Principal Component Analysis

Concept Module 7

Template by: Laurent Lessard

What is PCA?

Principal Component Analysis: (PCA) is a method for compressing (reducing the dimension) of a dataset while preserving parts of the data with the most variability.

- Simplify storage and representation
- Simplify visualization
- Expose structure in the data

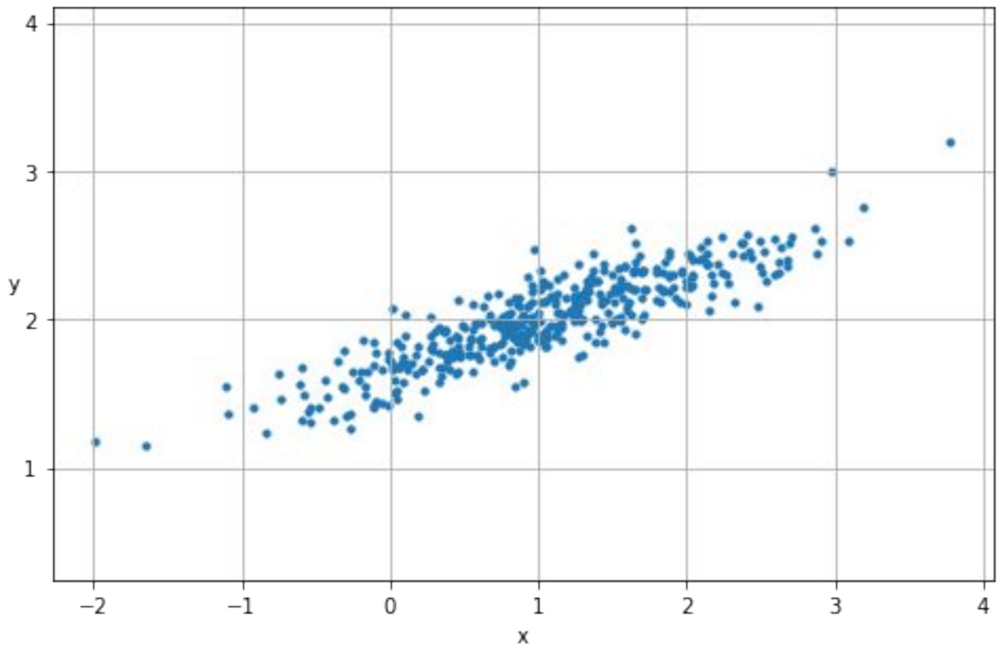
Example: 2D data

Goal: Use one number (instead of two) to represent each point.

	x	y
0	1.044267	2.110966
1	1.440636	2.372623
2	1.056300	1.872256
3	0.102490	2.025878
4	0.184354	1.352462

$\xrightarrow{???}$

	z
0	0.024377
1	0.169453
2	0.004115
3	-0.266665
4	-0.309448



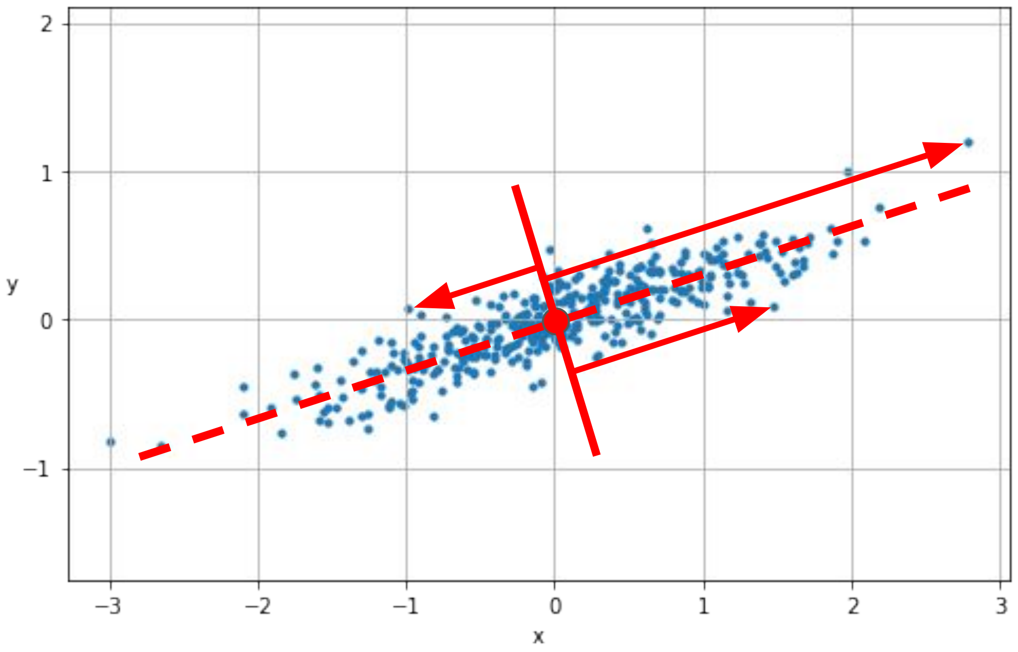
Source: https://jakevdp.github.io/PythonDataScienceHandbook/05.09-principal-component-analysis.html

