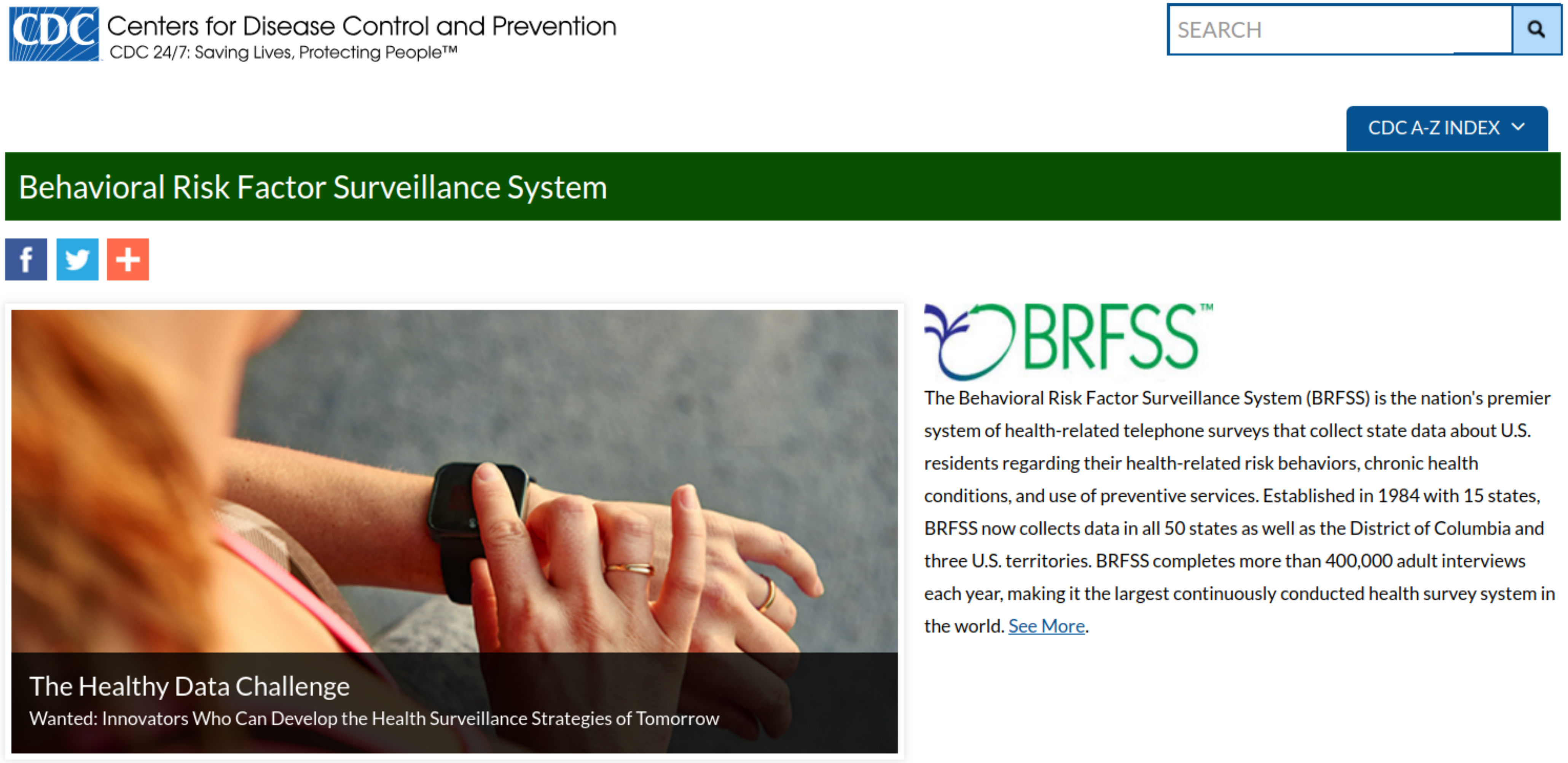
Exploring relationships

EXPLORATORY DATA ANALYSIS IN PYTHON

Allen Downey
Professor, Olin College

datacamp

Height and weight



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

Behavioral Risk Factor Surveillance System

The Healthy Data Challenge

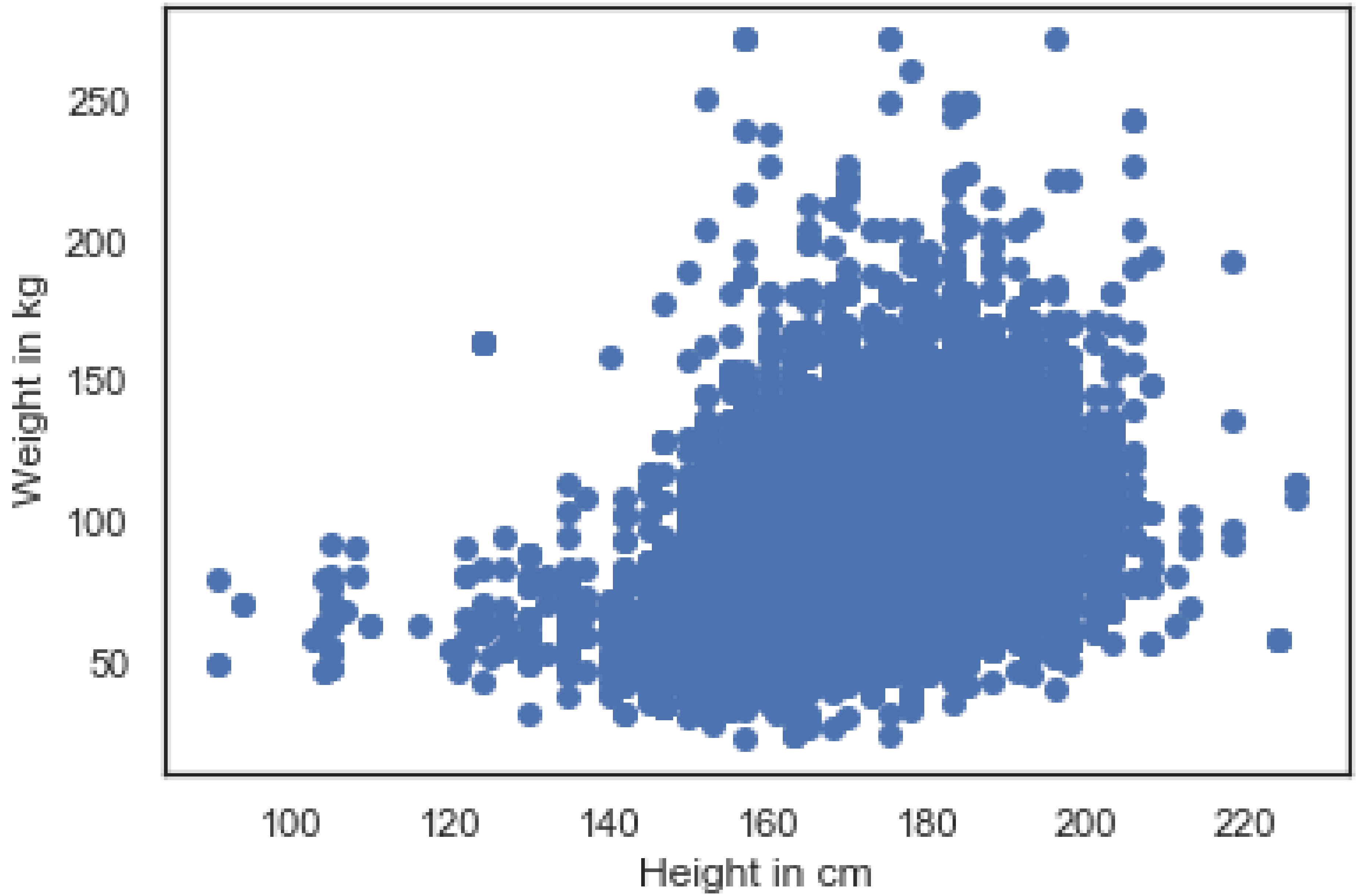
Wanted: Innovators Who Can Develop the Health Surveillance Strategies of Tomorrow

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.

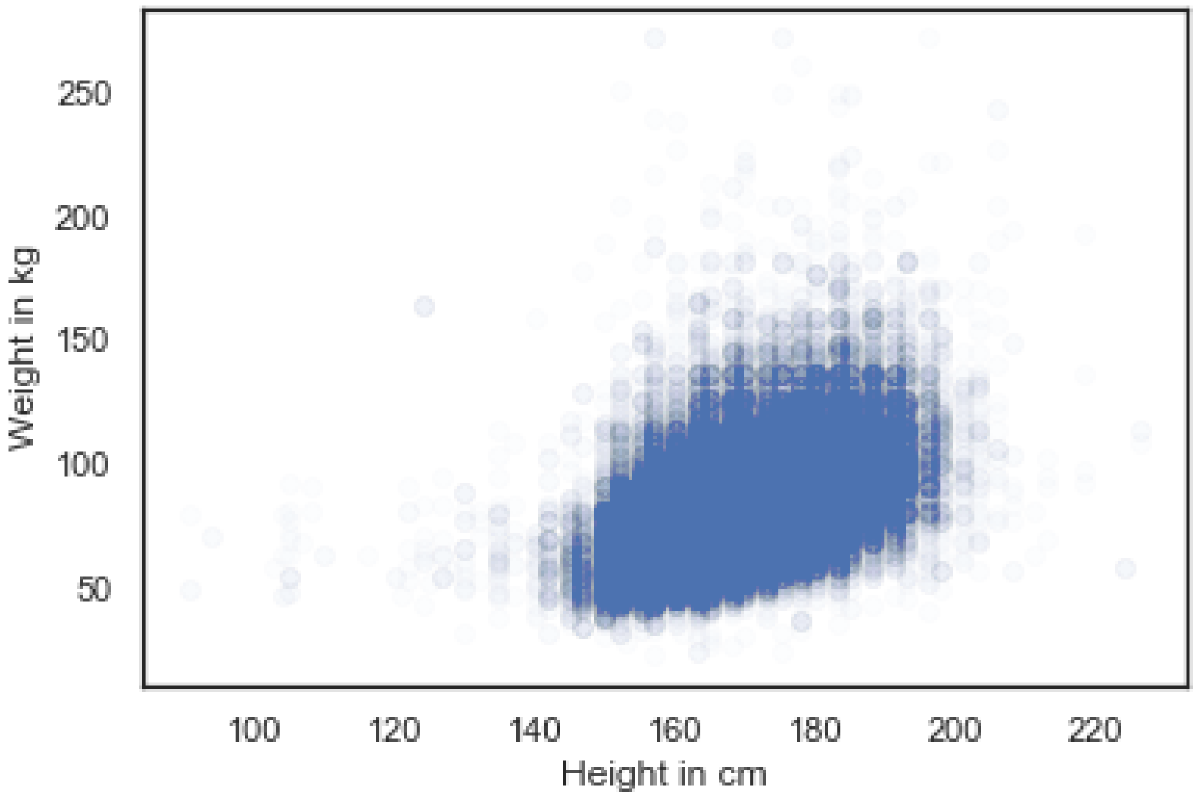
Scatter plot

```
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()
```


