SciComp with Py

Covariance and Correlation

Vladimir Kulyukin
Department of Computer Science
Utah State University

Standard Deviation: Population vs. Sample

- Suppose there is a certain population (e.g., the petals of all roses in your garden, the salaries of all employees in a company, etc.)
- We can either take measurements of the entire population or take a sample of the population
- Suppose that we want to measure how spread out the numbers are; if we measure the entire population, we have population standard deviation; if we measure just a sample, we have sample standard deviation

Population Standard Deviation

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

number of elements in population

population's mean

Sample Standard Deviation

$$s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2}$$

number of elements in sample

sample's mean

Population and Sample STD in Numpy

```
>>> np.std([5, 4, 9, 2, 12, 7])
3.304037933598349
>>> np.std([9, 2, 5, 4, 12, 7], ddof=1)
3.619392214170713
>>> lst = [5, 4, 9, 2, 12, 7]
>>> m = np.mean(lst)
>>> s = [(xi - m)**2 for xi in lst]
>>> math.sqrt(sum(s)/len(s))
3.304037933599835
>>> math.sqrt(sum(s)/(len(s)-1))
3.619392214170713
```

population STD

sample STD

verifying population STD

verifying sample STD

Population and Sample STD in Numpy

```
>>> import numpy as np
>>> np.std([5, 4, 9, 2, 12, 7])
3.3040379335998349
```

population STD

```
>>> np.std([5, 4, 9, 2, 12, 7], ddof=1)
3.6193922141707713
```

sample STD

Covariance and Correlation

- Covariance and correlation are mathematical means of measuring how the values of two variables are related to each other
- Example 1: T is the temperature inside a beehive; N is the number of the fanning worker bees
- Example 2: A is a person's age; M is the amount of money a person spends per year

Measuring Covariance

- Given two vectors of values, V1 and V2, of the same size where V1 consists of the values of the first variable (say X) and V2 consists of the values of the second variable (say Y)
- Compute the means of V1 and V2
- Compute DV1 as the vector of the deviations of each value in V1 from V1's mean
- Compute DV2 as the vector of the deviations of each value in V2 from V2's mean
- Compute the dot product between DV1 and DV2 and divide by the number of elements in DV1

Computing Covariance

pylab is another library that comes in handy in addition to numpy

```
import pylab as plb
def dev_from_mean(x):
  xmean = plb.mean(x)
  return [xi - xmean for xi in x]

def covariance(x, y):
  n = len(x)
  x_dev = dev_from_mean(x)
  y_dev = dev_from_mean(y)
  return plb.dot(x_dev, y_dev) / (n-1)
```

Interpreting Covariance

- Covariance can be positive or negative (this is the so called inverse covariance)
- A small covariance close to 0 means that there is not much relationship between the two variables
- But what if covariance is large?
- How large does it have to be for us to say that the two variables are correlated?

Enter Correlation

- Correlation is covariance divided by the standard deviations of both variables
- Correlation makes it easier for us to interpret the degree of relationship between two variables
- A correlation of 0 means there is no relationship
- A correlation of 1 means that the two variables are directly related
- A correlation of -1 means that the two variables are inversely related

Computing Correlation

standard deviation of a numpy vector

```
def correlation(x, y):
  stdx = x.std()
  stdy = y.std()
  return covariance(x, y) / stdx / stdy
```

Caveat: Correlation vs. Causality

Correlation does not necessarily imply causality

Determining Causality

Causality is determined by experiment or theorem proving.