Example: 2D data

1. Center the data by subtracting the mean from all points.

2. Find direction where data changes most.

3. Measure distance along that direction.

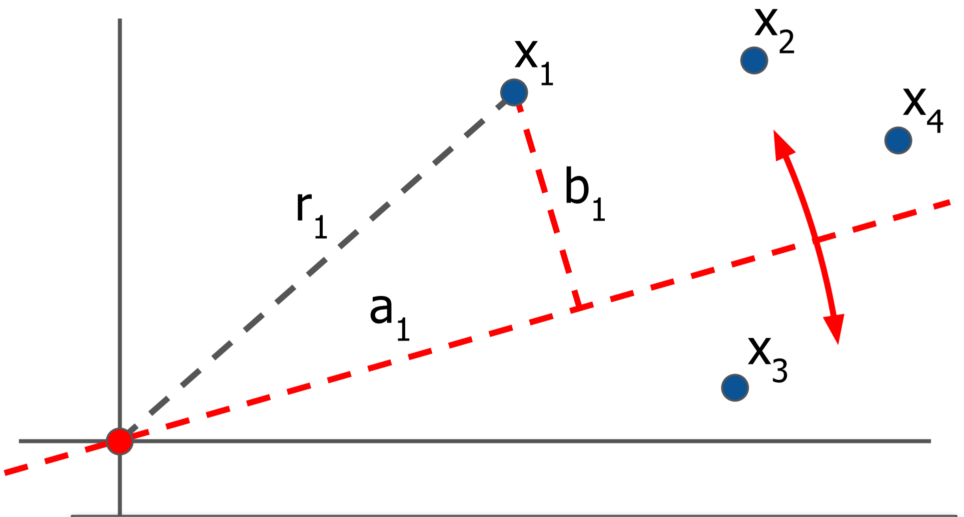


Source: https://jakevdp.github.io/PythonDataScienceHandbook/05.09-principal-component-analysis.html

Geometry of PCA

Choosing the Principal Component (PC1)

- Maximize $(a_1)^2 + \dots + (a_n)^2$
- Since points are centered, same as Maximizing std(a)
- Since $(a_k)^2 + (b_k)^2 = (r_k)^2$, this is the same as: **Minimizing** $(b_1)^2 + \dots + (b_n)^2$



$$s = \sqrt{\frac{(x[0]-m)**2 + \dots + (x[N-1]-m)**2}{N-1}}$$

How well does it do?

- We started off with points of the form (x_k,y_k).
- Pythagorean theorem: $(x_k)^2 + (y_k)^2 = (r_k)^2 = (a_k)^2 + (b_k)^2$
- Sum over k, divide by (N-1), obtain:

$$\underbrace{\mathrm{std}(x)^2 + \mathrm{std}(y)^2}_{\text{Total variance in the data}} = (sa)^2 + (sb)^2$$

Variance left over

Variance explained by PC1
(we maximized this!)