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

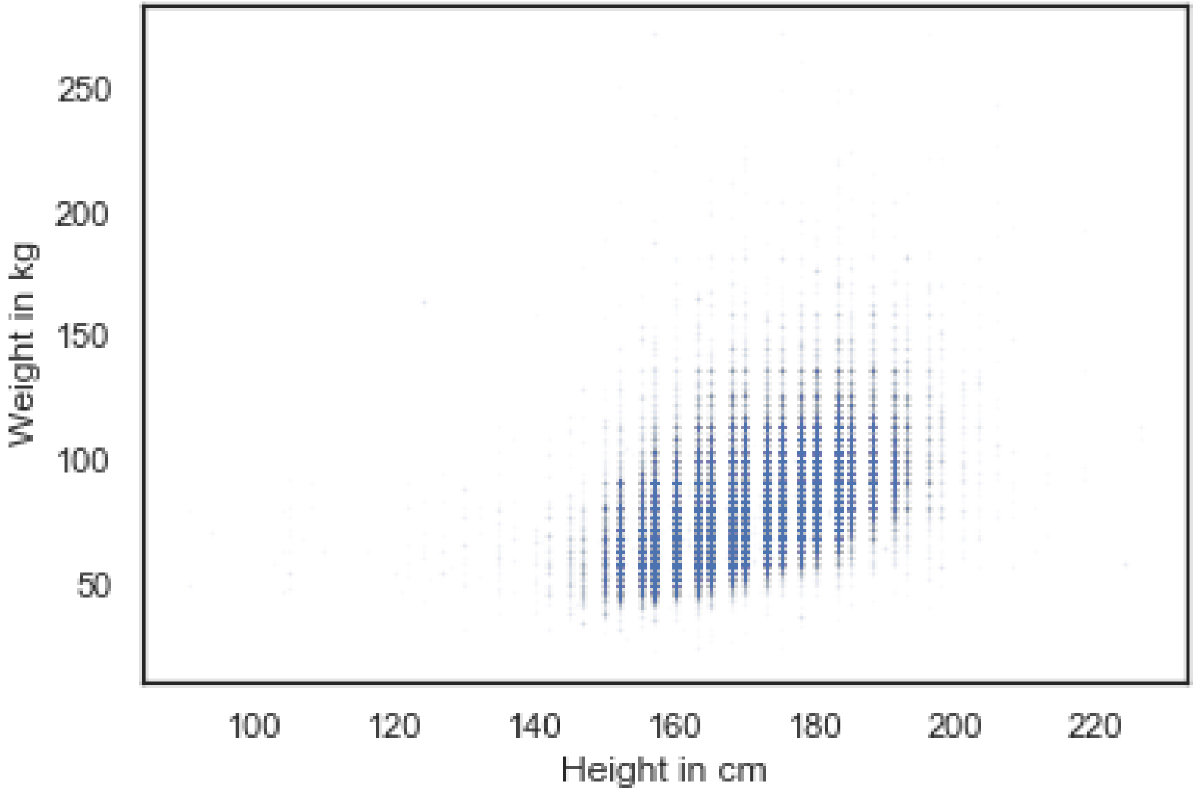
Transparency

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



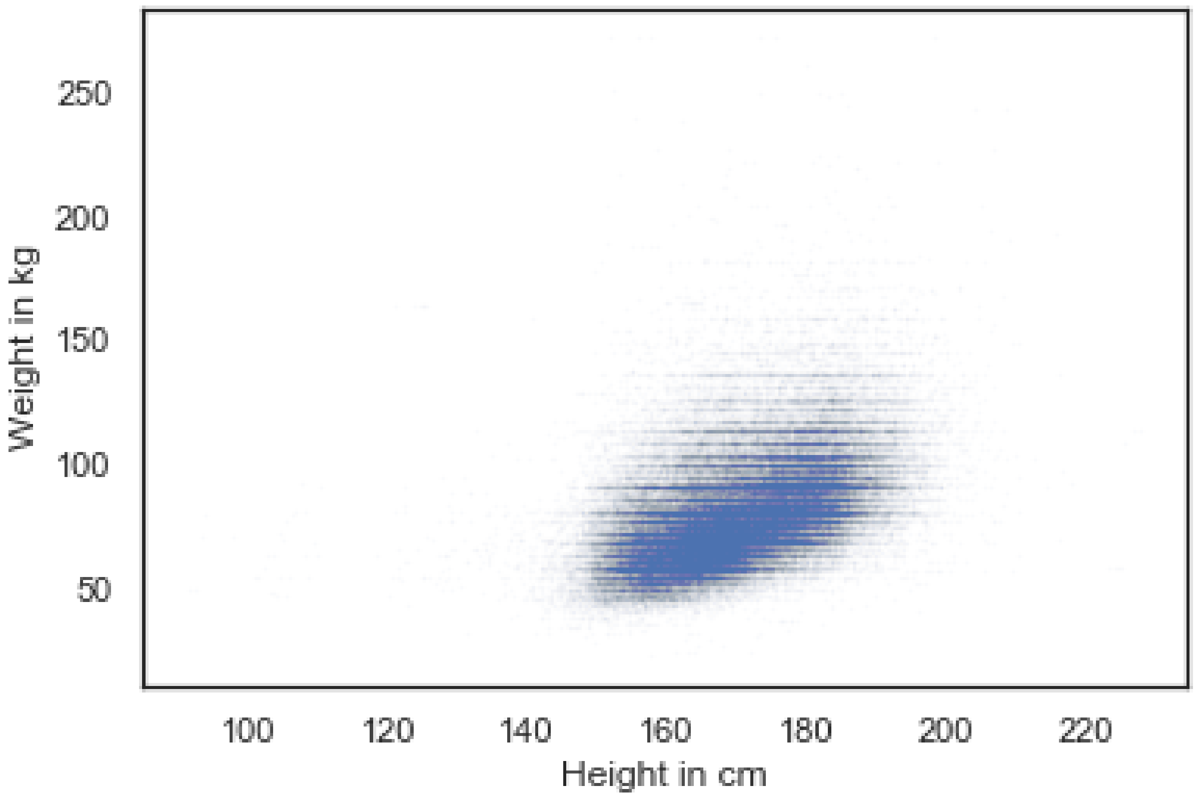
Marker size

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



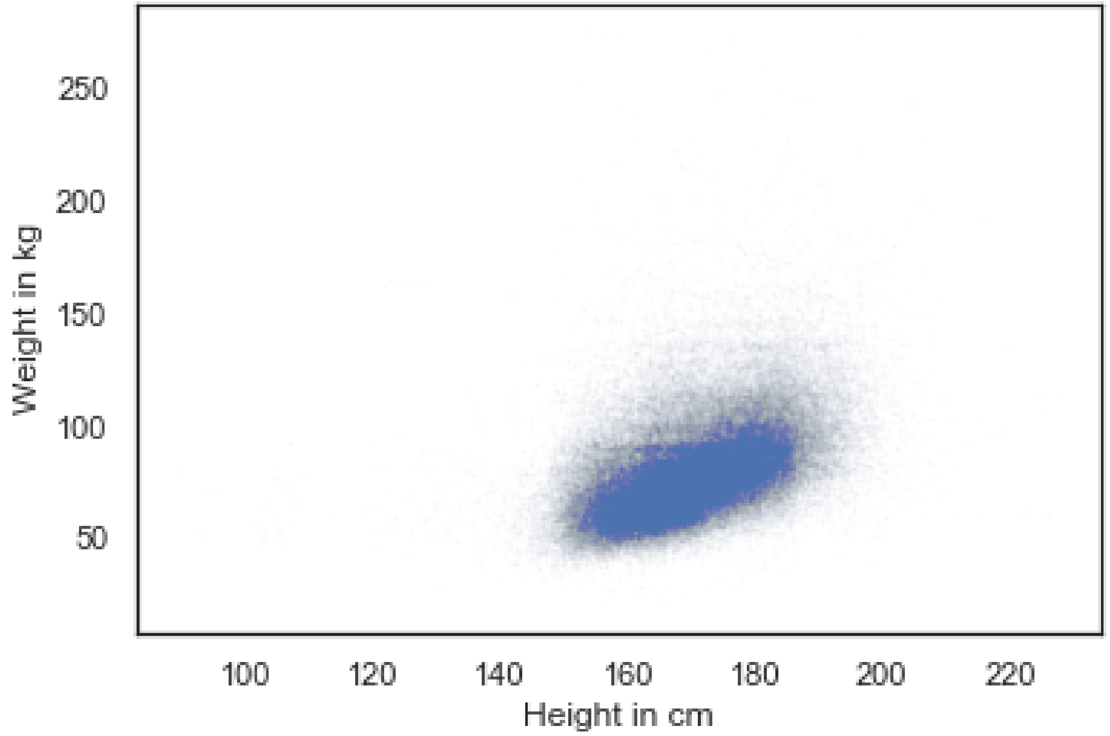
Jittering

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



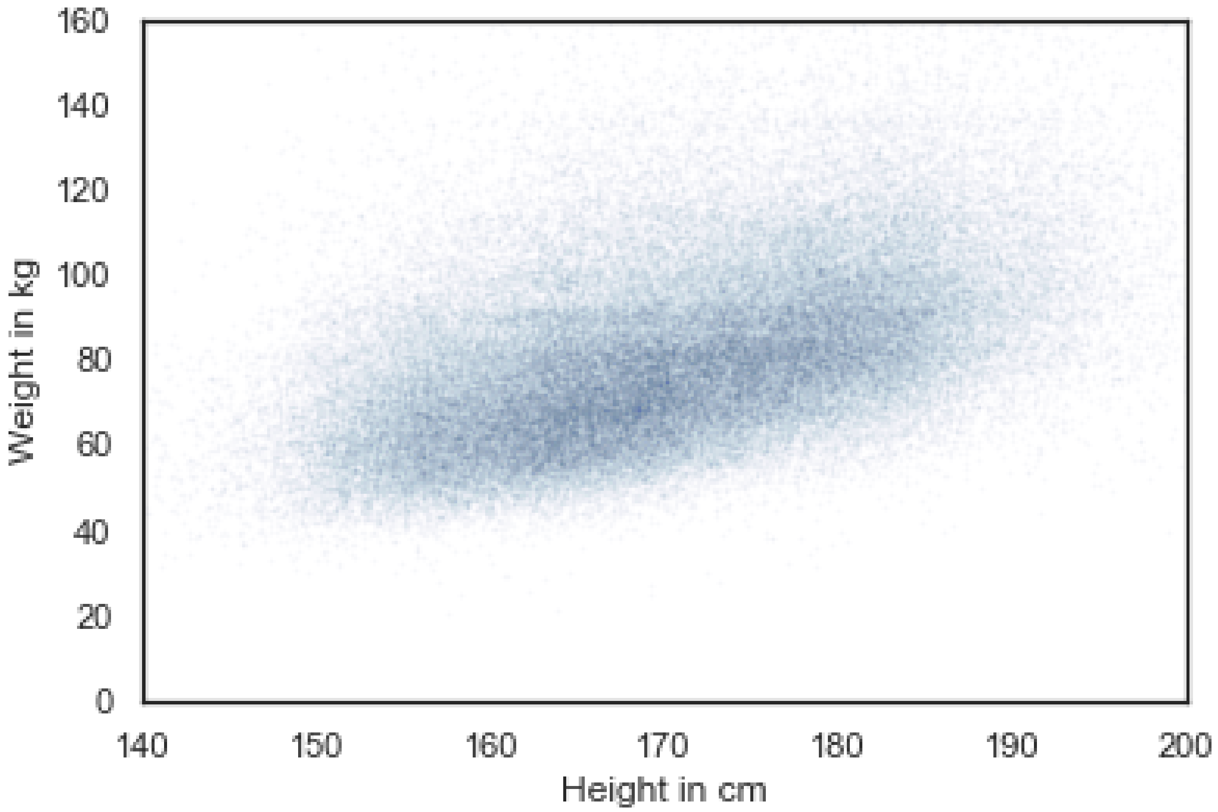
