MATH 250: Mathematical Data Visualization

Singular value decomposition and principal components analysis

Peter A. Gao

2025-02-26

San José State University

Warm-up: Ranks of common flags

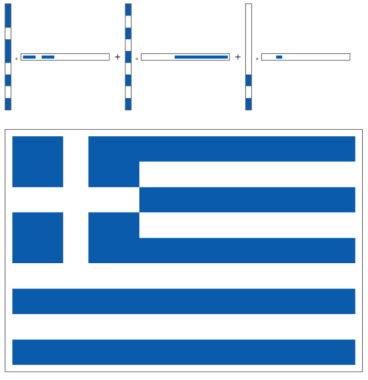


Figure 1: Rank-1 decomposition of Greek Flag by Michael E2

What would the rank of Denmark's flag be? 

Singular Value Decomposition

Review: SVD

Any matrix $A \in \mathbb{R}^{m \times n}$ can be factorized

$$A = U \Sigma V^\top$$

with

- $U \in \mathbb{R}^{m \times m}$ an orthogonal matrix of the **left singular vectors** of A
- $\Sigma \in \mathbb{R}^{m \times n}$ an $m \times n$ **singular value** matrix
- $V \in \mathbb{R}^{n \times n}$ an orthogonal matrix of the **right singular vectors** of A

Review: SVD

The columns of U form an orthonormal basis for the **column space** of A while the columns of V (rows of $V^\top$) form an orthonormal basis for the **row space**.

The key property to remember for the singular vectors:

$$A\mathbf{v}_i = \sigma_i \mathbf{u}_i, \quad i = 1, \ldots, r$$

Where $\mathbf{v}_i$ and $\mathbf{u}_i$ are the ith right and left singular vectors, respectively, σ_i is the ith singular value, and r is the rank of A.

Review: SVD

Example:

$$\begin{bmatrix} 3 & 0 \\ 4 & 5 \end{bmatrix} = \begin{bmatrix} 1/\sqrt{10} & -3/\sqrt{10} \\ 3/\sqrt{10} & 1/\sqrt{10} \end{bmatrix} \begin{bmatrix} 3\sqrt{5} & 0 \\ 0 & \sqrt{5} \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}$$

Decomposing linear transformations

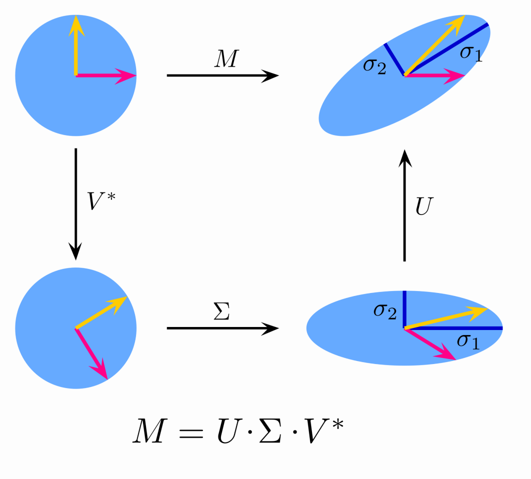


Figure 2: Geometric interpretation of SVD, by Georg-Johann

Review: SVD

In other words:

$$AV = U\Sigma \iff A\begin{bmatrix} \mathbf{v}_1 & \cdots & \mathbf{v}_n \end{bmatrix} = \begin{bmatrix} \mathbf{u}_1 & \cdots & \mathbf{u}_m \end{bmatrix} \left[\begin{array}{ccc|c} \sigma_1 & & & \\ & \ddots & & \mathbf{0} \\ & & \sigma_r & \\ \hline & \mathbf{0} & & \mathbf{0} \end{array}\right]$$

We can also write A in a reduced SVD form:

$$AV_r = U_r\Sigma_r$$

making Σ_r a diagonal $r \times r$ matrix and removing the last singular vectors from V and U.

Review: SVD as sum of rank-1 matrices

We can also write A as a sum of rank-1 matrices:

$$A = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^\top + \cdots + \sigma_r \mathbf{u}_r \mathbf{v}_r^\top$$

and obtain the best rank-k approximation:

$$A_k = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^\top + \cdots + \sigma_k \mathbf{u}_k \mathbf{v}_k^\top$$

Theorem

(Eckart-Young) If B has rank k, then $||A - A_k|| \leq ||A - B||$ in either the Frobenius or ℓ^2 norm.

