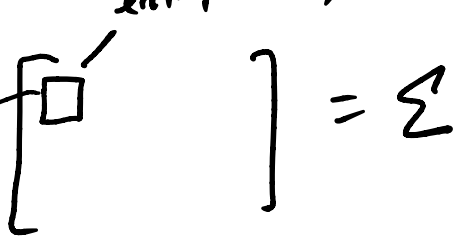
- How to compute covariance matrix for any dataset?

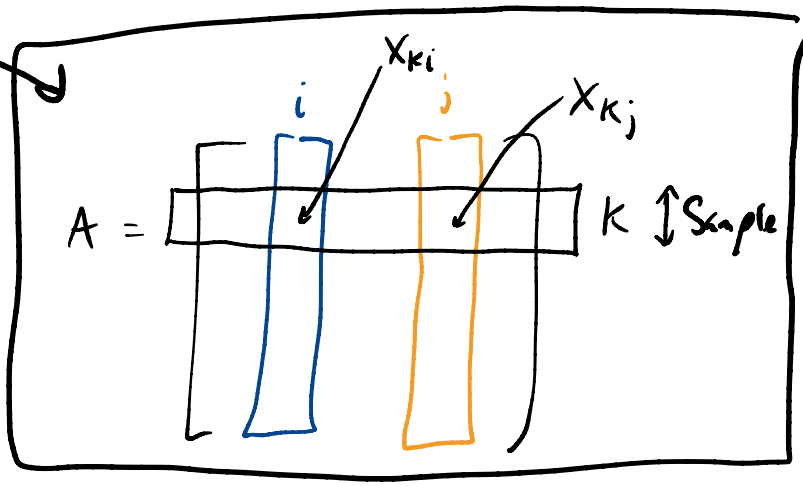
Scalar equation



Not that useful: Slow if implemented

$$\Sigma(i,j) = \sum_{k=1}^{N} \frac{(x_{ki} - \mu_i)(x_{kj} - \mu_j)}{N-1}$$

$X: (N, M)$



diagonals:

$$\Sigma(i,i) = \sum_{k=1}^{N} \frac{(x_{ki} - \mu_i)(x_{ki} - \mu_i)}{N-1} = \frac{(x_{ki} - \mu_i)^2}{N-1}$$

$= \sigma_i^2 \Rightarrow$ diagonal entries are ordinary variances

2D example: $\Sigma = \begin{bmatrix} \text{cov}(x,x) & \text{cov}(x,y) \\ \text{cov}(y,x) & \text{cov}(y,y) \end{bmatrix}$

$\text{cov}(x,x) = \sigma_x^2$

$\text{cov}(y,y) = \sigma_y^2$

$\Sigma(x,y)$

vectorized equation:

$$\Sigma = \frac{1}{N-1} \sum_{K=1}^{N} (\vec{x}_K - \vec{\mu})^T (\vec{x}_K - \vec{\mu})$$

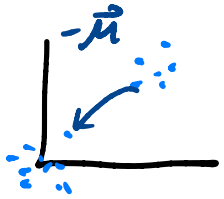
shape = (M, 1)

e.g. M = 3

$\vec{\mu} = (1, 2, 3)$

shape: (1, M) shape: (M, 1)

Centers data so that $\vec{x}_K$ has $\vec{0}$ mean:



Call Centered data matrix A_c : A_c shape = (N, M)

$$A_c = A - \vec{\mu}$$

$$A \begin{bmatrix} 1 & 2 & 3 \\ | & | & | \\ | & | & | \\ | & | & | \end{bmatrix} - [\mu_1, \mu_2, \mu_3]$$

$$A \begin{bmatrix} 1 & 2 & 3 \\ 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix} - [2, 5, 8]$$

$$= A_c \begin{bmatrix} | & | & | \\ | & | & | \\ | & | & | \end{bmatrix} = A_c \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

if data centered:

$$\Sigma = \frac{1}{N-1} A_c.T \,@\, A_c$$

[matrix equation]