More jittering

```
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()
```

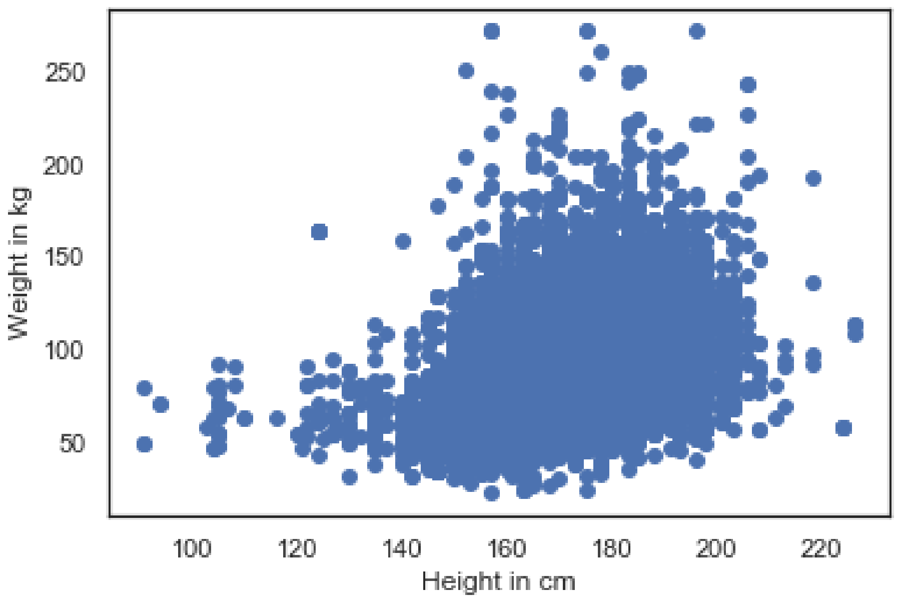


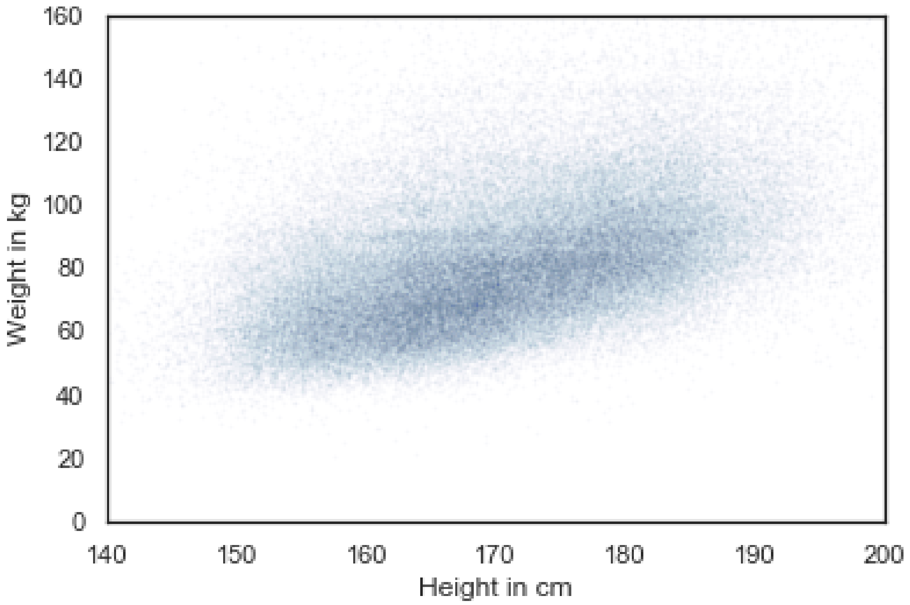
Zoom

```
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

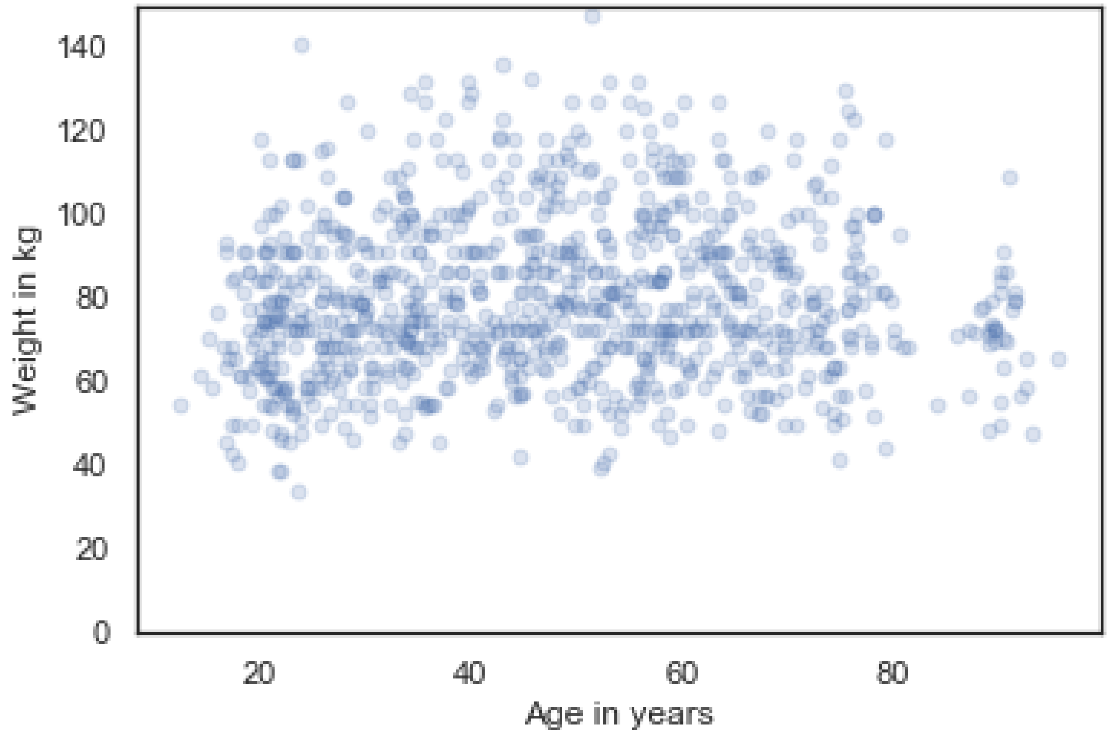
EXPLORATORY DATA ANALYSIS IN PYTHON

Allen Downey
Professor, Olin College

datacamp

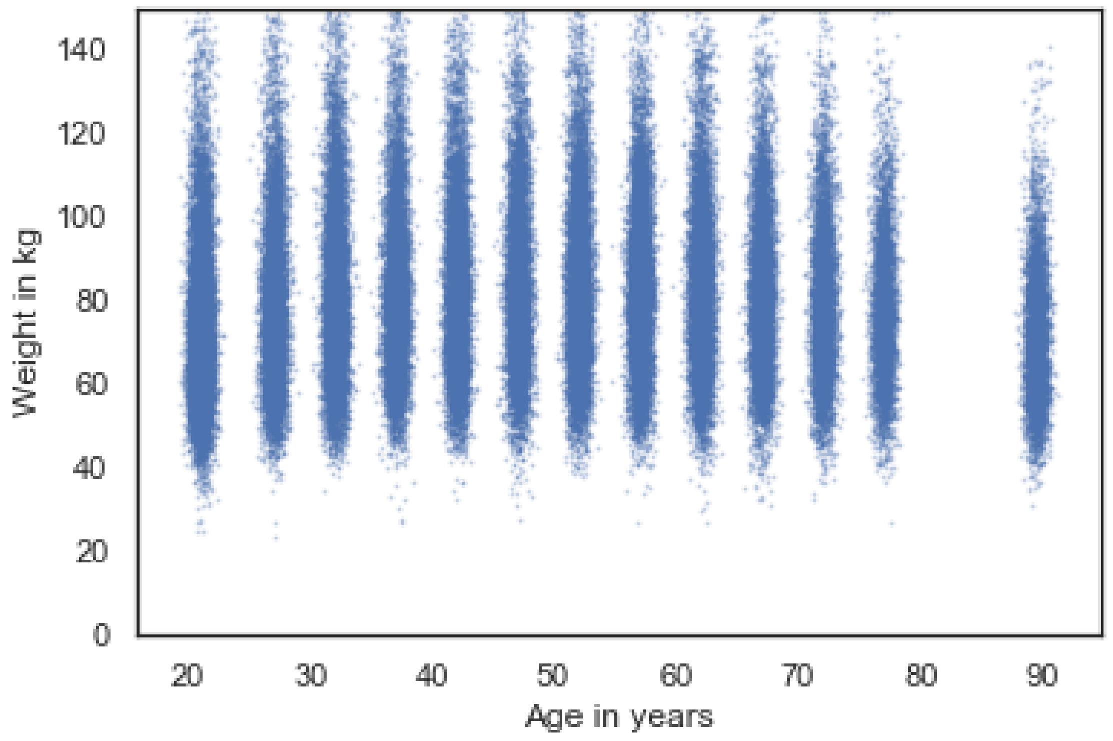
Weight and age

