**Prediction**

An important aspect of data science is to find out what data can tell us about the future. What do data about climate and pollution say about temperatures a few decades from now? Based on a person’s internet profile, which websites are likely to interest them? How can a patient’s medical history be used to judge how well he or she will respond to a treatment?

To answer such questions, data scientists have developed methods for making *predictions*. In this chapter we will study one of the most commonly used ways of predicting the value of one variable based on the value of another.

The foundations of the method were laid by [Sir Francis Galton](https://en.wikipedia.org/wiki/Francis_Galton). As we saw in Section 7.1, Galton studied how physical characteristics are passed down from one generation to the next. Among his best known work is the prediction of the heights of adults based on the heights of their parents. We have studied the dataset that Galton collected for this. The table heights contains his data on the midparent height and child’s height (all in inches) for a population of 934 adult “children”.

*# Galton's data on heights of parents and their adult children*

galton **=** Table**.**read\_table(path\_data **+** 'galton.csv')

heights **=** Table()**.**with\_columns(

'MidParent', galton**.**column('midparentHeight'),

'Child', galton**.**column('childHeight')

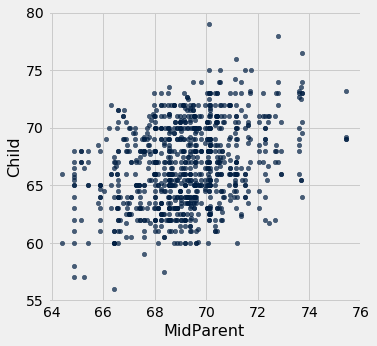
)

heights

| **MidParent** | **Child** |
| --- | --- |
| 75.43 | 73.2 |
| 75.43 | 69.2 |
| 75.43 | 69 |
| 75.43 | 69 |
| 73.66 | 73.5 |
| 73.66 | 72.5 |
| 73.66 | 65.5 |
| 73.66 | 65.5 |
| 72.06 | 71 |
| 72.06 | 68 |

... (924 rows omitted)

heights**.**scatter('MidParent')



The primary reason for collecting the data was to be able to predict the adult height of a child born to parents similar to those in the dataset. We made these predictions in Section 7.1, after noticing the positive association between the two variables.

Our approach was to base the prediction on all the points that correspond to a midparent height of around the midparent height of the new person. To do this, we wrote a function called predict\_child which takes a midparent height as its argument and returns the average height of all the children who had midparent heights within half an inch of the argument.

**def** **predict\_child**(mpht):

"""Return a prediction of the height of a child

whose parents have a midparent height of mpht.

The prediction is the average height of the children

whose midparent height is in the range mpht plus or minus 0.5 inches.

"""

close\_points **=** heights**.**where('MidParent', are**.**between(mpht**-**0.5, mpht **+** 0.5))

**return** close\_points**.**column('Child')**.**mean()

We applied the function to the column of Midparent heights, visualized our results.

*# Apply predict\_child to all the midparent heights*

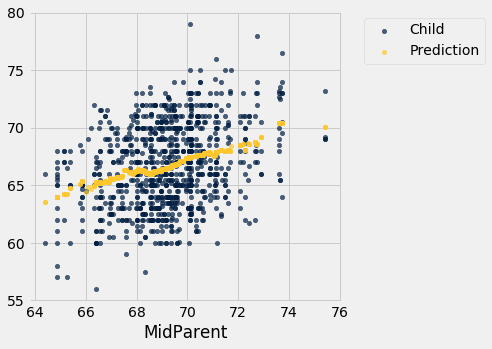
heights\_with\_predictions **=** heights**.**with\_column(

'Prediction', heights**.**apply(predict\_child, 'MidParent')

)

*# Draw the original scatter plot along with the predicted values*

heights\_with\_predictions**.**scatter('MidParent')



The prediction at a given midparent height lies roughly at the center of the vertical strip of points at the given height. This method of prediction is called *regression.* Later in this chapter we will see where this term came from. We will also see whether we can avoid our arbitrary definitions of “closeness” being “within 0.5 inches”. But first we will develop a measure that can be used in many settings to decide how good one variable will be as a predictor of another.

**Correlation**

In this section we will develop a measure of how tightly clustered a scatter diagram is about a straight line. Formally, this is called measuring *linear association*.

The table hybrid contains data on hybrid passenger cars sold in the United States from 1997 to 2013. The data were adapted from the online data archive of [Prof. Larry Winner](http://www.stat.ufl.edu/~winner/) of the University of Florida. The columns:

* vehicle: model of the car
* year: year of manufacture
* msrp: manufacturer’s suggested retail price in 2013 dollars
* acceleration: acceleration rate in km per hour per second
* mpg: fuel econonmy in miles per gallon
* class: the model’s class.

hybrid **=** Table**.**read\_table(path\_data **+** 'hybrid.csv')

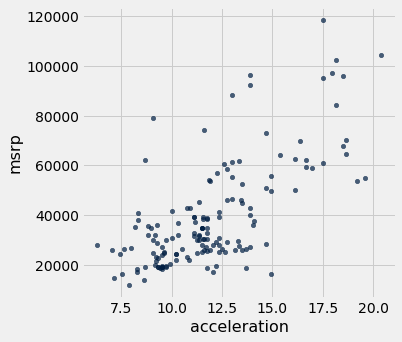
hybrid

| **vehicle** | **year** | **msrp** | **acceleration** | **mpg** | **class** |
| --- | --- | --- | --- | --- | --- |
| Prius (1st Gen) | 1997 | 24509.7 | 7.46 | 41.26 | Compact |
| Tino | 2000 | 35355 | 8.2 | 54.1 | Compact |
| Prius (2nd Gen) | 2000 | 26832.2 | 7.97 | 45.23 | Compact |
| Insight | 2000 | 18936.4 | 9.52 | 53 | Two Seater |
| Civic (1st Gen) | 2001 | 25833.4 | 7.04 | 47.04 | Compact |
| Insight | 2001 | 19036.7 | 9.52 | 53 | Two Seater |
| Insight | 2002 | 19137 | 9.71 | 53 | Two Seater |
| Alphard | 2003 | 38084.8 | 8.33 | 40.46 | Minivan |
| Insight | 2003 | 19137 | 9.52 | 53 | Two Seater |
| Civic | 2003 | 14071.9 | 8.62 | 41 | Compact |

... (143 rows omitted)

The graph below is a scatter plot of msrp *versus* acceleration. That means msrp is plotted on the vertical axis and accelaration on the horizontal.

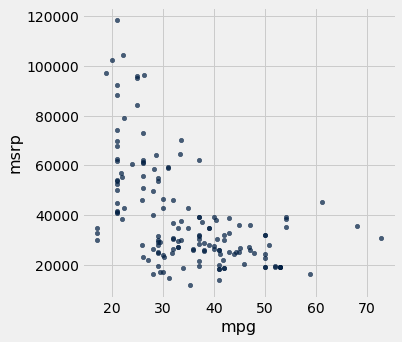
hybrid**.**scatter('acceleration', 'msrp')



Notice the positive association. The scatter of points is sloping upwards, indicating that cars with greater acceleration tended to cost more, on average; conversely, the cars that cost more tended to have greater acceleration on average.

The scatter diagram of MSRP versus mileage shows a negative association. Hybrid cars with higher mileage tended to cost less, on average. This seems surprising till you consider that cars that accelerate fast tend to be less fuel efficient and have lower mileage. As the previous scatter plot showed, those were also the cars that tended to cost more.

hybrid**.**scatter('mpg', 'msrp')

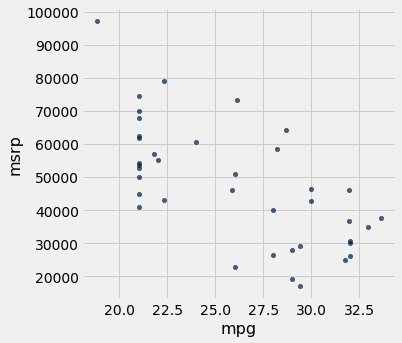


Along with the negative association, the scatter diagram of price versus efficiency shows a non-linear relation between the two variables. The points appear to be clustered around a curve, not around a straight line.

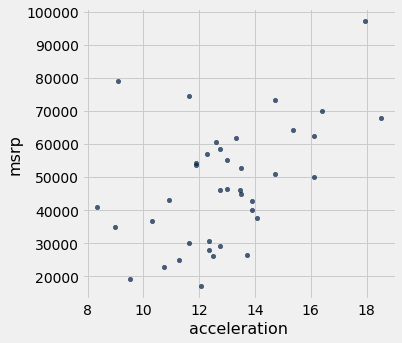
If we restrict the data just to the SUV class, however, the association between price and efficiency is still negative but the relation appears to be more linear. The relation between the price and acceleration of SUV’s also shows a linear trend, but with a positive slope.

suv **=** hybrid**.**where('class', 'SUV')

suv**.**scatter('mpg', 'msrp')



suv**.**scatter('acceleration', 'msrp')



You will have noticed that we can derive useful information from the general orientation and shape of a scatter diagram even without paying attention to the units in which the variables were measured.

Indeed, we could plot all the variables in standard units and the plots would look the same. This gives us a way to compare the degree of linearity in two scatter diagrams.

Recall that in an earlier section we defined the function standard\_units to convert an array of numbers to standard units.

**def** **standard\_units**(any\_numbers):

"Convert any array of numbers to standard units."

**return** (any\_numbers **-** np**.**mean(any\_numbers))**/**np**.**std(any\_numbers)

We can use this function to re-draw the two scatter diagrams for SUVs, with all the variables measured in standard units.

Table()**.**with\_columns(

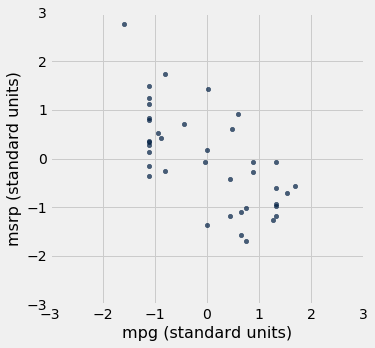
'mpg (standard units)', standard\_units(suv**.**column('mpg')),

'msrp (standard units)', standard\_units(suv**.**column('msrp'))

)**.**scatter(0, 1)

plots**.**xlim(**-**3, 3)

plots**.**ylim(**-**3, 3);



Table()**.**with\_columns(

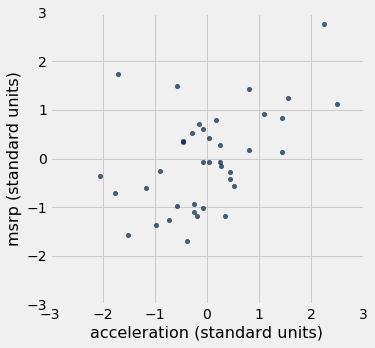
'acceleration (standard units)', standard\_units(suv**.**column('acceleration')),

'msrp (standard units)', standard\_units(suv**.**column('msrp'))

)**.**scatter(0, 1)

plots**.**xlim(**-**3, 3)

plots**.**ylim(**-**3, 3);



The associations that we see in these figures are the same as those we saw before. Also, because the two scatter diagrams are now drawn on exactly the same scale, we can see that the linear relation in the second diagram is a little more fuzzy than in the first.