Example: 2D data

Mean: (1,2)

PC1: (3,1)

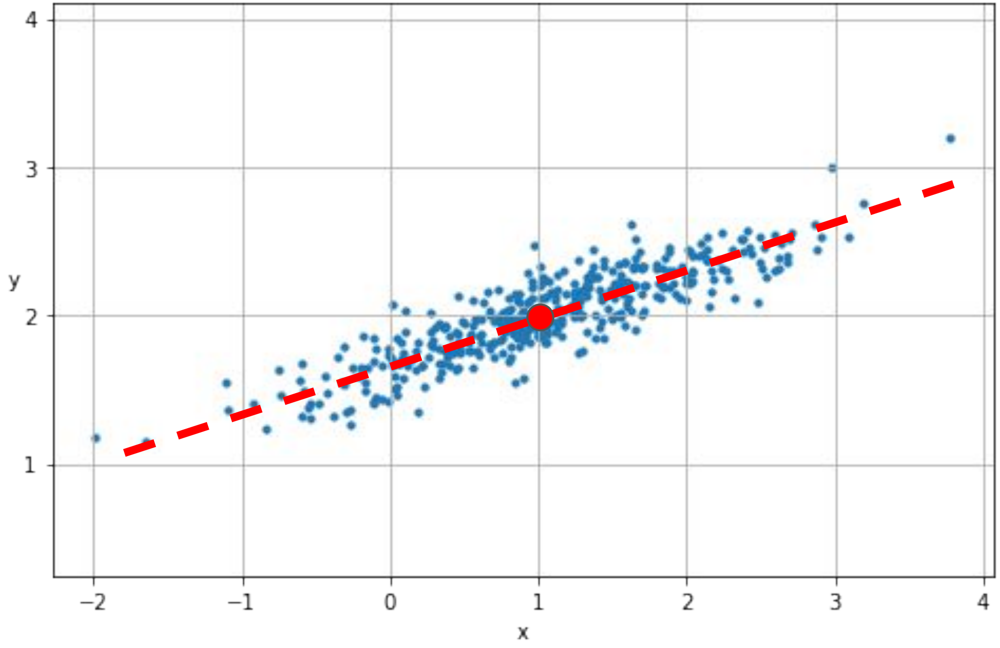
Any multiple works, e.g. (-6,-2) can be used instead

Data ≈ (1,2) + z(3,1)

Recipe!

	x	y
0	1.044267	2.110966
1	1.440636	2.372623
2	1.056300	1.872256
3	0.102490	2.025878
4	0.184354	1.352462

	z
0	0.024377
1	0.169453
2	0.004115
3	-0.266665
4	-0.309448



Source: https://jakevdp.github.io/PythonDataScienceHandbook/05.09-principal-component-analysis.html

Example: 2D data

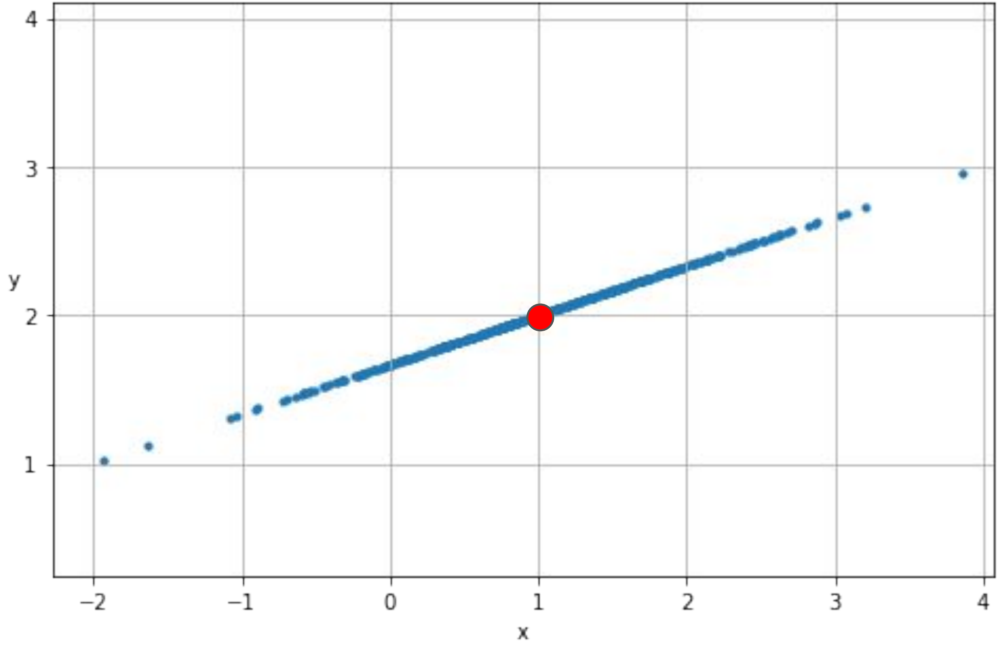
Recipe!