```
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

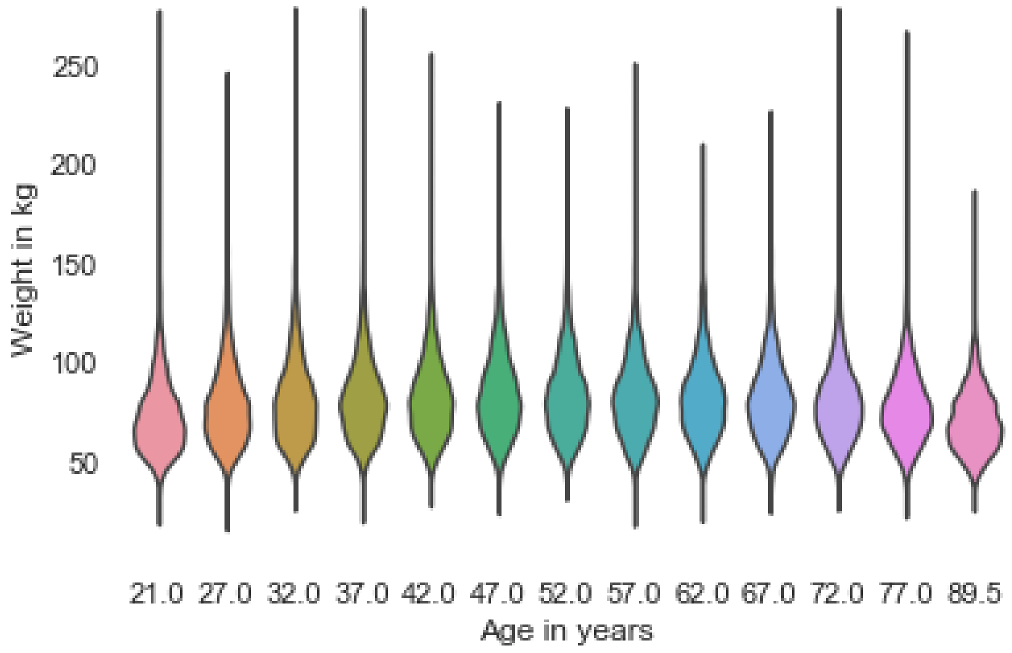
```
age = brfss['AGE'] + np.random.normal(0, 0.5, size=len(brfss))
weight = brfss['WTKG3'] + np.random.normal(0, 2, size=len(brfss))
plt.plot(age, weight, 'o', markersize=1, alpha=0.2)
plt.show()
```



3. More data
For the exercises, you worked with a small subset of the data. Now let's see what it looks like with more data. Here's the code. And here's the plot. I made a few changes in the code: * First, I reduced the marker size, because we have more data now, * Second, I jittered the weights, so the horizontal rows are not visible. * I jitter the ages, too, but less than in the exercise, so the data points are spread out, but there's still space between the columns. That makes it possible to see the shape of the distribution in each age group, and the differences between groups. If we take this idea one step farther, we can use KDE to estimate the density function in each column and plot it.

Violin plot

```
data = brfss.dropna(subset=['AGE', 'WTKG3'])
sns.violinplot(x='AGE', y='WTKG3', data=data, inner=None)
plt.show()
```

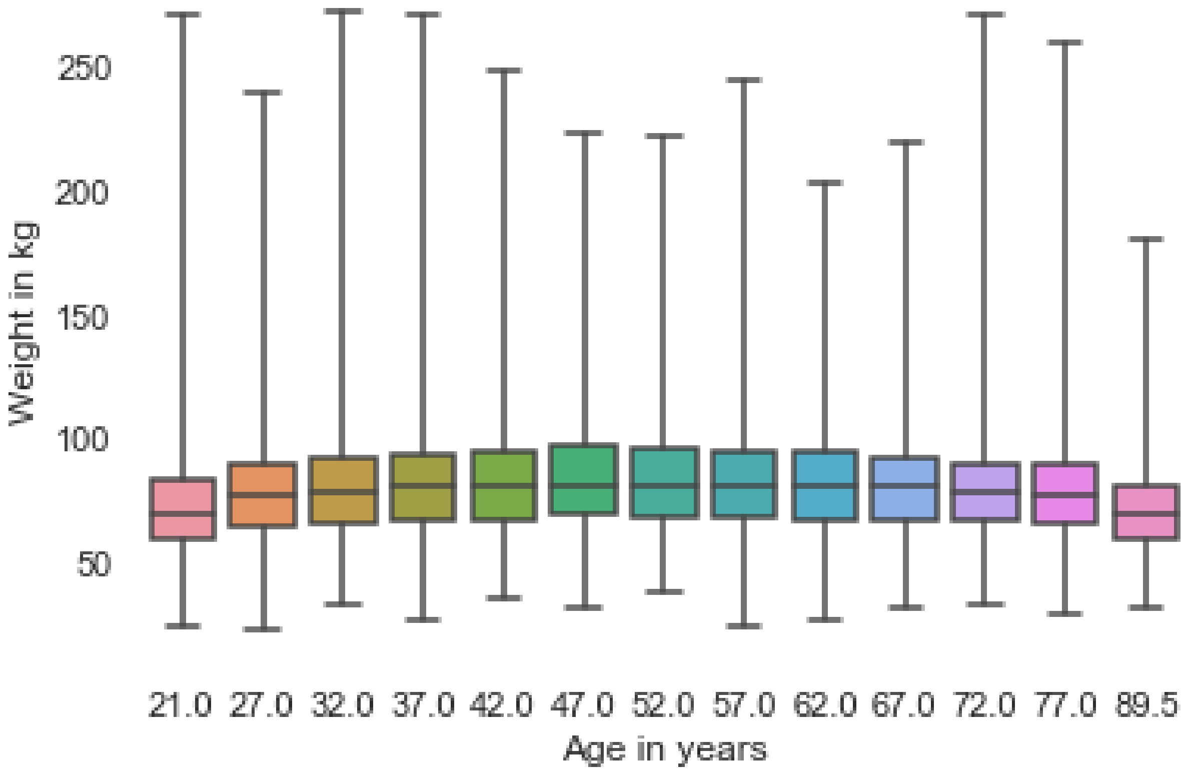


4. Violin plot

And there's a name for that; it's called a violin plot. Seaborn provides a function that makes violin plots, but before we can use it, we have to get rid of any rows with missing data. Here's how. dropna() creates a new DataFrame that contains the rows from brfss where AGE and WTKG3 are not NaN. Now we can call violinplot(). The x and y parameters mean we want AGE on the x-axis and WTKG3 on the y-axis. data is the DataFrame we just created, which contains the variables we're going to plot. The parameter inner=None simplifies the plot a little. Here's what it looks like. Each column is a graphical representation of the distribution of weight in one age group. The width of these shapes is proportional to the estimated density, so it's like two vertical PDFs plotted back to back, and filled in with nice colors. There's one other way to look at data like this, called a box plot.