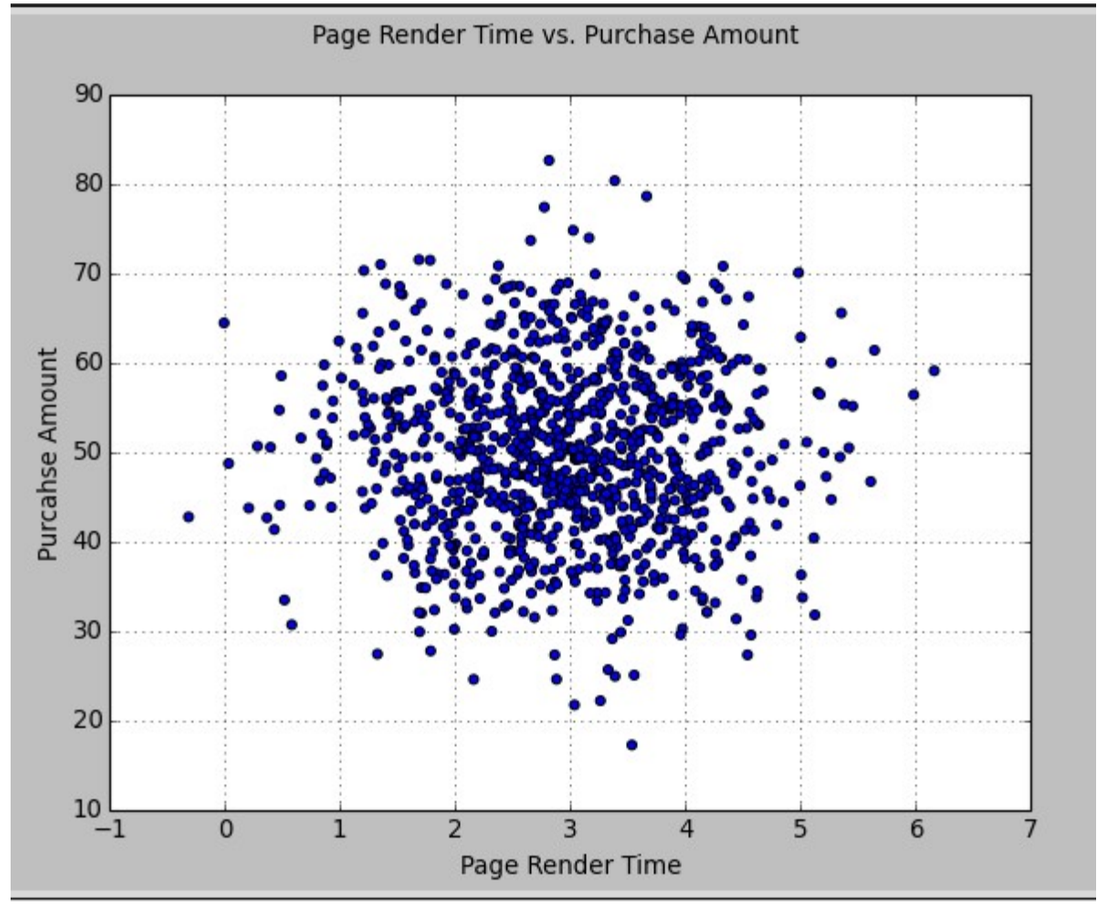
Correlation tells you what experiments you may want to run or what theorems you may want to prove.

Problem

You are a data scientist at an e-commerce company. The company is interested in finding a correlation between page rendition time (how fast a page displays to customer) and the amount of money a customer spends on that page. As the data are being collected, you start playing with simulated data to understand the types of feasible relationships between the two variables.

Simulated Solution: Low Covariance

```
page_render_time = np.random.normal(3.0, 1.0, 1000)
purchase_amount = np.random.normal(50.0, 10.0, 1000)


fig1 = plt.figure(1)
fig1.suptitle('Page Render Time vs. Purchase Amount')
plt.xlabel('Page Render Time')
plt.ylabel('Purcahse Amount')
plt.grid()
plt.scatter(page_render_time, purchase_amount)


print('covar = %f' % covariance(page_render_time,purchase_amount))
plt.show()
```