Review: Other SVD properties

- V contains orthornormal eigenvectors of $A^\top A$ and U contains orthonormal eigenvectors of $AA^\top$.
- $\sigma_1^2, \dots, \sigma_r^2$ are the nonzero eigenvalues of both $A^\top A$ and $AA^\top$.
- If S is symmetric positive definite, $U\Sigma V^\top = Q\Lambda Q^\top$.
- $\mathbf{v}_1$ maximizes $||A\mathbf{x}||/||\mathbf{x}||$, achieving a value of σ_1.

Application: Image compression

Figure 3: Stephan's quintet

Application: Image compression

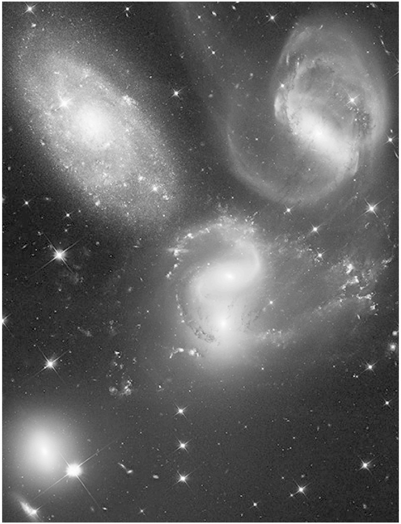
Each pixel is a value from 0-255 representing a color from white to black.

We can thus treat this image as a matrix, compute its SVD and the best rank-k approximation.

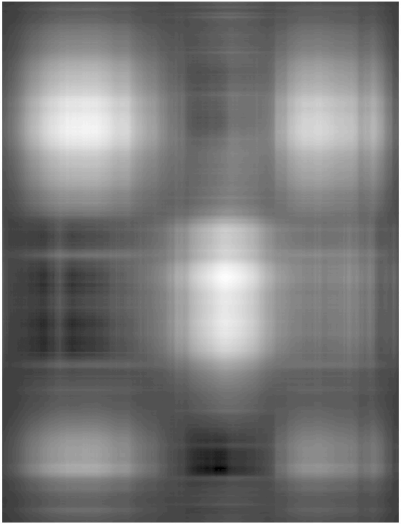
Plotting the rank-k approximation yields a compressed version of our original image.

Application: Image compression}

Original (2419 kb)

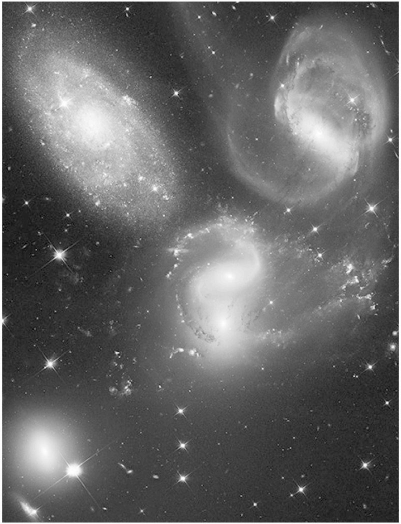


Rank-2 approximation (11 kb)

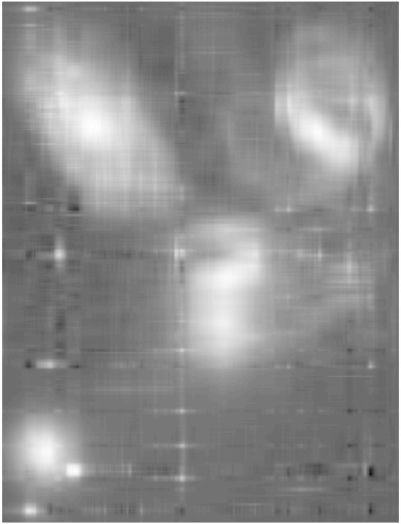


Application: Image compression

Original (2419 kb)

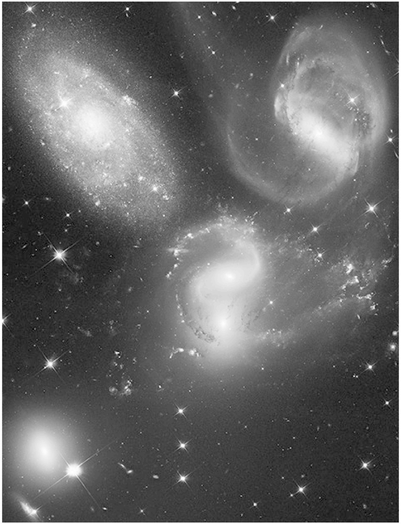


Rank-10 approximation (55.2 kb)