Box plot

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

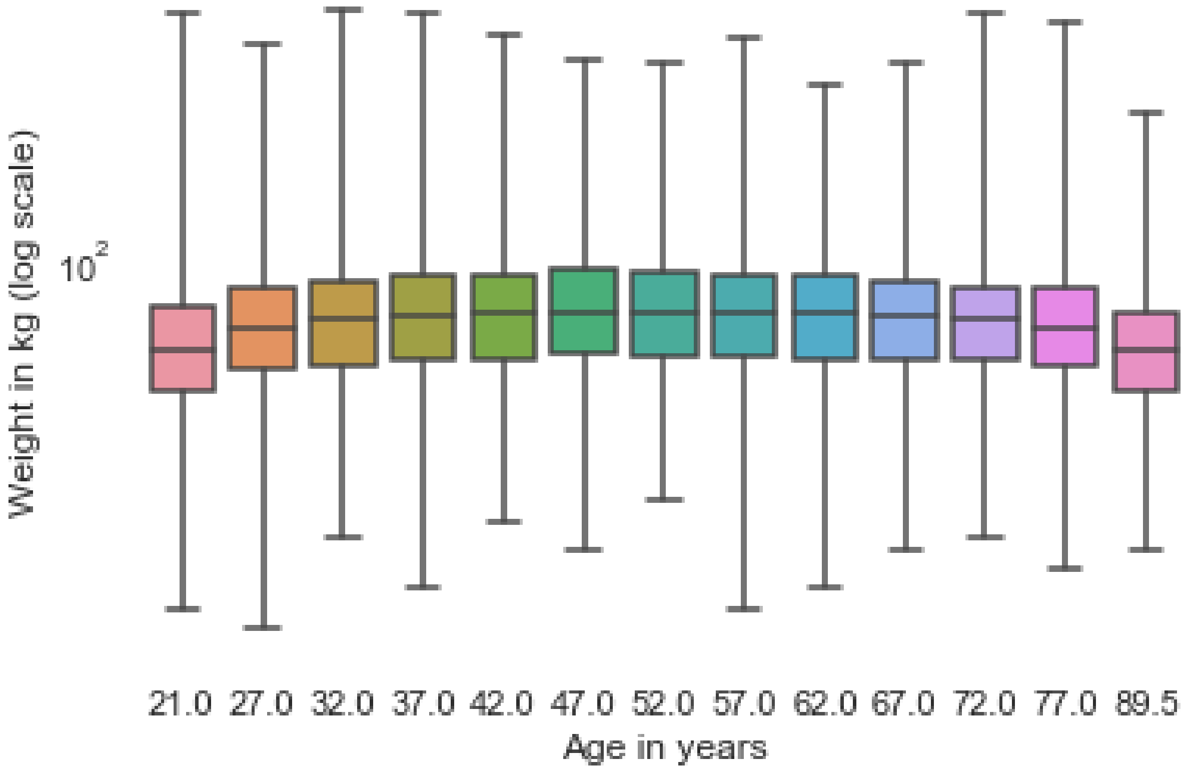


5. Box plot

The code to generate a box plot is very similar. I put in the parameter whis=10 to turn off a feature we don't need. If you are curious about it, you can read the documentation or check out DataCamp's Seaborn courses. Here's what it looks like. Each box represents the interquartile range, or IQR, from the 25th to the 75th percentile. The line in the middle of each box is the median. The spines sticking out of the top and bottom show the minimum and maximum values. In my opinion, this plot gives us the best view of the relationship between weight and age. Looking at the medians, it seems like people in their 40s are the heaviest; younger and older people are lighter. Looking at the sizes of the boxes, it seems like people in their 40s have the most variability in weight, too. These plots also show how skewed the distribution of weight is; that is, the heaviest people are much farther from the median than the lightest people.

Log scale

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



6. Log scale
For data that skews toward higher values, it is sometimes useful to look at it on a logarithmic scale. We can do that with the pyplot function yscale(). Here's what it looks like. To show the relationship between age and weight most clearly, this is probably the figure I would use.

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()
```

2. Correlation coefficient
When people say "correlation" casually, they might mean any relationship between two variables. In statistics, it usually means Pearson's correlation coefficient, which is a number between -1 and 1 that quantifies the strength of a linear relationship between variables. To demonstrate, I'll select three columns from the BRFSS dataset, like this. The result is a DataFrame with just those columns. Now we can use the corr() method, like this.

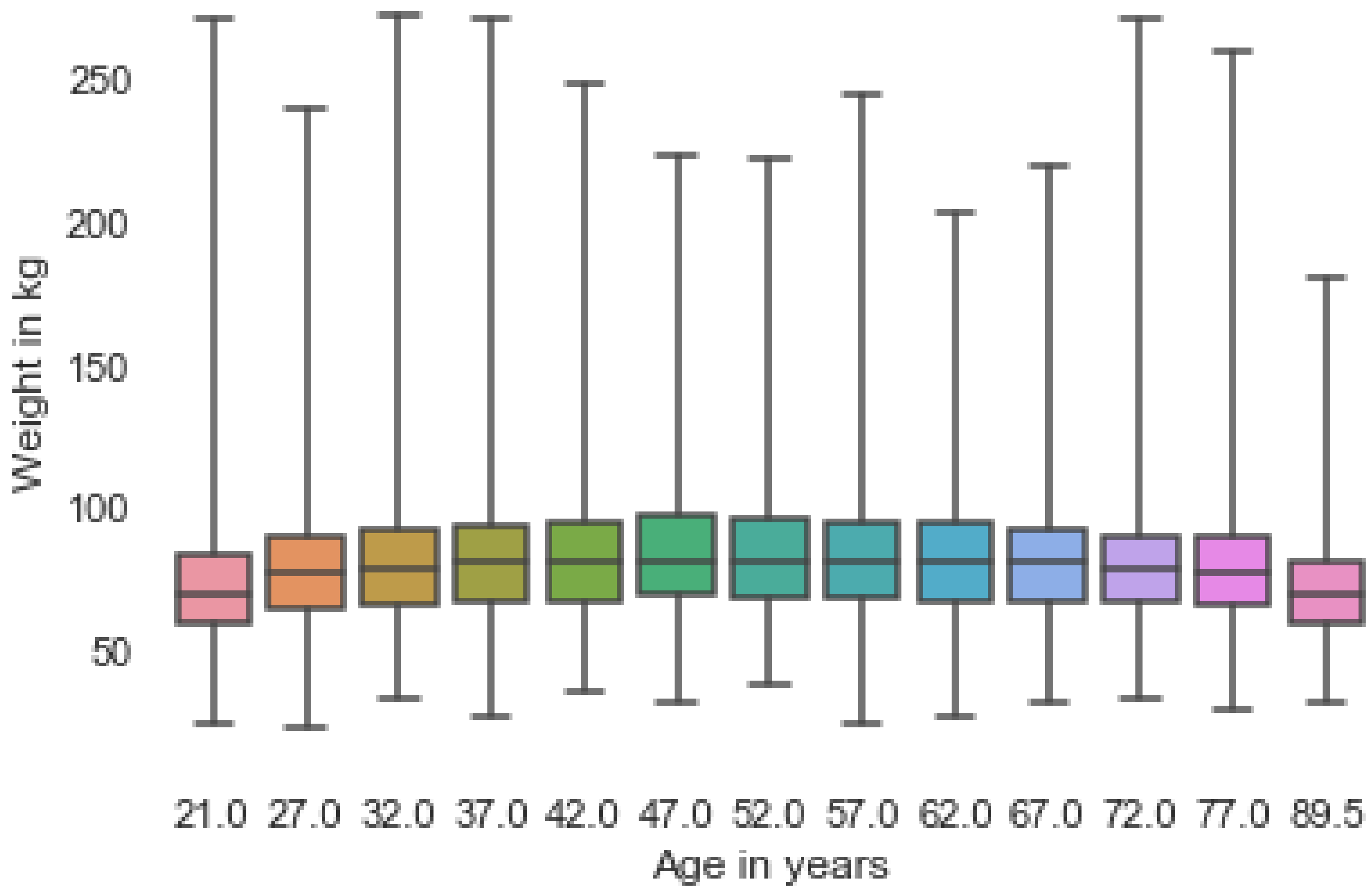
Correlation matrix

```
         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

3. Correlation matrix

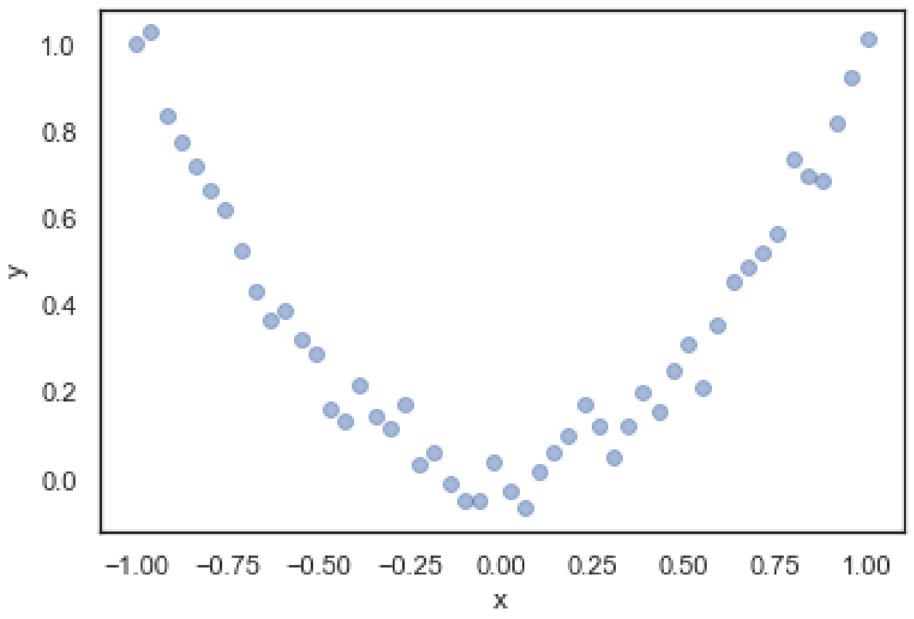
The result is a "correlation matrix". Reading across the first row, the correlation of HTM4 with itself is 1. That's expected; the correlation of anything with itself is 1. The next entry is more interesting; the correlation of height and weight is about 0 point 47. It's positive, which means taller people are heavier, and it is moderate in strength, which means it has some predictive value. If you know someone's height, you can make a better guess about their weight, and vice versa. The correlation between height and age is about -0 point 09. It's negative, which means that older people tend to be shorter, but it's weak, which means that knowing someone's age would not help much if you were trying to guess their height. The correlation between age and weight is even smaller. It is tempting to conclude that there is no relationship between age and weight, but we have already seen that there is. So why is the correlation so low?



4. Weight and age
Remember that the relationship between weight and age looks like this. People in their 40s are the heaviest; younger and older people are lighter. So this relationship is nonlinear.

```python
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.        ]])
```



5. Nonlinear relationships

But correlation only works for linear relationships. If the relationship is nonlinear, correlation generally underestimates how strong it is. To demonstrate, I'll generate some fake data: xs contains equally-spaced points between -1 and 1. ys is xs squared plus some random noise. Here's the scatter plot of xs and ys. It's clear that this is a strong relationship; if you are given `x`, you can make a much better guess about y. But here's the correlation matrix; the computed correlation is close to 0. In general, if correlation is high -- that is, close to 1 or -1, you can conclude that there is a strong linear relationship. But if correlation is close to 0, that doesn't mean there is no relationship; there might be a strong, non-linear relationship. This is one of the reasons I think correlation is not such a great statistic.