	x	y
0	1.044267	2.110966
1	1.440636	2.372623
2	1.056300	1.872256
3	0.102490	2.025878
4	0.184354	1.352462

	z
0	0.024377
1	0.169453
2	0.004115
3	-0.266665
4	-0.309448

	x	y
0	1.073130	2.024377
1	1.508359	2.169453
2	1.012346	2.004115
3	0.200004	1.733335
4	0.071657	1.690552

Inverse transform (shown on right) approximates the data



Source: https://jakevdp.github.io/PythonDataScienceHandbook/05.09-principal-component-analysis.html

Higher dimensions

- First principal component (PC1): direction of largest variation of the data (same as before)
- PC2: direction of largest variation once first PC1 has been removed. PC2 is always orthogonal (at right angles) to PC1
- PC3: and so on… PC3 will be orthogonal to PC1 and PC2.

$$\boxed{(\text{total variance}) = (s_{PC1})^2 + (s_{PC2})^2 + \dots}$$

PCA in Python

```
from sklearn.decomposition import PCA

pca = PCA(n_components = 2)
pca.fit(df)
```

Create PCA object. Specify the number of components to use.

Fit your data into the model. This performs PCA!

Principal components

```
# extract principal components
pca.components_

# how much variance is explained?
pca.explained_variance_
```

Transformed (compressed) data this is the recipe

```
# transform to reduced coordinates
# (find recipe in terms of PCs)
dfred = pca.transform(df)
```

	petalLength	petalWidth	sepalLength	sepalWidth
0	1.4	0.2	5.1	3.5
1	1.4	0.2	4.9	3.0
2	1.3	0.2	4.7	3.2
3	1.5	0.2	4.6	3.1
4	1.4	0.2	5.0	3.6
5	1.7	0.4	5.4	3.9
6	1.4	0.3	4.6	3.4
7	1.5	0.2	5.0	3.4

```
# reduced data (PC recipe)
pd.DataFrame(data=pca.transform(df),
             columns=['pc1','pc2'])
```

	pc1	pc2
0	-2.684126	0.319397
1	-2.714142	-0.177001
2	-2.888991	-0.144949
3	-2.745343	-0.318299
4	-2.728717	0.326755
5	-2.280860	0.741330
6	-2.820538	-0.089461
7	-2.626145	0.163385

```
# mean of the data
pd.DataFrame(index=['mean'],
             columns=df.columns,
             data=[pca.mean_])
```

	petalLength	petalWidth	sepalLength	sepalWidth
mean	3.758	1.199333	5.843333	3.057333

```
# principal components
pd.DataFrame(index=['pc1','pc2'],
             columns=df.columns,
             data=pca.components_)
```

	petalLength	petalWidth	sepalLength	sepalWidth
pc1	0.856671	0.358289	0.361387	-0.084523
pc2	-0.173373	-0.075481	0.656589	0.730161

	petalLength	petalWidth	sepalLength	sepalWidth
0	1.4	0.2	5.1	3.5
1	1.4	0.2	4.9	3.0
2	1.3	0.2	4.7	3.2
3	1.5	0.2	4.6	3.1
4	1.4	0.2	5.0	3.6
5	1.7	0.4	5.4	3.9
6	1.4	0.3	4.6	3.4
7	1.5	0.2	5.0	3.4

```
# reduced data (PC recipe)
pd.DataFrame(data=pca.transform(df),
             columns=['pc1','pc2'])
```

	pc1	pc2
0	-2.684126	0.319397
1	-2.714142	-0.177001
2	-2.888991	-0.144949
3	-2.745343	-0.318299
4	-2.728717	0.326755
5	-2.280860	0.741330
6	-2.820538	-0.089461
7	-2.626145	0.163385

Principal Components (PC1 and PC2)

	petalLength	petalWidth	sepalLength	sepalWidth
pc1	0.856671	0.358289	0.361387	-0.084523
pc2	-0.173373	-0.075481	0.656589	0.730161