$\vec{x}_i$

Example:

$$A = \begin{bmatrix} 1 & 2 & -1 \\ 10 & 2 & 1 \\ 3 & 4 & 5 \\ 3 & 2 & 1 \end{bmatrix}$$

$$- (\mu_1, \mu_2, \mu_3)$$

$$= A_c = \begin{bmatrix} -3.25 & -1/2 & -2.5 \\ 5.75 & -1/2 & -0.5 \\ -1.25 & 1.5 & 3.5 \\ -1.25 & -1/2 & -0.5 \end{bmatrix}$$

$$\vec{\mu} = \left(\frac{1+10+3+3}{4}, \frac{2+2+4+2}{4}, \frac{-1+1+5+1}{4} \right)$$

$$= (4.25, 2.5, 1.5)$$

$(3,4)$ $(4,3)$

$$\frac{1}{N-1} A_c^T A \longrightarrow (3,3)$$

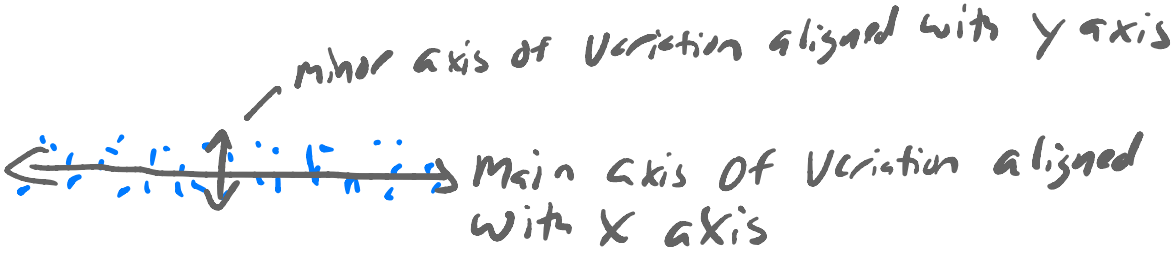
$$\frac{1}{3} \begin{bmatrix} -3.25 & 5.75 & -1.25 & -1.25 \\ -1/2 & -1/2 & 1.5 & -1/2 \\ -2.5 & -0.5 & 3.5 & -0.5 \end{bmatrix} \begin{bmatrix} -3.25 & -1/2 & -2.5 \\ 5.75 & -1/2 & -0.5 \\ -1.25 & 1.5 & 3.5 \\ -1.25 & -1/2 & -0.5 \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} 15.8 & -0.83 & 0.5 \\ -0.83 & 1 & 2.3 \\ 0.5 & 2.3 & 6.3 \end{bmatrix}$$

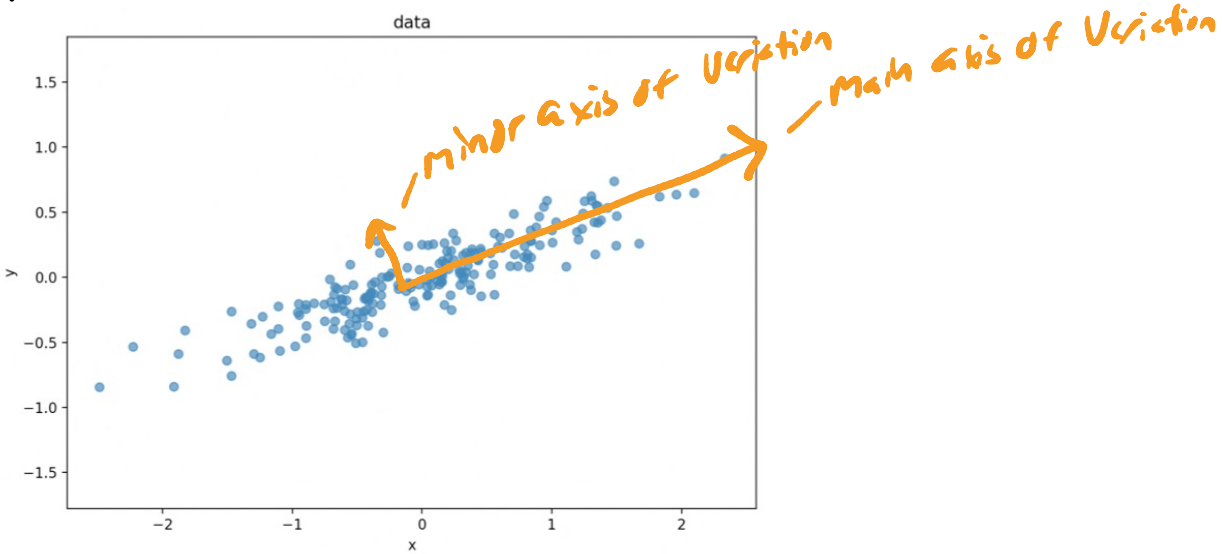
Principal Component Analysis (PCA)

Recall the example where we could toss out a variable (y) without losing much information:

In this case, the main axes of variation of data are aligned with the coordinate axes



⇒ Clearly this is not true in general! Axes of variation do not need to correspond to variables in dataset — e.g. x, y, etc:



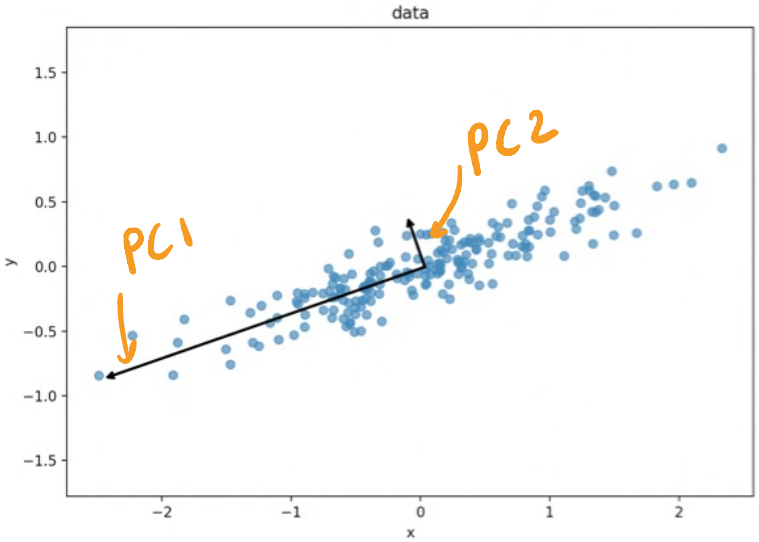
Above Main Variation axis != x axis!
Minor Variation axis != y axis!

We call these intrinsic axes of Variation in data

principal components (PCs)

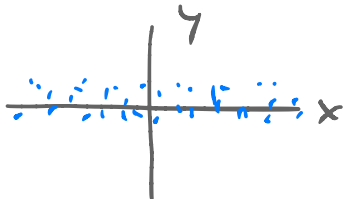
⇒ we number them in order based on the amount of Variation along each intrinsic axis

⇒ **larger Variation → smaller number.**



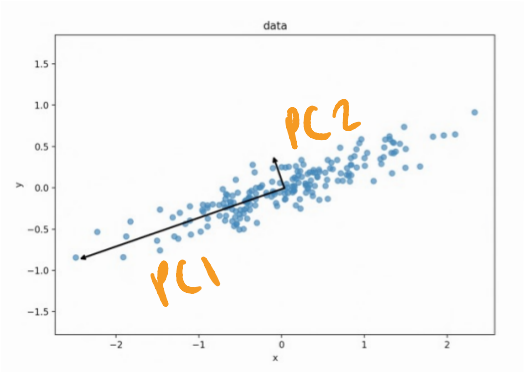
Goal of PCA Reduce dimensionality of dataset (e.g. M = 500 → 30) by dropping **intrinsic axes** with little Variation (least amount of info lost)

e.g. PC2 above, y Variable M

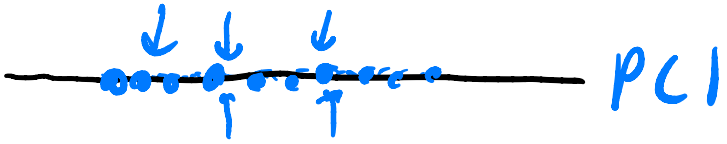


⚠ We do not want to drop data variables

e.g. ⊘ example drop PC2.



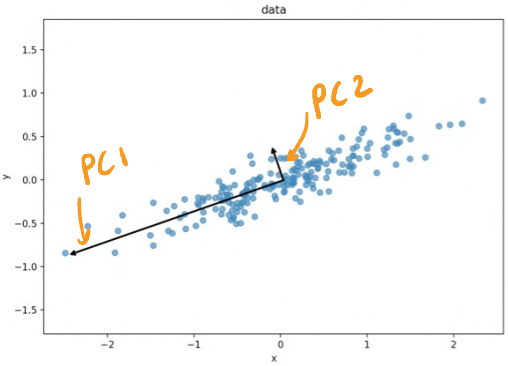
→



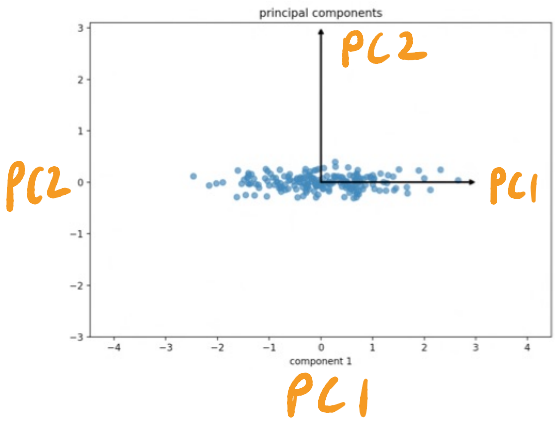
How do we "access" the PCs? — they can be oriented in any direction (but they are assumed to be orthogonal to each other).