You keep using that word

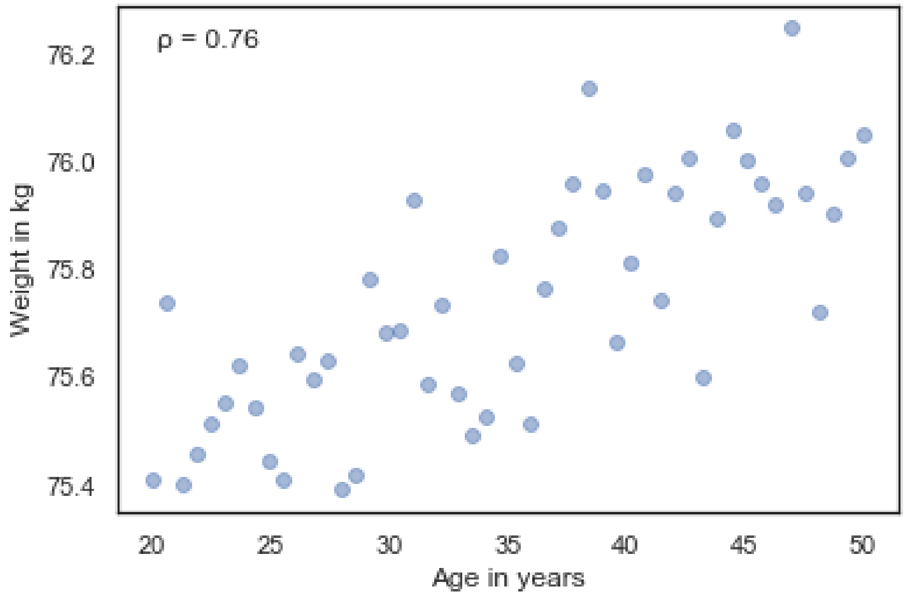
I do not think it means what you think it means

6. You keep using that word

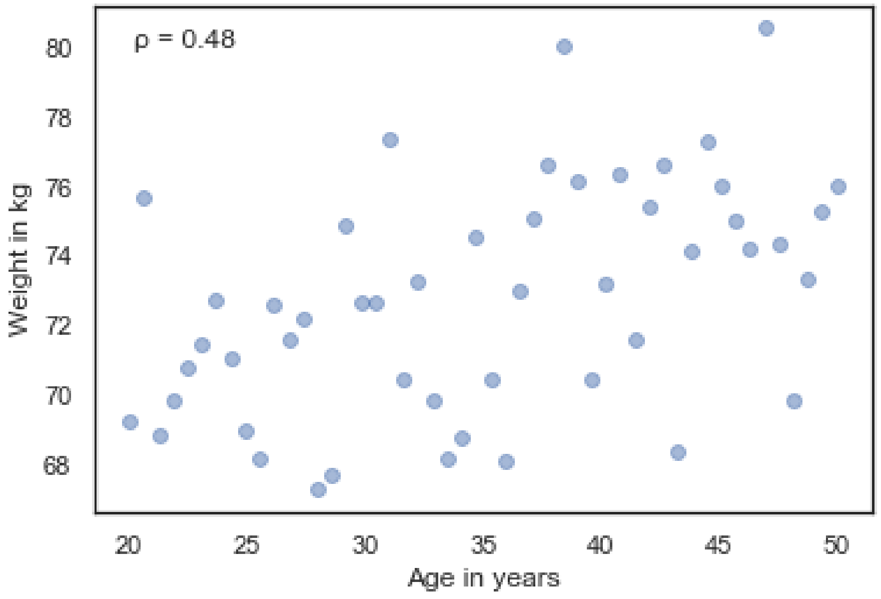
There's another reason to be careful with correlation; it doesn't mean what people take it to mean. Specifically, correlation says nothing about slope. If we say that two variables are correlated, that means we can use one to predict the other. But that might not be what we care about.

Strength of relationship

Hypothetical #1



Hypothetical #2



7. Strength of relationship

For example, suppose we are concerned about the health effects of weight gain, so we plot weight versus age, from 20 to 50 years old. Here are two fake datasets I generated. The one on the left has higher correlation, about 0 point 76 compared to 0 point 47. But on the left, the average weight gain over 30 years is less than 1 kg; on the right, it is almost 10 kilograms! In this scenario, the relationship on the right is probably more important, even though the correlation is lower. The statistic we really care about is the slope of the line.

Let's practice!

EXPLORATORY DATA ANALYSIS IN PYTHON

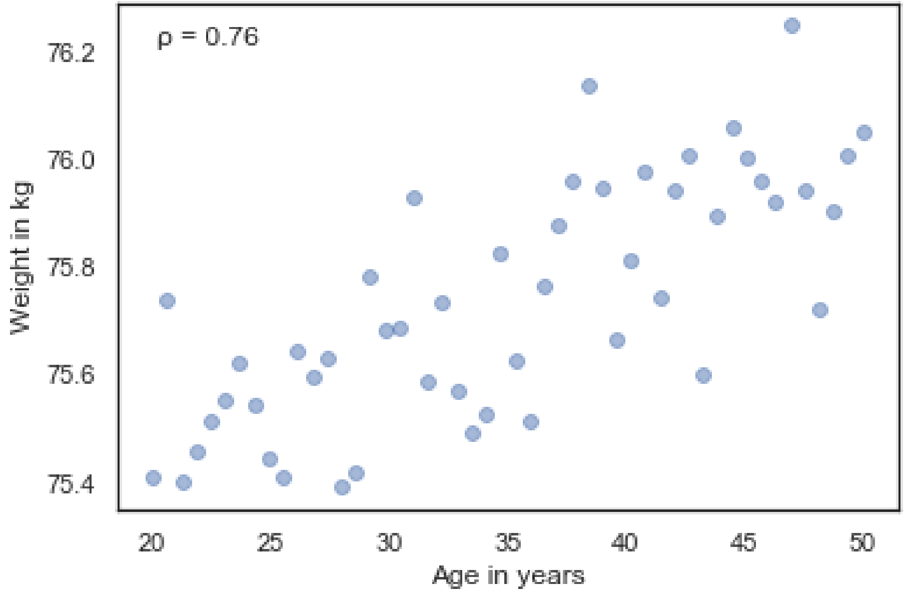
Simple regression

EXPLORATORY DATA ANALYSIS IN PYTHON

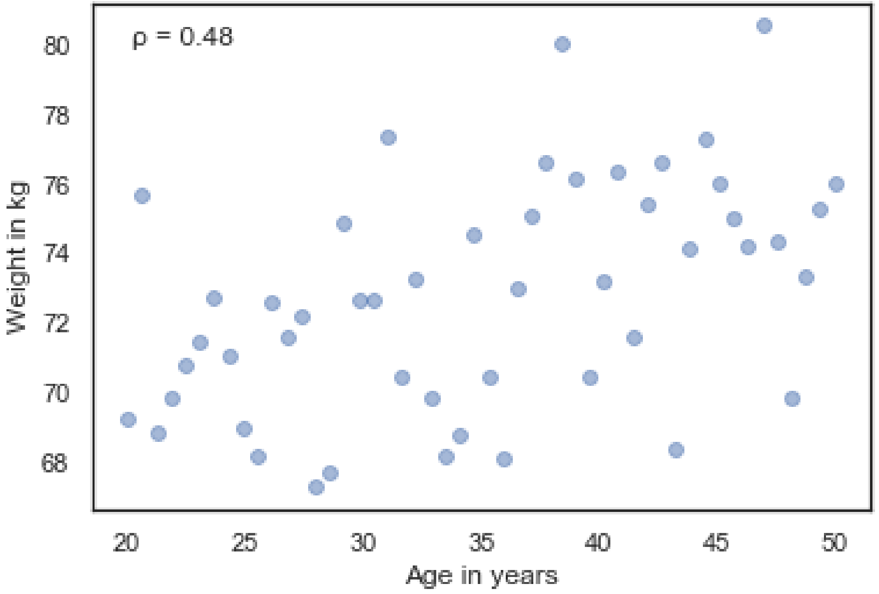
Allen Downey
Professor, Olin College

Strength of relationship

Hypothetical #1



Hypothetical #2



2. Strength of relationship

Let's look again at an example from the previous lesson, a hypothetical relationship between weight and age. I generated two fake datasets to make a point: The one on the left has higher correlation, about 0 point 76 compared to 0 point 48. But in the one on the left, the average weight gain over 30 years is less than 1 kg; on the right, it is almost 10 kilograms! In this context, the statistic we probably care about is the slope of the line, not the correlation coefficient.

Strength of effect

```python
from scipy.stats import linregress


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

