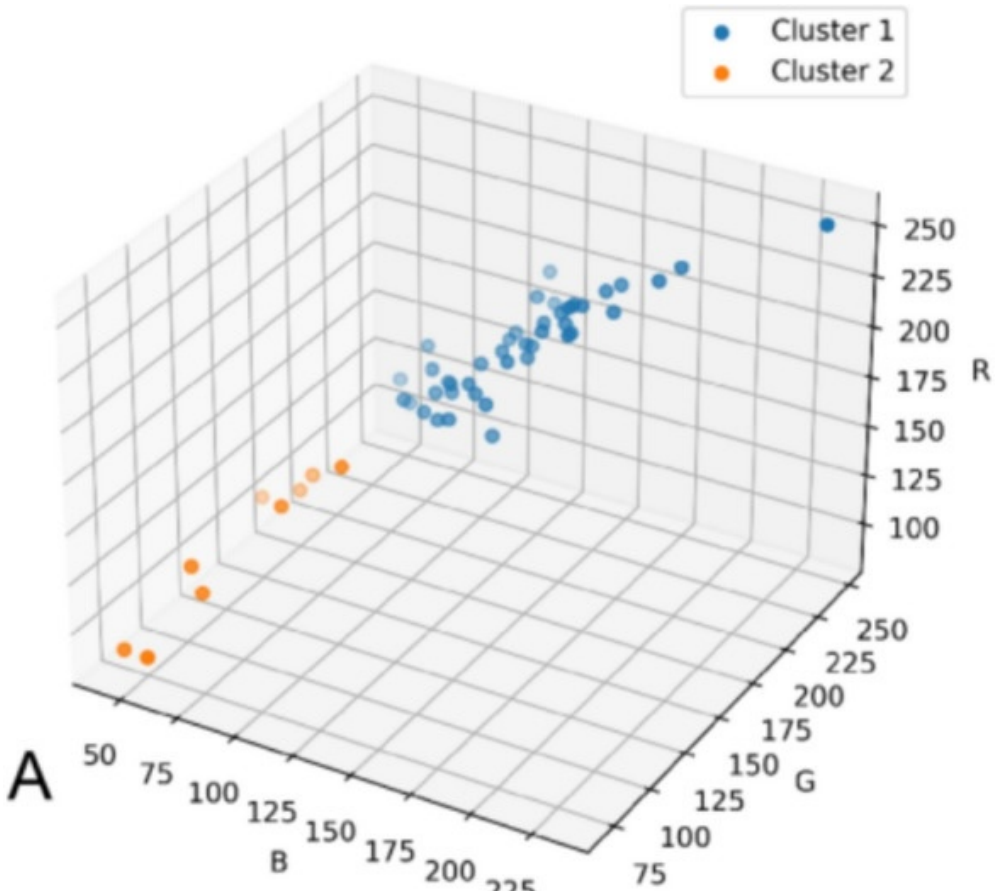
Principal Component Analysis

Jonathan Taylor
Stanford University

DATASCI 112

May 15, 2026



1. Motivation
2. Projection and Variance
3. Principal Components
4. PCA in Practice

1 Motivation

2 Projection and Variance

3 Principal Components

4 PCA in Practice

The Curse of Dimensionality

We have seen many datasets with many features (dimensions).

- Penguins: 4 features (bill length, bill depth, flipper length, body mass).
- Breast Cancer: 30 features.
- Bag of Words: 1000s of features.

Problems with high-dimensional data:

- Hard to visualize (we only have 2D/3D eyes).
- Redundant information (many features are correlated).
- Computationally expensive.

Dimensionality Reduction

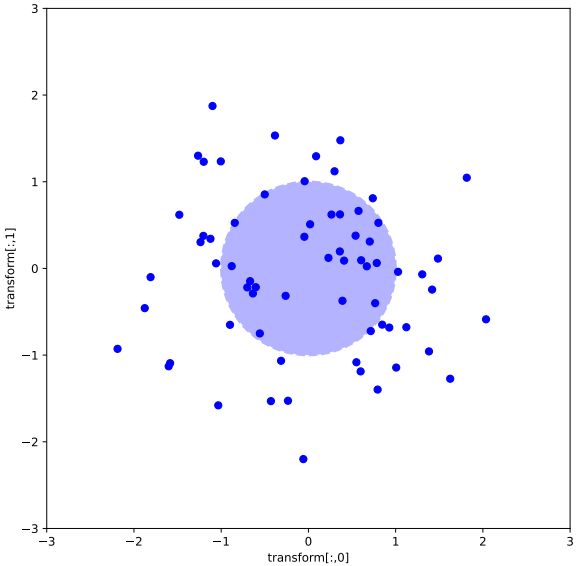
Goal: Find a lower-dimensional representation of the data that preserves its "essential" characteristics.

What do we mean by "essential"?
In **Principal Component Analysis (PCA)**, the goal is to keep much of the same *variation* intact.

