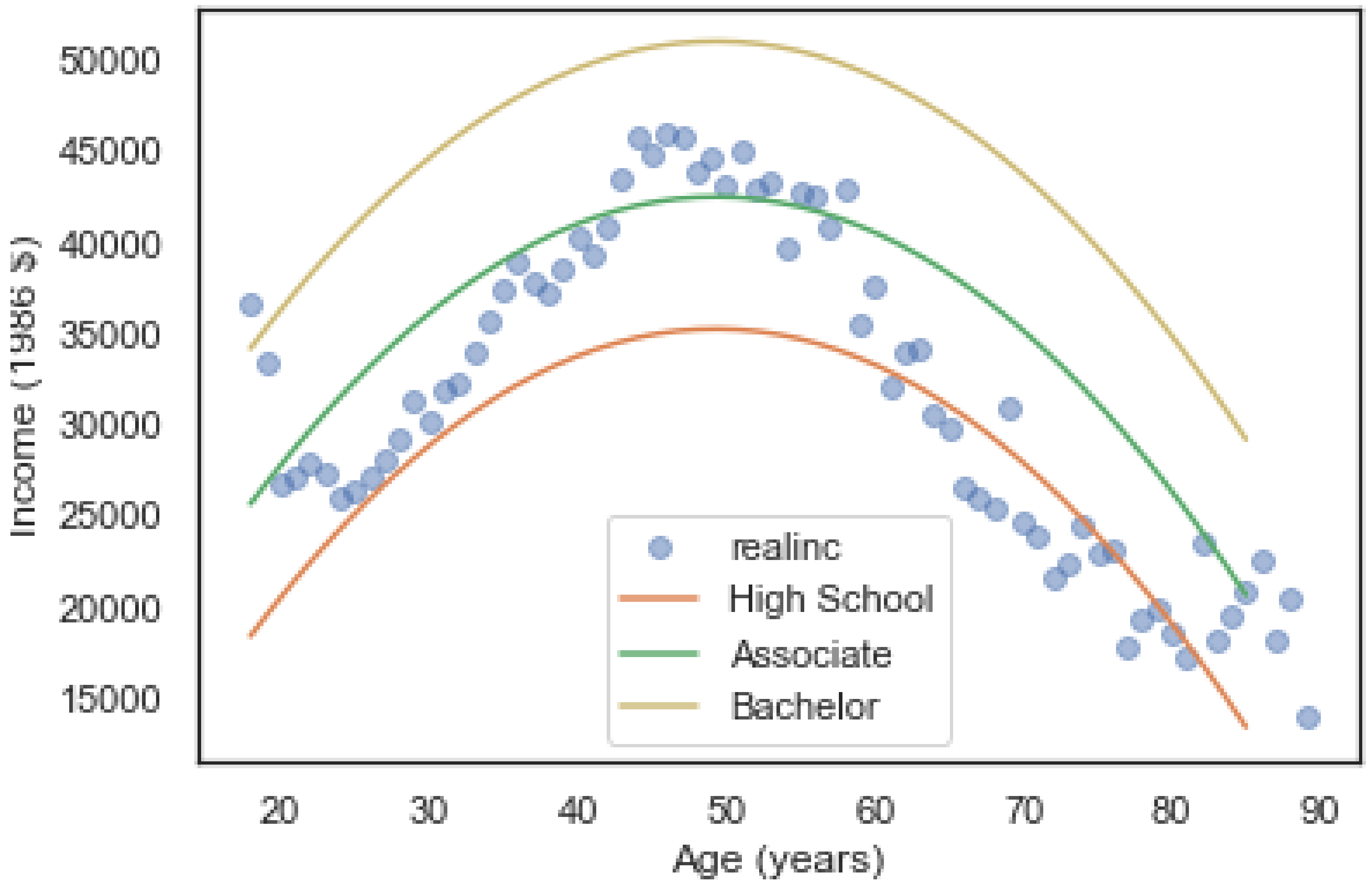
5. Comparing with data
Here are the results. The blue dots show the average income in each age group. The orange line shows the predictions generated by the model, holding education constant. This plot shows the shape of the model, a downward-facing parabola.

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'
```

6. Levels of education
We can do the same thing with other levels of education, like 14 years, which is the nominal time to earn an Associate's degree, and 16 years, which is the nominal time to earn a Bachelor's degree.



7. Interpreting the results
And here are 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!

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.

2. Categorical variables
To understand logistic regression, we have to start with categorical variables. Most of the variables we have used so far - like income, age, and education - are numerical. But variables like sex and race are categorical; that is, each respondent belongs to one of a specified set of categories.

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
```

3. Sex and income
With StatsModels, it is easy to include a categorical variable as part of a regression model. Here's how. In the formula string, the letter C indicates that sex is a categorical variable. And here are the results. The regression treats the value sex=1, which is male, as the default, and reports the difference associated with the value sex=2, which is female. So this result indicates that income for women is about $4100 less than for men, after controlling for age and education.