We will now define a measure that uses standard units to quantify the kinds of association that we have seen.

**The correlation coefficient**

The *correlation coefficient* measures the strength of the linear relationship between two variables. Graphically, it measures how clustered the scatter diagram is around a straight line.

The term *correlation coefficient* isn’t easy to say, so it is usually shortened to *correlation* and denoted by rr.

Here are some mathematical facts about rr that we will just observe by simulation.

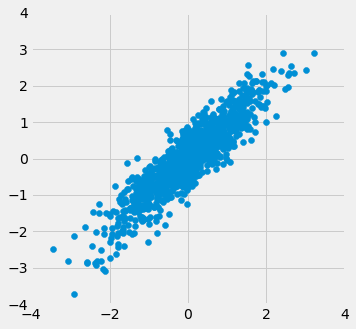
* The correlation coefficient rr is a number between −1−1 and 1.
* rr measures the extent to which the scatter plot clusters around a straight line.
* r=1r=1 if the scatter diagram is a perfect straight line sloping upwards, and r=−1r=−1 if the scatter diagram is a perfect straight line sloping downwards.

The function r\_scatter takes a value of rr as its argument and simulates a scatter plot with a correlation very close to rr. Because of randomness in the simulation, the correlation is not expected to be exactly equal to rr.

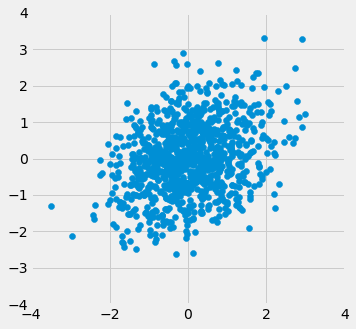
Call r\_scatter a few times, with different values of rr as the argument, and see how the scatter plot changes.

When r=1r=1 the scatter plot is perfectly linear and slopes upward. When r=−1r=−1, the scatter plot is perfectly linear and slopes downward. When r=0r=0, the scatter plot is a formless cloud around the horizontal axis, and the variables are said to be *uncorrelated*.

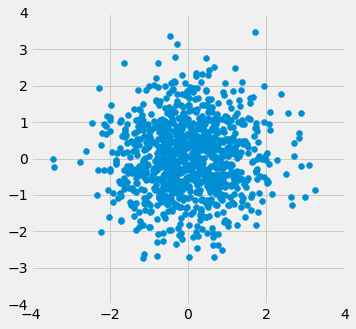
r\_scatter(0.9)



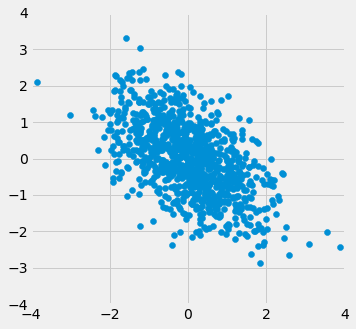
r\_scatter(0.25)



r\_scatter(0)



r\_scatter(**-**0.55)



**Calculating**rr

The formula for rr is not apparent from our observations so far. It has a mathematical basis that is outside the scope of this class. However, as you will see, the calculation is straightforward and helps us understand several of the properties of rr.

**Formula for**rr:

rr**is the average of the products of the two variables, when both variables are measured in standard units.**

Here are the steps in the calculation. We will apply the steps to a simple table of values of xx and yy.

x **=** np**.**arange(1, 7, 1)

y **=** make\_array(2, 3, 1, 5, 2, 7)

t **=** Table()**.**with\_columns(

'x', x,

'y', y

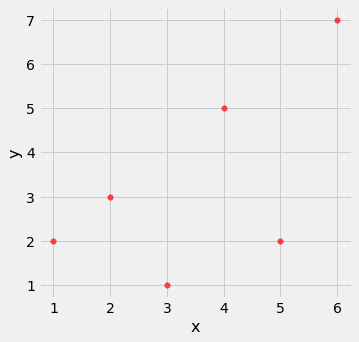
)

t

| **x** | **y** |
| --- | --- |
| 1 | 2 |
| 2 | 3 |
| 3 | 1 |
| 4 | 5 |
| 5 | 2 |
| 6 | 7 |

Based on the scatter diagram, we expect that rr will be positive but not equal to 1.

t**.**scatter(0, 1, s**=**30, color**=**'red')



**Step 1.** Convert each variable to standard units.

t\_su **=** t**.**with\_columns(

'x (standard units)', standard\_units(x),

'y (standard units)', standard\_units(y)

)

t\_su

| **x** | **y** | **x (standard units)** | **y (standard units)** |
| --- | --- | --- | --- |
| 1 | 2 | -1.46385 | -0.648886 |
| 2 | 3 | -0.87831 | -0.162221 |
| 3 | 1 | -0.29277 | -1.13555 |
| 4 | 5 | 0.29277 | 0.811107 |
| 5 | 2 | 0.87831 | -0.648886 |
| 6 | 7 | 1.46385 | 1.78444 |

**Step 2.** Multiply each pair of standard units.

t\_product **=** t\_su**.**with\_column('product of standard units', t\_su**.**column(2) **\*** t\_su**.**column(3))

t\_product

| **x** | **y** | **x (standard units)** | **y (standard units)** | **product of standard units** |
| --- | --- | --- | --- | --- |
| 1 | 2 | -1.46385 | -0.648886 | 0.949871 |
| 2 | 3 | -0.87831 | -0.162221 | 0.142481 |
| 3 | 1 | -0.29277 | -1.13555 | 0.332455 |
| 4 | 5 | 0.29277 | 0.811107 | 0.237468 |
| 5 | 2 | 0.87831 | -0.648886 | -0.569923 |
| 6 | 7 | 1.46385 | 1.78444 | 2.61215 |

**Step 3.** rr is the average of the products computed in Step 2.

*# r is the average of the products of standard units*

r **=** np**.**mean(t\_product**.**column(4))

r

0.6174163971897709

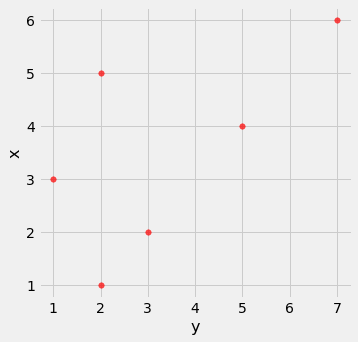
As expected, rr is positive but not equal to 1.

**Properties of**rr

The calculation shows that:

* rr is a pure number. It has no units. This is because rr is based on standard units.
* rr is unaffected by changing the units on either axis. This too is because rr is based on standard units.
* rr is unaffected by switching the axes. Algebraically, this is because the product of standard units does not depend on which variable is called xx and which yy. Geometrically, switching axes reflects the scatter plot about the line y=xy=x, but does not change the amount of clustering nor the sign of the association.

t**.**scatter('y', 'x', s**=**30, color**=**'red')



**The correlation function**

We are going to be calculating correlations repeatedly, so it will help to define a function that computes it by performing all the steps described above. Let’s define a function correlation that takes a table and the labels of two columns in the table. The function returns rr, the mean of the products of those column values in standard units.

**def** **correlation**(t, x, y):

**return** np**.**mean(standard\_units(t**.**column(x))**\***standard\_units(t**.**column(y)))

Let’s call the function on the x and y columns of t. The function returns the same answer to the correlation between xx and yy as we got by direct application of the formula for rr.

correlation(t, 'x', 'y')

0.6174163971897709

As we noticed, the order in which the variables are specified doesn’t matter.

correlation(t, 'y', 'x')

0.6174163971897709

Calling correlation on columns of the table suv gives us the correlation between price and mileage as well as the correlation between price and acceleration.

correlation(suv, 'mpg', 'msrp')

-0.6667143635709919

correlation(suv, 'acceleration', 'msrp')

0.48699799279959155

These values confirm what we had observed:

* There is a negative association between price and efficiency, whereas the association between price and acceleration is positive.
* The linear relation between price and acceleration is a little weaker (correlation about 0.5) than between price and mileage (correlation about -0.67).

Correlation is a simple and powerful concept, but it is sometimes misused. Before using rr, it is important to be aware of what correlation does and does not measure.

**Association is not Causation**

Correlation only measures association. Correlation does not imply causation. Though the correlation between the weight and the math ability of children in a school district may be positive, that does not mean that doing math makes children heavier or that putting on weight improves the children’s math skills. Age is a confounding variable: older children are both heavier and better at math than younger children, on average.

**Correlation Measures *Linear* Association**

Correlation measures only one kind of association – linear. Variables that have strong non-linear association might have very low correlation. Here is an example of variables that have a perfect quadratic relation y=x2y=x2 but have correlation equal to 0.

new\_x **=** np**.**arange(**-**4, 4.1, 0.5)

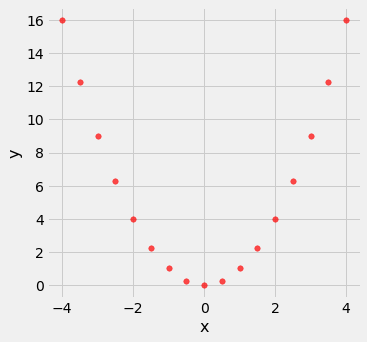
nonlinear **=** Table()**.**with\_columns(

'x', new\_x,

'y', new\_x**\*\***2

)

nonlinear**.**scatter('x', 'y', s**=**30, color**=**'r')



correlation(nonlinear, 'x', 'y')

0.0

**Correlation is Affected by Outliers**

Outliers can have a big effect on correlation. Here is an example where a scatter plot for which rr is equal to 1 is turned into a plot for which rr is equal to 0, by the addition of just one outlying point.