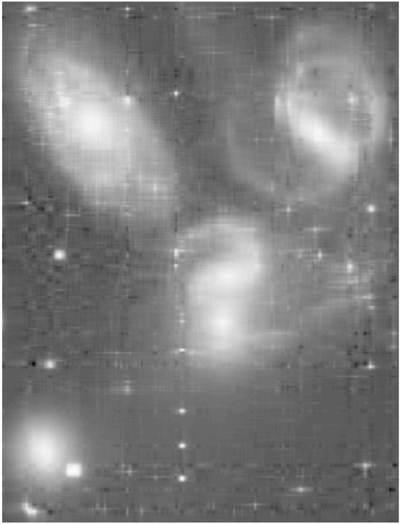


Application: Image compression

Original (2419 kb)

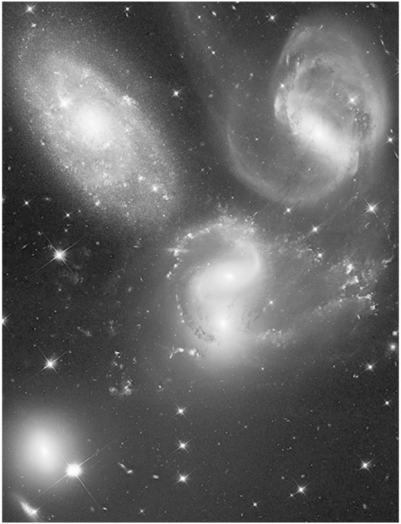


Rank-20 approximation (110.4 kb)



Application: Image compression

Original (2419 kb)



Rank-50 approximation (276 kb)

Application: Image compression

The SVD provides a crude approach to image compression, which is the "best" in the sense that it minimizes the matrix distance between these images.

However, when viewing two images, this may not be the right "distance" to be using.

When noise has been added to our image, the SVD can also be used to denoise and clean up images.

Application: Handwritten digits

USPS handwritten digits data:

- 9298 16 x 16 images of handwritten digits, split into training and test datasets.
- Centered and scaled to be the same size.

Example: handwritten digits

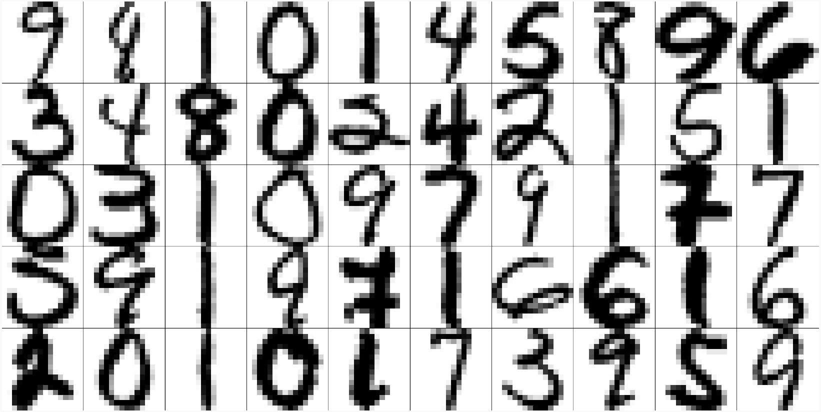


Figure 4: Handwritten 16 x 16 digits from USPS dataset Hull (1994)