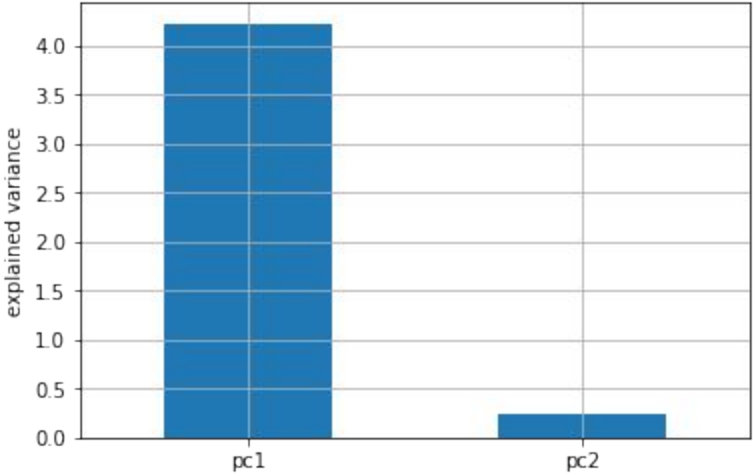
```
# inverse transform (back to original coords)
# NOTE: dft is the transformed data
pd.DataFrame(data=pca.inverse_transform(dft),
             columns=df.columns)
```

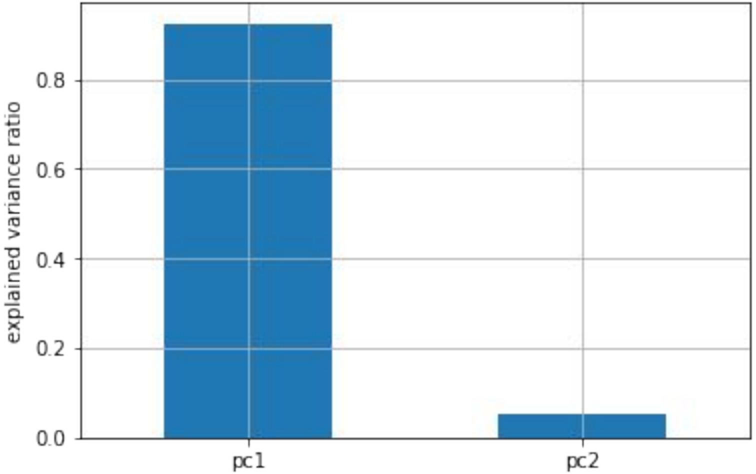
	petalLength	petalWidth	sepalLength	sepalWidth
0	1.403214	0.213532	5.083039	3.517414
1	1.463562	0.240246	4.746262	3.157500
2	1.308217	0.175180	4.704119	3.195682
3	1.461330	0.239732	4.642212	3.056967
4	1.363738	0.197000	5.071755	3.526555
5	1.675528	0.326170	5.505810	3.791408
6	1.357238	0.195518	4.765289	3.230411
7	1.479932	0.246081	5.001556	3.398599

Variance explained (scree plot)

```
dfvar = pd.DataFrame( data=pca.explained_variance_,index=['pc1','pc2'] )
dfvar.plot.bar(grid=True,legend=False,rot=0).set_ylabel('explained variance')
```

```
dfvar = pd.DataFrame( data=pca.explained_variance_ratio_,index=['pc1','pc2'] )
dfvar.plot.bar(grid=True,legend=False,rot=0).set_ylabel('explained variance ratio')
```





Variance explained

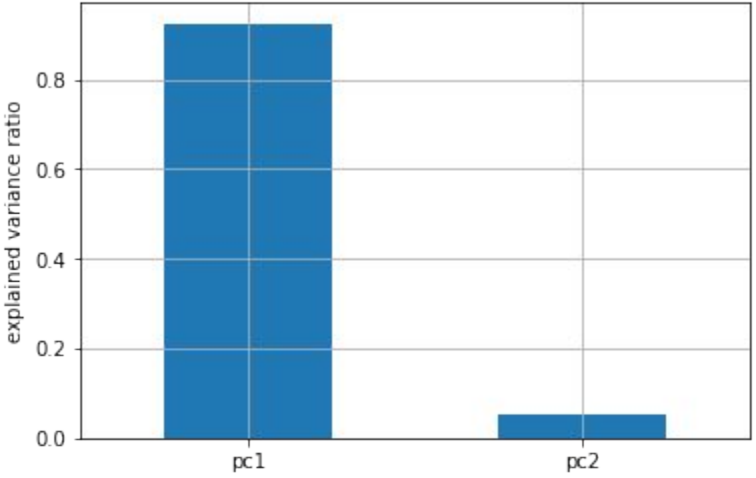
- PC1 and PC2 account for 97.8% of the variance explained!
- A very good 2D approximation to this 4D dataset.

```
pca.explained_variance_ratio_
```

```
array([ 0.92461872,  0.05306648])
```