⇒ Rotate the data so that the PCs become coordinate axes:

New coordinate axes: The axes of primary variation in the data



Rotate →



How is this done? A translation then a rotation matrix applied:

See plots above both centered at (0,0)

Rotated, centered data

rotation matrix

centered data matrix:

$A_c = A - \vec{\mu}$ ← translation

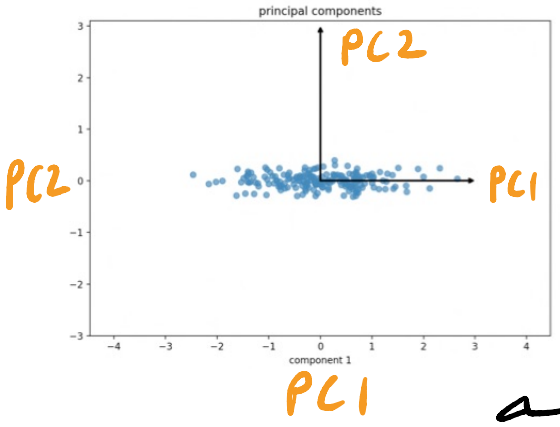
$$\hat{A}_c = (P.T \mathbin{@} A_c).T$$

Same thing

or more concisely:

$$\hat{A}_c = A_c \mathbin{@} P$$

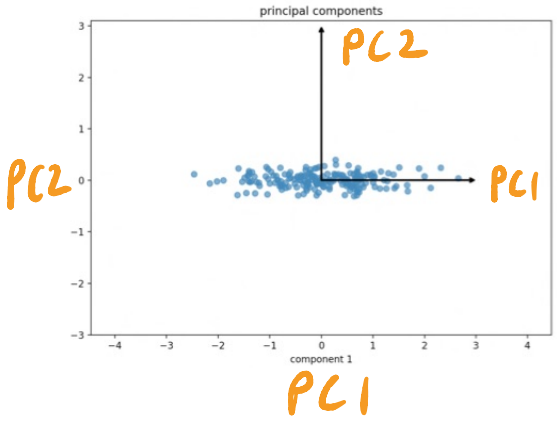
$\hat{A}_c$



$\hat{A}_c$ is centered data in PCA Space: coordinate axes are PC1 & PC2, not X, Y

... and to drop the insignificant PCs, we can project onto the large meaningful ones (e.g. PC1)

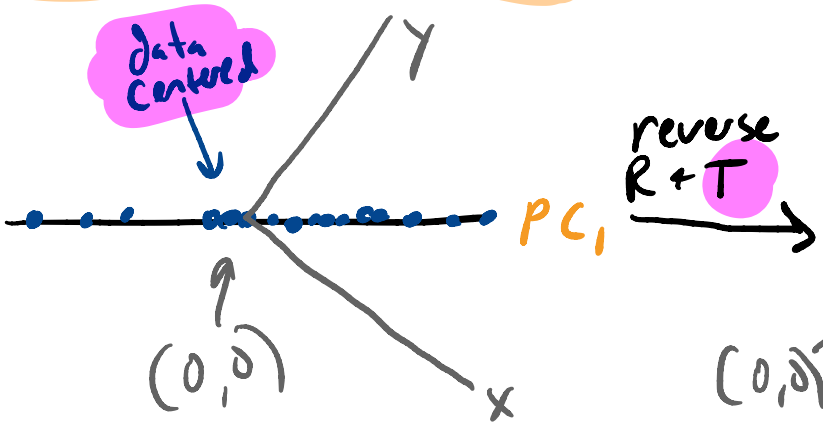
Remember: project = index vers — e.g. $\hat{A}_c[:, 0]$ (PC1 in index 0)