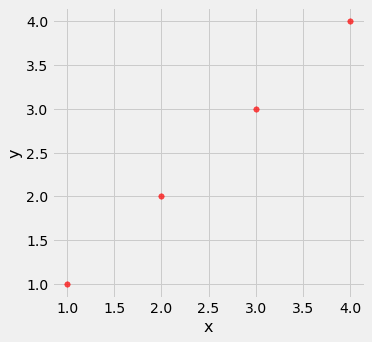
line **=** Table()**.**with\_columns(

'x', make\_array(1, 2, 3, 4),

'y', make\_array(1, 2, 3, 4)

)

line**.**scatter('x', 'y', s**=**30, color**=**'r')



correlation(line, 'x', 'y')

1.0

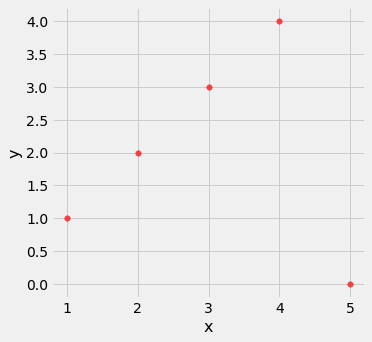
outlier **=** Table()**.**with\_columns(

'x', make\_array(1, 2, 3, 4, 5),

'y', make\_array(1, 2, 3, 4, 0)

)

outlier**.**scatter('x', 'y', s**=**30, color**=**'r')



correlation(outlier, 'x', 'y')

0.0

**Ecological Correlations Should be Interpreted with Care**

Correlations based on aggregated data can be misleading. As an example, here are data on the Critical Reading and Math SAT scores in 2014. There is one point for each of the 50 states and one for Washington, D.C. The column Participation Rate contains the percent of high school seniors who took the test. The next three columns show the average score in the state on each portion of the test, and the final column is the average of the total scores on the test.

sat2014 **=** Table**.**read\_table(path\_data **+** 'sat2014.csv')**.**sort('State')

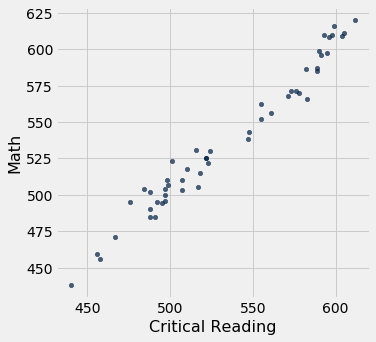
sat2014

| **State** | **Participation Rate** | **Critical Reading** | **Math** | **Writing** | **Combined** |
| --- | --- | --- | --- | --- | --- |
| Alabama | 6.7 | 547 | 538 | 532 | 1617 |
| Alaska | 54.2 | 507 | 503 | 475 | 1485 |
| Arizona | 36.4 | 522 | 525 | 500 | 1547 |
| Arkansas | 4.2 | 573 | 571 | 554 | 1698 |
| California | 60.3 | 498 | 510 | 496 | 1504 |
| Colorado | 14.3 | 582 | 586 | 567 | 1735 |
| Connecticut | 88.4 | 507 | 510 | 508 | 1525 |
| Delaware | 100 | 456 | 459 | 444 | 1359 |
| District of Columbia | 100 | 440 | 438 | 431 | 1309 |
| Florida | 72.2 | 491 | 485 | 472 | 1448 |

... (41 rows omitted)

The scatter diagram of Math scores versus Critical Reading scores is very tightly clustered around a straight line; the correlation is close to 0.985.

sat2014**.**scatter('Critical Reading', 'Math')



correlation(sat2014, 'Critical Reading', 'Math')

0.9847558411067434

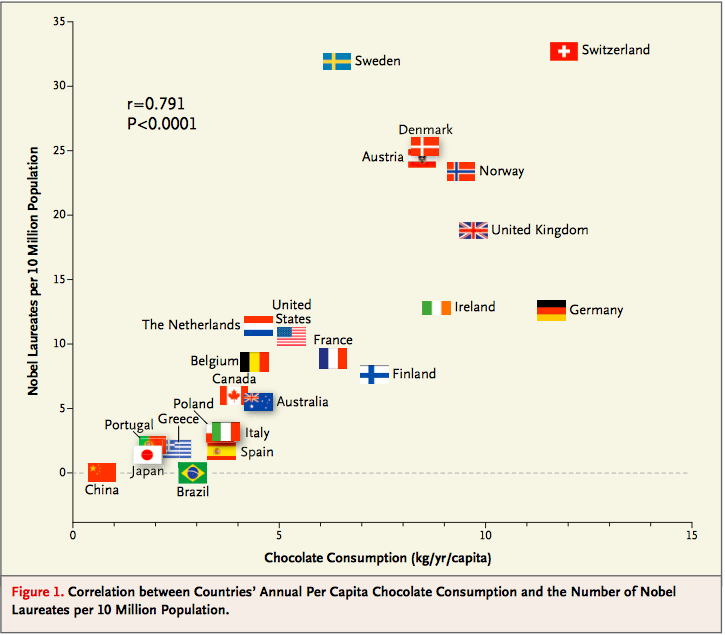
That’s an extremely high correlation. But it’s important to note that this does not reflect the strength of the relation between the Math and Critical Reading scores of *students*.

The data consist of average scores in each state. But states don’t take tests – students do. The data in the table have been created by lumping all the students in each state into a single point at the average values of the two variables in that state. But not all students in the state will be at that point, as students vary in their performance. If you plot a point for each student instead of just one for each state, there will be a cloud of points around each point in the figure above. The overall picture will be more fuzzy. The correlation between the Math and Critical Reading scores of the students will be *lower* than the value calculated based on state averages.

Correlations based on aggregates and averages are called *ecological correlations* and are frequently reported. As we have just seen, they must be interpreted with care.

**Serious or tongue-in-cheek?**

In 2012, a [paper](http://www.biostat.jhsph.edu/courses/bio621/misc/Chocolate%20consumption%20cognitive%20function%20and%20nobel%20laurates%20%28NEJM%29.pdf) in the respected New England Journal of Medicine examined the relation between chocolate consumption and Nobel Prizes in a group of countries. The [Scientific American](http://blogs.scientificamerican.com/the-curious-wavefunction/chocolate-consumption-and-nobel-prizes-a-bizarre-juxtaposition-if-there-ever-was-one/)responded seriously whereas [others](http://www.reuters.com/article/2012/10/10/us-eat-chocolate-win-the-nobel-prize-idUSBRE8991MS20121010#vFdfFkbPVlilSjsB.97) were more relaxed. You are welcome to make your own decision! The following graph, provided in the paper, should motivate you to go and take a look.



**The Regression Line**

The correlation coefficient rr doesn’t just measure how clustered the points in a scatter plot are about a straight line. It also helps identify the straight line about which the points are clustered. In this section we will retrace the path that Galton and Pearson took to discover that line.

Galton’s data on the heights of parents and their adult children showed a linear association. The linearity was confirmed when our predictions of the children’s heights based on the midparent heights roughly followed a straight line.

galton **=** Table**.**read\_table(path\_data **+** 'galton.csv')

heights **=** Table()**.**with\_columns(

'MidParent', galton**.**column('midparentHeight'),

'Child', galton**.**column('childHeight')

)

**def** **predict\_child**(mpht):

"""Return a prediction of the height of a child

whose parents have a midparent height of mpht.

The prediction is the average height of the children

whose midparent height is in the range mpht plus or minus 0.5 inches.

"""

close\_points **=** heights**.**where('MidParent', are**.**between(mpht**-**0.5, mpht **+** 0.5))

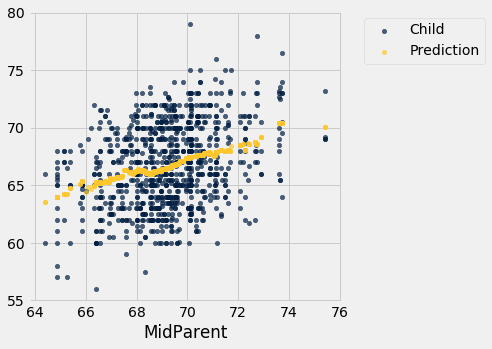
**return** close\_points**.**column('Child')**.**mean()

heights\_with\_predictions **=** heights**.**with\_column(

'Prediction', heights**.**apply(predict\_child, 'MidParent')

)

heights\_with\_predictions**.**scatter('MidParent')



**Measuring in Standard Units**

Let’s see if we can find a way to identify this line. First, notice that linear association doesn’t depend on the units of measurement – we might as well measure both variables in standard units.

**def** **standard\_units**(xyz):

"Convert any array of numbers to standard units."

**return** (xyz **-** np**.**mean(xyz))**/**np**.**std(xyz)

heights\_SU **=** Table()**.**with\_columns(

'MidParent SU', standard\_units(heights**.**column('MidParent')),

'Child SU', standard\_units(heights**.**column('Child'))

)

heights\_SU

| **MidParent SU** | **Child SU** |
| --- | --- |
| 3.45465 | 1.80416 |
| 3.45465 | 0.686005 |
| 3.45465 | 0.630097 |
| 3.45465 | 0.630097 |
| 2.47209 | 1.88802 |
| 2.47209 | 1.60848 |
| 2.47209 | -0.348285 |
| 2.47209 | -0.348285 |
| 1.58389 | 1.18917 |
| 1.58389 | 0.350559 |

... (924 rows omitted)

On this scale, we can calculate our predictions exactly as before. But first we have to figure out how to convert our old definition of “close” points to a value on the new scale. We had said that midparent heights were “close” if they were within 0.5 inches of each other. Since standard units measure distances in units of SDs, we have to figure out how many SDs of midparent height correspond to 0.5 inches.

One SD of midparent heights is about 1.8 inches. So 0.5 inches is about 0.28 SDs.

sd\_midparent **=** np**.**std(heights**.**column(0))

sd\_midparent

1.8014050969207571

0.5**/**sd\_midparent

0.277561110965367

We are now ready to modify our prediction function to make predictions on the standard units scale. All that has changed is that we are using the table of values in standard units, and defining “close” as above.

**def** **predict\_child\_su**(mpht\_su):

"""Return a prediction of the height (in standard units) of a child

whose parents have a midparent height of mpht\_su in standard units.

"""

close **=** 0.5**/**sd\_midparent

close\_points **=** heights\_SU**.**where('MidParent SU', are**.**between(mpht\_su**-**close, mpht\_su **+** close))

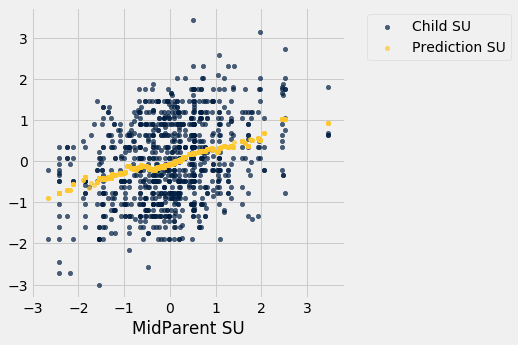
**return** close\_points**.**column('Child SU')**.**mean()

heights\_with\_su\_predictions **=** heights\_SU**.**with\_column(

'Prediction SU', heights\_SU**.**apply(predict\_child\_su, 'MidParent SU')

)

heights\_with\_su\_predictions**.**scatter('MidParent SU')

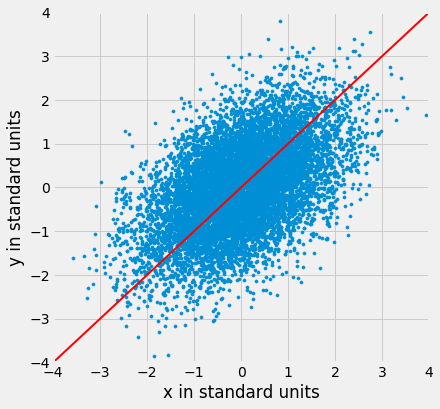


This plot looks exactly like the plot drawn on the original scale. Only the numbers on the axes have changed. This confirms that we can understand the prediction process by just working in standard units.

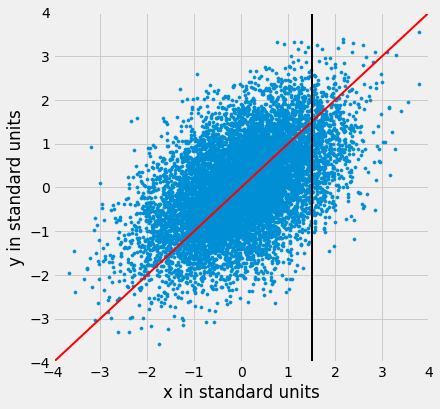
**Identifying the Line in Standard Units**

Galton’s scatter plot has a *football* shape – that is, it is roughly oval like an American football. Not all scatter plots are football shaped, not even those that show linear association. But in this section we will pretend we are Galton and work only with football shaped scatter plots. In the next section, we will generalize our analysis to other shapes of plots.