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
```

4. Boolean variable

If a categorical variable has only two values, it's called a boolean variable. For example, one of the questions in the General Social Survey asks "Would you favor or oppose a law which would require a person to obtain a police permit before he or she could buy a gun?" The variable is called gunlaw, and here are the values. 1 means yes and 2 means no, so most respondents are in favor. To explore the relationship between this variable and factors like age, sex, and education, we can use logistic regression. StatsModels provides logistic regression, but to use it, we have to recode the variable so 1 means yes and 0 means no. We can do that by replacing 2 with 0. And we can check the results.

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
```

5. Logistic regression
Now we can run the regression. Instead of ols(), we use logit(), which is named for the logit function, which is related to logistic regression. Other than that, everything is the same as what we have seen before. And here are the results. The parameters are in the form of "log odds", which you may or may not be familiar with. I won't explain them in detail here, except to say that positive values are associated with things that make the outcome more likely; negative values make the outcome less likely. For example, the parameter associated with sex=2 is 0.75, which indicates that women are more likely to support this form of gun control. To see how much more likely, we can generate and plot predictions, as we did with linear regression.

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)
```

6. Generating predictions
As an example, I'll generate predictions for different ages and sexes, with education held constant. First we need a DataFrame with age and educ. Then we can compute age2 and educ2. We can generate predictions for men like this. And for women like this.

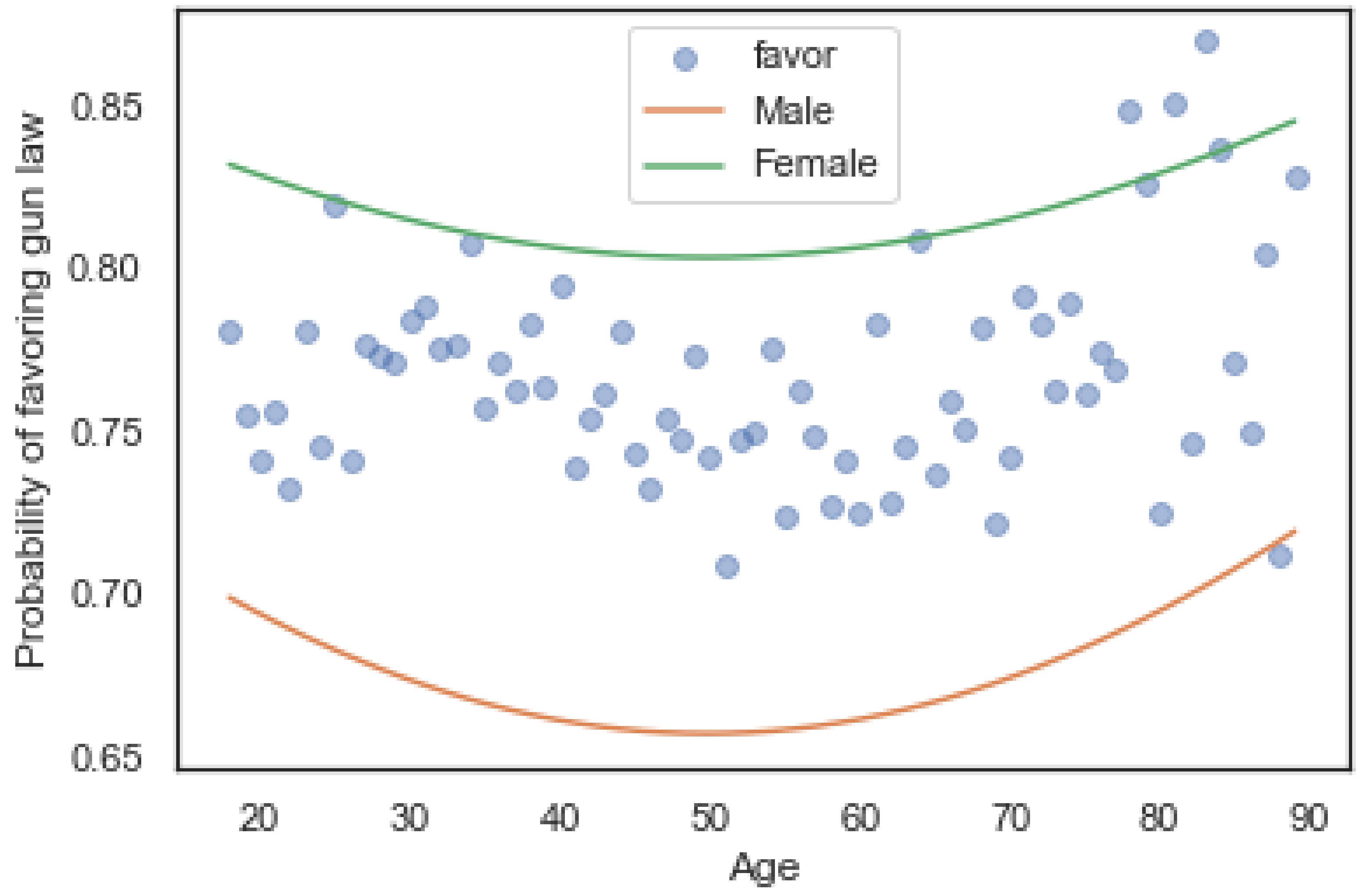
Visualizing results

```
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()
```

7. Visualizing results
Now, to visualize the results, I start by plotting the data. As we've done before, we'll divide the respondents into age groups and compute the mean in each group. The mean of a binary variable is the fraction of people in favor. Now we can plot the predictions, for men and women, as a function of age. And label the axes.



