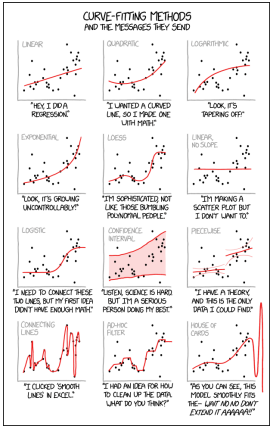
# PCA: Seeing Through the Noise

## A Simple Guide to Principal Component Analysis

Methods and Algorithms

Spring 2026

# When You Have Too Many Dimensions to Plot




- Your dataset has 50 columns
- You can only plot 2 axes
- What do you do?

## Today's Question

How do we compress many features into a few while keeping the important patterns?

**XKCD #2048 by Randall Munroe (CC BY-NC 2.5). This is the problem we solve today.**

# What Happens When You Have 50 Features?

A bank tracks **50 risk metrics** for each portfolio. A data scientist needs to visualize risk clusters.

- Plotting 50 axes is impossible – humans see at most 3 dimensions
- Dropping features loses information – which ones are safe to remove?
- Many features are correlated – they carry redundant information

## The Key Insight

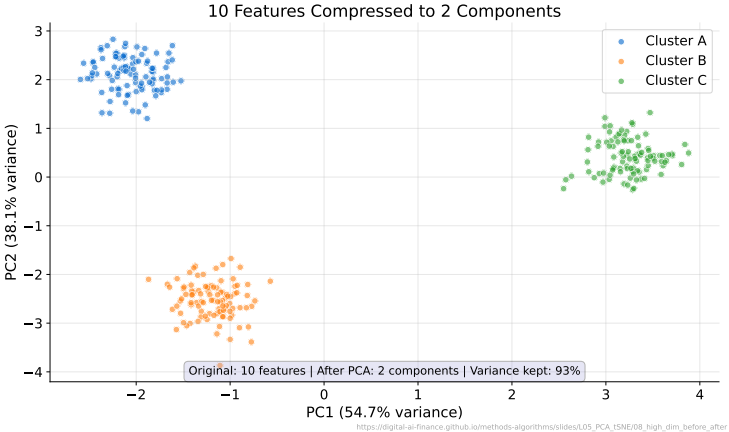
What if there were **directions** in the data that capture most of the variation? Then we could project onto just those directions and lose almost nothing.

This is exactly what **Principal Component Analysis (PCA)** does: it finds the directions of maximum variance and projects your data onto them.

---

**High-dimensional data is everywhere: genomics (20k genes), NLP (50k words), finance (hundreds of factors).**

# What Does Compressed Data Look Like?




- Each dot was originally described by **10 numbers** (features)
- PCA compressed it to just **2 numbers** (principal components)
- The three clusters are still clearly visible after compression

**PCA found the 2 directions that preserve the most spread in the data – no human picked them.**

# Why Not Just Pick Two Features?

**Cherry-Picking Features**

- Pick features 1 and 2 arbitrarily
- Lose all information from features 3–10
- The "best" pair depends on your task
- You must try all $\binom{p}{2}$ pairs

**PCA Approach**

- Combine ALL features into 2 new axes
- Each new axis is a weighted blend of all originals
- Maximizes retained information automatically
- No manual selection needed

## Key Difference

PCA does not discard features – it **blends** them. Each principal component is a linear combination of all original features.

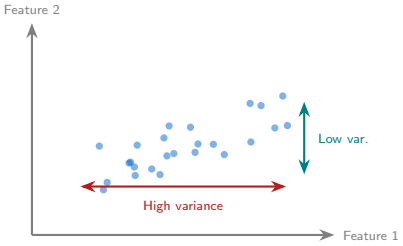
**Selecting features = supervised (you choose). PCA = unsupervised (the data chooses).**

# What Does “Variance” Mean Here?

**The Formula**

$$\mathrm{Var}(x) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

- Variance measures **spread** in the data
- High variance = lots of information
- Low variance = mostly noise




In PCA, “variance” and “information” are treated as synonyms. This is an assumption, not a universal fact.

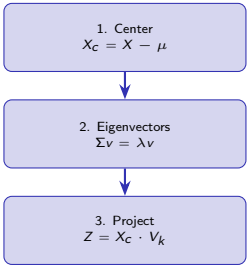
**In PCA, “variance” and “information” are treated as synonyms. This is an assumption, not a fact.**

# Road Map: Three Steps of PCA

1. **Center** the data
   Subtract the column mean from every feature so the cloud sits at the origin.

