*7.6 LINEAR REGRESSION

Suppose you work in college admissions, and would like to determine how well various admissions data predict success in college. For example, if an applicant earned a 3.1 GPA in high school, is that a predictor of cumulative college GPA? Are SAT scores better or worse predictors?

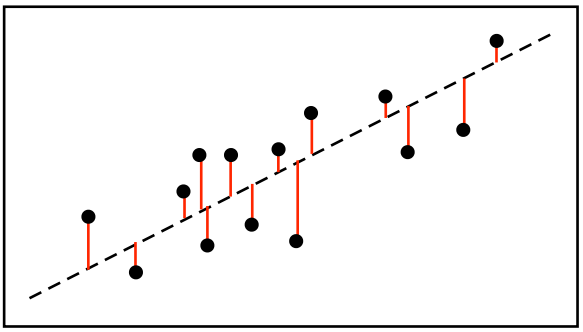
Analyses such as these, looking for relationships between two (or more) variables, are called *regression analyses*. In a regression analysis, we would like to find a function or formula that accurately predicts the value of a *dependent variable* based on the value of an *independent variable*. In the college admissions problem, the independent variables are high school GPA and SAT scores, and the dependent variable is cumulative GPA in college. A regression analysis would choose one independent variable (e.g., high school GPA) and try to find a function that accurately predicts the value of one dependent variable (e.g., college GPA). Regression is a fundamental technique of ***data mining***, which seeks to extract patterns and other meaningful information from large data sets.

The input to a regression problem is a data set consisting of n pairs of values. The first value in each pair is a value of the independent variable and the second value is the corresponding value of the dependent variable. For example, consider the following miniscule data set.

High school GPA	College GPA
3.1	2.8
3.8	3.7
3.4	3.2

We can also think of each row in the table, which represents one student, as an (x,y) point where x is the value of the independent variable and y is the value of the dependent variable.

The most common type of regression analysis is called ***linear regression***. A linear regression finds a straight line that most closely approximates the relationship between the two variables. The most commonly used linear regression technique, called the *least squares* method, finds the line that minimizes the sum of the squares of the vertical distances between the line and the data points. This is represented graphically below.



The red line segments in the figure represent the vertical distances between the points and the dashed line. This dashed line represents the line that results in the minimum total squared vertical distance for these points.

Mathematically, we are trying to find the line $y = mx + b$ (with slope m and y-intercept b)

for which

$$\sum_{(x,y)} (y - (mx + b))^2$$

is the minimum. The x and y in this notation represent any one of the points (x,y) in our data set; $(y-(mx+b))$ represents the vertical distance between the height of (x,y) (given by y) and the height of the line at (x,y) (given by $mx+b$). The uppercase Greek letter sigma (Σ) with (x,y) below it means that we are taking the sum over all points (x,y) in our data set.

To find the least squares line for a data set, we could test all of the possible lines, and choose the one with the minimum total squared distance. However, since there are an infinite number of such lines, this "brute force" approach would take a very long time. Fortunately, the least squares line can be found exactly using calculus. The slope m of this line is given by

$$m = \frac{n \cdot \sum(xy) - \sum x \cdot \sum y}{n \cdot \sum(x^2) - (\sum x)^2}$$

and the y-intercept b is given by

$$b = \frac{\sum y - m \sum x}{n}.$$

Although the notation is these formulas may look imposing, the quantities are really quite simple:

- n is the number of points
- $\sum x$ is the sum of the x coordinates of all of the points (x,y)
- $\sum y$ is the sum of the y coordinates of all of the points (x,y)
- $\sum(xy)$ is the sum of x times y for all of the points (x,y)
- $\sum(x^2)$ is the sum of the squares of the x coordinates of all of the points (x,y)

For example, suppose we had only three points: $(5,4)$, $(3,2)$, and $(8,3)$. Then

- $\sum x = 5 + 3 + 8 = 16$
- $\sum y = 4 + 2 + 3 = 9$
- $\sum(xy) = (5 \cdot 4) + (3 \cdot 2) + (8 \cdot 3) = 20 + 6 + 24 = 50$
- $\sum(x^2) = 5^2 + 3^2 + 8^2 = 25 + 9 + 64 = 98$

Therefore,

$$m = \frac{n \cdot \sum(xy) - \sum x \cdot \sum y}{n \cdot \sum(x^2) - (\sum x)^2} = \frac{3 \cdot 50 - 16 \cdot 9}{3 \cdot 98 - 16^2} = \frac{3}{19}$$

and

$$b = \frac{\sum y - m \sum x}{n} = \frac{9 - (3/19) \cdot 16}{3} = \frac{41}{19}.$$

Plugging in these values, we find that the formula for the least squares line is

$$y = \left(\frac{3}{19}\right)x + \frac{41}{19},$$

which is plotted below.