8. Gun laws and age

Here's what it looks like. According to the model, people near age 50 are least likely to support gun control (at least as this question was posed). And women are more likely to support it than men, by almost 15 percentage points.

Let's practice!

EXPLORATORY DATA ANALYSIS IN PYTHON

9. Let's practice!
Logistic regression is a powerful tool for exploring relationships between a binary variable and the factors that predict it. In the exercises, you'll explore the factors that predict support for legalizing marijuana.

Next steps

EXPLORATORY DATA ANALYSIS IN PYTHON

Allen Downey
Professor, Olin College

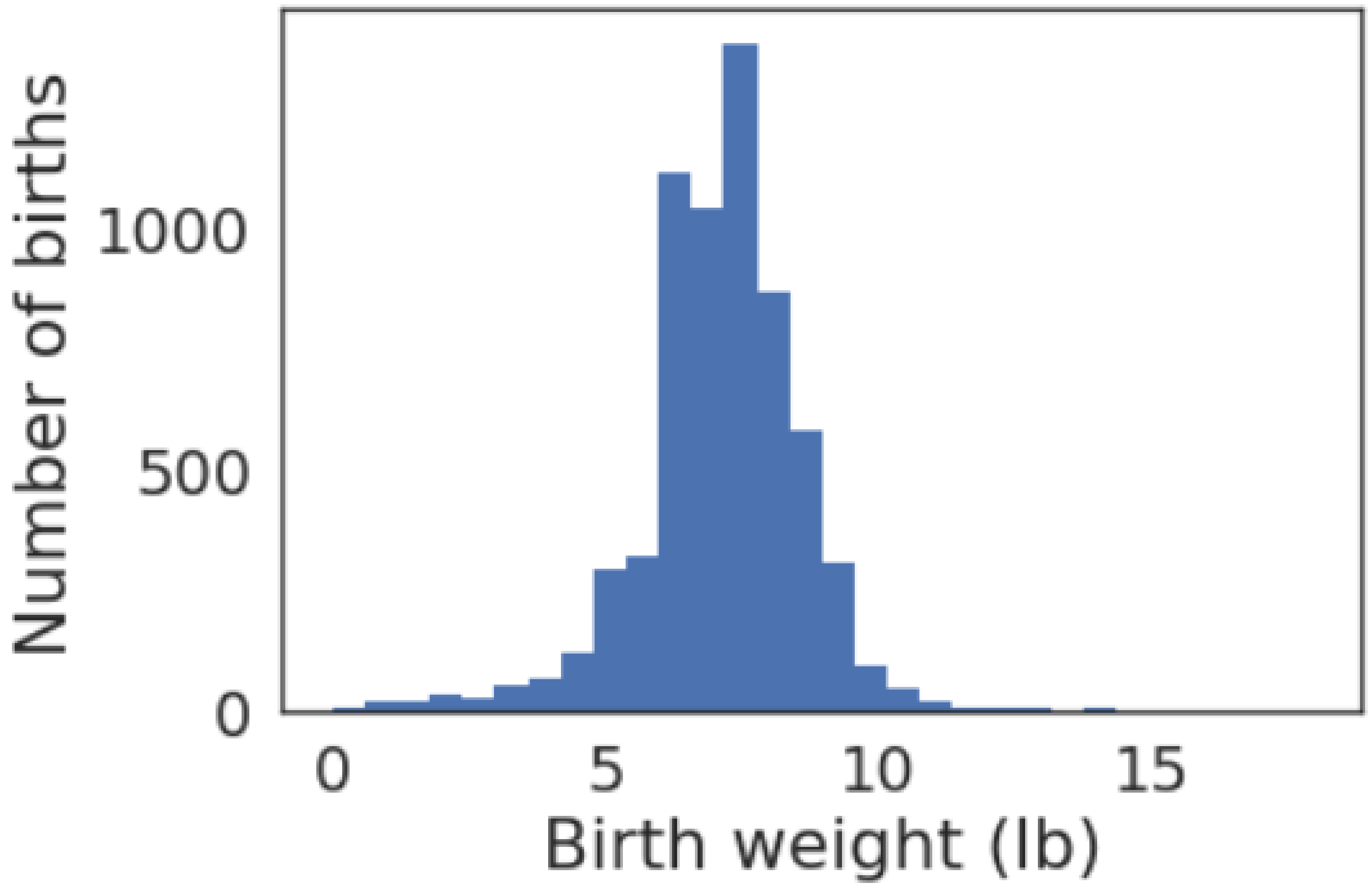
Exploratory Data Analysis

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

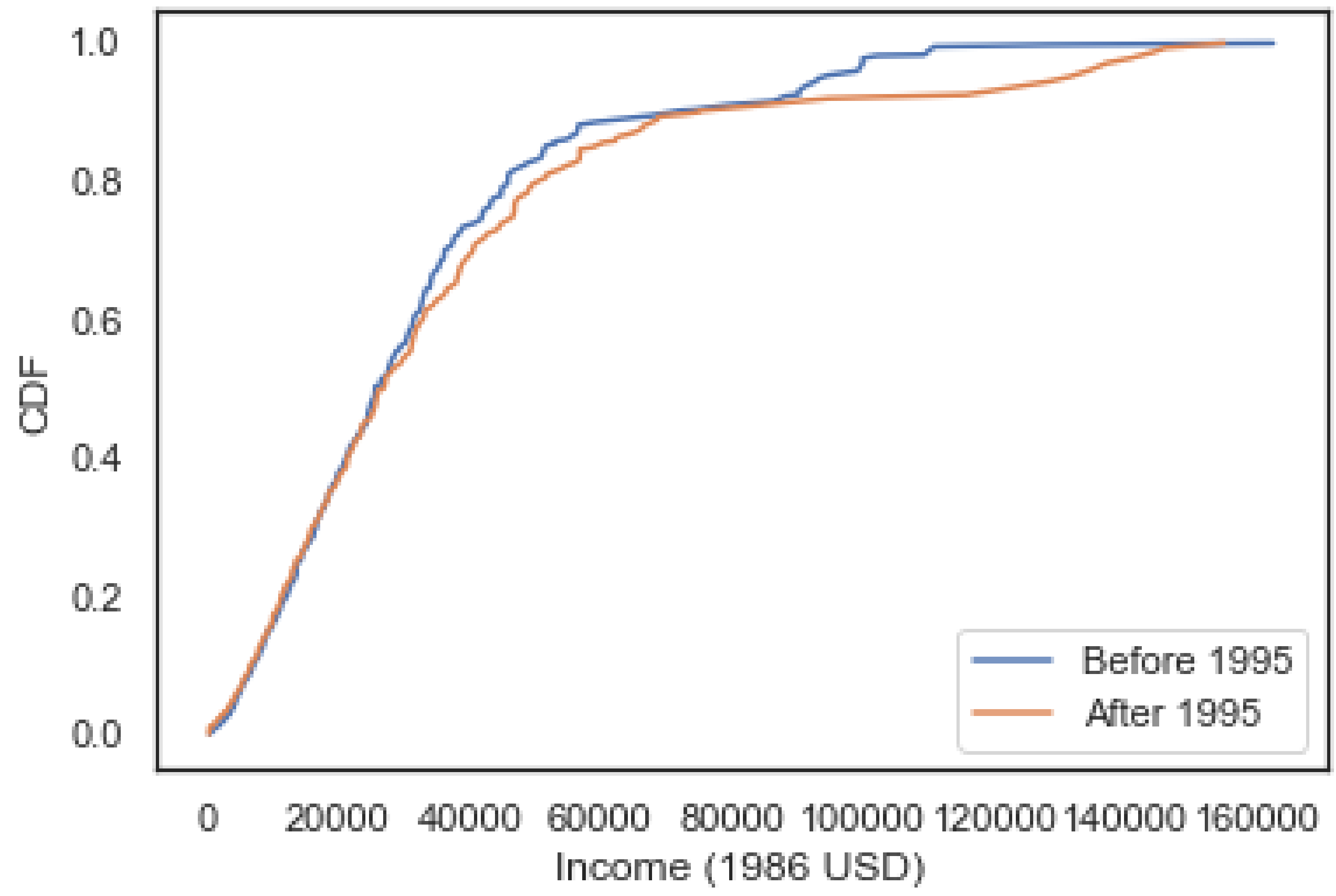
2. Exploratory Data Analysis

The organizing theme of this course is exploratory data analysis, which is a process and a set of techniques for working with data, especially in the early stages of a project, or when you are working with a new data set....