2. **Find the directions** of maximum variance
   Compute the covariance matrix and its eigenvectors – these are the principal components.

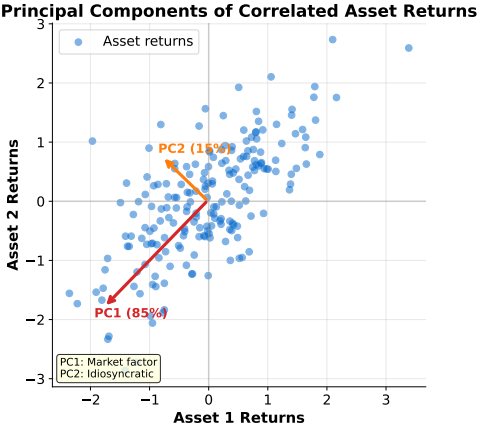
3. **Project** onto those directions
   Multiply your data by the top $k$ eigenvectors to get $k$ new features.




That is the **entire algorithm**. We will see each step visually before any formulas.

**We will see each step visually before we see any formulas.**

# Which Direction Captures the Most Spread?




https://github.com/Digital-AI-Finance/methods-algorithms/tree/master/slides/L05_PCA_tSNE/02_principal_components

- The **first arrow (PC1)** points along the main cloud – maximum spread
- The **second arrow (PC2)** is perpendicular – remaining spread
- Together they form new coordinate axes for the data

**PCA is a rotation: it does not distort your data, it views it from a better angle.**

# Step 1: Center the Data

**Definition**

$$X_c = X - \mu$$

where $\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$ is the column mean vector.

- Subtract the mean of each feature
- The centered cloud sits at the origin
- Ensures PC1 passes through the center of the data

**Mini-Example**

Data: $(2, 4)$, $(4, 6)$, $(6, 8)$, $(8, 10)$
Mean: $\mu = (5, 7)$

| | $x_1 - 5$ | $x_2 - 7$ |
|---|---|---|
| Centered: | $-3$ | $-3$ |
| | $-1$ | $-1$ |
| | $+1$ | $+1$ |
| | $+3$ | $+3$ |

The cloud shifts so its center is at $(0, 0)$.

**Centering ensures PC1 passes through the origin. Without centering, the mean dominates the first component.**

# Step 2: The Covariance Matrix

**Formula**

$$\Sigma = \frac{1}{n-1} X_c^T X_c$$

This is a $p \times p$ matrix where $p$ is the number of features.

- Diagonal entries = variance of each feature
- Off-diagonal entries = how features move together

**Interpretation**

$$\Sigma = \begin{pmatrix} \text{Var}(x_1) & \text{Cov}(x_1, x_2) \\ \text{Cov}(x_2, x_1) & \text{Var}(x_2) \end{pmatrix}$$

- Large off-diagonal = features are redundant
- Redundancy = PCA can compress
- All zeros off-diagonal = features already independent, PCA does nothing

**If all off-diagonals were zero, PCA would do nothing – the features are already independent.**

# Step 3: Eigenvalues and Eigenvectors (The Key Equation)

**The Equation**

$$\Sigma v = \lambda v$$

- $v$ is an **eigenvector**: a direction that $\Sigma$ only stretches, never rotates
- $\lambda$ is the **eigenvalue**: how much it stretches (= variance along $v$)
- Largest $\lambda$ = most important direction

**Intuition**

Imagine pulling a rubber sheet anchored at the center:

- The eigenvectors are the directions that **only stretch**, not twist
- The eigenvalues tell you **how far** each direction stretches
- We keep the directions that stretch the most

This is the ONLY equation you need to remember. Everything else follows from it.

**This is the ONLY equation you need to remember. Everything else follows from it.**

# Derivation: Why Does PCA Maximize Variance?

We want the direction $v$ that maximizes the variance of the projected data:

1. **Objective:** Maximize $\text{Var}(X_c\, v) = v^T \Sigma\, v$
2. **Constraint:** $\|v\| = 1$ (unit length, so the answer is a direction, not a scale)
3. **Lagrangian:** $\mathcal{L} = v^T \Sigma\, v - \lambda(v^T v - 1)$
4. **Set derivative to zero:** $\frac{\partial \mathcal{L}}{\partial v} = 2\Sigma v - 2\lambda v = 0 \quad \Rightarrow \quad \Sigma v = \lambda v$

## Result

The eigenvectors of $\Sigma$ are the principal components. The eigenvalues are their variances. The **largest eigenvalue** corresponds to the direction of maximum variance (PC1).