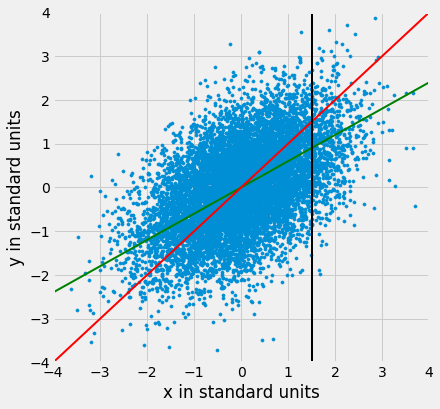
Here is a football shaped scatter plot with both variables measured in standard units. The 45 degree line is shown in red.



But the 45 degree line is not the line that picks off the centers of the vertical strips. You can see that in the figure below, where the vertical line at 1.5 standard units is shown in black. The points on the scatter plot near the black line all have heights roughly in the -2 to 3 range. The red line is too high to pick off the center.



So the 45 degree line is not the “graph of averages.” That line is the green one shown below.



Both lines go through the origin (0, 0). The green line goes through the centers of the vertical strips (at least roughly), and is *flatter* than the red 45 degree line.

The slope of the 45 degree line is 1. So the slope of the green “graph of averages” line is a value that is positive but less than 1.

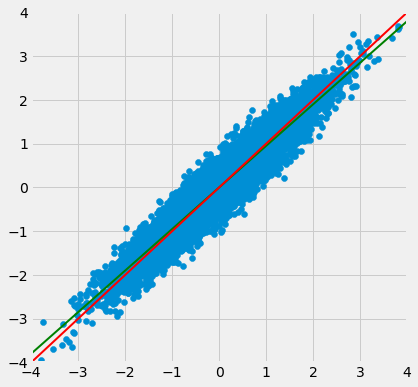
What value could that be? You’ve guessed it – it’s rr.

**The Regression Line, in Standard Units**

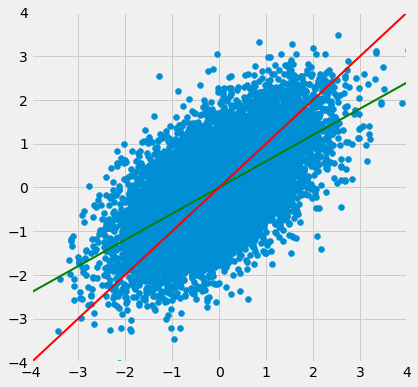
The green “graph of averages” line is called the *regression line*, for reasons we will explain shortly. But first, let’s simulate some football shaped scatter plots with different values of rr, and see how the line changes. In each case, the red 45 degree line has been drawn for comparison.

The function that performs the simulation is called regression\_line and takes rr as its argument.

regression\_line(0.95)



regression\_line(0.6)



When rr is close to 1, the scatter plot, the 45 degree line, and the regression line are all very close to each other. But for more moderate values of rr, the regression line is noticeably flatter.

**The Regression Effect**

In terms of prediction, this means that for a parents whose midparent height is at 1.5 standard units, our prediction of the child’s height is somewhat *less* than 1.5 standard units. If the midparent height is 2 standard units, we predict that the child’s height will be somewhat less than 2 standard units.

In other words, we predict that the child will be somewhat closer to average than the parents were.

This didn’t please Sir Francis Galton. He had been hoping that exceptionally tall parents would have children who were just as exceptionally tall. However, the data were clear, and Galton realized that the tall parents have children who are not quite as exceptionally tall, on average. Frustrated, Galton called this phenomenon “regression to mediocrity.”

Galton also noticed that exceptionally short parents had children who were somewhat taller relative to their generation, on average. In general, individuals who are away from average on one variable are expected to be not quite as far away from average on the other. This is called the *regression effect*.

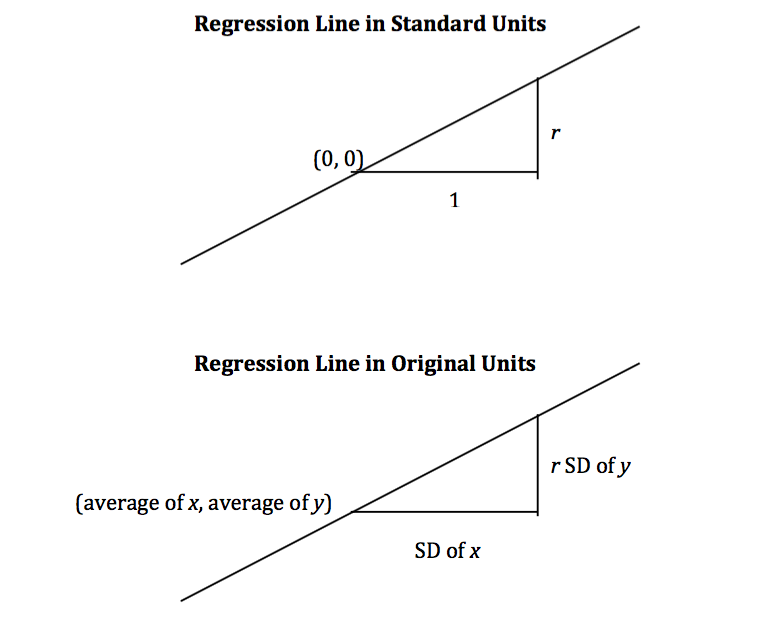
**The Equation of the Regression Line**

In regression, we use the value of one variable (which we will call xx) to predict the value of another (which we will call yy). When the variables xx and yy are measured in standard units, the regression line for predicting yy based on xx has slope rr and passes through the origin. Thus the equation of the regression line can be written as:

estimate of y = r⋅x   when both variables are measured in standard unitsestimate of y = r⋅x   when both variables are measured in standard units

In the original units of the data, this becomes

estimate of y − average of ySD of y = r×the given x − average of xSD of xestimate of y − average of ySD of y = r×the given x − average of xSD of x



The slope and intercept of the regression line in original units can be derived from the diagram above.

slope of the regression line = r⋅SD of ySD of xslope of the regression line = r⋅SD of ySD of xintercept of the regression line = average of y − slope⋅average of xintercept of the regression line = average of y − slope⋅average of x

The three functions below compute the correlation, slope, and intercept. All of them take three arguments: the name of the table, the label of the column containing xx, and the label of the column containing yy.

**def** **correlation**(t, label\_x, label\_y):

**return** np**.**mean(standard\_units(t**.**column(label\_x))**\***standard\_units(t**.**column(label\_y)))

**def** **slope**(t, label\_x, label\_y):

r **=** correlation(t, label\_x, label\_y)

**return** r**\***np**.**std(t**.**column(label\_y))**/**np**.**std(t**.**column(label\_x))

**def** **intercept**(t, label\_x, label\_y):

**return** np**.**mean(t**.**column(label\_y)) **-** slope(t, label\_x, label\_y)**\***np**.**mean(t**.**column(label\_x))

**The Regression Line and Galton’s Data**

The correlation between midparent height and child’s height is 0.32:

galton\_r **=** correlation(heights, 'MidParent', 'Child')

galton\_r

0.32094989606395924

We can also find the equation of the regression line for predicting the child’s height based on midparent height.

galton\_slope **=** slope(heights, 'MidParent', 'Child')

galton\_intercept **=** intercept(heights, 'MidParent', 'Child')

galton\_slope, galton\_intercept

(0.637360896969479, 22.63624054958975)

The equation of the regression line is

estimate of child's height = 0.64⋅midparent height + 22.64estimate of child's height = 0.64⋅midparent height + 22.64

This is also known as the *regression equation.* The principal use of the regression equation is to predict yy based on xx.

For example, for a midparent height of 70.48 inches, the regression equation predicts the child’s height to be 67.56 inches.

galton\_slope**\***70.48 **+** galton\_intercept

67.55743656799862

Our original prediction, created by taking the average height of all children who had midparent heights close to 70.48, came out to be pretty close: 67.63 inches compared to the regression line’s prediction of 67.55 inches.

heights\_with\_predictions**.**where('MidParent', are**.**equal\_to(70.48))**.**show(3)

| **MidParent** | **Child** | **Prediction** |
| --- | --- | --- |
| 70.48 | 74 | 67.6342 |
| 70.48 | 70 | 67.6342 |
| 70.48 | 68 | 67.6342 |

... (5 rows omitted)

Here are all of the rows in Galton’s table, along with our original predictions and the new regression predictions of the children’s heights.

heights\_with\_predictions **=** heights\_with\_predictions**.**with\_column(

'Regression Prediction', galton\_slope**\***heights**.**column('MidParent') **+** galton\_intercept

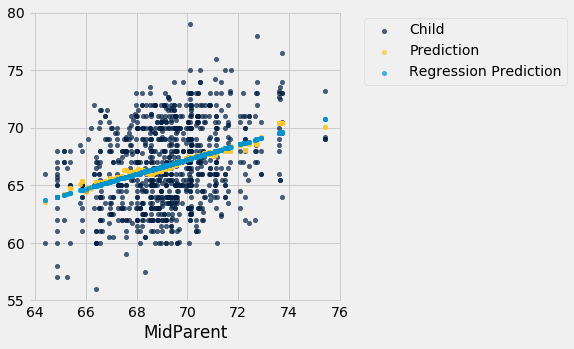
)

heights\_with\_predictions

| **MidParent** | **Child** | **Prediction** | **Regression Prediction** |
| --- | --- | --- | --- |
| 75.43 | 73.2 | 70.1 | 70.7124 |
| 75.43 | 69.2 | 70.1 | 70.7124 |
| 75.43 | 69 | 70.1 | 70.7124 |
| 75.43 | 69 | 70.1 | 70.7124 |
| 73.66 | 73.5 | 70.4158 | 69.5842 |
| 73.66 | 72.5 | 70.4158 | 69.5842 |
| 73.66 | 65.5 | 70.4158 | 69.5842 |
| 73.66 | 65.5 | 70.4158 | 69.5842 |
| 72.06 | 71 | 68.5025 | 68.5645 |
| 72.06 | 68 | 68.5025 | 68.5645 |

... (924 rows omitted)

heights\_with\_predictions**.**scatter('MidParent')



The grey dots show the regression predictions, all on the regression line. Notice how the line is very close to the gold graph of averages. For these data, the regression line does a good job of approximating the centers of the vertical strips.

**Fitted Values**

The predictions all lie on the line and are known as the “fitted values”. The function fit takes the name of the table and the labels of xx and yy, and returns an array of fitted values, one fitted value for each point in the scatter plot.

**def** **fit**(table, x, y):

"""Return the height of the regression line at each x value."""

