01_27_correlations_sort_files

Unknown Author

January 28, 2014

Part I

Correlations, sorting, file input/output

0.1 Correlations

Given a data consisting of a pair of measurements/variables correlation gives a statistical measure of relationship between the two variables. Standard examples are height/weight, scores etc. Let us denote the data by (x_i, y_i), i runs from 1 to N say. If they are linearly related the plot of x_i, y_i would lie on a straight line. We want points which more or less lie on a straight line to have correlation 1 or -1 depending upon whether he slope is posiive or negaive, and if he data is random, the correlation should be zero. (x_i, y_i) lying on a line means that $(x_i - \mu_x, y_i - \mu_y)$ lie on a line passing through origin. In other words, the vector $(y_1 - \mu_y, \ldots, y_N - \mu_y)$ is a multiple of the vector $(x_1 - \mu_x, \ldots, x_N - \mu_N)$. Cauchy Shwartz inequality for $\mathbb{R}^N$ will then tell us : $\sum_i (x_i - \mu_x)(y_i - \mu_y)/(\sqrt{(\sum_i (x_i - \mu_x)^2)(\sum_i (y_i - \mu_y)^2}$ will lie between -1 and 1, and it attains -1 or 1 only when the vectors are multiples of each other. This motivates the following definition of correlation.

$$\text{Correlation}(X_i, Y_i) = \frac{\text{Covariance}(X_i, Y_i)}{\text{s.d.}(X_i)\text{s.d.}(Y_i)} \tag{1}$$

where

$$\text{Covariance}(X_i, Y_i) = \frac{1}{N}\sum_{i=1}^{N}(X_i - \mu_X)(Y_i - \mu_Y), \tag{2}$$

μ_X being the mean of X_i's and μ_Y being the mean of the Y_i's.As before we we try to simplify the formula so that we can compute using just one loop to compute.

$$\sum_i (X_i - \mu_X)(Y_i - \mu_Y) = \sum_i X_i Y_i - \mu_X \sum_i Y_i - \mu_Y \sum_i X_i + N\mu_X\mu_Y = \sum_i X_i Y_i - N\mu_X\mu_Y - N\mu_X\mu_Y + N\mu_X\mu_Y \tag{3}$$

$$= \sum_i X_i Y_i - \frac{1}{N}\left(\sum_i X_i\right)\left(\sum_i X_i\right). \tag{4}$$

Let us try this out

In [1]:
```
# For finding standard deviation, we need
from math import sqrt
```

In [2]:
```
def my_corr(lst_of_2_tuples) :
    """Given a list of 2-tuples, this functions computes the correlation between the f
    second entries."""

    # As before we use a huge bunch of variables.
    sumx = 0.0
    sumy = 0.0
    sumxy = 0.0
    sumx2 = 0.0
    sumy2 = 0.0

    # Now loop
    for (x, y) in lst_of_2_tuples :

        # Now accumulate
        sumx += x
        sumy += y
        sumxy += x * y
        sumx2 += x * x
        sumy2 += y * y

    # Now we got all the ingredients to compute covariance and s.d. except n :
    n = len(lst_of_2_tuples)

    # Now compute
    covariance = sumxy - sumx * sumy / n
    sdx = sqrt(sumx2 - sumx**2/n)
    sdy = sqrt(sumy2 - sumy**2/n)

    if sdx == 0 or sdy == 0 :
        print "\nError: Correlation: One of the variables is constant. Cannot compute
        correlation = None
    else :
        correlation = covariance / (sdx * sdy)

    return correlation
```

In [3]:
```
data_mid = [23, 45, 83, 90, 12, 87, 67, 69, 74, 36, 43, 69, 66, 70]
data_end = [45, 44, 95, 87, 24, 100, 45, 70, 66, 32, 50, 55, 80, 81]

zipped_data = zip(data_mid, data_end)
print "Zipped data : ", zipped_data
```

```
Zipped data :  [(23, 45), (45, 44), (83, 95), (90, 87), (12, 24), (87,
100), (67, 45), (69, 70), (74, 66), (36, 32), (43, 50), (69, 55), (66,
80), (70, 81)]
```

In [4]:
```
print "Correlation is", my_corr(zipped_data)
```

```
Correlation is 0.867523870625
```

We can experiment

In [5]:
```
def test_corr(lst_of_2_tups) :
    print "Correlation of", lst_of_2_tups, "is", my_corr(lst_of_2_tups)
```