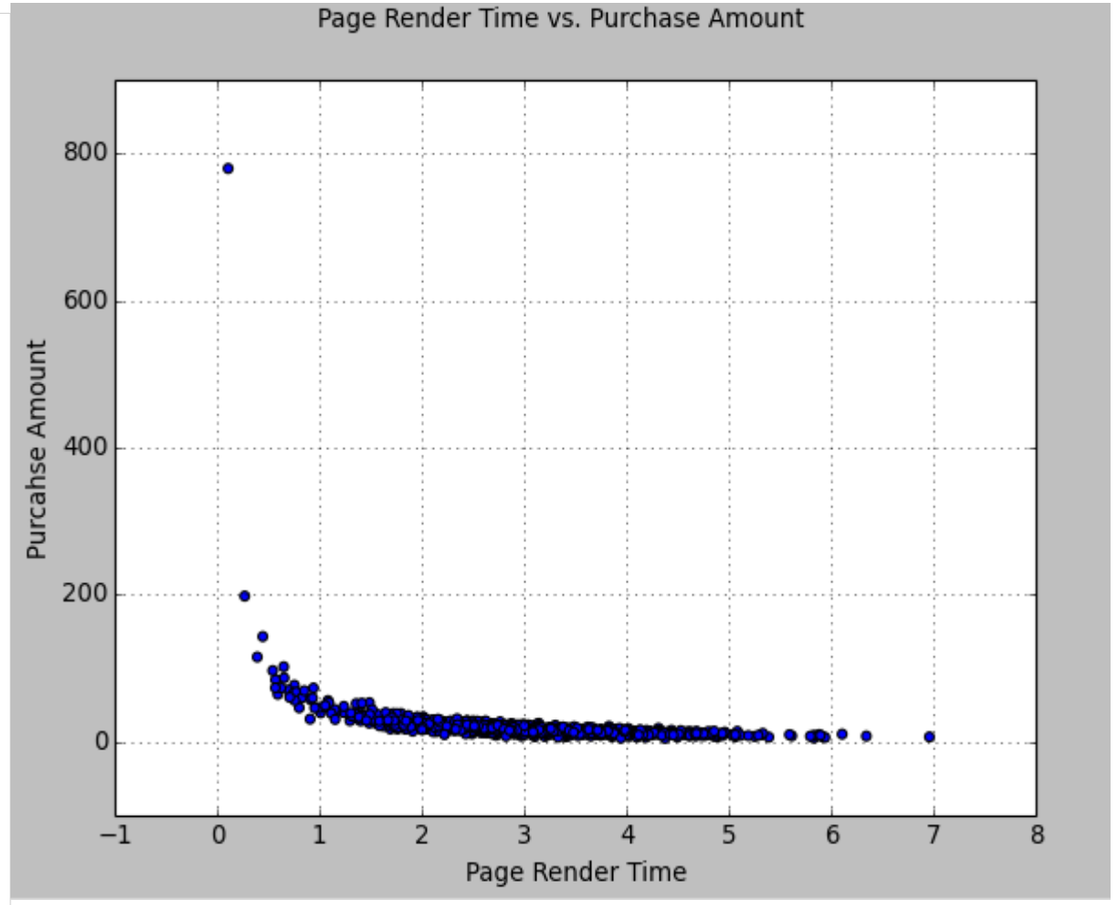


covar = 0.038134

source in covar_correl.py

Simulated Solution: Large Inverse Covariance

```
page_render_time = np.random.normal(3.0, 1.0, 1000)

purchase_amount = np.random.normal(50.0, 10.0, 1000) /page_render_time


fig1 = plt.figure(1)

fig1.suptitle('Page Render Time vs. Purchase Amount')

plt.xlabel('Page Render Time')

plt.ylabel('Purcahse Amount')

plt.grid()

plt.scatter(page_render_time, purchase_amount)


print(covariance(page_render_rime, purchase_amount))

plt.show()
```



covar = -11.935704878

source in covar_correl_02.py

Simulated Solution: Using Numpy

Py

```
pageRenderTime = np.random.normal(3.0, 1.0, 1000)
purchaseAmount = np.random.normal(50.0, 10.0, 1000) /pageRenderTime


print('covar=%f' % covariance(pageRenderTime, purchaseAmount))
print('correl=%f' % correlation(pageRenderTime, purchaseAmount))


## here is how you can compute correlation and covariance w/ numpy.
print('Numpy Correlation')
print(np.corrcoef(pageRenderTime, purchaseAmount))
print('Numpy Covariance')
print(np.cov(pageRenderTime, purchaseAmount))
```