**Pearson (1901) and Hotelling (1933). The constrained optimization is standard Lagrange multipliers.**

# Worked Example: 2D to 1D by Hand

**Data:** 5 points in 2D: $(1,2),\ (3,4),\ (5,6),\ (2,3),\ (4,5)$

**Step 1 – Center:** $\mu = (3,4)$. Centered: $(-2,-2),\ (0,0),\ (2,2),\ (-1,-1),\ (1,1)$

**Step 2 – Covariance:** $\Sigma = \frac{1}{4}\begin{pmatrix} 10 & 10 \\ 10 & 10 \end{pmatrix} = \begin{pmatrix} 2.5 & 2.5 \\ 2.5 & 2.5 \end{pmatrix}$

**Step 3 – Eigenvalues:** $\det(\Sigma - \lambda I) = 0 \ \Rightarrow\ \lambda_1 = 5,\ \lambda_2 = 0$

**Eigenvector for** $\lambda_1 = 5$**:** $v_1 = \frac{1}{\sqrt{2}}(1,1)$ (the 45-degree line)

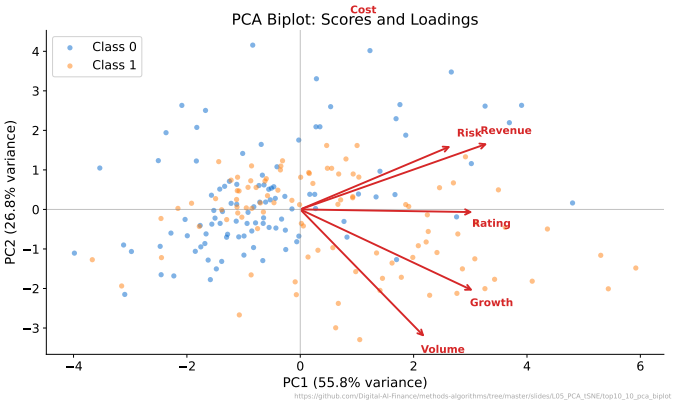
**Project:** $z_i = v_1^T x_{c,i}$: $-2\sqrt{2},\ 0,\ 2\sqrt{2},\ -\sqrt{2},\ \sqrt{2}$

## Result

We reduced 2D to 1D and kept **100%** of the variance ($\lambda_2 = 0$ means the second direction had zero spread – the data was perfectly correlated).

**Try this yourself: change the correlation and watch how the PC direction rotates.**

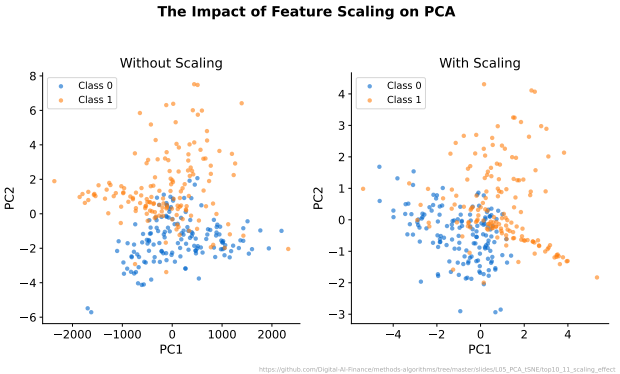
# How Do You Read a Biplot?




- **Dots** = data points in PC space
- **Arrows** = original features. Long arrow = feature contributes strongly
- Arrows pointing the same direction = correlated features

**Biplots are the main diagnostic tool for interpreting what PCA did. Always plot one.**

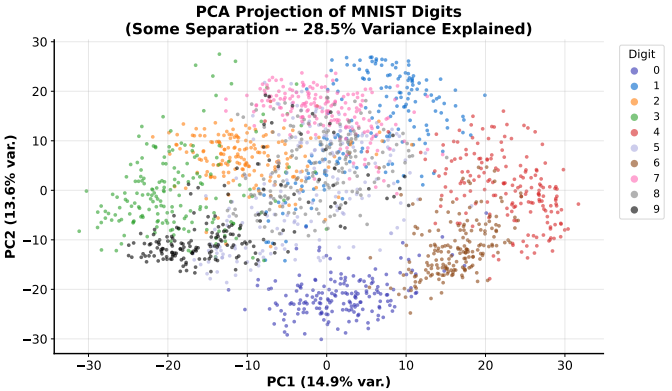
# What Happens If You Forget to Scale?