Example: handwritten digits

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[,9]	[,10]	[,11]	[,12]	[,13]	[,14]	[,15]	[,16]
[1,]	0	0	0	0	0	11	167	197	29	0	0	0	0	0	0	0
[2,]	0	0	0	0	22	207	255	204	0	0	0	0	0	0	0	0
[3,]	0	0	0	95	248	255	160	7	0	0	0	0	0	0	0	0
[4,]	0	58	175	255	255	226	117	146	117	88	88	29	0	0	0	0
[5,]	84	255	255	255	204	145	145	204	145	174	229	255	197	80	0	0
[6,]	32	65	36	7	0	0	0	0	0	0	3	160	255	255	65	0
[7,]	0	0	0	0	0	0	0	0	0	0	0	204	255	236	21	0
[8,]	0	0	0	0	0	0	0	0	0	59	175	255	225	40	0	0
[9,]	0	0	0	0	0	0	22	110	226	255	233	145	0	0	0	0
[10,]	0	0	22	132	190	219	255	255	255	255	190	132	73	0	0	0
[11,]	0	0	7	101	130	72	14	14	14	72	101	159	251	212	37	0
[12,]	0	0	0	0	0	0	0	0	0	0	0	0	25	255	255	44
[13,]	0	0	0	0	0	0	0	0	0	0	0	0	26	255	255	101
[14,]	0	0	116	95	0	0	0	0	0	0	0	44	193	255	247	0
[15,]	0	0	0	138	154	37	37	37	66	125	212	255	236	130	21	0
[16,]	0	0	0	0	50	108	166	196	196	196	137	79	10	0	0	0

Figure 5: Handwritten digit from USPS dataset, as matrix

The “typical” digits

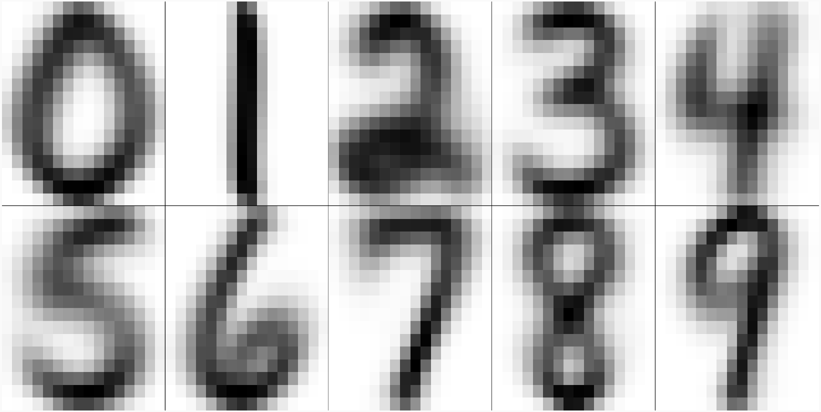


Figure 6: Centroids from USPS dataset (element-wise means)

A naive classification algorithm

1. For each new image i, calculate its distance from the centroids 0-9.
2. Label the new image i based on the closest centroid.

This achieves 75% accuracy. Note the work needed to create all these black-and-white, centered images.

Creating SVD-based representations of each digit

Let n_i be the number of images of digit i in the training set. For each digits, construct a $n_i \times 256$ matrix:

n_i rows $\boxed{A}$

256 columns

The right singular vectors $\mathbf{v}_i$ of A form an orthonormal basis in the space of images.

For a given digit, the first few singular vectors can be used to reconstruct each image in the training set.

Creating SVD-based representations of each digit

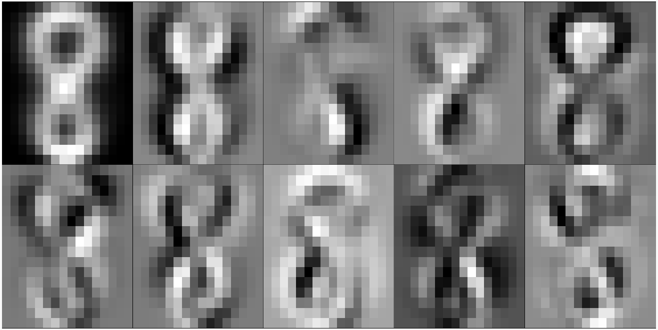


Figure 7: First 10 basis images for USPS 8 digits

Creating SVD-based representations of each digit

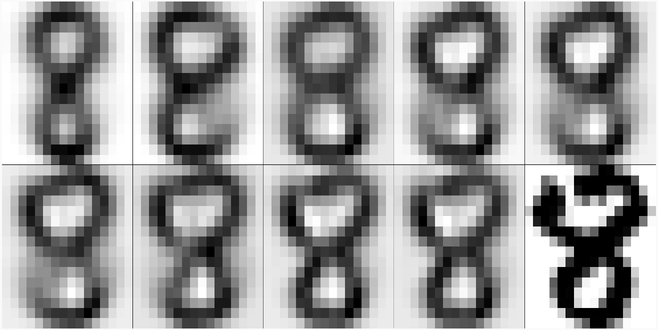


Figure 8: A single 8 image (lower right), projected on the first k basis images, for $k = 1$ to $k = 9$

Creating SVD-based representations of each digit

Figure 9: A single 8 image (lower right), projected on the first k basis images **for the digit 9**, for $k = 1$ to $k = 9$

SVD basis classification

1. For each new image i, calculate its representation in the SVD basis for each digit.
2. Label the new image i based on the most accurate representation.