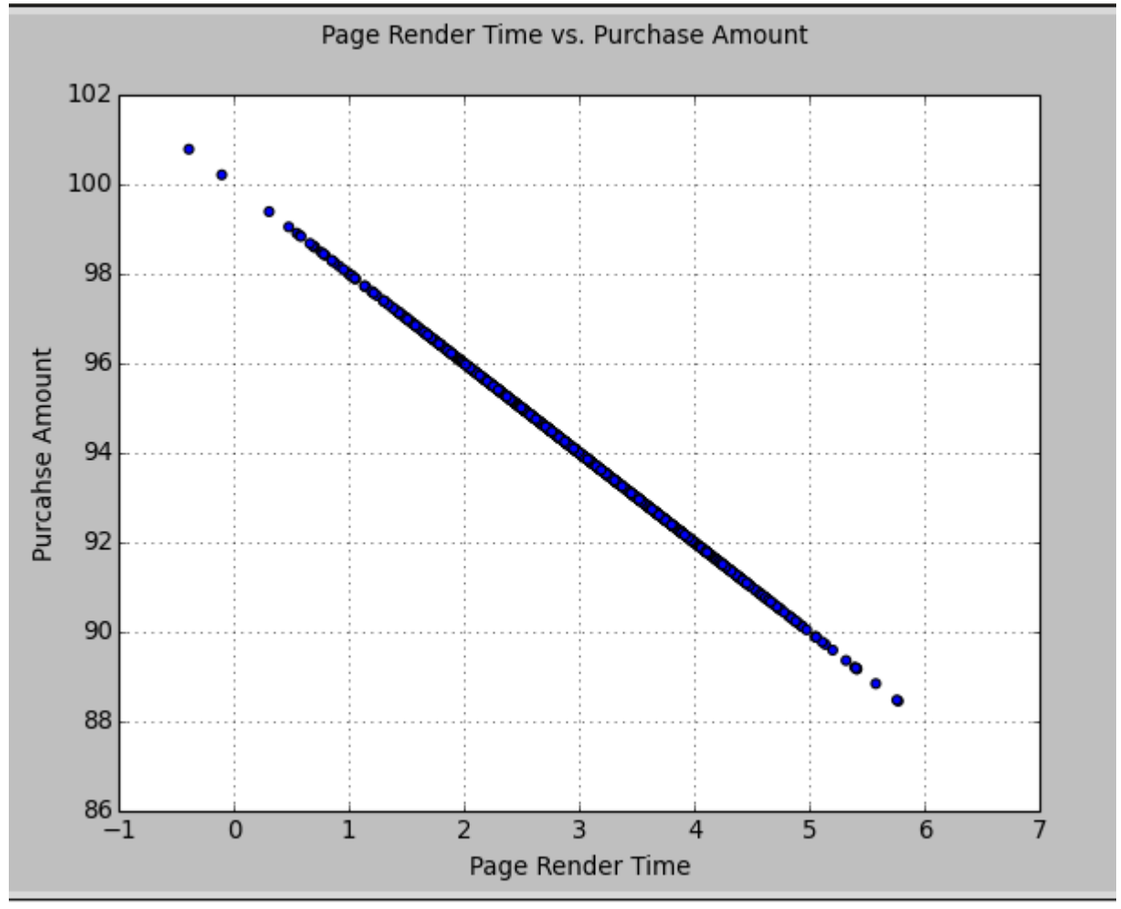
Output

```
covar=-11.346364
correl=-0.461441
Numpy Correlation
[[ 1.         -0.46097982]
 [-0.46097982  1.        ]]
Numpy Covariance
[[   0.95645996  -11.34636365]
 [ -11.34636365  633.40687855]]
```

source in covar_correl_03.py

Simulated Solution: Perfect Inverse Correlation

```
## an example of strong negative correlation

pageRenderTime = np.random.normal(3.0, 1.0, 1000)

purchaseAmount = 100 - pageRenderTime * 2


fig1 = plt.figure(1)

fig1.suptitle('Page Render Time vs. Purchase Amount')

plt.xlabel('Page Render Time')

plt.ylabel('Purcahse Amount')

plt.grid()

plt.scatter(pageRenderTime, purchaseAmount)


print('covar=%f' % covariance(pageRenderTime, purchaseAmount))

print('correl=%f' % correlation(pageRenderTime, purchaseAmount))

plt.show()
```