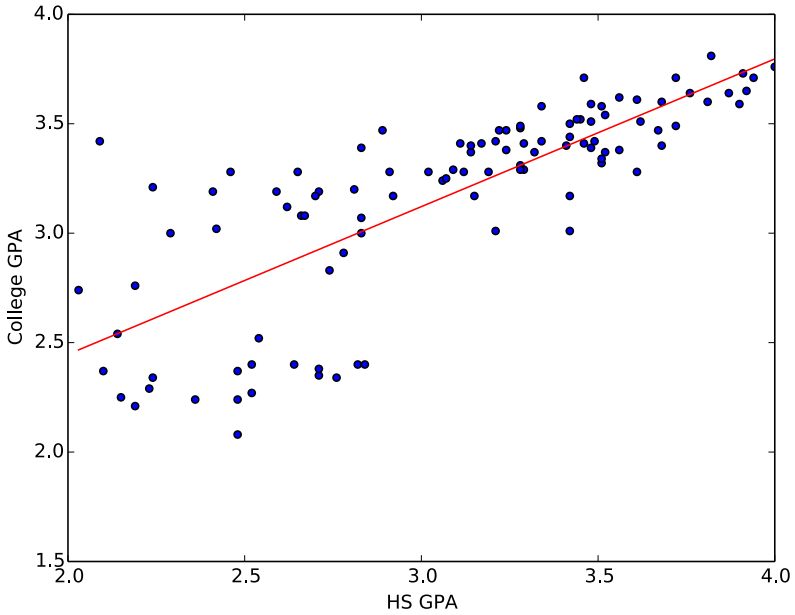
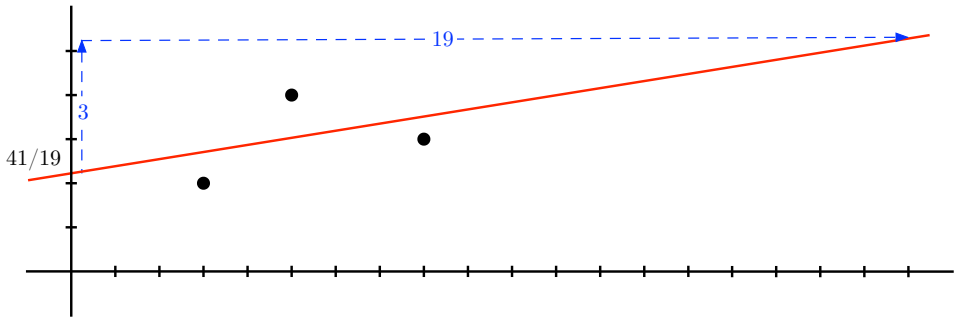


Figure 1 High school GPA and corresponding college GPA with regression line.



Given these formulas, it is fairly easy to write a `linearRegression` function to find the least squares regression line. Suppose the function takes as parameters a list of x coordinates named `x` and a list of y coordinates named `y` (just like `pyplot.plot`). The x coordinates are values of the independent variable and the y coordinates are the values of the dependent variable. We can use four accumulators to compute the four sums above. For example, $\sum x$ can be computed with

```
n = len(x)              # number of points
sumx = 0                # sum of x coordinates
for index in range(n):
    sumx = sumx + x[index]
```

Exercise 7.6.1 asks you to complete the implementation of this function.

Once we have a function that performs linear regression, it is fairly simple to plot a set of points with the regression line:

```
import matplotlib.pyplot as pyplot

def plotRegression(x, y, xLabel, yLabel):
    """Plot points in x and y with a linear regression line.

    Parameters:
        x:      a list of x values (independent variable)
        y:      a list of y values (dependent variable)
        xLabel: a string to label the x axis
        yLabel: a string to label the y axis

    Return value: None
    """

    pyplot.scatter(x, y)            # plot the points

    m, b = linearRegression(x, y)   # find the regression line

    # plot the regression line
    minX = min(x)
    maxX = max(x)
    pyplot.plot([minX, maxX], [m * minX + b, m * maxX + b], color = 'red')
    pyplot.xlabel(xLabel)
    pyplot.ylabel(yLabel)
    pyplot.show()
```

Returning to our college admissions problem, suppose the high school GPA values are in a list named `hsGPA` and the college GPA values are in a list named `collegeGPA`. Then we can get our regression line by calling

```
plotRegression(hsGPA, collegeGPA, 'HS GPA', 'College GPA')
```

An example plot with real data is shown in Figure 1.

Reflection 1 *What can you discern from this plot? Does high school GPA do a good job of predicting college GPA?*

In the exercises below, and in Project 7.4, you will have the opportunity to investigate this problem in more detail. Projects 7.3 and 7.5 also use linear regression to approximate the demand curve for an economics problem and predict flood levels on the Snake River, respectively.

Exercises

7.6.1* Complete the function

```
linearRegression(x, y)
```

The function should return the slope `m` and y-intercept `b` of the least squares regression line for the points whose x and y coordinates are stored in the lists `x` and `y`, respectively. (Your function should use only one loop.)