For an unknown image $\mathbf{z}$, we can approximate it in a basis using a least squares solution:

$$\min_{\mathbf{c}} ||\mathbf{z} - \sum_{i=1}^{k} c_i \mathbf{u}_i||$$

We can repeat this process for each basis for 0-9 and identify the digit that yields the most accurate representation.

SVD basis classification

The accuracy of this classification method depends on the dimension k of the basis:

# basis images	1	2	4	6	8	10
accuracy	80	86	90	90.5	92	93

Figure 10: Accuracy by number of basis images (Elden 2007)

Principal components analysis

PCA vs. SVD

You may often hear "PCA is just SVD." It is–sort of.

SVD

- a matrix method
- $m \times n$ matrix

PCA

- a data analysis method
- $n \times p$ data matrix

Let's start with PCA and show how it relates to the SVD.

Reviewing statistics

Definition

The **covariance** between two random variables X and Y is defined as

$$\operatorname{Cov}(X, Y) = E[(X - E(X))(Y - E(Y))]$$

Remark

If $\mathbf{x}$ is a p-dimensional random vector, its **covariance matrix** is defined to be

$$\operatorname{Cov}(\mathbf{x}) = E[(\mathbf{x} - E(\mathbf{x}))(\mathbf{x} - E(\mathbf{x}))^\top]$$

Thus, the covariance matrix is positive semidefinite.

Reviewing statistics

$$\text{Cov}(\mathbf{x}) = E[(\mathbf{x} - E(\mathbf{x}))(\mathbf{x} - E(\mathbf{x}))^\top]$$

$$= \begin{bmatrix} \text{Var}(X_1) & \text{Cov}(X_1, X_2) & \cdots & \text{Cov}(X_1, X_p) \\ \text{Cov}(X_1, X_2) & \text{Var}(X_2) & \cdots & \text{Cov}(X_2, X_p) \\ \vdots & \vdots & \ddots & \vdots \\ \text{Cov}(X_1, X_p) & \text{Cov}(X_2, X_p) & \cdots & \text{Var}(X_p) \end{bmatrix}$$

Remark

As a result,

$$\text{Cov}(A\mathbf{x} + \mathbf{b}) = A[\text{Cov}(\mathbf{x})]A^\top$$

Reviewing statistics

If we have a sample of iid random vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n \sim \mathbf{x}$, we can combine them into an $n \times p$ data matrix:

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix}$$

Definition

The **sample covariance** of $\mathbf{x}$ is defined as

$$\widehat{S} = \frac{1}{n} \sum_{i=1}^{n} (\mathbf{x}_i - \overline{\mathbf{x}})(\mathbf{x}_i - \overline{\mathbf{x}})^\top = \frac{1}{n} X_c^\top X_c$$

where $\overline{\mathbf{x}}$ is the sample mean and X_c is a centered version of X (Exercise).

PCA: One-dimensional case

Motivation

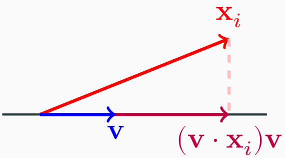
We want to project our p-dimensional data into a simpler q-dimensional space. We will try to choose the ”most important” q dimensions along which the data have maximum variance.

We can start with the example where $q = 1$. What does it mean to find the “optimal” one-dimensional projection of our data? Assume all of our data is centered, so the mean of each column of X is zero.