```
LinregressResult(slope=0.018821034903244386,
            intercept=75.08049023710964,
                rvalue=0.7579660563439402,
                pvalue=1.8470158725246148e-10,
                stderr=0.002337849260560818)
```

Strength of effect

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

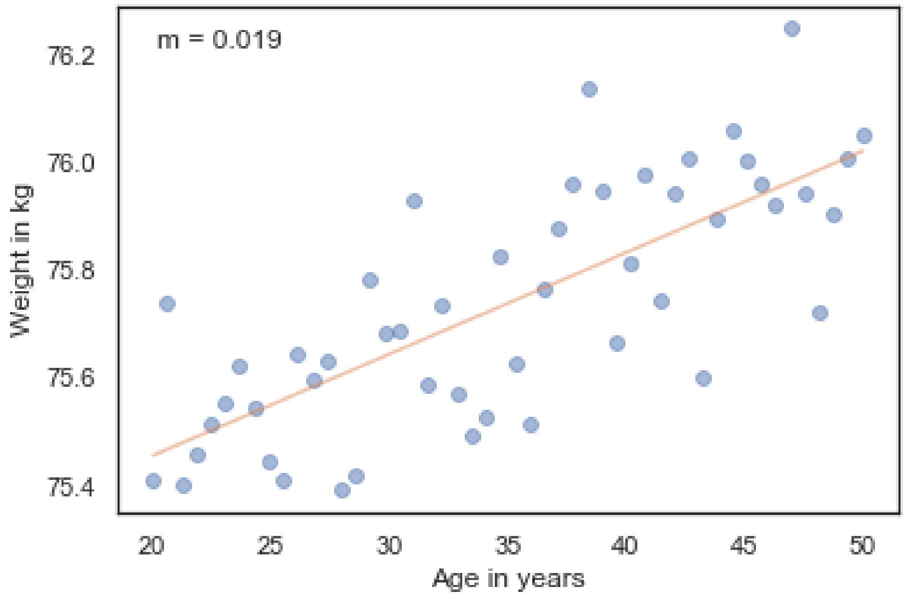
```
LinregressResult(slope=0.17642069806488855,
            intercept=66.60980474219305,
                rvalue=0.47827769765763173,
                pvalue=0.0004430600283776241,
                stderr=0.04675698521121631)
```

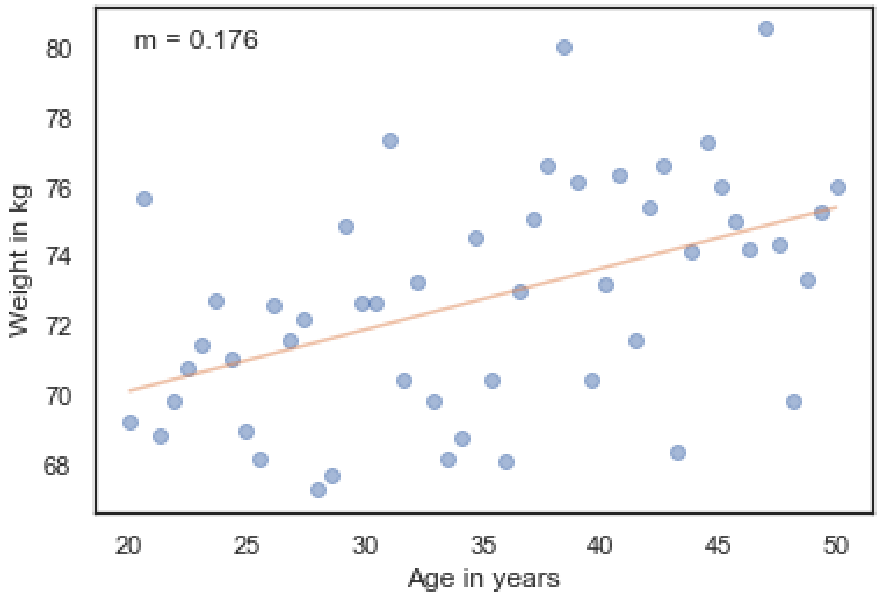
4. Strength of effect
Here are the results for Hypothetical #2. The estimated slope is about 10 times higher: about 0 point 18 kilograms per year or 6 kilograms per 30 years, What's called rvalue here is correlation, which confirms what we saw before; the first example has higher correlation, about 0 point 76 compared to 0 point 48. But the strength of the effect, as measured by the slope of the line, is about 10 times higher in the second example.

Regression lines

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

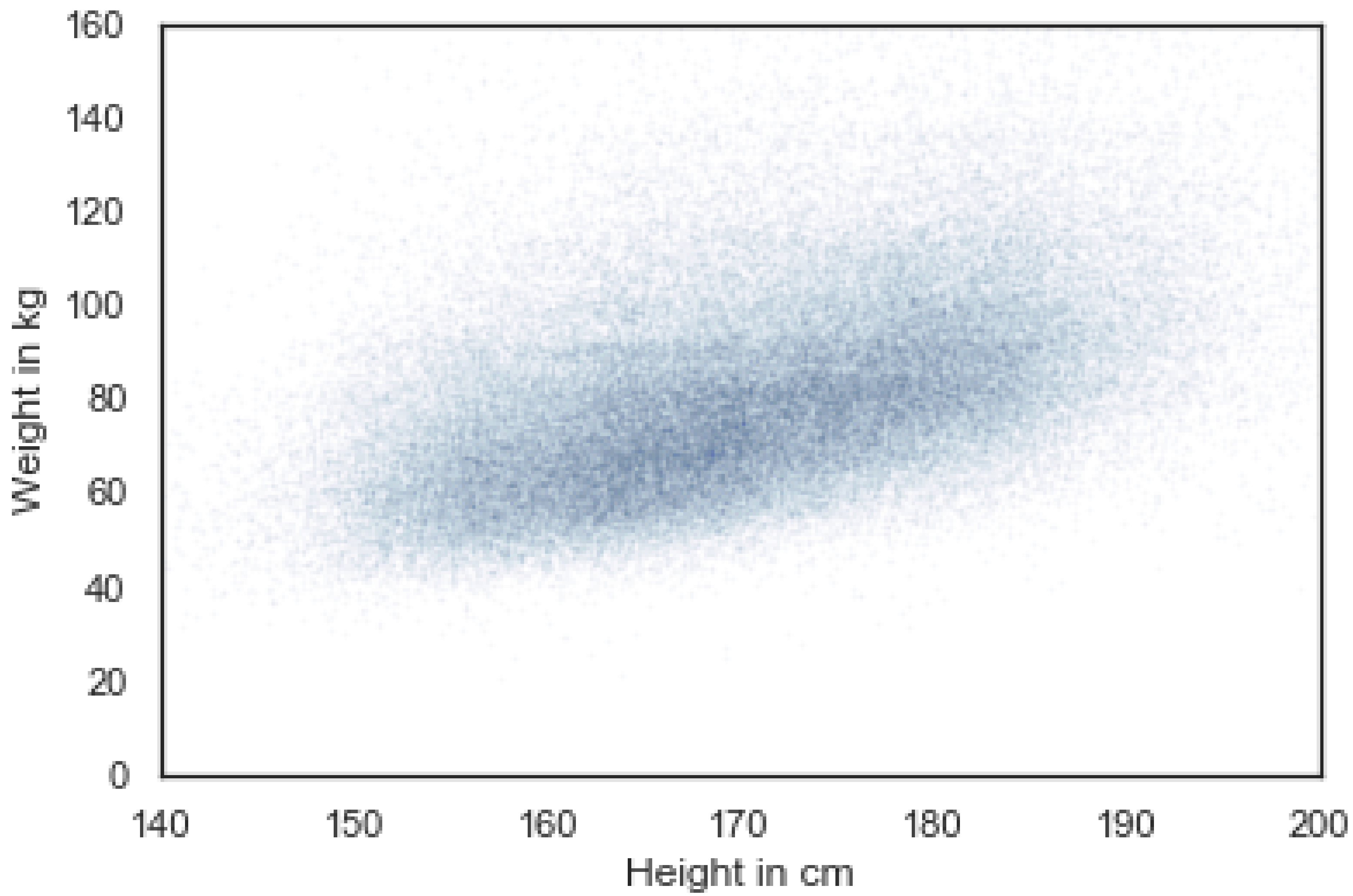
```
fx = ...
fy = ...
plt.plot(fx, fy, '-')
```





5. Regression lines

We can use the results from linregress() to compute the line of best fit: first we get the min and max of the observed xs; then we multiply by the slope and add the intercept. And plot the line. Here's what that looks like for the first example. And the same thing for the second example. The visualization here might be misleading unless you look closely at the vertical scales; the slope on the right is almost 10 times higher.



6. Height and weight
Now let's look at an example with real data. Here's the scatter plot of height and weight again, from Lesson 1.

Regression line

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

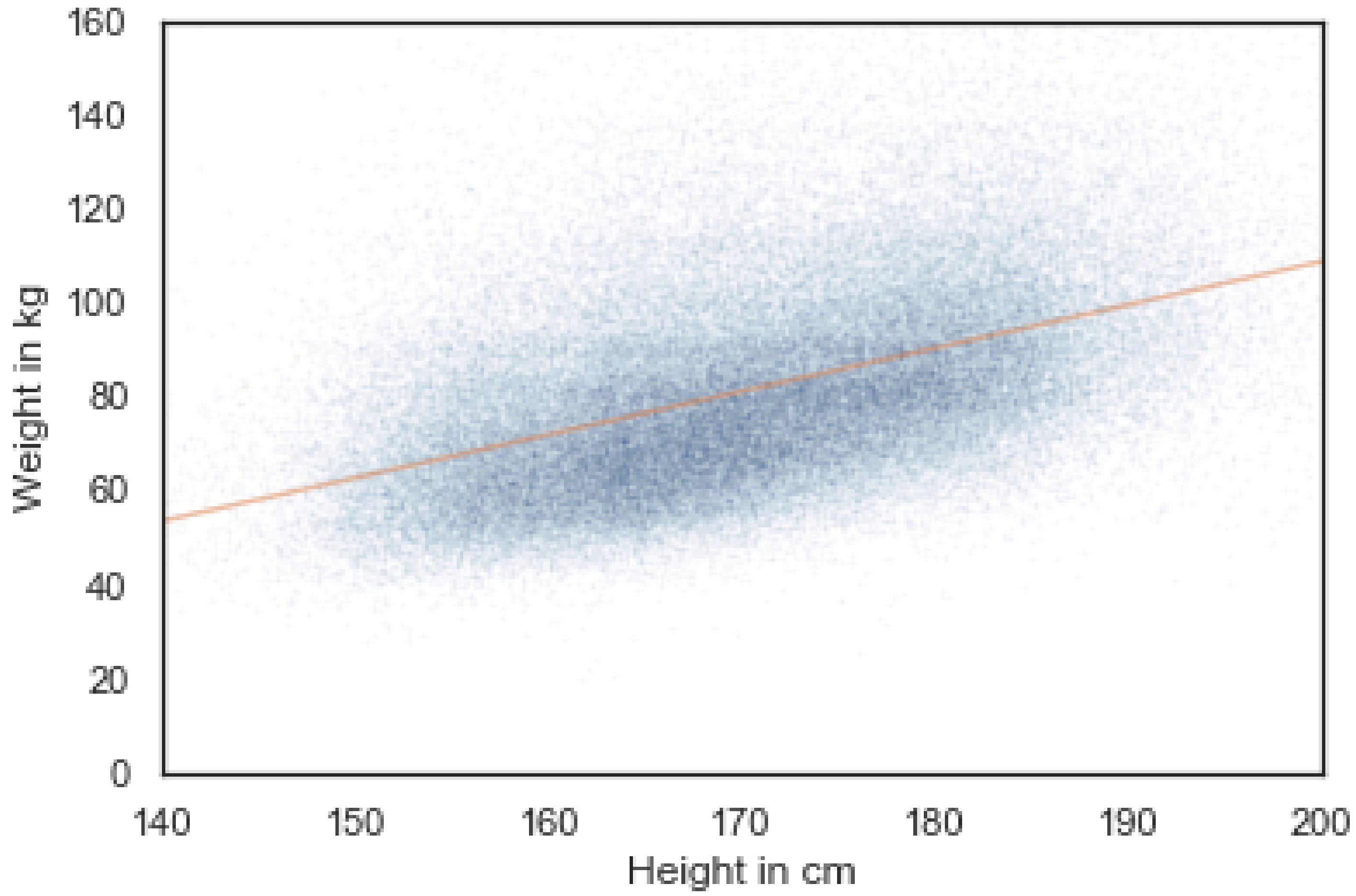
```
xs = subset['HTM4']
ys = subset['WTKG3']
res = linregress(xs, ys)
```