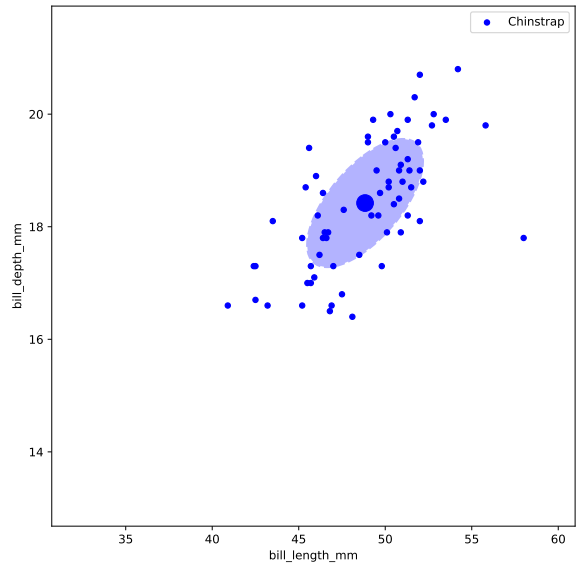
We want to find directions in the feature space along which the data varies the most.

Recap

Transformed features



Original features



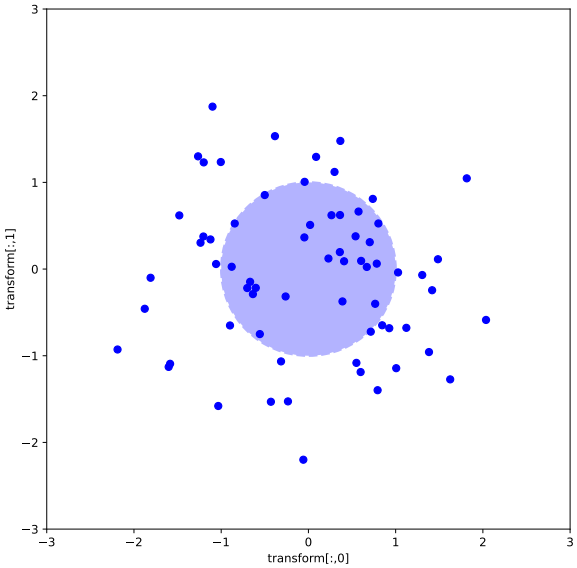
Recall that we were looking at how different linear transformations change the *shape* of the distance in the original space.

In the transform shape the data looks circular.

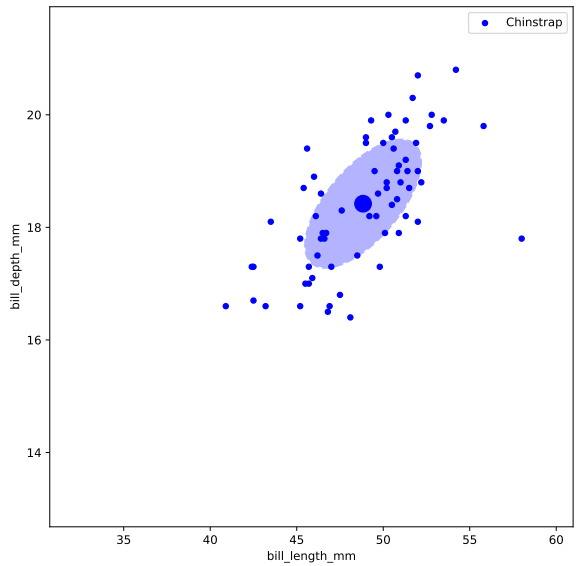
The cloud chose here “fits” the data better in original space than scaling alone.

Recap

Transformed features



Original features



The x coordinate in the transformed variables matches up with the major axis of the ellipse.

The y matches the minor axis of the ellipse.

Covariance

The ellipse is determined by the *covariance* of the features.

```
X_train[df['species'] == 'Chinstrap'].cov()
```

The correlation `.corr()` is a function of the `.cov()`.

	bill_length_mm	bill_depth_mm
bill_length_mm	11.150630	2.477801
bill_depth_mm	2.477801	1.289122

1 Motivation

2 Projection and Variance

3 Principal Components

4 PCA in Practice

Projecting onto a Line

Suppose we have 2D data (x, y). We want to reduce it to 1D.

Usually we'll first put the data into unitless coordinates by centering and scaling.

We can choose a line and *project* every point onto that line.

This will now depend on `.corr()`.

Each projected point is now just a single number (its position along the line).

Which Line to Choose?

We want to choose the line such that the projected points are as "spread out" as possible.

If the points are spread out in this direction, we have preserved as much of the original information (variance) as possible.

If we project onto a line where the points are all squashed together, we've lost that information.

This strategy will choose the major axis of the ellipse corresponding to the "point cloud".

Which ellipse? The one determined by `.cov()`.

1 Motivation

2 Projection and Variance

3 Principal Components

4 PCA in Practice

First Principal Component (PC1)

The **First Principal Component** is the direction (a unit vector $\mathbf{v}$) that maximizes the variance of the projected data.