Idea (from Shalizi)

Choose the unit vector $\mathbf{v}$ such that when we project our data vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$ onto $\mathbf{v}$, the residual error is minimized:

$$\mathrm{MSE}(\mathbf{v}) = \frac{1}{n}\sum_{i=1}^{n} ||\mathbf{x}_i - (\mathbf{v}\cdot\mathbf{x}_i)\mathbf{v}||^2$$



PCA: Outline

Idea (from Shalizi)

This turns out to be equivalent to maximizing the sample variance of lengths of the projections onto $\mathbf{v}$ (since the columns of X are centered):

$$
\begin{aligned}
\widehat{\mathrm{Var}}(\mathbf{v} \cdot \mathbf{x}_i) &= \frac{1}{n} \sum_{i=1}^{n} (\mathbf{v} \cdot \mathbf{x}_i)^2 \\
&= \frac{1}{n} (X\mathbf{v})^\top (X\mathbf{v}) \\
&= \frac{1}{n} \mathbf{v}^\top X^\top X \mathbf{v} \\
&= \mathbf{v}^\top \widehat{S} \mathbf{v}
\end{aligned}
$$

PCA: One-dimensional case

In other words, we are simply maximizing the Rayleigh quotient $\mathbf{v}^\top \widehat{S}\mathbf{v}$. How do we find the maximizing $\mathbf{v}$?

PCA: One-dimensional case

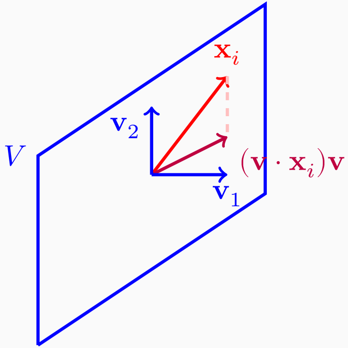
In other words, we are simply maximizing the Rayleigh quotient $\mathbf{v}^\top \widehat{S}\mathbf{v}$. How do we find the maximizing $\mathbf{v}$?

From last week, the maximizing $\mathbf{v}$ is the eigenvector of $\widehat{S}$ with the largest eigenvalue λ_1.

Thus, $\mathbf{v}^\top \widehat{S}\mathbf{v}$ achieves maximum value λ_1.

PCA: Multi-dimensional case

For $q > 1$, we can generalize our approach. Instead of the single vector along which the projected data has maximum variance, we are looking for a k-dimensional plane along which our projected data has maximum variance.



PCA: Multi-dimensional case

Theorem

The q-dimensional plane along which our projected data has maximum variance has an orthonormal basis given by the first q eigenvectors of $\widehat{S}$ and the total variance of the projections is given by $\lambda_1 + \cdots + \lambda_k$.

From PCA to SVD

Computing $X^\top X$ is potentially expensive and can lead to an ill-conditioned matrix.

Luckily, the first q eigenvectors of $X^\top X$ are also given by…

From PCA to SVD

Computing $X^\top X$ is potentially expensive and can lead to an ill-conditioned matrix.

Luckily, the first q eigenvectors of $X^\top X$ are also given by... the first q right singular vectors of X (remember, X is centered). The variance captured by the q-dimensional projection plane is $\lambda_1 + \cdots + \lambda_q = \sigma_1^2 + \cdots + \sigma_q^2$.

Summary

Principal components analysis typically involves identifying a set of maximum-variance directions

$$V_q = \begin{bmatrix} \mathbf{v}_1 & \cdots & \mathbf{v}_q \end{bmatrix} \in \mathbb{R}^{p \times q}$$

and the corresponding coordinates of each of the observations in the new basis

$$Y_q = \begin{bmatrix} \mathbf{y}_1 & \cdots & \mathbf{y}_q \end{bmatrix} \in \mathbb{R}^{n \times q}$$

where

$$Y_q = XV_q = U_q\Sigma_q$$

Definitions

1. **Principal directions**: The unit eigenvectors $\mathbf{v}_1, \dots, \mathbf{v}_q$ that form a basis for the q-dimensional subspace on which the projected data have maximum variance.
2. **Variable loadings**: Usually also referring to the eigenvectors $\mathbf{v}_1, \dots, \mathbf{v}_q$ and in particular the scalar components. For example, v_{11}, the first component of $\mathbf{v}_1$ represents the contribution (loading) of $\mathbf{x}_1$ to the first principal direction.
3. **Principal components**: The columns of the rotated data matrix XV_q representing the new coordinates after projected the data onto the q-dimensional subspace spanned by the principal directions.

Summary

We can say

- The unit eigenvector $\mathbf{v}_j$ is the jth **principal direction** of the data;
- The **principal components** $Y \in \mathbb{R}^{n \times q}$ are the coefficients obtained by projecting X on the first q **principal directions** of the data.