```
LinregressResult(slope=0.9192115381848297,
           intercept=-75.12704250330233,
                rvalue=0.47420308979024584,
                pvalue=0.0,
                stderr=0.005632863769802998)
```

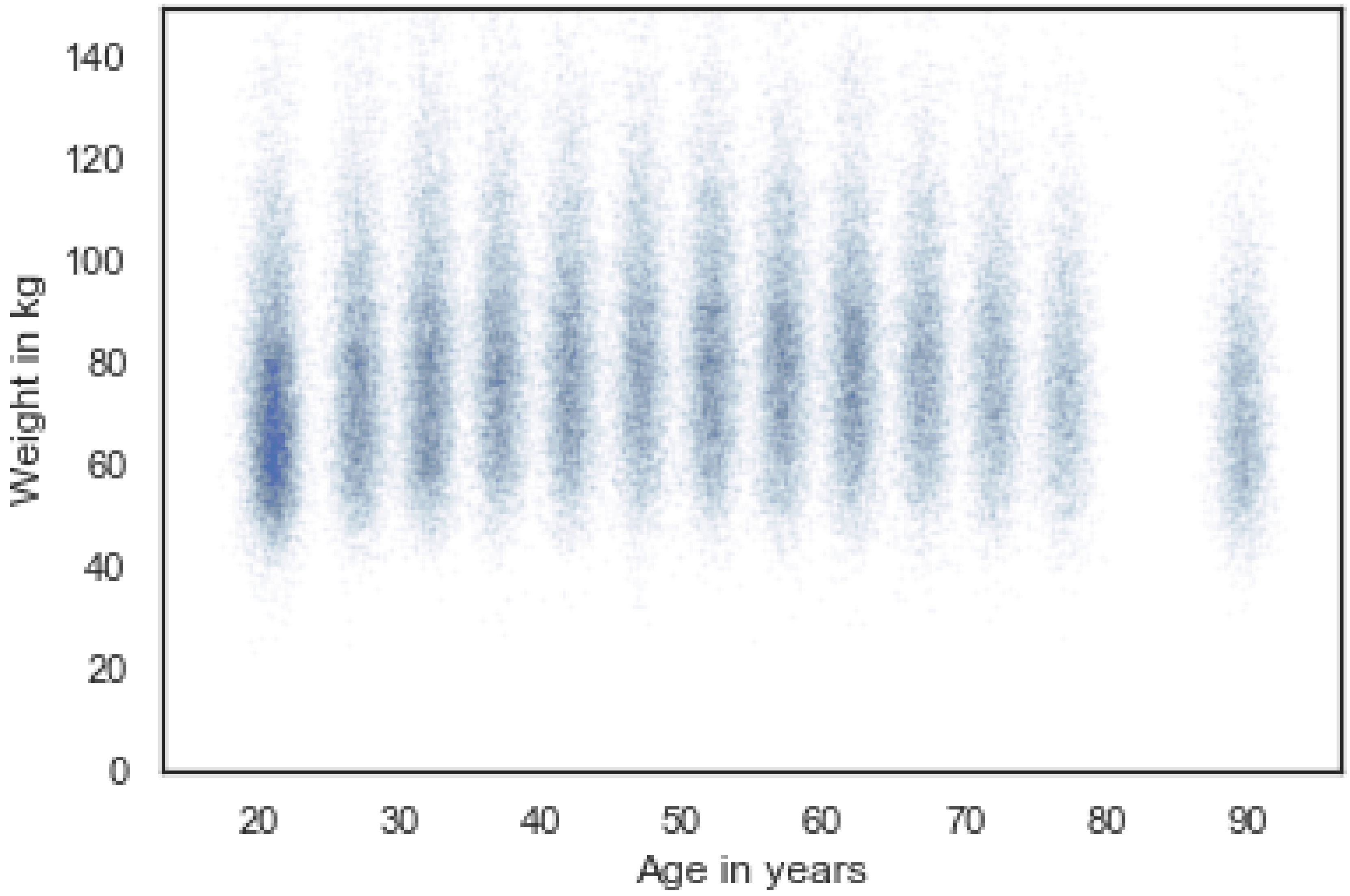
7. Regression line

Now we can compute the regression line. linregress() can't handle NaNs, so we have to use dropna() to remove rows that are missing the data we need. Now we can compute the linear regression. And here are the results. The slope is about 0 point 9 kilograms per centimeter, which means that we expect a person one centimeter taller to be almost a kilogram heavier. That's quite a lot.

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



Linear relationships



9. Linear relationships
However, linear regression has the same problem as correlation; it only measures the strength of a linear relationship. Here's the scatter plot of weight versus age, which you saw in a previous exercise. People in their 40s are the heaviest; younger and older people are lighter. So the relationship is nonlinear.

Nonlinear relationships

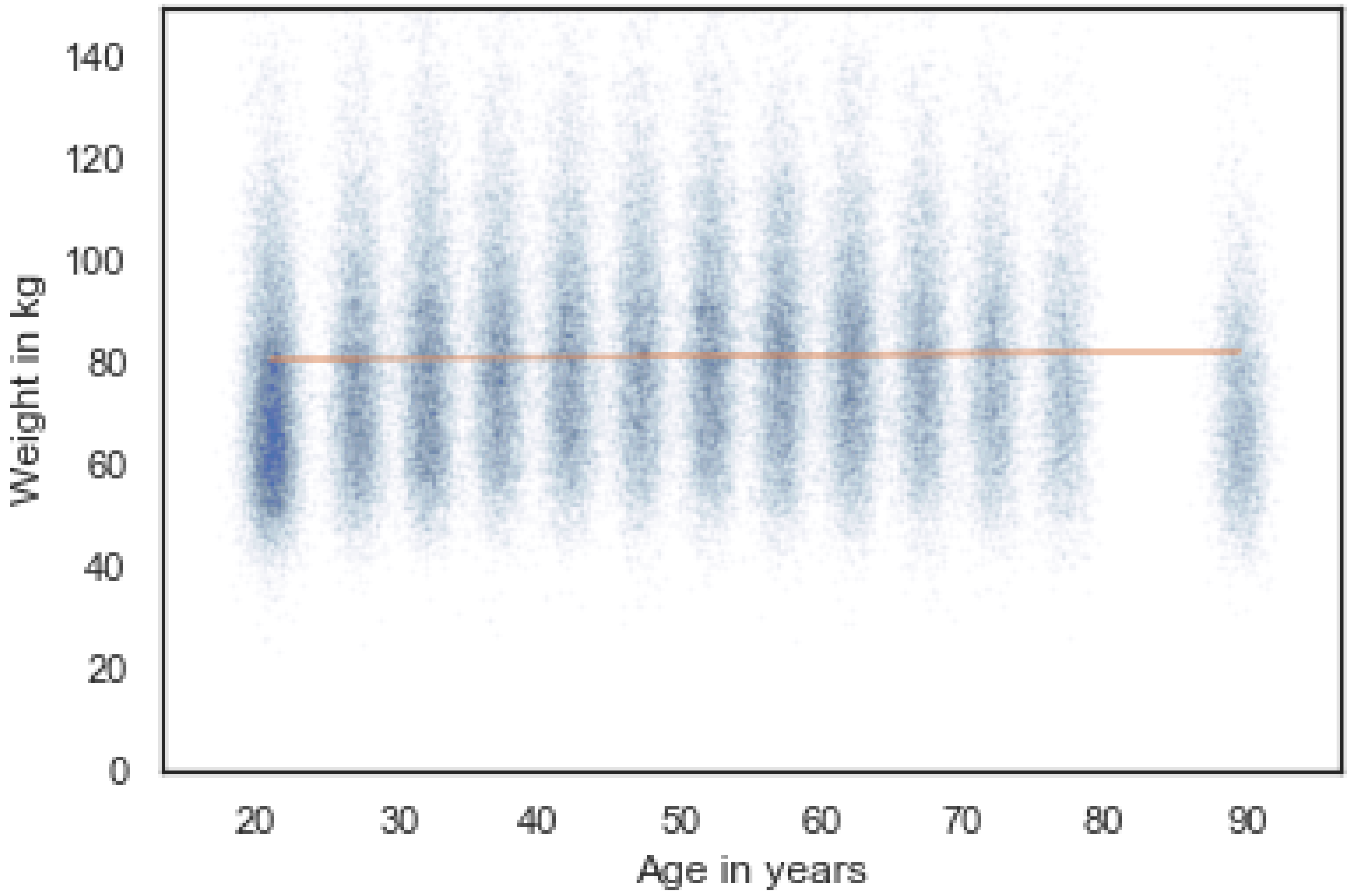
```python
subset = brfss.dropna(subset=['WTKG3', 'AGE'])
xs = subset['AGE']
ys = subset['WTKG3']

res = linregress(xs, ys)
```

```
LinregressResult(slope=0.023981159566968724,
            intercept=80.07977583683224,
                rvalue=0.021641432889064068,
                pvalue=4.374327493007566e-11,
                stderr=0.0036381394107421867)
```

10. Nonlinear relationships

If we don't look at the scatter plot and blindly compute the regression line, here's what we get. The estimated slope is only 0 point 02 kilograms per year, or 0 point 6 kilograms in 30 years.

Not a good fit



11. Not a good fit
And here's what the line of best fit looks like. A straight line does not capture the relationship between these variables well.

Let's practice!

EXPLORATORY DATA ANALYSIS IN PYTHON