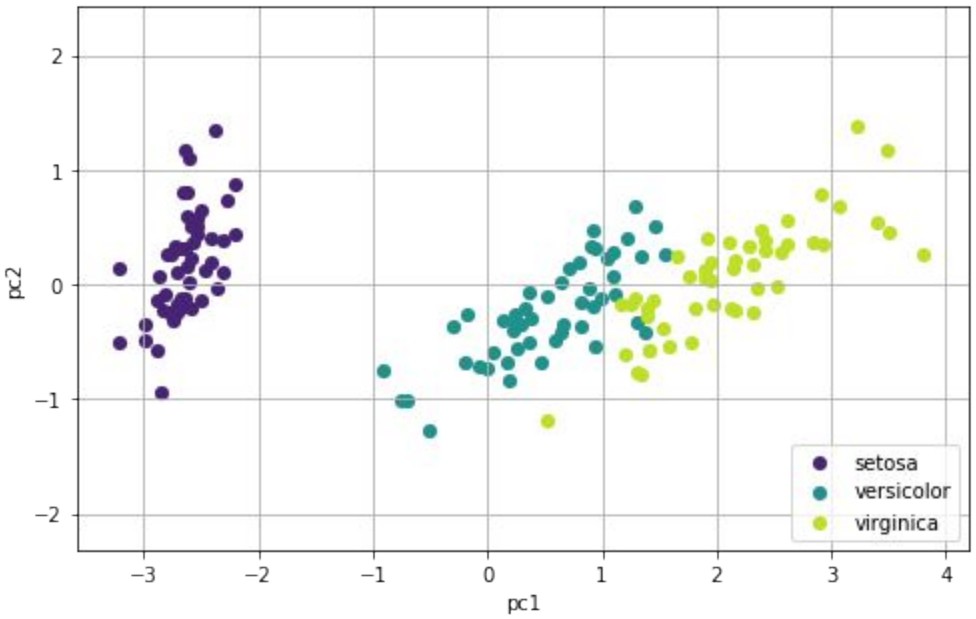
```
pca.explained_variance_ratio_.sum()
```

```
0.97768520631879496
```

