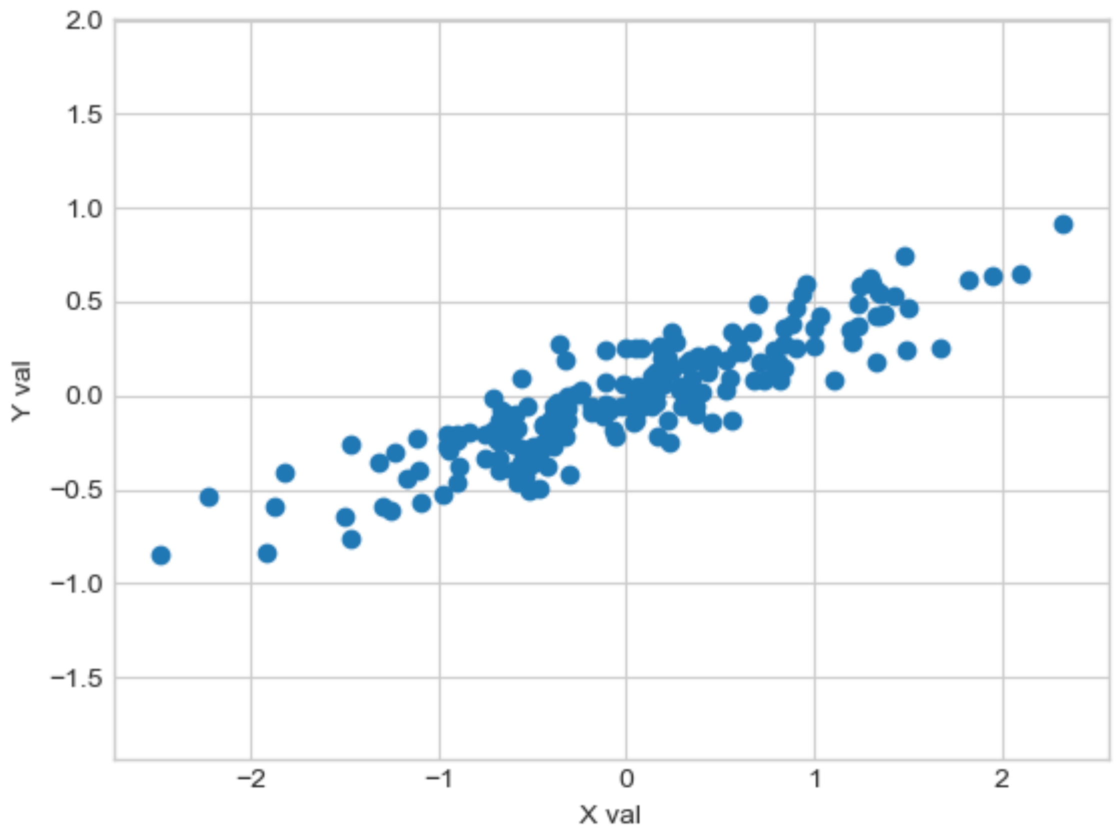
Python for Scientific Data Analysis

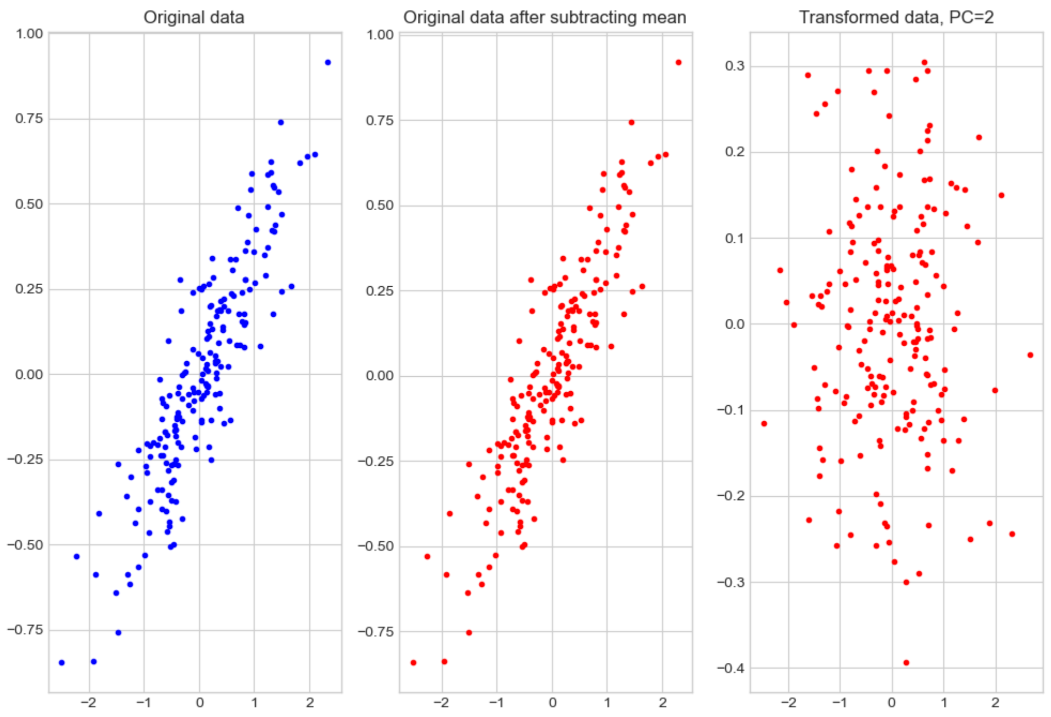
Homework - Week 5

1. PCA (in class)

- Take the data shown in our first plot of the PCA lecture notes:
  `rng = np.random.RandomState(1) X = np.dot(rng.rand(2, 2), rng.randn(2, 200)).T`



Perform PCA on these data to produce the following plot:



Use the source code in `pcademo3.py` for plotting and guidance.

Hints

- look at the notes and the source code for `pcademo3.py`. In particular, look at where the PCA steps start (denoted by "Step 1"). What variable do you need to drop in here and then mean-subtract?
- There should be a block of code that produces the plot. You can figure it out if you know what each variable stands for (i.e. if you can read the source code). If not, putting in some commment lines to figure out where it is.
- To make the appearance of the plot match what we want, you need to set the x and y limits properly. The block of code that will do this is:

```
# Set x ticks
xl=ax[0].get_xlim()
yl=ax[0].get_ylim()
ax[1].set_xlim(xl)
ax[1].set_ylim(yl)
ax[2].set_xlim(-2.75,2.75)
ax[2].set_ylim(-0.425,0.325)
```

2. PCA and SVD (in class)

Part 1

- Create an array of random numbers with dimensions (7,3) ...

- perform PCA on this array (remember all of the steps to PCA).

To produce a (3,3) covariance matrix, I recommend you do `np.matmul(array.T,array)` where "array" is your mean-subtracted array.

- starting with the *mean_subtracted* version of the above array, perform SVD
- Compare the eigenvectors computed from PCA to the matrices computed from SVD. Are there any similarities?
- Compare the eigenvalues computed from PCA to the singular values computed from SVD. Are there any similarities or correlations?

Part 2 - Repeat part 1 except for a random (9,3) array.

What conclusions can you draw about PCA and eigendecomposition vs SVD from these results?

3. A Simple PCA Calculation (in class)

Start with the following array:

```
data=np.array([[7., 4., 3.],
               [4., 1., 8.],
               [6., 3., 5.],
               [8., 6., 1.],
               [8., 5., 7.],
               [7., 2., 9.],
               [5., 3., 3.],
               [9., 5., 8.],
               [7., 4., 5.],
               [8., 2., 2.]])
```

Create a Python script (a file that will end with ".py") containing a single Python **function** (you know, the things that start with "def") that performs PCA (steps 1--5), including the number of principal components as a keyword variable.

Demonstrate how to execute the function. Run it where you retain i) 3 principal components and ii) 5 principal components. Include a print statement within the function to print out the results.

(**Note**: don't overthink this question. I'm just asking you to write code to do a simple PCA calculation. It's really just making sure you remember how to do PCA -- which you should since we are just covering it -- *and* write Python functions, which we covered during the first week of class.)