In essence, PCA is a change of coordinate system, where the new axes are the principal directions/axes of the data and the new coordinates the principal components. These principal directions are orthogonal.

How many principal components?

If $n >= p$, then as long as the columns of X are linearly independent, the rank of X is p, so we can have up to p principal components/directions.

What if $n < p$? Still feasible and as long as the columns of X are linearly independent, but the maximum number of principal components is equal to n.

Geometric interpretation

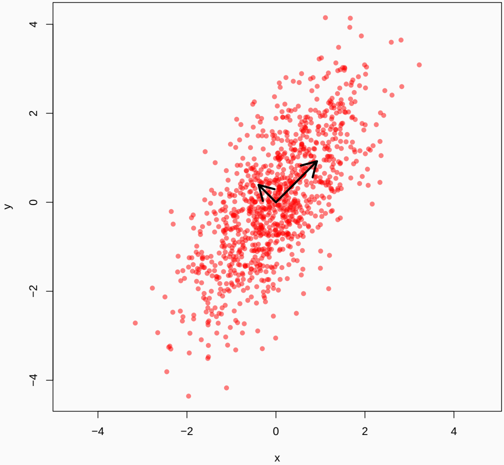


Figure 11: Example data with **principal directions**

Geometric interpretation

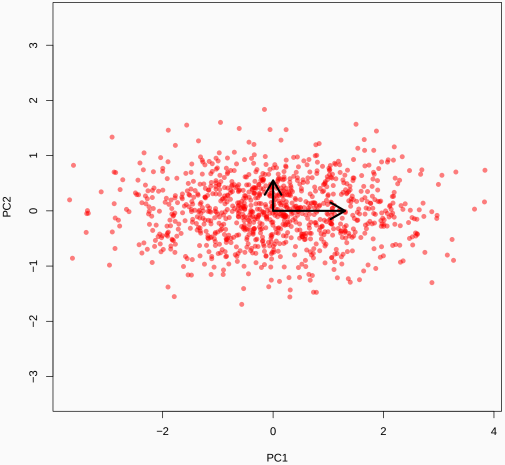


Figure 12: Example **principal components**

Steps

Given a data matrix $X \in \mathbb{R}^{n \times p}$ and an integer q,

1. Center X by subtracting out the mean, if necessary.
2. Carry out rank-q SVD on $X \approx U_q \Sigma_q V_q^\top$ (using an efficient algorithm)
3. Compute the principal components $Y = U_q \Sigma_q$.

Application: NWSL Teams

In honor of San Jose’s new soccer team Bay FC, we’ll take a look at data from the National Women’s Soccer League, using the `nwslR` package.

In 2023, there were 12 teams. We can download team-level data for each team including variables like **goals**, **assists**, and **goals allowed**.

Application: NWSL Teams

```r
library(nwslR)
library(tidyverse)
teams <- load_teams() |>
  filter(last_season >= 2023 & first_season <= 2023 & team_abbreviation != "UTA")
nwsl23 <-
  lapply(
    teams$team_abbreviation,
    function(x) load_team_season_stats(team_id = x, season = "2023")
  ) |>
  bind_rows()

nwsl23_mat <- nwsl23[, c(-1, -2)] |>
  dplyr::select("possession_pct",
                "goals", "assists", "pass_pct",
                "goal_conversion_pct", "clean_sheets",
                "shot_accuracy", "shots_total", "goals_conceded",
                "tackled") |>
  as.matrix(nrow = nrow(nwsl23))
rownames(nwsl23_mat) <-
  teams$team_abbreviation[match(nwsl23$team_id, teams$team_id)]
```

Application: NWSL Teams

```r
head(nwsl23_mat)
```

```
    possession_pct goals assists pass_pct goal_conversion_pct clean_sheets
CHI             47    19      11    75.15               17.59            3
HOU             48    10       5    71.08                7.87            6
NJY             53    17       8    73.67               11.26            4
RGN             49    23      18    73.97               17.04            4
ORL             47    15       9    75.07               10.20            4
POR             53    34      27    77.53               15.25            4
    shot_accuracy shots_total goals_conceded tackled
CHI         46.30         108             33     300
HOU         44.88         127             12     259
NJY         48.34         151             14     303
RGN         47.41         135             18     220
ORL         47.62         147             21     314
POR         52.02         223             21     274
```