In [6]:
```
test_corr([(1, 5), (3, 9), (10, 23), (-2, -1), (0, 3)])
test_corr([(1, 0), (0, 1), (1, 1), (0, 0)])
test_corr([(1, 0), (0, 1), (1, 1)])
test_corr([(x, 1) for x in range(5)])
test_corr([(x**2, x) for x in range(0, 100)])
```

```
Correlation of [(1, 5), (3, 9), (10, 23), (-2, -1), (0, 3)] is 1.0
Correlation of [(1, 0), (0, 1), (1, 1), (0, 0)] is 0.0
Correlation of [(1, 0), (0, 1), (1, 1)] is -0.5
Correlation of [(0, 1), (1, 1), (2, 1), (3, 1), (4, 1)] is
```

```
Error: Correlation: One of the variables is constant. Cannot compute
correlation.
None
Correlation of [(0, 0), (1, 1), (4, 2), (9, 3), (16, 4), (25, 5), (36,
6), (49, 7), (64, 8), (81, 9), (100, 10), (121, 11), (144, 12), (169,
13), (196, 14), (225, 15), (256, 16), (289, 17), (324, 18), (361, 19),
(400, 20), (441, 21), (484, 22), (529, 23), (576, 24), (625, 25),
(676, 26), (729, 27), (784, 28), (841, 29), (900, 30), (961, 31),
(1024, 32), (1089, 33), (1156, 34), (1225, 35), (1296, 36), (1369,
37), (1444, 38), (1521, 39), (1600, 40), (1681, 41), (1764, 42),
(1849, 43), (1936, 44), (2025, 45), (2116, 46), (2209, 47), (2304,
48), (2401, 49), (2500, 50), (2601, 51), (2704, 52), (2809, 53),
(2916, 54), (3025, 55), (3136, 56), (3249, 57), (3364, 58), (3481,
59), (3600, 60), (3721, 61), (3844, 62), (3969, 63), (4096, 64),
(4225, 65), (4356, 66), (4489, 67), (4624, 68), (4761, 69), (4900,
70), (5041, 71), (5184, 72), (5329, 73), (5476, 74), (5625, 75),
(5776, 76), (5929, 77), (6084, 78), (6241, 79), (6400, 80), (6561,
81), (6724, 82), (6889, 83), (7056, 84), (7225, 85), (7396, 86),
(7569, 87), (7744, 88), (7921, 89), (8100, 90), (8281, 91), (8464,
92), (8649, 93), (8836, 94), (9025, 95), (9216, 96), (9409, 97),
(9604, 98), (9801, 99)] is 0.967644392713
```

0.2 Sorting

Given a list of numbers (or any list of sortable elements) we can sort them using the following simple algorithmStart at the beginning of the list. Compare the adjacent entries. If they are in wrong order swap. Advance by one place. Repeat till nothing is swapped in on full sweep.

In [7]:
```
def horrible_sort(somelist, showstep=False) :
    swapped_during_pass = True
    while (swapped_during_pass) :
        swapped_during_pass = False
        for i in range(len(somelist) - 1) :
            if somelist[i] > somelist[i+1] :
                k = somelist[i]
                somelist[i] = somelist[i+1]
                somelist[i+1] = k
                swapped_during_pass = True
            if showstep :
                print somelist
    return somelist
```

In [8]:
```
print horrible_sort([3,1,4,2,5,0])
print horrible_sort([1,3,1,3,1,3,1], True)
```

```
[0, 1, 2, 3, 4, 5]
[1, 3, 1, 3, 1, 3, 1]
[1, 1, 3, 3, 1, 3, 1]
[1, 1, 3, 3, 1, 3, 1]
[1, 1, 3, 1, 3, 3, 1]
[1, 1, 3, 1, 3, 3, 1]
[1, 1, 3, 1, 3, 1, 3]
[1, 1, 3, 1, 3, 1, 3]
[1, 1, 3, 1, 3, 1, 3]
[1, 1, 1, 3, 3, 1, 3]
[1, 1, 1, 3, 3, 1, 3]
[1, 1, 1, 3, 1, 3, 3]
```

```
[1, 1, 1, 3, 1, 3, 3]
[1, 1, 1, 3, 1, 3, 3]
[1, 1, 1, 3, 1, 3, 3]
[1, 1, 1, 3, 1, 3, 3]
[1, 1, 1, 1, 3, 3, 3]
[1, 1, 1, 1, 3, 3, 3]
[1, 1, 1, 1, 3, 3, 3]
[1, 1, 1, 1, 3, 3, 3]
[1, 1, 1, 1, 3, 3, 3]
[1, 1, 1, 1, 3, 3, 3]
[1, 1, 1, 1, 3, 3, 3]
[1, 1, 1, 1, 3, 3, 3]
[1, 1, 1, 1, 3, 3, 3]
```

0.3 Reading from files

In [9]:
```
data_file = open("files/01_27_data.txt", "r")
for line in data_file :
    print line
data_file.close()
```

```
Temperature   Ice Cream Sales

        14.2            215

        16.4            325

        11.9            185

        15.2            332

        18.5            406

        22.1            522

        19.4            412

        25.1            614

        23.4            544

        18.1            421

        22.6            445

        17.2            408
```

Now we can extract the data using the split() function as follows:

In [10]:
```
data_file = open("files/01_27_data.txt", 'r')
for line in data_file :
    print line.split()
data_file.close()
```

```
['Temperature', 'Ice', 'Cream', 'Sales']
['14.2', '215']
['16.4', '325']
['11.9', '185']
['15.2', '332']
['18.5', '406']
['22.1', '522']
['19.4', '412']
['25.1', '614']
['23.4', '544']
['18.1', '421']
['22.6', '445']
['17.2', '408']
```

However the entries are strings and the first line has to be discarded. We do this as follows. i keeps track of which line we are in. If it is not the first line, we convert the strings into float an store them.

In [11]:
```
data_file = open("files/01_27_data.txt", 'r')
i = 0
ice_cream_data = []
for line in data_file :
    if i > 0 :
        ice_cream_data.append((float(line.split()[0]), float(line.split()[1])))
    i += 1
data_file.close()
print ice_cream_data
```

```
[(14.2, 215.0), (16.4, 325.0), (11.9, 185.0), (15.2, 332.0), (18.5,
406.0), (22.1, 522.0), (19.4, 412.0), (25.1, 614.0), (23.4, 544.0),
(18.1, 421.0), (22.6, 445.0), (17.2, 408.0)]
```

Okay! Now that we have a list of tuples, we can find the correlation!

In [12]:
```
data_file = open("files/01_27_data.txt", 'r')
i = 0
ice_cream_data = []
for line in data_file :
    if i > 0 :
        ice_cream_data.append((float(line.split()[0]), float(line.split()[1])))
    i += 1
data_file.close()
print "Correlation for the icecream data is %6.4f" % my_corr(ice_cream_data)
```

```
Correlation for the icecream data is 0.9575
```