- **Without scaling**: a feature measured in thousands dominates PC1 alone
- **After scaling**: all features contribute fairly to each component
- Always standardize unless features share the same units

**StandardScaler (zero mean, unit variance) is the default. Skip it only when features are already on the same scale.**

# Can PCA Separate Clusters?




- Each color is a different class. PCA separates some but not all
- Well-separated classes remain distinct; overlapping classes stay mixed
- PCA is **linear** – it cannot untangle nonlinear structure

**For better cluster separation on complex data, consider t-SNE (visualization) or UMAP.**

# When Does PCA Fail?

**Strengths**

- Linear relationships between features
- Correlated features (high redundancy)
- Preprocessing for ML pipelines
- Interpretable components (loadings)

**Weaknesses**

- Nonlinear structure (e.g., swiss roll)
- Sensitive to outliers (they inflate variance)
- All features must be numeric
- Components are blends, not single features

## The Linear Assumption

PCA assumes linear relationships. If your data lives on a curved surface, PCA will cut through it rather than follow the curve.

**PCA assumes linear relationships. If your data lives on a curved surface, PCA will cut through it.**

# PCA as Preprocessing: The 95% Rule

**Practical Recipe for Using PCA in a Machine Learning Pipeline**

1. **Scale:** `StandardScaler().fit_transform(X)`
2. **Reduce:** `PCA(n_components=0.95).fit_transform(X_scaled)`
3. **Model:** Feed the reduced data into your classifier or regressor

## How It Works

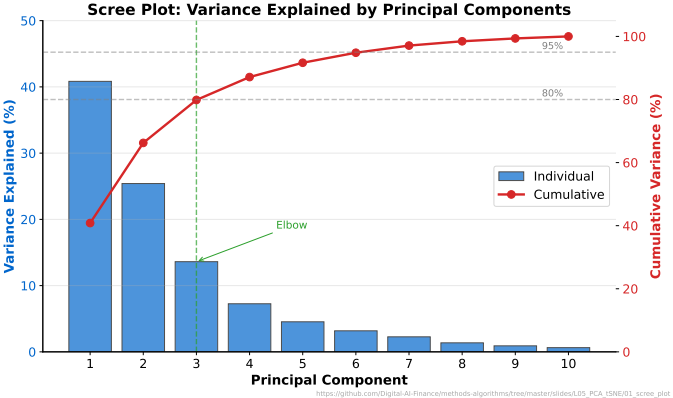
When you pass `n_components=0.95`, sklearn **automatically selects $k$** – the smallest number of components that explain at least 95% of total variance. No manual scree plot needed.

**Why does this help?** PCA removes noise dimensions, so downstream models train faster and often achieve *better* accuracy (less overfitting from irrelevant features).

---

**PCA before KNN or logistic regression often improves speed AND accuracy by removing noise dimensions.**

# How Many Components Do We Keep?




- **Bars** = variance explained by each PC. The first few tower over the rest
- **Line** = cumulative variance. It crosses 80% quickly
- The **elbow** says: keep components before the curve flattens, discard the rest

**Named after geological "scree" (rubble at a cliff base). You stop where the cliff flattens into rubble.**

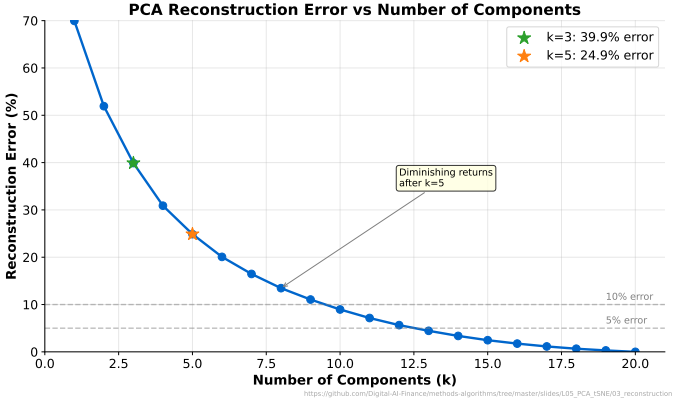
# Rules of Thumb for Choosing $k$

1. **Elbow Method**
   - Look at the scree plot
   - Find where the curve "bends"
   - Keep components before the elbow
2. **Threshold Method**
   - Set a target: 90–95% variance
   - Count components needed to reach it
   - Simple and widely used
3. **Cross-Validation**
   - Pick $k$ that maximizes downstream task performance
   - Most principled approach
   - Requires a supervised task

No single rule is always best. In practice, try 2–3 methods and see if they agree.