7.6.2* The table below lists the average homework and exam scores for a class (one row per student). Write a program that uses the completed `linearRegression`

function from Exercise 7.6.1 and the `plotRegression` function to plot a linear regression line for this data.

HW	Exam
63	73
91	99
81	98
67	82
100	97
87	99
91	96
74	77
26	33
100	98
78	100
59	81
85	38
69	74

7.6.3. On the book website, you will find a CSV data file named `sat.csv` that contains GPA and SAT data for 105 students. Write a function

```
readData(filename)
```

that reads the data from this file and returns a tuple of two lists containing the data in the first and fourth columns of the file (high school GPAs and college GPAs). Then use the `plotRegression` function (which will call your completed `linearRegression` function from Exercise 7.6.1) to plot this data with a linear regression line to determine whether there is a correlation between high school GPA and college GPA. (Your plot should look like Figure 1.)

7.6.4. A standard way to measure how well a regression line fits a set of data is to compute the coefficient of determination, often called the R^2 coefficient, of the regression line. R^2 is defined to be

$$R^2 = 1 - \frac{S}{T}$$

where S and T are defined as follows:

$$S = \sum_{(x,y)} (y - (mx + b))^2$$

$$T = \sum_{(x,y)} (y - \bar{y})^2, \text{ where } \bar{y} \text{ is the mean } y \text{ value}$$

For example, the three points in the text $(5, 4)$, $(3, 2)$, and $(8, 3)$ have regression line

$$y = \left(\frac{3}{19}\right)x + \frac{41}{19}.$$

So $m = 3/19$ and $b = 41/19$. Therefore,

- $\bar{y} = (4 + 2 + 3)/3 = 3$
- $T = \sum(y - \bar{y})^2 = (4 - 3)^2 + (2 - 3)^2 + (3 - 3)^2 = 2$

- $S = \sum(y - (mx + b))^2 = (4 - 56/19)^2 + (2 - 50/19)^2 + (3 - 65/19)^2 = 608/361$
- $R^2 = 1 - (608/361)/2 = 1 - 304/361 = 57/361 \approx 0.15789$

The R^2 coefficient is always between 0 and 1, with values closer to 1 indicating a better fit.

Write a function

```
rSquared(x, y, m, b)
```

that returns the R^2 coefficient for the set of points whose x and y coordinates are stored in the lists `x` and `y`, respectively. The third and fourth parameters are the slope and y-intercept of the regression line for the set of points. For example, to apply your function to the example above, you would call

```
rSquared([5, 3, 8], [4, 2, 3], 3/19, 41/19)
```

7.6.5. An alternative linear regression method, called a *Deming regression* finds the line that minimizes the squares of the perpendicular distances between the points and the line rather than the vertical distances. While the traditional least squares method accounts only for errors in the y values, this technique accounts for errors in both x and y. The slope and y-intercept of the line found by this method are[4]

$$m = \frac{s_{yy} - s_{xx} + \sqrt{(s_{yy} - s_{xx})^2 + 4(s_{xy})^2}}{2s_{xy}}$$

and

$$b = \bar{y} - m\bar{x}$$

where

- $\bar{x} = (1/n)\sum x$, the mean of the x coordinates of all of the points (x,y)
- $\bar{y} = (1/n)\sum y$, the mean of the y coordinates of all of the points (x,y)
- $s_{xx} = (1/(n-1))\sum(x - \bar{x})^2$
- $s_{yy} = (1/(n-1))\sum(y - \bar{y})^2$
- $s_{xy} = (1/(n-1))\sum(x - \bar{x})(y - \bar{y})$

Write a function

```
linearRegressionDeming(x, y)
```

that computes these values of m and b. Test your function by using it in the `plotRegression` function, applied to a data set from a previous exercise in this section.

[4] These formulas assume that the variances of the x and y errors are equal.

Selected Exercise Solutions

7.6.1
```
def linearRegression(x, y):
    n = len(x) # number of points
    sumx = 0   # sum of x coordinates
    sumy = 0   # sum of y coordinates
    sumxy = 0  # sum of products of x and y coordinates
    sumxx = 0  # sum of squares of x coordinates
    for index in range(n):
        sumx = sumx + x[index]
        sumy = sumy + y[index]
        sumxy = sumxy + x[index] * y[index]
        sumxx = sumxx + x[index] * x[index]
    sumx2 = sumx ** 2   # square of sum of x coordinates

    m = (n * sumxy - sumx * sumy) / (n * sumxx - sumx2)  # slope
    b = (sumy - m * sumx) / n                            # y intercept
    return m, b
```

7.6.2
```
hw = [63, 91, 81, 67, 100, 87, 91, 74, 26, 100,78, 59, 85, 69]
exam = [73, 99, 98, 82, 97, 99, 96, 77, 33, 98, 100, 81, 38, 74]
plotRegression(hw, exam, 'Average HW Score', 'Average Exam Score')
```