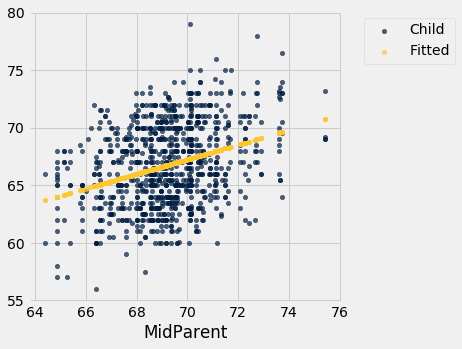
a **=** slope(table, x, y)

b **=** intercept(table, x, y)

**return** a **\*** table**.**column(x) **+** b

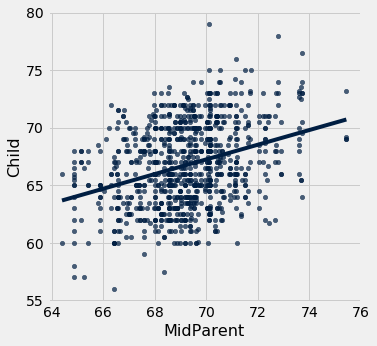
It is easier to see the line in the graph below than in the one above.

heights**.**with\_column('Fitted', fit(heights, 'MidParent', 'Child'))**.**scatter('MidParent')



Another way to draw the line is to use the option fit\_line=True with the Table method scatter.

heights**.**scatter('MidParent', fit\_line**=**True)

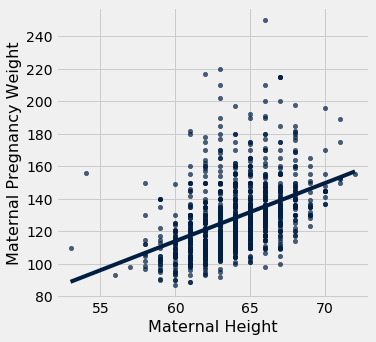


**Units of Measurement of the Slope**

The slope is a ratio, and it worth taking a moment to study the units in which it is measured. Our example comes from the familiar dataset about mothers who gave birth in a hospital system. The scatter plot of pregnancy weights versus heights looks like a football that has been used in one game too many, but it’s close enough to a football that we can justify putting our fitted line through it. In later sections we will see how to make such justifications more formal.

baby **=** Table**.**read\_table(path\_data **+** 'baby.csv')

baby**.**scatter('Maternal Height', 'Maternal Pregnancy Weight', fit\_line**=**True)



slope(baby, 'Maternal Height', 'Maternal Pregnancy Weight')

3.572846259275056

The slope of the regression line is **3.57 pounds per inch**. This means that for two women who are 1 inch apart in height, our prediction of pregnancy weight will differ by 3.57 pounds. For a woman who is 2 inches taller than another, our prediction of pregnancy weight will be 2×3.57 = 7.142×3.57 = 7.14pounds more than our prediction for the shorter woman.

Notice that the successive vertical strips in the scatter plot are one inch apart, because the heights have been rounded to the nearest inch. Another way to think about the slope is to take any two consecutive strips (which are necessarily 1 inch apart), corresponding to two groups of women who are separated by 1 inch in height. The slope of 3.57 pounds per inch means that the average pregnancy weight of the taller group is about 3.57 pounds more than that of the shorter group.

**Example**

Suppose that our goal is to use regression to estimate the height of a basset hound based on its weight, using a sample that looks consistent with the regression model. Suppose the observed correlation rr is 0.5, and that the summary statistics for the two variables are as in the table below:

|  | **average** | **SD** |
| --- | --- | --- |
| height | 14 inches | 2 inches |
| weight | 50 pounds | 5 pounds |

To calculate the equation of the regression line, we need the slope and the intercept.

slope = r⋅SD of ySD of x = 0.5⋅2 inches5 pounds = 0.2 inches per poundslope = r⋅SD of ySD of x = 0.5⋅2 inches5 pounds = 0.2 inches per poundintercept = average of y−slope⋅average of x = 14 inches − 0.2 inches per pound⋅50 pounds = 4 inchesintercept = average of y−slope⋅average of x = 14 inches − 0.2 inches per pound⋅50 pounds = 4 inches

The equation of the regression line allows us to calculate the estimated height, in inches, based on a given weight in pounds:

estimated height = 0.2⋅given weight + 4estimated height = 0.2⋅given weight + 4

The slope of the line is measures the increase in the estimated height per unit increase in weight. The slope is positive, and it is important to note that this does not mean that we think basset hounds get taller if they put on weight. The slope reflects the difference in the average heights of two groups of dogs that are 1 pound apart in weight. Specifically, consider a group of dogs whose weight is ww pounds, and the group whose weight is w+1w+1 pounds. The second group is estimated to be 0.2 inches taller, on average. This is true for all values of ww in the sample.

In general, the slope of the regression line can be interpreted as the average increase in yy per unit increase in xx. Note that if the slope is negative, then for every unit increase in xx, the average of yydecreases.

**Endnote**

Even though we won’t establish the mathematical basis for the regression equation, we can see that it gives pretty good predictions when the scatter plot is football shaped. It is a surprising mathematical fact that no matter what the shape of the scatter plot, the same equation gives the “best” among all straight lines. That’s the topic of the next section.

**The Method of Least Squares**

We have retraced the steps that Galton and Pearson took to develop the equation of the regression line that runs through a football shaped scatter plot. But not all scatter plots are football shaped, not even linear ones. Does every scatter plot have a “best” line that goes through it? If so, can we still use the formulas for the slope and intercept developed in the previous section, or do we need new ones?

To address these questions, we need a reasonable definition of “best”. Recall that the purpose of the line is to *predict* or *estimate* values of yy, given values of xx. Estimates typically aren’t perfect. Each one is off the true value by an *error*. A reasonable criterion for a line to be the “best” is for it to have the smallest possible overall error among all straight lines.

In this section we will make this criterion precise and see if we can identify the best straight line under the criterion.

Our first example is a dataset that has one row for every chapter of the novel “Little Women.” The goal is to estimate the number of characters (that is, letters, spaces punctuation marks, and so on) based on the number of periods. Recall that we attempted to do this in the very first lecture of this course.

little\_women **=** Table**.**read\_table(path\_data **+** 'little\_women.csv')

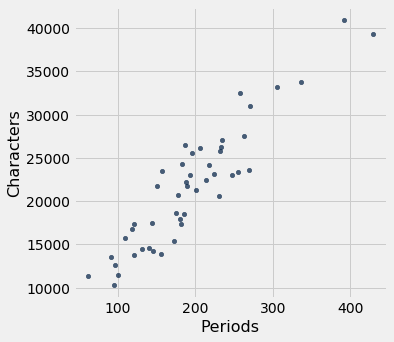
little\_women **=** little\_women**.**move\_to\_start('Periods')

little\_women**.**show(3)

| **Periods** | **Characters** |
| --- | --- |
| 189 | 21759 |
| 188 | 22148 |
| 231 | 20558 |

... (44 rows omitted)

little\_women**.**scatter('Periods', 'Characters')



To explore the data, we will need to use the functions correlation, slope, intercept, and fitdefined in the previous section.

correlation(little\_women, 'Periods', 'Characters')

0.9229576895854816

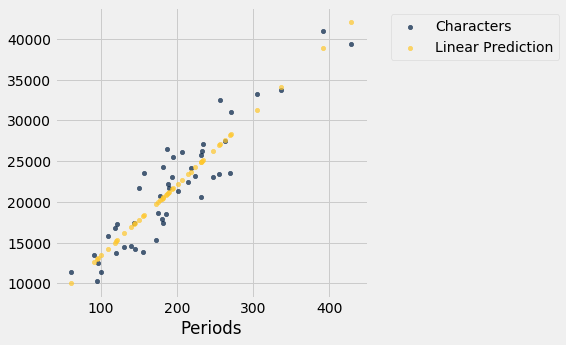
The scatter plot is remarkably close to linear, and the correlation is more than 0.92.

**Error in Estimation**

The graph below shows the scatter plot and line that we developed in the previous section. We don’t yet know if that’s the best among all lines. We first have to say precisely what “best” means.

lw\_with\_predictions **=** little\_women**.**with\_column('Linear Prediction', fit(little\_women, 'Periods', 'Characters'))

lw\_with\_predictions**.**scatter('Periods')



Corresponding to each point on the scatter plot, there is an error of prediction calculated as the actual value minus the predicted value. It is the vertical distance between the point and the line, with a negative sign if the point is below the line.

actual **=** lw\_with\_predictions**.**column('Characters')

predicted **=** lw\_with\_predictions**.**column('Linear Prediction')

errors **=** actual **-** predicted

lw\_with\_predictions**.**with\_column('Error', errors)

| **Periods** | **Characters** | **Linear Prediction** | **Error** |
| --- | --- | --- | --- |
| 189 | 21759 | 21183.6 | 575.403 |
| 188 | 22148 | 21096.6 | 1051.38 |
| 231 | 20558 | 24836.7 | -4278.67 |
| 195 | 25526 | 21705.5 | 3820.54 |
| 255 | 23395 | 26924.1 | -3529.13 |
| 140 | 14622 | 16921.7 | -2299.68 |
| 131 | 14431 | 16138.9 | -1707.88 |
| 214 | 22476 | 23358 | -882.043 |
| 337 | 33767 | 34056.3 | -289.317 |
| 185 | 18508 | 20835.7 | -2327.69 |

... (37 rows omitted)

We can use slope and intercept to calculate the slope and intercept of the fitted line. The graph below shows the line (in light blue). The errors corresponding to four of the points are shown in red. There is nothing special about those four points. They were just chosen for clarity of the display. The function lw\_errors takes a slope and an intercept (in that order) as its arguments and draws the figure.

lw\_reg\_slope **=** slope(little\_women, 'Periods', 'Characters')

lw\_reg\_intercept **=** intercept(little\_women, 'Periods', 'Characters')

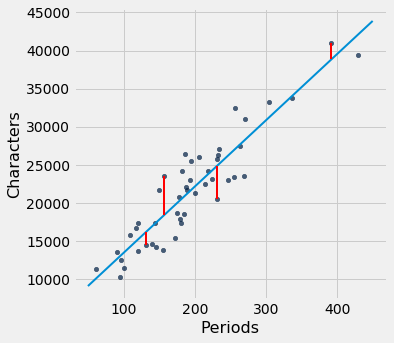
**print**('Slope of Regression Line: ', np**.**round(lw\_reg\_slope), 'characters per period')

**print**('Intercept of Regression Line:', np**.**round(lw\_reg\_intercept), 'characters')

lw\_errors(lw\_reg\_slope, lw\_reg\_intercept)

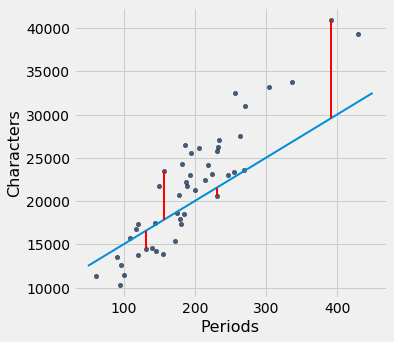
Slope of Regression Line: 87.0 characters per period

Intercept of Regression Line: 4745.0 characters

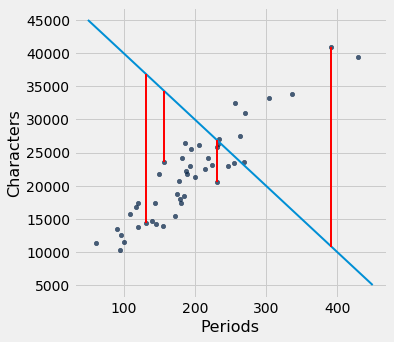


Had we used a different line to create our estimates, the errors would have been different. The graph below shows how big the errors would be if we were to use another line for estimation. The second graph shows large errors obtained by using a line that is downright silly.

lw\_errors(50, 10000)



lw\_errors(**-**100, 50000)



**Root Mean Squared Error**

What we need now is one overall measure of the rough size of the errors. You will recognize the approach to creating this – it’s exactly the way we developed the SD.