Import, clean, and validate



Visualize distributions

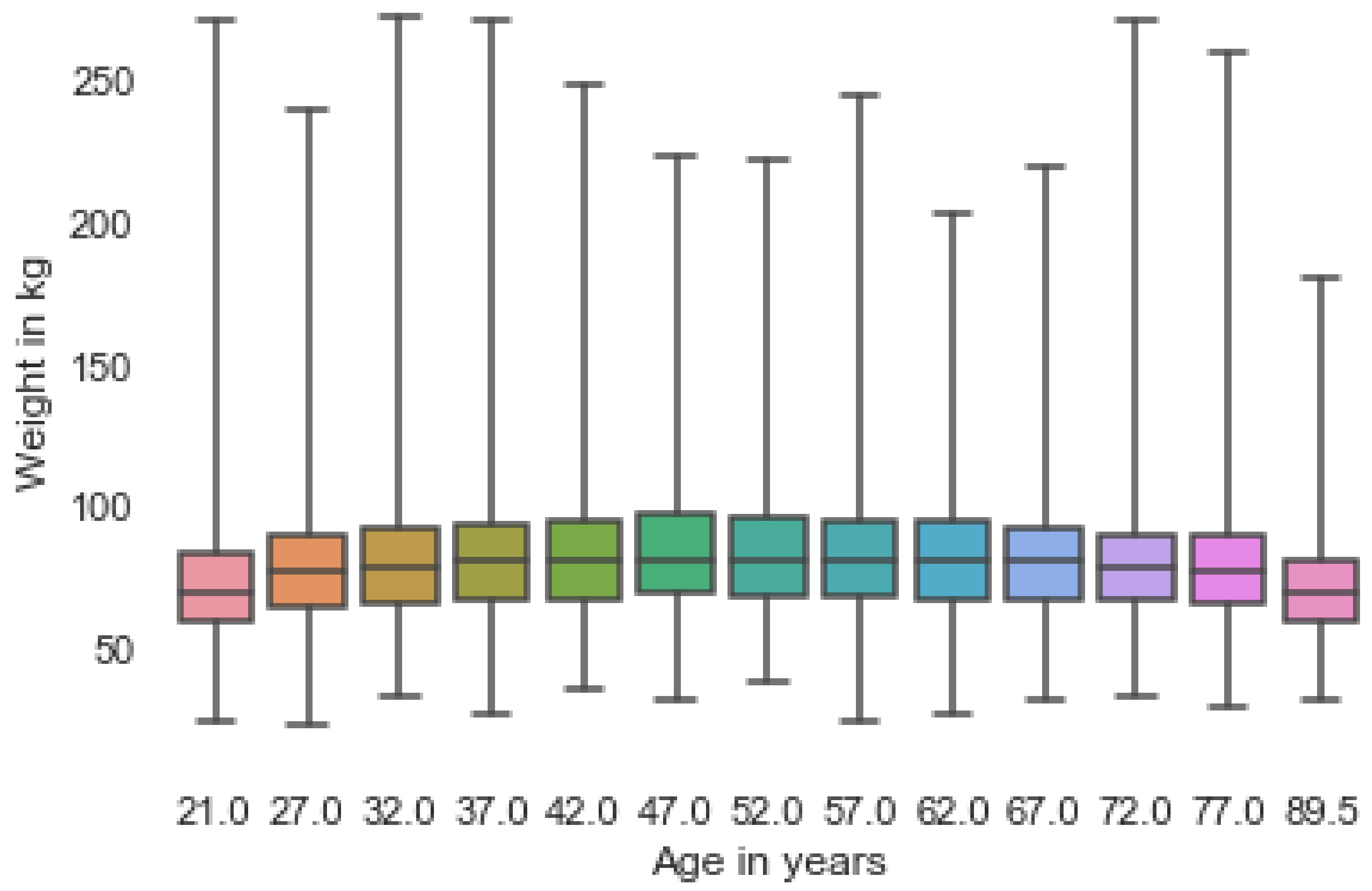


CDF, PMF, and KDE

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

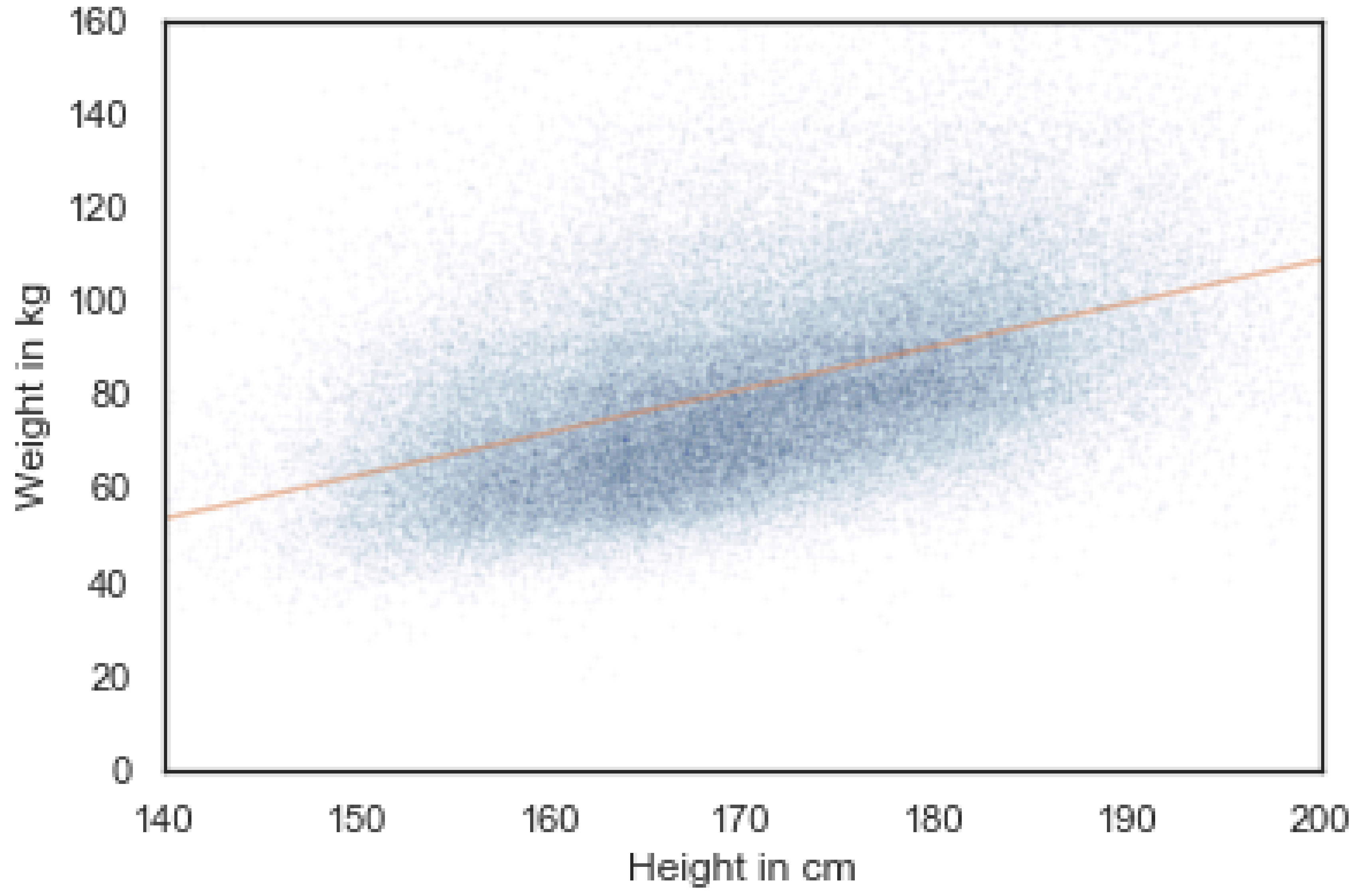
5. CDF, PMF, and KDE
I recommended using CDFs as the best way to explore distributions. But when you present to audiences that are not familiar with CDFs, you can use PMFs if there are a small number of unique values, and KDE if there are a lot