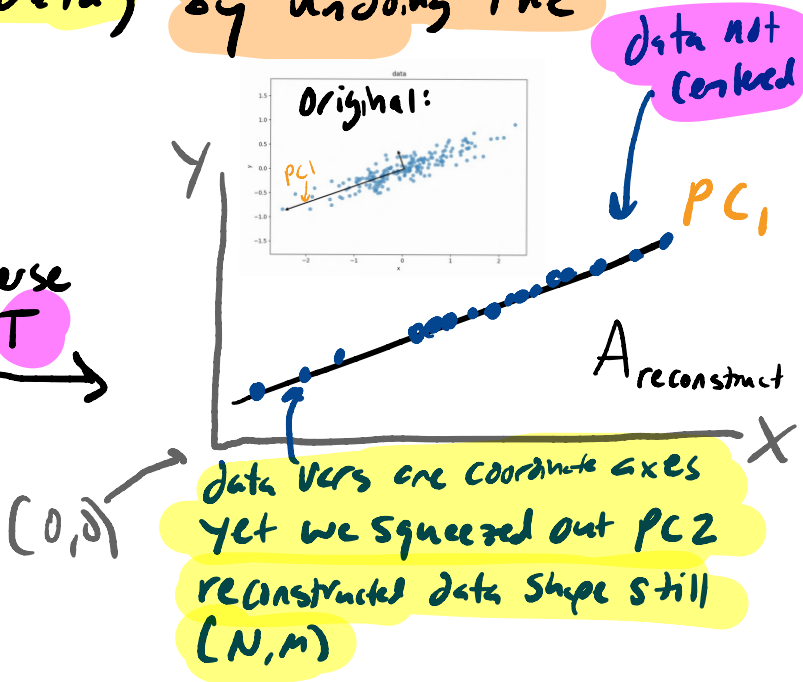
$\hat{A}_c[:, 0]$ →

This achieves what we wanted, but the coordinate axes are PCs — not data variables, which might not what we want.

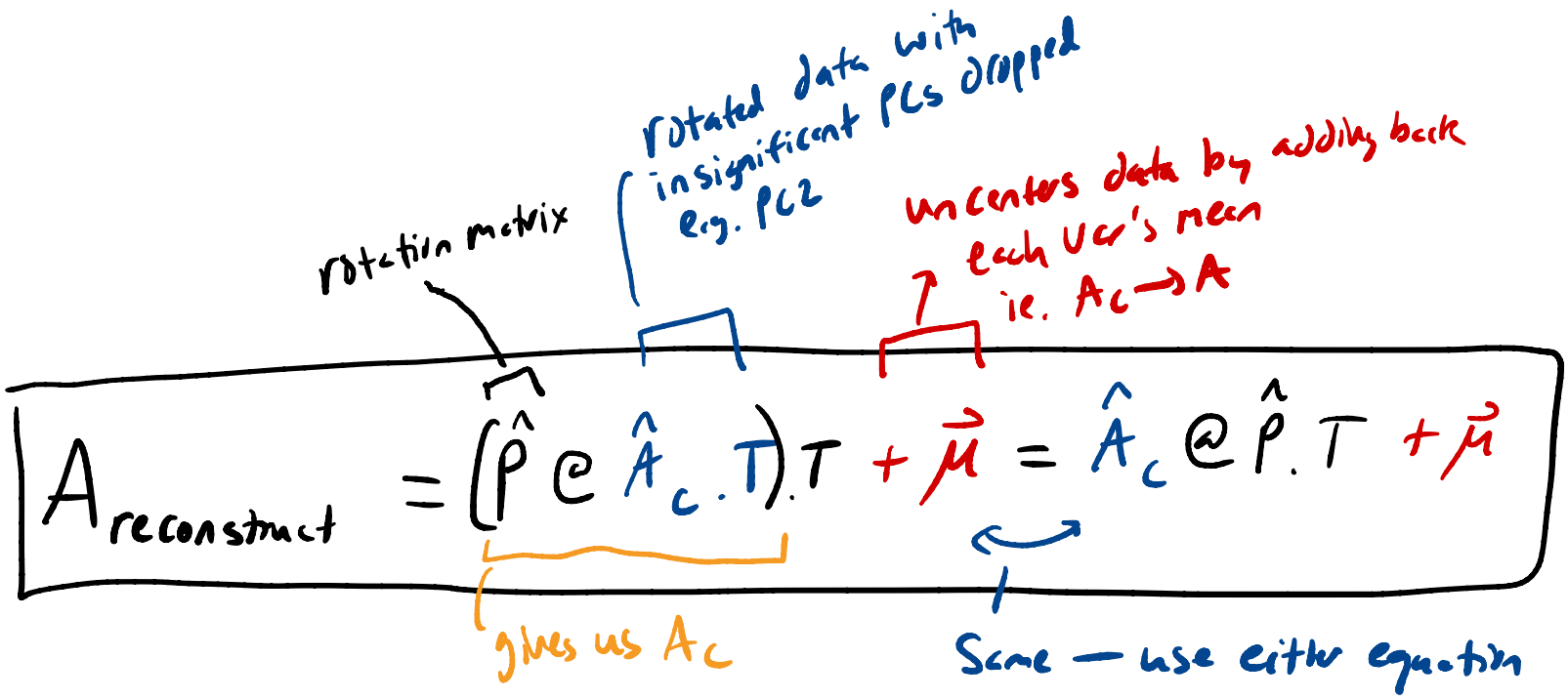
We can express this result in terms of the original data variables (reconstruct the data) by undoing the rotation + translation!



