CSE1729 - Introduction to Programming | March 4, 2022

Problem Set 6

1. Define a SCHEME function, `zip`, which takes as arguments two lists $(a_1 \ldots a_n)$ and $(b_1 \ldots b_n)$ of the same length, and produces the list of pairs $((a_1.b_1) \ldots (a_n.b_n))$.

2. Define a SCHEME function, `unzip`, which takes a list of pairs $((a_1.b_1) \ldots (a_n.b_n))$ and returns a *pair* consisting of the two lists $(a_1 \ldots a_n)$ and $(b_1 \ldots b_n)$. Note that these functions are not exactly inverses of each other, since `zip` takes its lists as a two arguments, while `unzip` produces a pair of lists. Finally, it is wise to realize how SCHEME prints out a pair of lists. Consider the statement `(cons (list 1 2) (list 3 4))` which denotes a pair of lists. It prints as `((1 2) 3 4)`.

3. *Pearson's Coefficient* is widely used in the natural sciences as a measure of "correlation" between two variables. For example, it is reasonable to expect that human height and weight are correlated, which is to say that—on average—taller humans tend to weigh more. As an example, consider the heights and weights of a small population of humans, as shown in the table below:

	A	B	C	D	E	F	G
height (cm)	160	186	172	202	177	186	191
weight (kg)	51	79	69	100	66	80	83

   Examining the data you can see that, as a rule, weight *does* increase with height, though there are some exceptions.

   The Pearson coefficient (of a collection of samples with two features, like height and weight above) is always between -1 and 1. A Pearson coefficient of exactly 1 (or exactly -1) indicates a perfectly linear relationship between the two quantities, whereas a coefficient near zero indicates that there is no such nice relationship. The data above has a Pearson coefficient of $\approx .98$, indicating an extremely strong relationship between the variables.

   Given two lists, $X = (x_1\ x_2\ \ldots\ x_n)$ and $Y = (y_1\ y_2\ \ldots\ y_n)$, each containing n values, Pearson's Coefficient is defined by the expression:

$$\frac{\sum_{i=1}^{n}(x_i - \bar{X})(y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{Y})^2}},$$

   where $\bar{X}$ and $\bar{Y}$ denote the averages of the x_i and the y_i, which is to say that

$$\bar{X} = \frac{1}{n}\sum_i x_i \quad \text{and} \quad \bar{Y} = \frac{1}{n}\sum_i y_i\,.$$

   (a) Write a function, `list-sum`, that takes a list and returns its sum.

   (b) Write a function that takes a list $X = (x_1\ x_2\ \ldots\ x_n)$ and returns the average $\bar{X}$. (Note: You will have to compute the length of the list in order to compute the average.) Call your function `average`.

(c) Write a function, `var-map`, that *maps* a list X to the "square of its deviation." Thus the list $X = (x_1\ x_2\ \ldots\ x_n)$ should be carried to the list

$$((x_1 - \bar{X})^2\ \ \ldots\ \ (x_n - \bar{X})^2).$$

You should use the `map` function. (For example, your function, when evaluated on the list (1 2 3 4 5), should return (4 1 0 1 4).)

(d) Write a function `stdev` that returns the *standard deviation* of a list. Specifically, given the list $(x_1 \ldots x_n)$, your functions should return

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{X})^2}.$$

This should be easy, you already have the functions you need.

(e) Write a new version (called `map2`) of the `map` function which operates on two lists. Your function should accept three parameters, a function f and two lists X and Y, and return a new list composed of the function f applied to corresponding elements of X and Y. For concreteness, place the function first among the parameters, so that a call to the function appears as `(map2 f X Y)`. Your function may assume that the two lists have the same length. In particular, given two lists $(x_1\ x_2\ \ldots\ x_n)$ and $(y_1\ y_2\ \ldots\ y_n)$ (and the function f), `map2` should return

$$(f(x_1, y_1)\ \ldots\ f(x_n, y_n)).$$

(f) Write a function, `covar-elements`, that, given two lists $X = (x_1\ x_2\ \ldots\ x_n)$ and $Y = (y_1\ y_2\ \ldots\ y_n)$, returns the covariance list:

$$((x_1 - \bar{X})(y_1 - \bar{Y})\ \ \ldots\ \ (X_n - \bar{X})(Y_n - \bar{Y})).$$

Note that the resulting list should have the same length as the two input lists; the ith element of the resulting list is the product $(x_i - \overline{X})(y_i - \overline{Y})$.

(g) Write a function, `pearson`, that computes Pearson's Coefficient. It might be useful to observe that an equivalent way to write Pearson's coefficient, by dividing the top and bottom by n, is

$$\frac{{}^1\!/_n \sum_{i=1}^{n}(x_i - \bar{X})(y_i - \bar{Y})}{\sigma(X)\sigma(Y)}.$$

(Thus, your function should take two lists as arguments, and return the value indicated above.)

4. *Least-Squares Line Fitting.* Line fitting refers to the process of finding the "best" fitting line for a set of points. This is one of the most fundamental tools that natural scientists use to fit mathematical models to experimental data.

As an example, consider the red points in the scatter plot of Figure 1. They do not lie on a line, but there is a line that "fits" them very well, shown in black. This line has been chosen—among all possible lines—to be the one that minimizes the sum of *squares* of the vertical distances from the points to the line. (This may seem like an odd thing to minimize, but it turns out that there are several reasons that it is the "right" thing minimize.)

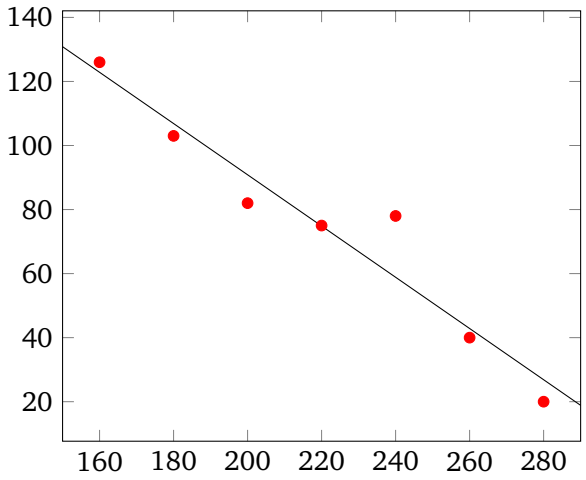

Figure 1: An example of a scatter plot and a best-fit line.

Since the equation of a line is $y = ax + b$, this boils down to finding a slope a and x-intercept b for given lists X and Y corresponding to point data. It turns out that the best fitting line can be found with the following two equations:

$$a = r \cdot \frac{\sigma(Y)}{\sigma(X)}, \quad \text{and} \quad b = \bar{Y} - a\bar{X},$$

where $\sigma(X)$ and $\sigma(Y)$ refer to the standard deviations of X and Y and r is Pearson's Coefficient.

(a) Write a function `best-fit` that takes two lists X and Y (of the same length) and returns a pair (a, b) corresponding to the (slope and intercept of the) best fit line for the data.

(b) Write an alternate version of `best-fit` that returns the best fitting line as a *function*. Specifically, write a new function `best-fit-fn` so that `(best-fit-fn pX pY)` returns the *function* which, given a number x, returns $ax + b$ where a and b are the best fit coefficients.

Use this to the best fit line for the following data:

$$X = \{160, 180, 200, 220, 240, 260, 280\}, \qquad Y = \{126, 103, 82, 75, 78, 40, 20\}.$$