If you use any arbitrary line to calculate your estimates, then some of your errors are likely to be positive and others negative. To avoid cancellation when measuring the rough size of the errors, we will take the mean of the squared errors rather than the mean of the errors themselves.

The mean squared error of estimation is a measure of roughly how big the squared errors are, but as we have noted earlier, its units are hard to interpret. Taking the square root yields the root mean square error (rmse), which is in the same units as the variable being predicted and therefore much easier to understand.

**Minimizing the Root Mean Squared Error**

Our observations so far can be summarized as follows.

* To get estimates of yy based on xx, you can use any line you want.
* Every line has a root mean squared error of estimation.
* “Better” lines have smaller errors.

Is there a “best” line? That is, is there a line that minimizes the root mean squared error among all lines?

To answer this question, we will start by defining a function lw\_rmse to compute the root mean squared error of any line through the Little Women scatter diagram. The function takes the slope and the intercept (in that order) as its arguments.

**def** **lw\_rmse**(slope, intercept):

lw\_errors(slope, intercept)

x **=** little\_women**.**column('Periods')

y **=** little\_women**.**column('Characters')

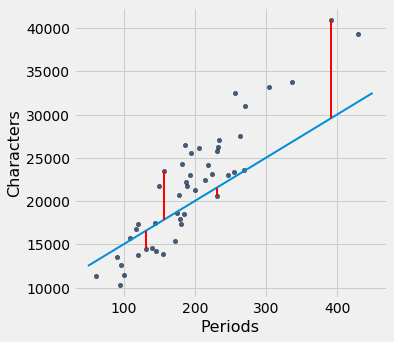
fitted **=** slope **\*** x **+** intercept

mse **=** np**.**mean((y **-** fitted) **\*\*** 2)

**print**("Root mean squared error:", mse **\*\*** 0.5)

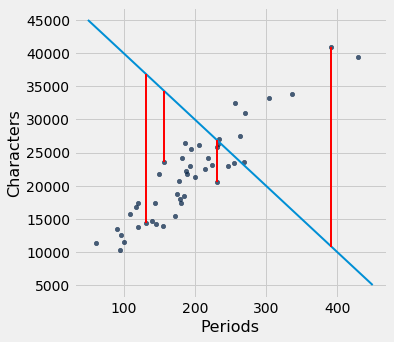
lw\_rmse(50, 10000)

Root mean squared error: 4322.167831766537



lw\_rmse(**-**100, 50000)

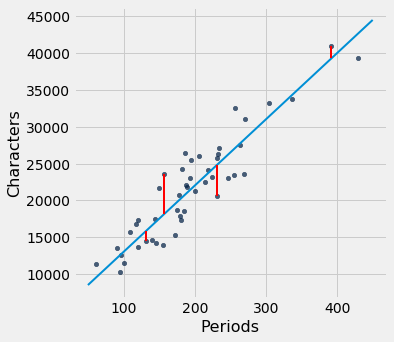
Root mean squared error: 16710.11983735375



Bad lines have big values of rmse, as expected. But the rmse is much smaller if we choose a slope and intercept close to those of the regression line.

lw\_rmse(90, 4000)

Root mean squared error: 2715.5391063834586

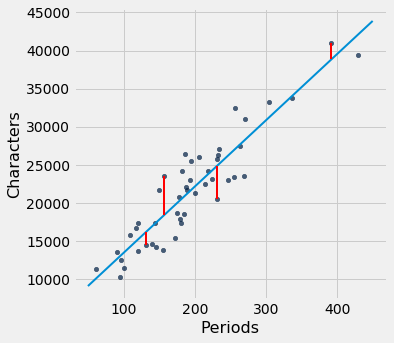


Here is the root mean squared error corresponding to the regression line. By a remarkable fact of mathematics, no other line can beat this one.

* **The regression line is the unique straight line that minimizes the mean squared error of estimation among all straight lines.**

lw\_rmse(lw\_reg\_slope, lw\_reg\_intercept)

Root mean squared error: 2701.690785311856



The proof of this statement requires abstract mathematics that is beyond the scope of this course. On the other hand, we do have a powerful tool – Python – that performs large numerical computations with ease. So we can use Python to confirm that the regression line minimizes the mean squared error.

**Numerical Optimization**

First note that a line that minimizes the root mean squared error is also a line that minimizes the squared error. The square root makes no difference to the minimization. So we will save ourselves a step of computation and just minimize the mean squared error (mse).

We are trying to predict the number of characters (yy) based on the number of periods (xx) in chapters of Little Women. If we use the line prediction = ax+bprediction = ax+b it will have an mse that depends on the slope aa and the intercept bb. The function lw\_mse takes the slope and intercept as its arguments and returns the corresponding mse.

**def** **lw\_mse**(any\_slope, any\_intercept):

x **=** little\_women**.**column('Periods')

y **=** little\_women**.**column('Characters')

fitted **=** any\_slope**\***x **+** any\_intercept

**return** np**.**mean((y **-** fitted) **\*\*** 2)

Let’s check that lw\_mse gets the right answer for the root mean squared error of the regression line. Remember that lw\_mse returns the mean squared error, so we have to take the square root to get the rmse.

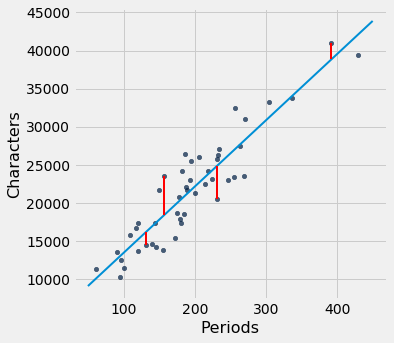
lw\_mse(lw\_reg\_slope, lw\_reg\_intercept)**\*\***0.5

2701.690785311856

That’s the same as the value we got by using lw\_rmse earlier:

lw\_rmse(lw\_reg\_slope, lw\_reg\_intercept)

Root mean squared error: 2701.690785311856



You can confirm that lw\_mse returns the correct value for other slopes and intercepts too. For example, here is the rmse of the extremely bad line that we tried earlier.

lw\_mse(**-**100, 50000)**\*\***0.5

16710.11983735375

And here is the rmse for a line that is close to the regression line.

lw\_mse(90, 4000)**\*\***0.5

2715.5391063834586

If we experiment with different values, we can find a low-error slope and intercept through trial and error, but that would take a while. Fortunately, there is a Python function that does all the trial and error for us.

The minimize function can be used to find the arguments of a function for which the function returns its minimum value. Python uses a similar trial-and-error approach, following the changes that lead to incrementally lower output values.

The argument of minimize is a function that itself takes numerical arguments and returns a numerical value. For example, the function lw\_mse takes a numerical slope and intercept as its arguments and returns the corresponding mse.

The call minimize(lw\_mse) returns an array consisting of the slope and the intercept that minimize the mse. These minimizing values are excellent approximations arrived at by intelligent trial-and-error, not exact values based on formulas.

best **=** minimize(lw\_mse)

best

array([ 86.97784117, 4744.78484535])

These values are the same as the values we calculated earlier by using the slope and interceptfunctions. We see small deviations due to the inexact nature of minimize, but the values are essentially the same.

**print**("slope from formula: ", lw\_reg\_slope)

**print**("slope from minimize: ", best**.**item(0))

**print**("intercept from formula: ", lw\_reg\_intercept)

**print**("intercept from minimize: ", best**.**item(1))

slope from formula: 86.97784125829821

slope from minimize: 86.97784116615884

intercept from formula: 4744.784796574928

intercept from minimize: 4744.784845352655

**The Least Squares Line**

Therefore, we have found not only that the regression line minimizes mean squared error, but also that minimizing mean squared error gives us the regression line. The regression line is the only line that minimizes mean squared error.

That is why the regression line is sometimes called the “least squares line.”

**Least Squares Regression**

In an earlier section, we developed formulas for the slope and intercept of the regression line through a *football shaped* scatter diagram. It turns out that the slope and intercept of the least squares line have the same formulas as those we developed, *regardless of the shape of the scatter plot*.

We saw this in the example about Little Women, but let’s confirm it in an example where the scatter plot clearly isn’t football shaped. For the data, we are once again indebted to the rich [data archive of Prof. Larry Winner](http://www.stat.ufl.edu/~winner/datasets.html) of the University of Florida. A [2013 study](http://digitalcommons.wku.edu/ijes/vol6/iss2/10/) in the International Journal of Exercise Science studied collegiate shot put athletes and examined the relation between strength and shot put distance. The population consists of 28 female collegiate athletes. Strength was measured by the the biggest amount (in kilograms) that the athlete lifted in the “1RM power clean” in the pre-season. The distance (in meters) was the athlete’s personal best.

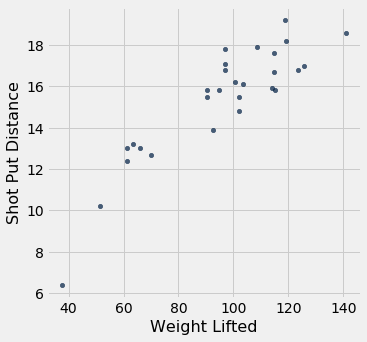
shotput **=** Table**.**read\_table(path\_data **+** 'shotput.csv')

shotput

| **Weight Lifted** | **Shot Put Distance** |
| --- | --- |
| 37.5 | 6.4 |
| 51.5 | 10.2 |
| 61.3 | 12.4 |
| 61.3 | 13 |
| 63.6 | 13.2 |
| 66.1 | 13 |
| 70 | 12.7 |
| 92.7 | 13.9 |
| 90.5 | 15.5 |
| 90.5 | 15.8 |

... (18 rows omitted)

shotput**.**scatter('Weight Lifted')



That’s not a football shaped scatter plot. In fact, it seems to have a slight non-linear component. But if we insist on using a straight line to make our predictions, there is still one best straight line among all straight lines.

Our formulas for the slope and intercept of the regression line, derived for football shaped scatter plots, give the following values.

slope(shotput, 'Weight Lifted', 'Shot Put Distance')

0.09834382159781997

intercept(shotput, 'Weight Lifted', 'Shot Put Distance')

5.959629098373952

Does it still make sense to use these formulas even though the scatter plot isn’t football shaped? We can answer this by finding the slope and intercept of the line that minimizes the mse.

We will define the function shotput\_linear\_mse to take an arbirtary slope and intercept as arguments and return the corresponding mse. Then minimize applied to shotput\_linear\_msewill return the best slope and intercept.

