

 This repository

[Pull requests](#) [Issues](#) [Gist](#)

 [quantopian](#) / [research\\_public](#)

 Watch ▾

95

 Star

181

 Fork

106

 Code

 Issues 0

 Pull requests 0

 Projects 0

 Wiki

 Pulse


 Graphs

Branch: master ▾

[research\\_public](#) / [lectures](#) / [Variances.ipynb](#)

Find file

Copy path

 **CaptainKanuk** ENH: Added 3 new lectures. a61beaf on Jul 21


1 contributor


316 lines (316 sloc) 10.4 KB


Raw

Blame

History







# Measures of Dispersion

By Evgenia "Jenny" Nitishinskaya, Maxwell Margenot, and Delaney Granizo-Mackenzie.

Part of the Quantopian Lecture Series:

- [www.quantopian.com/lectures](https://www.quantopian.com/lectures) (<https://www.quantopian.com/lectures>)
- [github.com/quantopian/research\\_public](https://github.com/quantopian/research_public) ([https://github.com/quantopian/research\\_public](https://github.com/quantopian/research_public))

Notebook released under the Creative Commons Attribution 4.0 License.

Dispersion measures how spread out a set of data is. This is especially important in finance because one of the main ways risk is measured is in how spread out returns have been historically. If returns have been very tight around a central value, then we have less reason to worry. If returns have been all over the place, that is risky.

Data with low dispersion is heavily clustered around the mean, while high dispersion indicates many very large and very small values.

Let's generate an array of random integers to work with.

```
In [1]: # Import Libraries
import numpy as np

np.random.seed(121)
```

```
In [2]: # Generate 20 random integers < 100
X = np.random.randint(100, size=20)

# Sort them
X = np.sort(X)
print 'X: %s' %(X)

mu = np.mean(X)
print 'Mean of X:', mu

X: [ 3  8 34 39 46 52 52 52 54 57 60 65 66 75 83 85 88 94 95 96]
Mean of X: 60.2
```

## Range

Range is simply the difference between the maximum and minimum values in a dataset. Not surprisingly, it is very sensitive to outliers. We'll use numpy's peak to peak (ptp) function for this.

```
In [3]: print 'Range of X: %s' %(np.ptp(X))

Range of X: 93
```

## Mean Absolute Deviation (MAD)

The mean absolute deviation is the average of the distances of observations from the arithmetic mean. We use the absolute value of the deviation, so that 5 above the mean and 5 below the mean both contribute 5, because otherwise the deviations always sum to 0.

$$MAD = \frac{\sum_{i=1}^n |X_i - \mu|}{n}$$

where  $n$  is the number of observations and  $\mu$  is their mean.

```
In [4]: abs_dispersion = [np.abs(mu - x) for x in X]
MAD = np.sum(abs_dispersion)/len(abs_dispersion)
print 'Mean absolute deviation of X:', MAD

Mean absolute deviation of X: 20.52
```

## Variance and standard deviation

The variance  $\sigma^2$  is defined as the average of the squared deviations around the mean:

$$\sigma^2 = \frac{\sum_{i=1}^n (X_i - \mu)^2}{n}$$

This is sometimes more convenient than the mean absolute deviation because absolute value is not differentiable, while squaring is smooth, and some optimization algorithms rely on differentiability.

Standard deviation is defined as the square root of the variance,  $\sigma$ , and it is the easier of the two to interpret because it is in the same units as the observations.

```
In [5]: print 'Variance of X:', np.var(X)
        print 'Standard deviation of X:', np.std(X)

Variance of X: 670.16
Standard deviation of X: 25.8874486962
```

One way to interpret standard deviation is by referring to Chebyshev's inequality. This tells us that the proportion of samples within  $k$  standard deviations (that is, within a distance of  $k \cdot$  standard deviation) of the mean is at least  $1 - 1/k^2$  for all  $k > 1$ .

Let's check that this is true for our data set.

```
In [6]: k = 1.25
        dist = k*np.std(X)
        l = [x for x in X if abs(x - mu) <= dist]
        print 'Observations within', k, 'stds of mean:', l
        print 'Confirming that', float(len(l))/len(X), '>', 1 - 1/k**2

Observations within 1.25 stds of mean: [34, 39, 46, 52, 52, 52, 54, 57, 60, 65, 66, 75, 83, 85, 88]
Confirming that 0.75 > 0.36
```

The bound given by Chebyshev's inequality seems fairly loose in this case. This bound is rarely strict, but it is useful because it holds for all data sets and distributions.

## Semivariance and semideviation

Although variance and standard deviation tell us how volatile a quantity is, they do not differentiate between deviations upward and deviations downward. Often, such as in the case of returns on an asset, we are more worried about deviations downward. This is addressed by semivariance and semideviation, which only count the observations that fall below the mean. Semivariance is defined as

$$\frac{\sum_{X_i < \mu} (X_i - \mu)^2}{n_{<}}$$

where  $n_{<}$  is the number of observations which are smaller than the mean. Semideviation is the square root of the semivariance.

```
In [7]: # Because there is no built-in semideviation, we'll compute it ourselves
        lows = [e for e in X if e <= mu]

        semivar = np.sum( (lows - mu) ** 2 ) / len(lows)

        print 'Semivariance of X:', semivar
        print 'Semideviation of X:', np.sqrt(semivar)

Semivariance of X: 689.512727273
Semideviation of X: 26.2585743572
```

A related notion is target semivariance (and target semideviation), where we average the distance from a target of values which fall below that target:

$$\frac{\sum_{X_i < B} (X_i - B)^2}{n_{<B}}$$

```
In [8]: B = 19
        lows_B = [e for e in X if e <= B]
        semivar_B = sum(map(lambda x: (x - B)**2, lows_B))/len(lows_B)

        print 'Target semivariance of X:', semivar_B
        print 'Target semideviation of X:', np.sqrt(semivar_B)

Target semivariance of X: 188
Target semideviation of X: 13.7113092008
```

## These are Only Estimates

All of these computations will give you sample statistics, that is standard deviation of a sample of data. Whether or not this reflects the current true population standard deviation is not always obvious, and more effort has to be put into determining that. This is especially problematic in finance because all data are time series and the mean and variance may change over time. There are many different techniques and subtleties here, some of which are address in other lectures in the [Quantopian Lecture Series \(https://www.quantopian.com/lectures\)](https://www.quantopian.com/lectures).

In general do not assume that because something is true of your sample, it will remain true going forward

in general do not assume that because something is true of your sample, it will remain true going forward.

*This presentation is for informational purposes only and does not constitute an offer to sell, a solicitation to buy, or a recommendation for any security; nor does it constitute an offer to provide investment advisory or other services by Quantopian, Inc. ("Quantopian"). Nothing contained herein constitutes investment advice or offers any opinion with respect to the suitability of any security, and any views expressed herein should not be taken as advice to buy, sell, or hold any security or as an endorsement of any security or company. In preparing the information contained*