reverse R+T →



To reverse R+T and reconstruct data from $\hat{A}_c$:



What is the rotation matrix P?

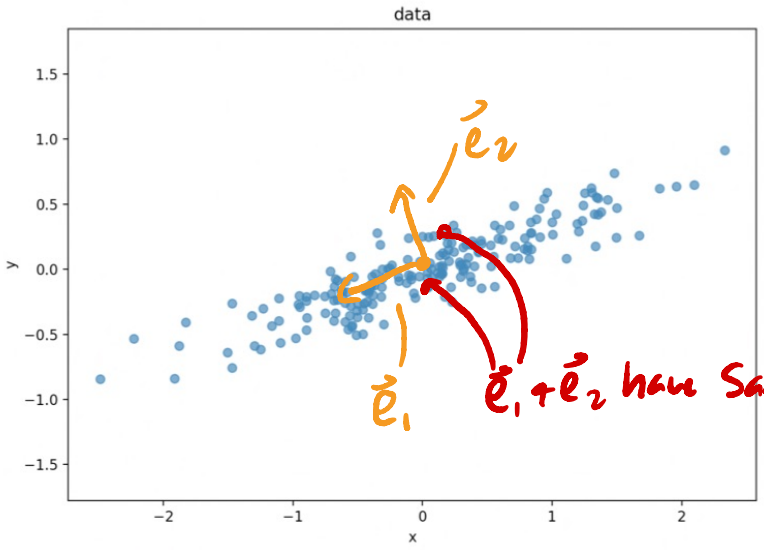
P is the **eigenvectors** of the **covariance matrix** Σ of the data.

Recall: $\Sigma = \frac{1}{N-1} A_c.T @ A_c$

Shape: (M, M)

M of them — one per data variable

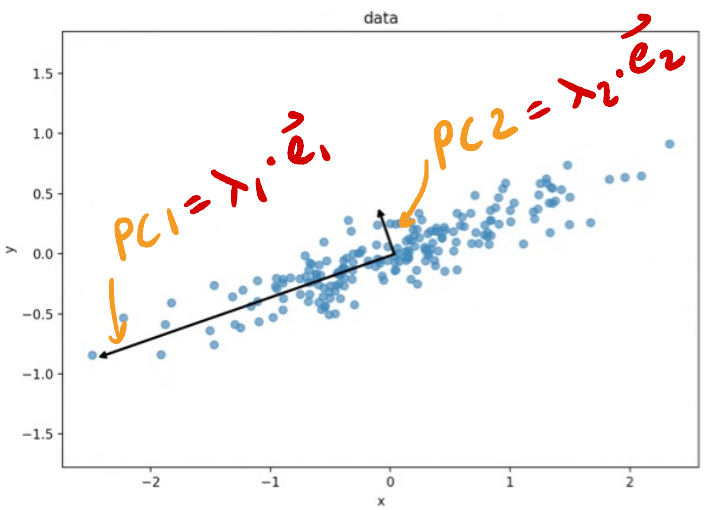
Eigen vectors $P = [\vec{e}_1, \vec{e}_2, \ldots, \vec{e}_m]$ tell us the **direction** but not amount of the corresponding PCs.



★ Rotating data by P aligns coordinate axes with PCs

How do we know the amount of variation in each PC direction? **Eigen values** $= [\lambda_1, \lambda_2, \ldots, \lambda_m]$

M scalars/floats
One per PC/eigenvector/data variable.



← λ_1 is much bigger than λ_2!