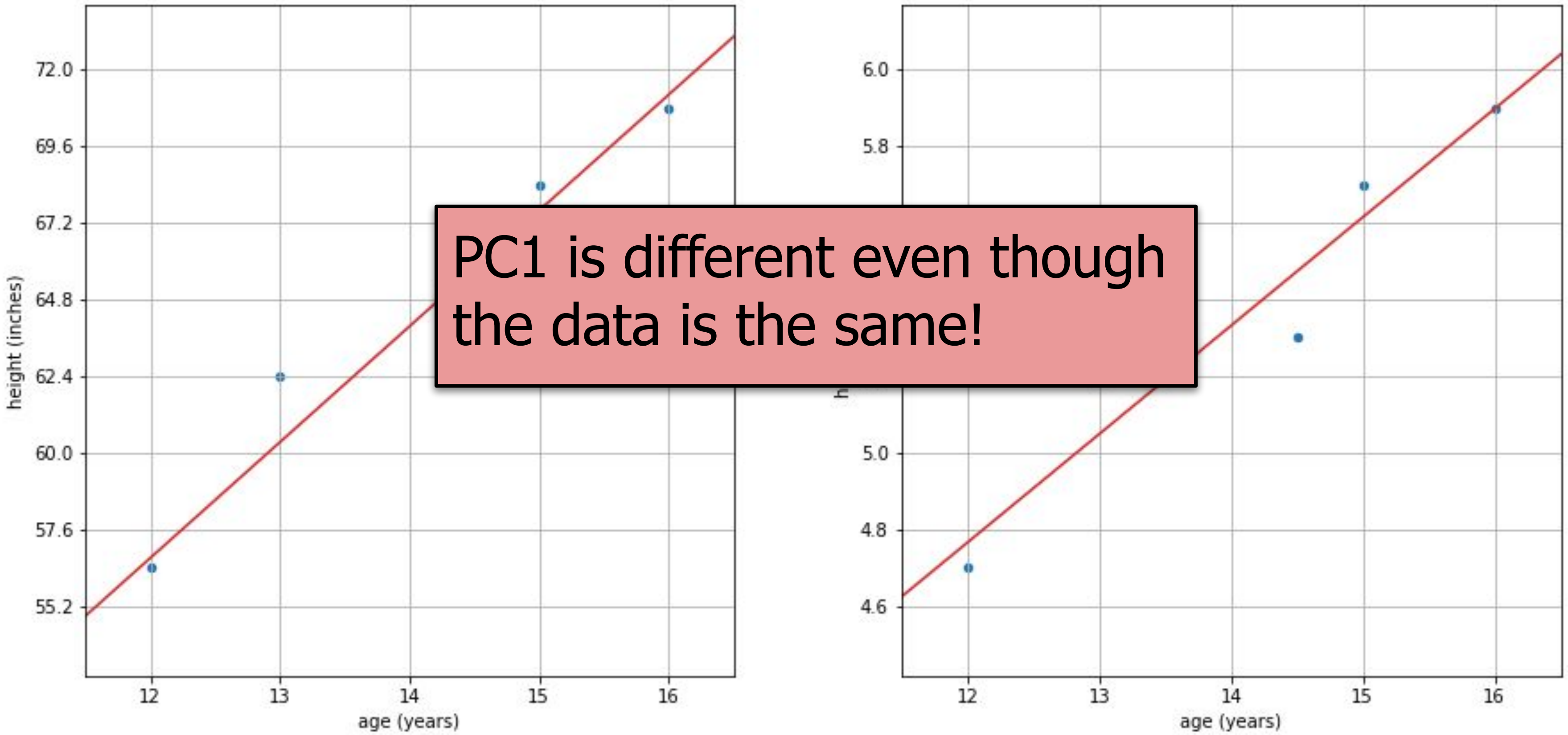


PC plot

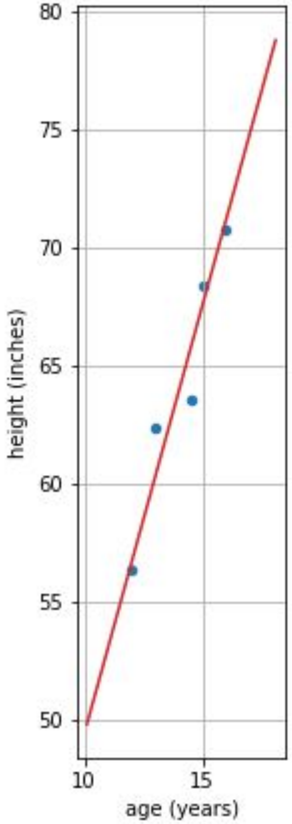
- Scatter plot of PC1 vs PC2 (with labels)
- Can also be drawn in 3D (including PC3)



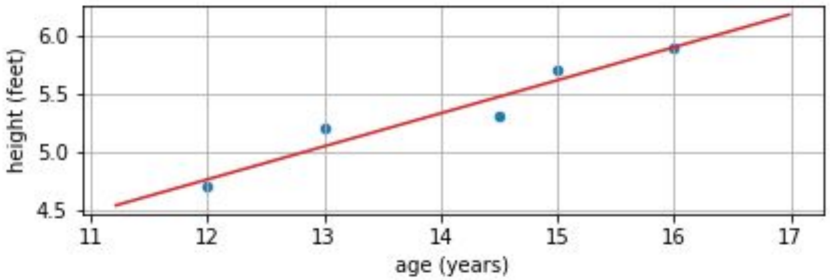
WARNING: Beware of the scale!



WARNING: Beware of the scale!



- Perpendicular distances change if data is stretched. Results depend on scale!
- Often useful to "normalize" data to a common scale.



Summary

- PCA finds the directions with the most variation in the data. These are called Principal Components (PC).
- Total variance in the data is the sum of contributions from each PC. Can use a scree plot to compare them.
- If the first couple PCs account for a significant proportion of the total variance, data is "essentially" low-dimensional.
- PCA rotates your frame of reference so the most "interesting" (highly variable) dimensions come first!
- The PCs can change depending on how your data is scaled.