Visualizing relationships



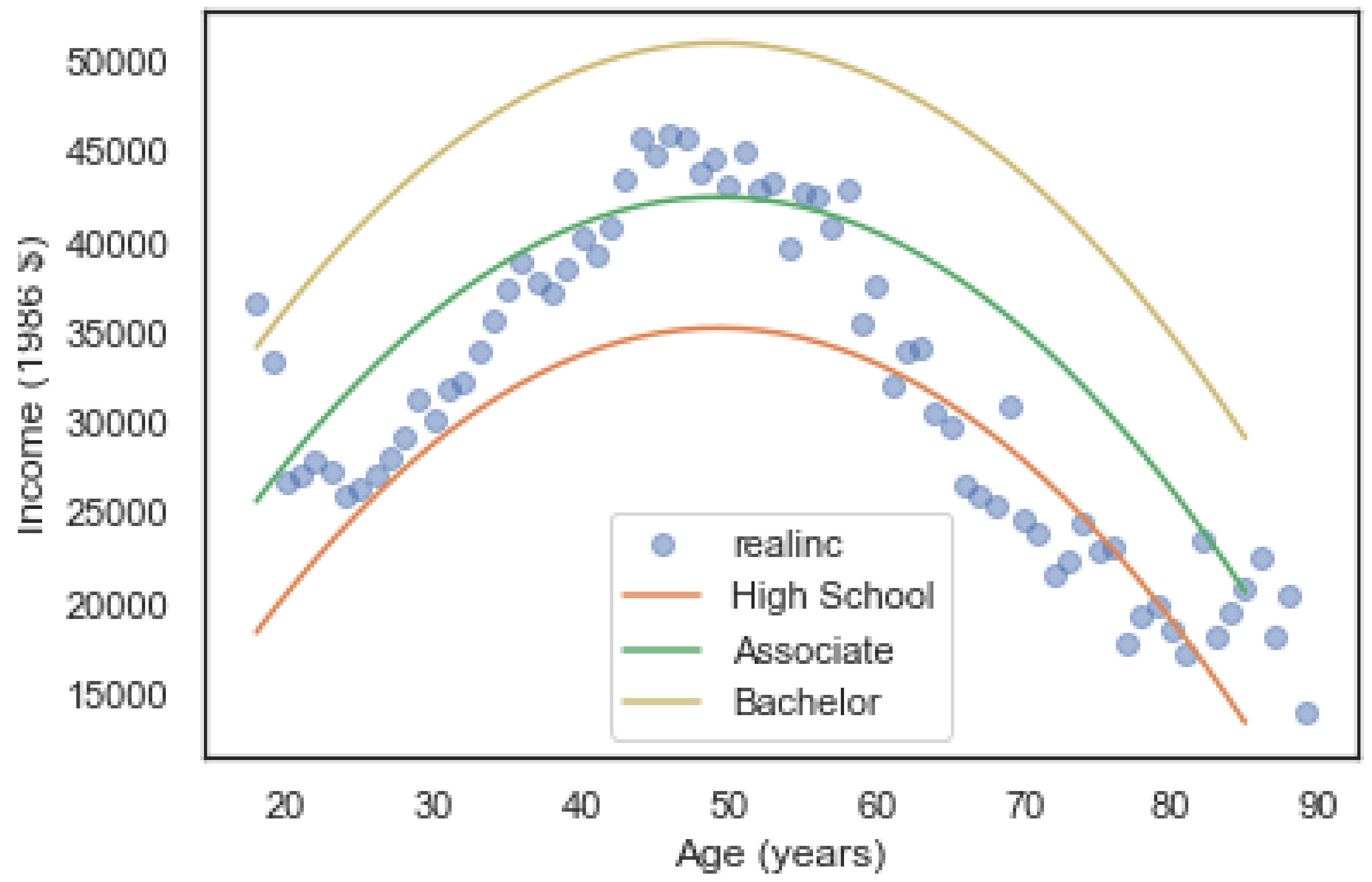
6. Visualizing relationships
In Chapter 3 we looked at heights and weights from the BRFSS, and developed several ways to visualize relationships between variables, including scatter plots, violin plots, and box plots like this one.

Quantifying correlation



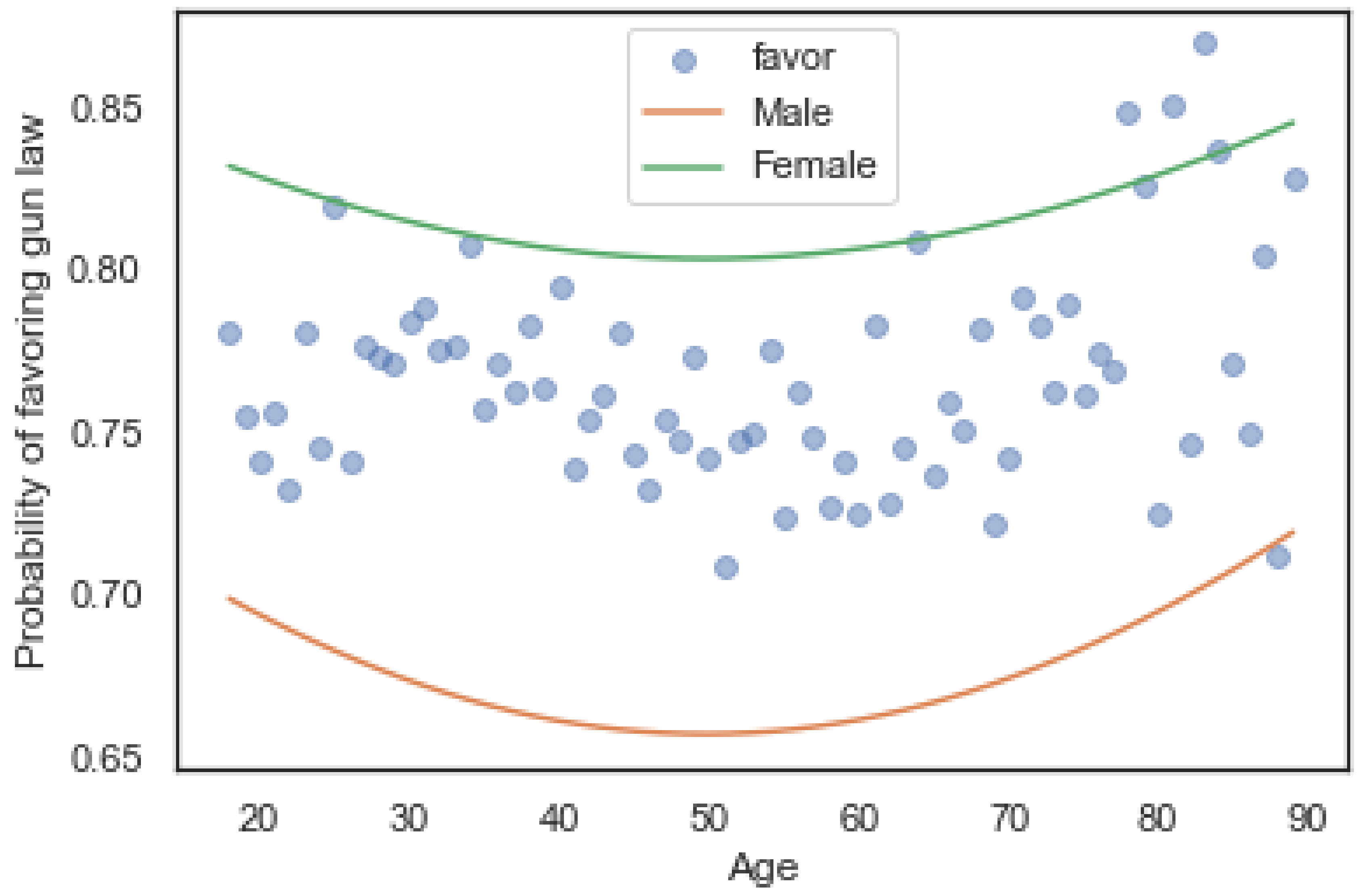
7. Quantifying correlation
We used the coefficient of correlation to quantify the strength of a relationship. We also used simple regression to find the line of best fit, like the one here that shows weight as a function of height. But remember that both of these methods only capture linear relationships; 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



8. Multiple regression
In Chapter 4 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



9. Logistic regression
And finally 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?