Mathematically, if X is our centered data matrix, we want to find $\mathbf{v}$ to maximize:

$$\max_{\|\mathbf{v}\|=1} \text{Var}(X\mathbf{v})$$

Second Principal Component (PC2)

In 2-dimensions, it is the direction perpendicular to PC1.
In >2 dimensions, it is **Second Principal Component** is the direction that:

- Is orthogonal (perpendicular) to PC1.
- Maximizes the remaining variance.

For 2D data, this will be the minor axis of the ellipse.

For D-dimensional data, the ellipse has up to D axes leading to D scores. Each is orthogonal to all previous ones and captures as much of the remaining variance as possible.

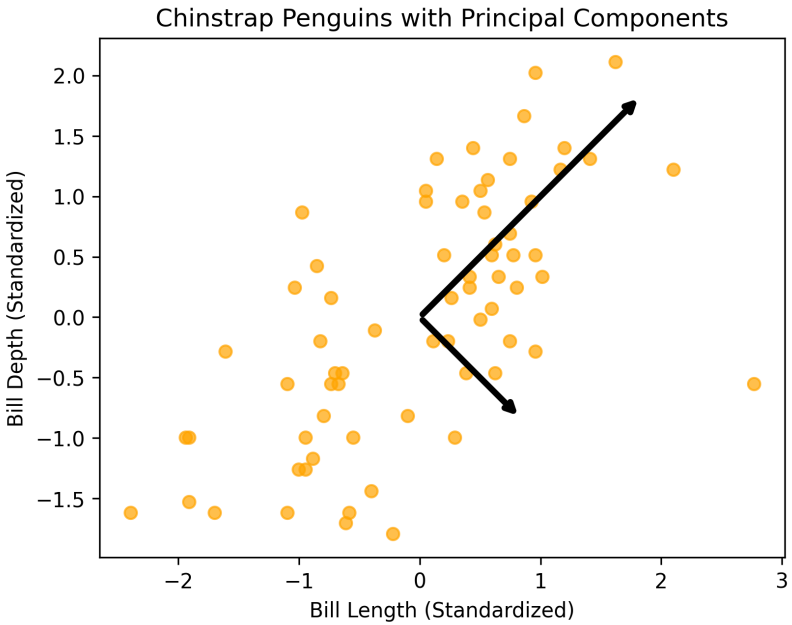
1 Motivation

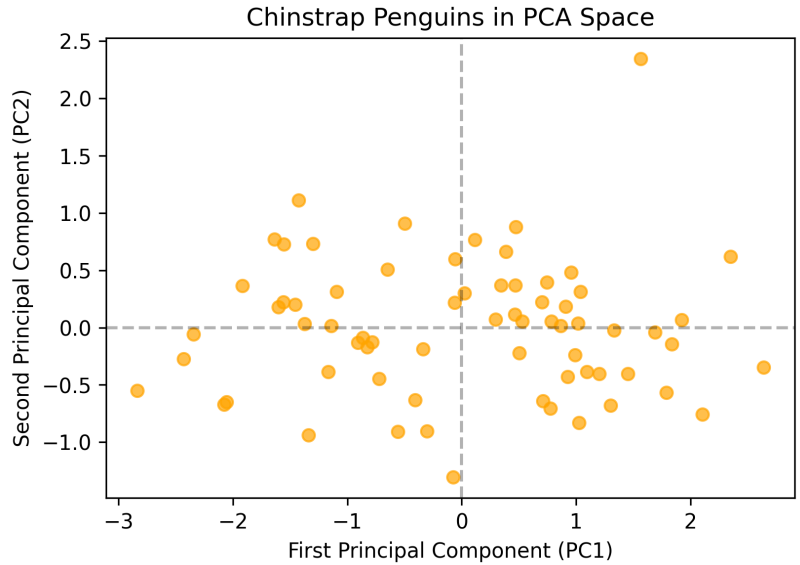
2 Projection and Variance

3 Principal Components

4 PCA in Practice

Visualizing PCA on Chinstrap Penguins





The original data (left) is rotated so that the principal components align with the x and y axes (right).

Choosing the Number of Components

If we have 30 features, how many PCs should we keep? 2? 5? 10?

We look at the **Explained Variance Ratio**: the percentage of total variance captured by each PC.

A **scree plot** shows the explained variance for each component.

We often look for an “elbow” in the plot where adding more components doesn’t add much more information.

PCA in scikit-learn

```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

# Create a pipeline that scales the data and then applies PCA
pca_pipeline = make_pipeline(
    StandardScaler(),   # very common practice to scale data first
    PCA(n_components=2)
)

# Fit and transform the data
X_pca = pca_pipeline.fit_transform(X)

# Convert to a DataFrame for easier plotting
df_pca = pd.DataFrame(X_pca, columns=["PC1", "PC2"])
df_pca["species"] = y.values
```

Summary

- PCA is a tool for **dimensionality reduction**.
- It finds directions (Principal Components) that maximize **variance**.
- PCs are **orthogonal** and ordered by importance.
- Essential for **visualization** and overcoming the **curse of dimensionality**.