**def** **shotput\_linear\_mse**(any\_slope, any\_intercept):

x **=** shotput**.**column('Weight Lifted')

y **=** shotput**.**column('Shot Put Distance')

fitted **=** any\_slope**\***x **+** any\_intercept

**return** np**.**mean((y **-** fitted) **\*\*** 2)

minimize(shotput\_linear\_mse)

array([0.09834382, 5.95962911])

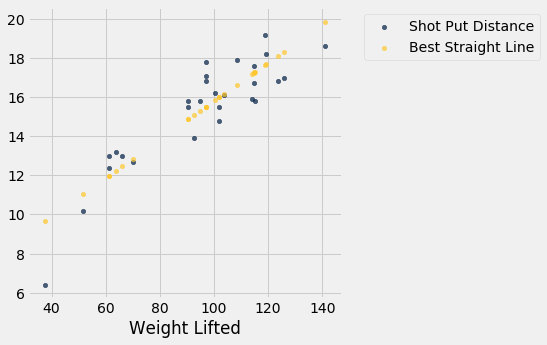
These values are the same as those we got by using our formulas. To summarize:

**No matter what the shape of the scatter plot, there is a unique line that minimizes the mean squared error of estimation. It is called the regression line, and its slope and intercept are given by**

slope of the regression line = r⋅SD of ySD of xslope of the regression line = r⋅SD of ySD of xintercept of the regression line = average of y − slope⋅average of xintercept of the regression line = average of y − slope⋅average of x

fitted **=** fit(shotput, 'Weight Lifted', 'Shot Put Distance')

shotput**.**with\_column('Best Straight Line', fitted)**.**scatter('Weight Lifted')



**Nonlinear Regression**

The graph above reinforces our earlier observation that the scatter plot is a bit curved. So it is better to fit a curve than a straight line. The [study](http://digitalcommons.wku.edu/ijes/vol6/iss2/10/) postulated a quadratic relation between the weight lifted and the shot put distance. So let’s use quadratic functions as our predictors and see if we can find the best one.

We have to find the best quadratic function among all quadratic functions, instead of the best straight line among all straight lines. The method of least squares allows us to do this.

The mathematics of this minimization is complicated and not easy to see just by examining the scatter plot. But numerical minimization is just as easy as it was with linear predictors! We can get the best quadratic predictor by once again using minimize. Let’s see how this works.

Recall that a quadratic function has the form

f(x) = ax2+bx+cf(x) = ax2+bx+c for constants aa, bb, and cc.

To find the best quadratic function to predict distance based on weight lifted, using the criterion of least squares, we will first write a function that takes the three constants as its arguments, calculates the fitted values by using the quadratic function above, and then returns the mean squared error.

The function is called shotput\_quadratic\_mse. Notice that the definition is analogous to that of lw\_mse, except that the fitted values are based on a quadratic function instead of linear.

**def** **shotput\_quadratic\_mse**(a, b, c):

x **=** shotput**.**column('Weight Lifted')

y **=** shotput**.**column('Shot Put Distance')

fitted **=** a**\***(x**\*\***2) **+** b**\***x **+** c

**return** np**.**mean((y **-** fitted) **\*\*** 2)

We can now use minimize just as before to find the constants that minimize the mean squared error.

best **=** minimize(shotput\_quadratic\_mse)

best

array([-1.04004838e-03, 2.82708045e-01, -1.53182115e+00])

Our prediction of the shot put distance for an athlete who lifts xx kilograms is about −0.00104x2 + 0.2827x−1.5318−0.00104x2 + 0.2827x−1.5318 meters. For example, if the athlete can lift 100 kilograms, the predicted distance is 16.33 meters. On the scatter plot, that’s near the center of a vertical strip around 100 kilograms.

(**-**0.00104)**\***(100**\*\***2) **+** 0.2827**\***100 **-** 1.5318

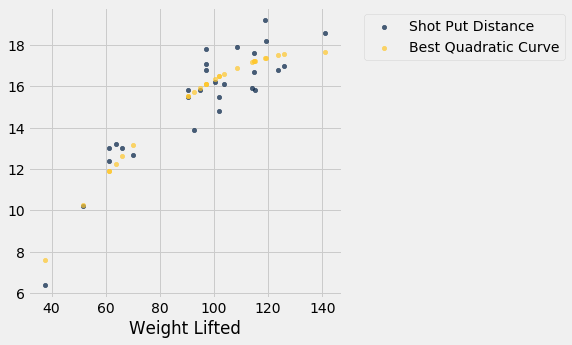
16.3382

Here are the predictions for all the values of Weight Lifted. You can see that they go through the center of the scatter plot, to a rough approximation.

x **=** shotput**.**column(0)

shotput\_fit **=** best**.**item(0)**\***(x**\*\***2) **+** best**.**item(1)**\***x **+** best**.**item(2)

shotput**.**with\_column('Best Quadratic Curve', shotput\_fit)**.**scatter(0)



**Visual Diagnostics**

Suppose a data scientist has decided to use linear regression to estimate values of one variable (called the response variable) based on another variable (called the predictor). To see how well this method of estimation performs, the data scientist must how far off the estimates are from the actual values. These differences are called *residuals*.

residual = observed value − regression estimateresidual = observed value − regression estimate

A residual is what’s left over – the residue – after estimation.

Residuals are the vertical distances of the points from the regression line. There is one residual for each point in the scatter plot. The residual is the difference between the observed value of yy and the fitted value of yy, so fr the point (x,y)(x,y),

residual  =  y − fitted value of y  =  y − height of regression line at xresidual  =  y − fitted value of y  =  y − height of regression line at x

The function residual calculates the residuals. The calculation assumes all the relevant functions we have already defined: standard\_units, correlation, slope, intercept, and fit.

**def** **residual**(table, x, y):

**return** table**.**column(y) **-** fit(table, x, y)

Continuing our example of using Galton’s data to estimate the heights of adult children (the response) based on the midparent height (the predictor), let us calculate the fitted values and the residuals.

heights **=** heights**.**with\_columns(

'Fitted Value', fit(heights, 'MidParent', 'Child'),

'Residual', residual(heights, 'MidParent', 'Child')

)

heights

| **MidParent** | **Child** | **Fitted Value** | **Residual** |
| --- | --- | --- | --- |
| 75.43 | 73.2 | 70.7124 | 2.48763 |
| 75.43 | 69.2 | 70.7124 | -1.51237 |
| 75.43 | 69 | 70.7124 | -1.71237 |
| 75.43 | 69 | 70.7124 | -1.71237 |
| 73.66 | 73.5 | 69.5842 | 3.91576 |
| 73.66 | 72.5 | 69.5842 | 2.91576 |
| 73.66 | 65.5 | 69.5842 | -4.08424 |
| 73.66 | 65.5 | 69.5842 | -4.08424 |
| 72.06 | 71 | 68.5645 | 2.43553 |
| 72.06 | 68 | 68.5645 | -0.564467 |

... (924 rows omitted)

When there are so many variables to work with, it is always helpful to start with visualization. The function scatter\_fit draws the scatter plot of the data, as well as the regression line.

**def** **scatter\_fit**(table, x, y):

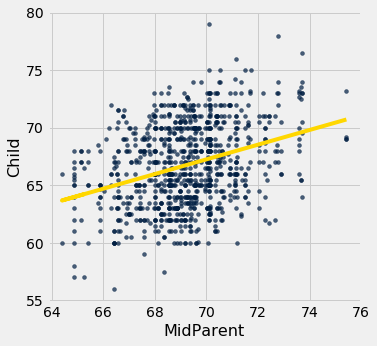
table**.**scatter(x, y, s**=**15)

plots**.**plot(table**.**column(x), fit(table, x, y), lw**=**4, color**=**'gold')

plots**.**xlabel(x)

plots**.**ylabel(y)

scatter\_fit(heights, 'MidParent', 'Child')



A *residual plot* can be drawn by plotting the residuals against the predictor variable. The function residual\_plot does just that.

**def** **residual\_plot**(table, x, y):

x\_array **=** table**.**column(x)

t **=** Table()**.**with\_columns(

x, x\_array,

'residuals', residual(table, x, y)

)

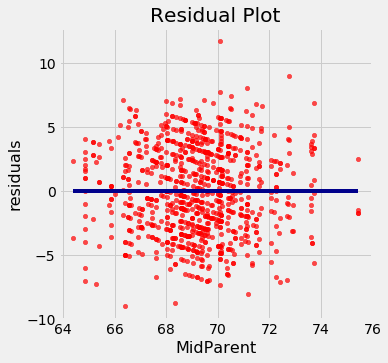
t**.**scatter(x, 'residuals', color**=**'r')

xlims **=** make\_array(min(x\_array), max(x\_array))

plots**.**plot(xlims, make\_array(0, 0), color**=**'darkblue', lw**=**4)

plots**.**title('Residual Plot')

residual\_plot(heights, 'MidParent', 'Child')



The midparent heights are on the horizontal axis, as in the original scatter plot. But now the vertical axis shows the residuals. Notice that the plot appears to be centered around the horizontal line at the level 0 (shown in dark blue). Notice also that the plot shows no upward or downward trend. We will observe later that this is true of all regressions.

**Regression Diagnostics**

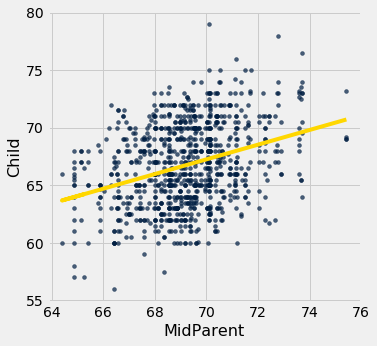
Residual plots help us make visual assessments of the quality of a linear regression analysis. Such assessments are called *diagnostics*. The function regression\_diagnostic\_plots draws the original scatter plot as well as the residual plot for ease of comparison.

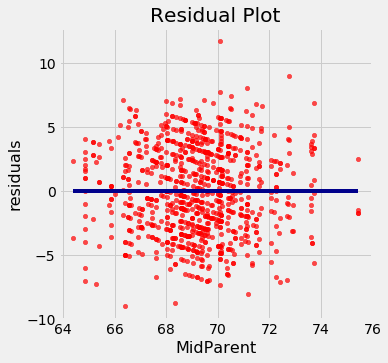
**def** **regression\_diagnostic\_plots**(table, x, y):

scatter\_fit(table, x, y)

residual\_plot(table, x, y)

regression\_diagnostic\_plots(heights, 'MidParent', 'Child')





This residual plot indicates that linear regression was a reasonable method of estimation. Notice how the residuals are distributed fairly symmetrically above and below the horizontal line at 0, corresponding to the original scatter plot being roughly symmetrical above and below. Notice also that the vertical spread of the plot is fairly even across the most common values of the children’s heights. In other words, apart from a few outlying points, the plot isn’t narrower in some places and wider in others.

In other words, the accuracy of the regression appears to be about the same across the observed range of the predictor variable.

**The residual plot of a good regression shows no pattern. The residuals look about the same, above and below the horizontal line at 0, across the range of the predictor variable.**

**Detecting Nonlinearity**

Drawing the scatter plot of the data usually gives an indication of whether the relation between the two variables is non-linear. Often, however, it is easier to spot non-linearity in a residual plot than in the original scatter plot. This is usually because of the scales of the two plots: the residual plot allows us to zoom in on the errors and hence makes it easier to spot patterns.