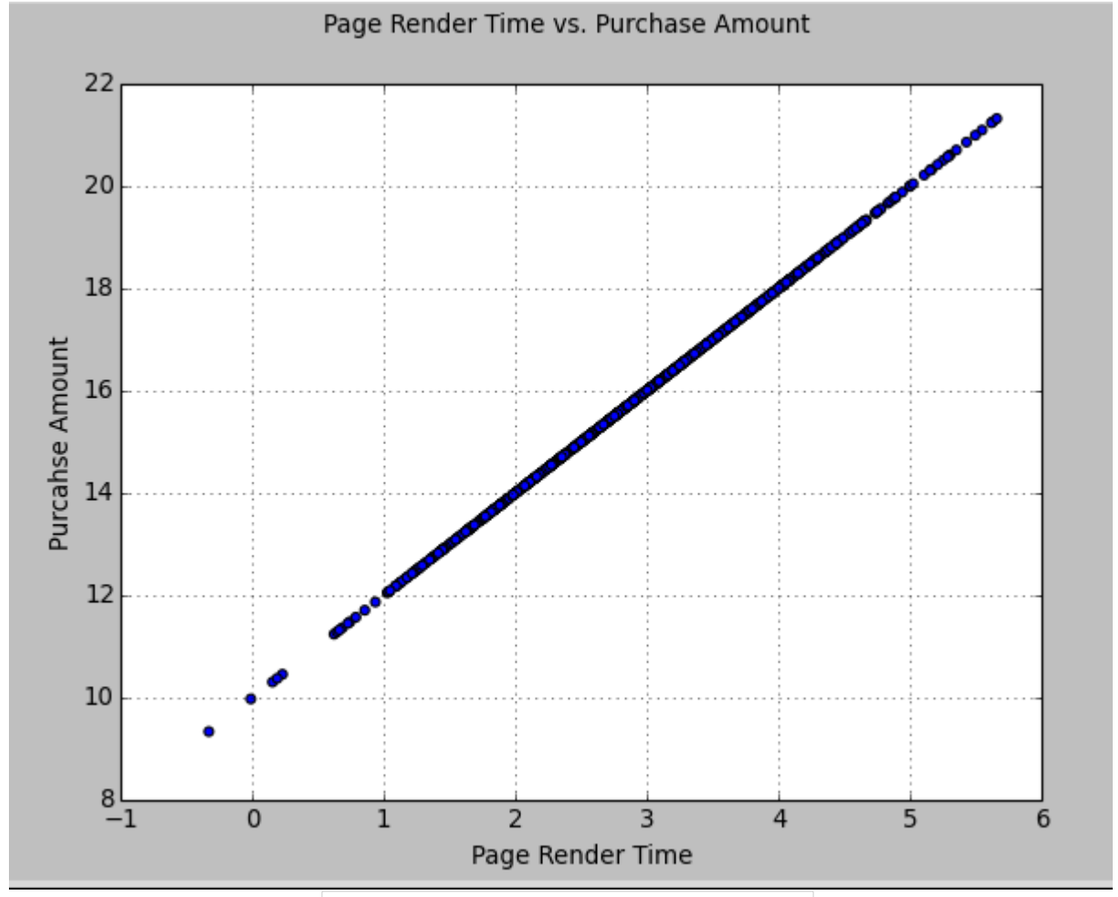
covar=-2.015354

correl=-1.001001

source in covar_correl_04.py

Simulated Solution: Perfect Direct Correlation

```
## an example of perfect direct correlation

pageRenderTime = np.random.normal(3.0, 1.0, 1000)

purchaseAmount = 10 + pageRenderTime * 2


fig1 = plt.figure(1)

fig1.suptitle('Page Render Time vs. Purchase Amount')

plt.xlabel('Page Render Time')

plt.ylabel('Purcahse Amount')

plt.grid()

plt.scatter(pageRenderTime, purchaseAmount)


print('covar=%f' % covariance(pageRenderTime, purchaseAmount))

print('correl=%f' % correlation(pageRenderTime, purchaseAmount))

plt.show()
```



covar=1.847916

correl=1.001001

source in covar_correl_05.py