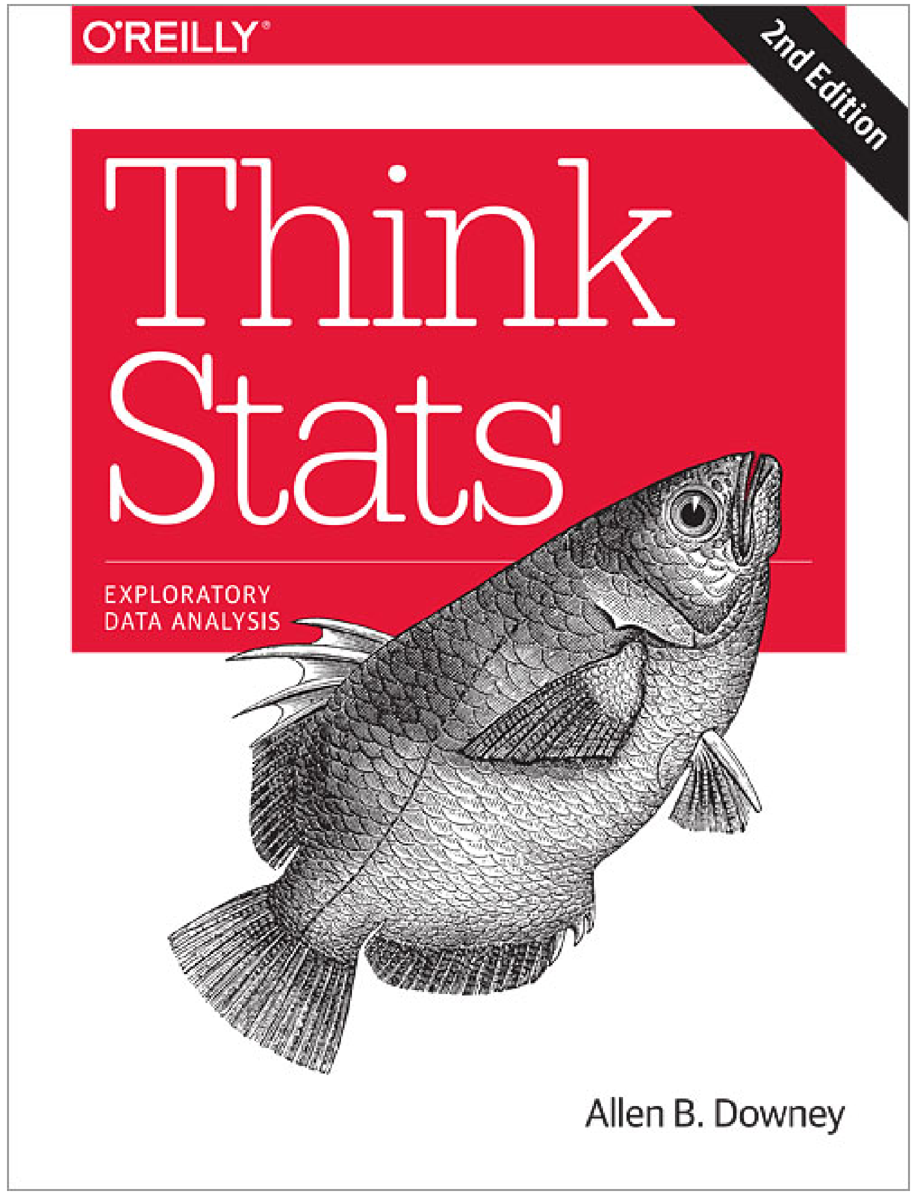
- Statistical Thinking in Python
- pandas Foundations
- Improving Your Data Visualizations in Python
- Introduction to Linear Modeling in Python

10. Where to next?
This course makes a great foundation for other DataCamp courses. Maybe you may want to learn more about CDFs, or data visualization, or pandas. DataCamp has courses that cover all of these topics and more, and I encourage you to check them out. On this slide are a few recommended next courses.

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