Our data are a [dataset](http://www.statsci.org/data/oz/dugongs.html) on the age and length of dugongs, which are marine mammals related to manatees and sea cows (image from [Wikimedia Commons](https://commons.wikimedia.org/wiki/File:Dugong_dugon.jpg)). The data are in a table called dugong. Age is measured in years and length in meters. Because dugongs tend not to keep track of their birthdays, ages are estimated based on variables such as the condition of their teeth.

dugong **=** Table**.**read\_table('http://www.statsci.org/data/oz/dugongs.txt')

dugong **=** dugong**.**move\_to\_start('Length')

dugong

| **Length** | **Age** |
| --- | --- |
| 1.8 | 1 |
| 1.85 | 1.5 |
| 1.87 | 1.5 |
| 1.77 | 1.5 |
| 2.02 | 2.5 |
| 2.27 | 4 |
| 2.15 | 5 |
| 2.26 | 5 |
| 2.35 | 7 |
| 2.47 | 8 |

... (17 rows omitted)

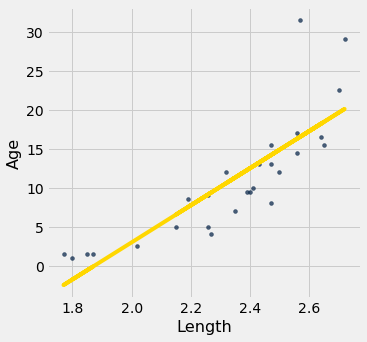
If we could measure the length of a dugong, what could we say about its age? Let’s examine what our data say. Here is a regression of age (the response) on length (the predictor). The correlation between the two variables is substantial, at 0.83.

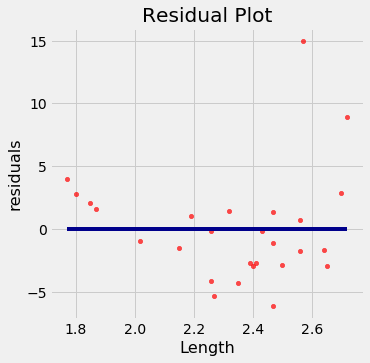
correlation(dugong, 'Length', 'Age')

0.8296474554905714

High correlation notwithstanding, the plot shows a curved pattern that is much more visible in the residual plot.

regression\_diagnostic\_plots(dugong, 'Length', 'Age')





While you can spot the non-linearity in the original scatter, it is more clearly evident in the residual plot.

At the low end of the lengths, the residuals are almost all positive; then they are almost all negative; then positive again at the high end of lengths. In other words the regression estimates have a pattern of being too high, then too low, then too high. That means it would have been better to use a curve instead of a straight line to estimate the ages.

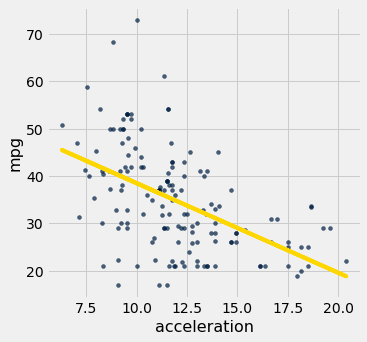
**When a residual plot shows a pattern, there may be a non-linear relation between the variables.**

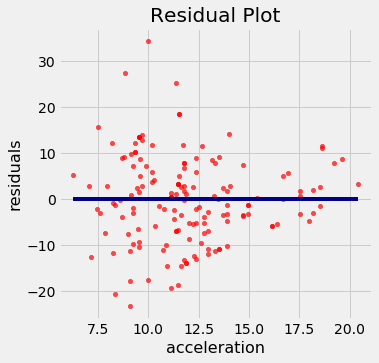
**Detecting Heteroscedasticity**

*Heteroscedasticity* is a word that will surely be of interest to those who are preparing for Spelling Bees. For data scientists, its interest lies in its meaning, which is “uneven spread”.

Recall the table hybrid that contains data on hybrid cars in the U.S. Here is a regression of fuel efficiency on the rate of acceleration. The association is negative: cars that accelearate quickly tend to be less efficient.

regression\_diagnostic\_plots(hybrid, 'acceleration', 'mpg')





Notice how the residual plot flares out towards the low end of the accelerations. In other words, the variability in the size of the errors is greater for low values of acceleration than for high values. Uneven variation is often more easily noticed in a residual plot than in the original scatter plot.

**If the residual plot shows uneven variation about the horizontal line at 0, the regression estimates are not equally accurate across the range of the predictor variable.**

**Numerical Diagnostics**

In addition to visualization, we can use numerical properties of residuals to assess the quality of regression. We will not prove these properties mathematically. Rather, we will observe them by computation and see what they tell us about the regression.

All of the facts listed below hold for all shapes of scatter plots, whether or not they are linear.

**Residual Plots Show No Trend**

**For every linear regression, whether good or bad, the residual plot shows no trend. Overall, it is flat. In other words, the residuals and the predictor variable are uncorrelated.**

You can see this in all the residual plots above. We can also calculate the correlation between the predictor variable and the residuals in each case.

correlation(heights, 'MidParent', 'Residual')

-2.719689807647064e-16

That doesn’t look like zero, but it is a tiny number that is 0 apart from rounding error due to computation. Here it is again, correct to 10 decimal places. The minus sign is because of the rounding that above.

round(correlation(heights, 'MidParent', 'Residual'), 10)

-0.0

dugong **=** dugong**.**with\_columns(

'Fitted Value', fit(dugong, 'Length', 'Age'),

'Residual', residual(dugong, 'Length', 'Age')

)

round(correlation(dugong, 'Length', 'Residual'), 10)

0.0

**Average of Residuals**

**No matter what the shape of the scatter diagram, the average of the residuals is 0.**

This is analogous to the fact that if you take any list of numbers and calculate the list of deviations from average, the average of the deviations is 0.

In all the residual plots above, you have seen the horizontal line at 0 going through the center of the plot. That is a visualization of this fact.

As a numerical example, here is the average of the residuals in the regression of children’s heights based on parents’ heights in Galton’s dataset.

round(np**.**mean(heights**.**column('Residual')), 10)

0.0

The same is true of the average of the residuals in the regression of the age of dugongs on their length. The mean of the residuals is 0, apart from rounding error.

round(np**.**mean(dugong**.**column('Residual')), 10)

0.0

**SD of the Residuals**

**No matter what the shape of the scatter plot, the SD of the residuals is a fraction of the SD of the response variable. The fraction is**√1−r21−r2**.**

SD of residuals = √1−r2⋅SD of ySD of residuals = 1−r2⋅SD of y

We will soon see how this measures the accuracy of the regression estimate. But first, let’s confirm it by example.

In the case of children’s heights and midparent heights, the SD of the residuals is about 3.39 inches.

np**.**std(heights**.**column('Residual'))

3.3880799163953426

That’s the same as √1−r21−r2 times the SD of response variable:

r **=** correlation(heights, 'MidParent', 'Child')

np**.**sqrt(1 **-** r**\*\***2) **\*** np**.**std(heights**.**column('Child'))

3.388079916395342

The same is true for the regression of mileage on acceleration of hybrid cars. The correlation rr is negative (about -0.5), but r2r2 is positive and therefore √1−r21−r2 is a fraction.

r **=** correlation(hybrid, 'acceleration', 'mpg')

r

-0.5060703843771186

hybrid **=** hybrid**.**with\_columns(

'fitted mpg', fit(hybrid, 'acceleration', 'mpg'),

'residual', residual(hybrid, 'acceleration', 'mpg')

)

np**.**std(hybrid**.**column('residual')), np**.**sqrt(1 **-** r**\*\***2)**\***np**.**std(hybrid**.**column('mpg'))

(9.43273683343029, 9.43273683343029)

Now let us see how the SD of the residuals is a measure of how good the regression is. Remember that the average of the residuals is 0. Therefore the smaller the SD of the residuals is, the closer the residuals are to 0. In other words, if the SD of the residuals is small, the overall size of the errors in regression is small.

The extreme cases are when r=1r=1 or r=−1r=−1. In both cases, √1−r2=01−r2=0. Therefore the residuals have an average of 0 and an SD of 0 as well, and therefore the residuals are all equal to 0. The regression line does a perfect job of estimation. As we saw earlier in this chapter, if r=±1r=±1, the scatter plot is a perfect straight line and is the same as the regression line, so indeed there is no error in the regression estimate.

But usually rr is not at the extremes. If rr is neither ±1±1 nor 0, then √1−r21−r2 is a proper fraction, and the rough overall size of the error of the regression estimate is somewhere between 0 and the SD of yy.

The worst case is when r=0r=0. Then √1−r2=11−r2=1, and the SD of the residuals is equal to the SD of yy. This is consistent with the observation that if r=0r=0 then the regression line is a flat line at the average of yy. In this situation, the root mean square error of regression is the root mean squared deviation from the average of yy, which is the SD of yy. In practical terms, if r=0r=0 then there is no linear association between the two variables, so there is no benefit in using linear regression.

**Another Way to Interpret**rr

We can rewrite the result above to say that no matter what the shape of the scatter plot,

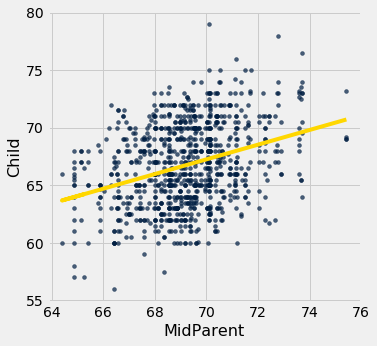
SD of residualsSD of y = √1−r2SD of residualsSD of y = 1−r2

|  |  |  |
| --- | --- | --- |
| A complentary result is that no matter what the shape of the scatter plot, the SD of the fitted values is a fraction of the SD of the observed values of yy. The fraction is | r | . |

SD of fitted valuesSD of y = |r|SD of fitted valuesSD of y = |r|

To see where the fraction comes in, notice that the fitted values are all on the regression line whereas the observed values of yy are the heights of all the points in the scatter plot and are more variable.

scatter\_fit(heights, 'MidParent', 'Child')



The fitted values range from about 64 to about 71, whereas the heights of all the children are quite a bit more variable, ranging from about 55 to 80.

To verify the result numerically, we just have to calculate both sides of the identity.

correlation(heights, 'MidParent', 'Child')

0.32094989606395924

Here is ratio of the SD of the fitted values and the SD of the observed values of birth weight:

np**.**std(heights**.**column('Fitted Value'))**/**np**.**std(heights**.**column('Child'))

0.32094989606395957

The ratio is equal to rr, confirming our result.

Where does the absolute value come in? First note that as SDs can’t be negative, nor can a ratio of SDs. So what happens when rr is negative? The example of fuel efficiency and acceleration will show us.

correlation(hybrid, 'acceleration', 'mpg')

-0.5060703843771186

np**.**std(hybrid**.**column('fitted mpg'))**/**np**.**std(hybrid**.**column('mpg'))

0.5060703843771186

|  |  |  |
| --- | --- | --- |
| The ratio of the two SDs is $ | r | $. |

A more standard way to express this result is to recall that

variance = mean squared deviation from average = SD2variance = mean squared deviation from average = SD2

and therefore, by squaring both sides of our result,

variance of fitted valuesvariance of y = r2