**No single rule is always best. In practice, try 2–3 methods and see if they agree.**

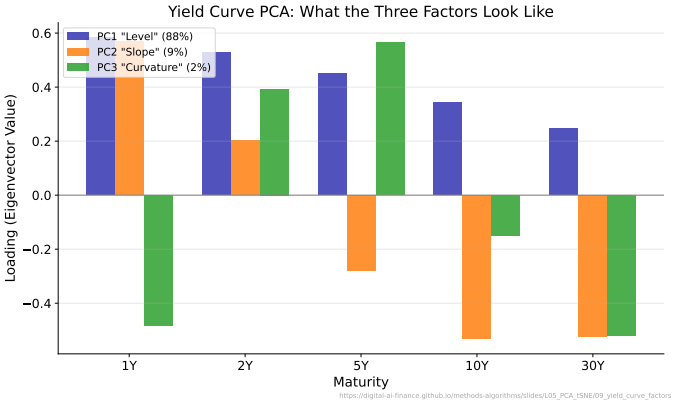
# Reconstruction Error: What Did We Lose?




- Error drops steeply for the first few components
- After $k \approx 5$, adding components barely helps
- The gap between $k$=3 and $k$=20 is only a few percent

**Reconstruction:** $\hat{X} = Z \cdot V_k^T + \mu$. **The error is** $\|X - \hat{X}\|^2$.

# Finance Application: Yield Curve PCA




- **PC1** (flat bars) = all maturities move together = "Level"
- **PC2** (sloped bars) = short and long rates diverge = "Slope"
- **PC3** (U-shaped bars) = belly moves differently = "Curvature"

**Litterman & Scheinkman (1991): 3 PCA factors explain >98% of US Treasury yield curve movements.**

# Yield Curve PCA: The Worked Numbers

**Portfolio Risk with PCA**

Instead of tracking 5+ maturity exposures, use 3 factor sensitivities:

$$\Delta P \approx D_1 \cdot \Delta\text{PC1} + D_2 \cdot \Delta\text{PC2} + D_3 \cdot \Delta\text{PC3}$$

where $D_k$ = sensitivity ("factor duration") to factor $k$.

**Example**

Suppose:

- Level shifts +10bp (PC1 = +10)
- Slope steepens (PC2 = +5)
- Curvature unchanged (PC3 = 0)
- $D_1 = 0.8,\ D_2 = 0.3,\ D_3 = 0.1$

Then:

$$\Delta P \approx 0.8 \times 10 + 0.3 \times 5 + 0.1 \times 0$$
$$= 8.0 + 1.5 + 0 = \textbf{9.5bp}$$

3 numbers replace 5+ maturity exposures. This is why every fixed income desk uses PCA.

**3 numbers replace 5+ maturity exposures. This is why every fixed income desk uses PCA.**

# PCA Decision Checklist

**Use PCA When...**

- Features are correlated (large off-diagonals in $\Sigma$)
- Too many dimensions for your model or plot
- Preprocessing to speed up ML and reduce noise
- You want interpretable compression (loadings)

**Skip PCA When...**

- Few features already ($p < 10$)
- Nonlinear structure (use kernel PCA or t-SNE)
- You need original feature names in the output
- Data is categorical (PCA needs numeric inputs)

## Quick Test

The simplest test: if your **correlation matrix has large off-diagonal entries**, PCA will help. If features are already independent, PCA adds nothing.

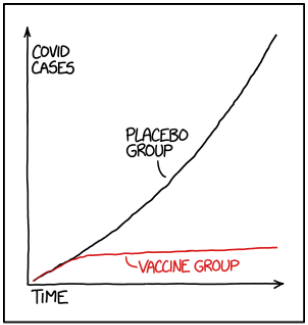
**The simplest test: if your correlation matrix has large off-diagonal entries, PCA will help.**

# PCA in Four Sentences

1. High-dimensional data is hard to visualize and model
2. PCA finds orthogonal directions of maximum variance
3. Keep $k$ components that explain 90–95% of variance
4. In finance, 3 PCA factors explain nearly all yield curve movement

## Next Steps

Explore t-SNE for nonlinear visualization in the deep dive. Try the Jupyter notebook to run PCA on real financial data.




**XKCD #2400 by Randall Munroe (CC BY-NC 2.5). Next: explore t-SNE for nonlinear visualization in the deep dive.**