Application: NWSL Teams

```r
pca <- prcomp(nwsl23_mat, scale = T)
options(width = 60)
summary(pca)
```

```
Importance of components:
                          PC1    PC2    PC3    PC4     PC5
Standard deviation     1.9471 1.5613 1.1729 1.1076 0.84948
Proportion of Variance 0.3791 0.2438 0.1376 0.1227 0.07216
Cumulative Proportion  0.3791 0.6229 0.7605 0.8832 0.95532
                           PC6     PC7     PC8     PC9
Standard deviation     0.50289 0.36546 0.20599 0.13384
Proportion of Variance 0.02529 0.01336 0.00424 0.00179
Cumulative Proportion  0.98061 0.99396 0.99821 1.00000
                           PC10
Standard deviation     0.002521
Proportion of Variance 0.000000
Cumulative Proportion  1.000000
```

Application: NWSL Teams

```r
str(pca) # structure of pca object
```

```
List of 5
 $ sdev    : num [1:10] 1.947 1.561 1.173 1.108 0.849 ...
 $ rotation: num [1:10, 1:10] 0.368 0.422 0.415 0.386 0.337 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:10] "possession_pct" "goals" "assists" "pass_pct" ...
  .. ..$ : chr [1:10] "PC1" "PC2" "PC3" "PC4" ...
 $ center  : Named num [1:10] 49.9 19.7 12.2 74.6 13.9 ...
  ..- attr(*, "names")= chr [1:10] "possession_pct" "goals" "assists" "pass_pct
 $ scale   : Named num [1:10] 3.63 5.71 5.91 3.03 3.19 ...
  ..- attr(*, "names")= chr [1:10] "possession_pct" "goals" "assists" "pass_pct
 $ x       : num [1:12, 1:10] -1.046 -2.551 -0.679 0.891 -1.577 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:12] "CHI" "HOU" "NJY" "RGN" ...
  .. ..$ : chr [1:10] "PC1" "PC2" "PC3" "PC4" ...
 - attr(*, "class")= chr "prcomp"
```

Application: NWSL Teams

```r
# square root of eigenvalues
pca$sdev
```

```
 [1] 1.947118570 1.561340876 1.172896087 1.107625762
 [5] 0.849476477 0.502890350 0.365461414 0.205993340
 [9] 0.133840335 0.002521167
```

Application: NWSL Teams

```r
# first three principal directions
pca$rotation[, 1:3]
```

```
                           PC1         PC2         PC3
possession_pct       0.3675695 -0.25738992 -0.07820678
goals                0.4218606  0.33869010 -0.09242574
assists              0.4152512  0.29222969 -0.09100741
pass_pct             0.3862252 -0.15782340  0.07035432
goal_conversion_pct  0.3368898  0.17920622  0.57472153
clean_sheets         0.2122449 -0.55909064  0.03992048
shot_accuracy        0.3389031 -0.07611405  0.04417828
shots_total          0.2224901  0.26001949 -0.66237229
goals_conceded      -0.1001183  0.51019377  0.36817595
tackled             -0.1799982  0.17416846 -0.25291961
```

How should we interpret the loadings for the first two principal directions?

Application: NWSL Teams

```
# original column standard deviations
pca$scale
```

```
     possession_pct               goals             assists 
           3.629634            5.710172            5.905827 
           pass_pct goal_conversion_pct         clean_sheets 
           3.028963            3.185115            1.443376 
      shot_accuracy         shots_total      goals_conceded 
           3.481862           29.283877            5.944185 
            tackled 
          32.577530 
```

Application: NWSL Teams

```r
unscaled_pca <- prcomp(nwsl23_mat, scale = F)
unscaled_pca$rotation[, 1:3]
```

```
                             PC1          PC2         PC3
possession_pct       0.001564728 -0.018339648 -0.12307260
goals                0.017632048 -0.145233100 -0.48496553
assists              0.001849595 -0.125199988 -0.54508628
pass_pct            -0.002864390 -0.015535596 -0.17098315
goal_conversion_pct  0.036297376  0.001347064 -0.38041324
clean_sheets         0.016597302  0.005625277  0.03499502
shot_accuracy        0.030876779 -0.043509933 -0.10711208
shots_total         -0.230507228 -0.954278241  0.12837166
goals_conceded      -0.064831781  0.040496550 -0.49884966
tackled             -0.969428672  0.220093406 -0.02377243
```

What changed? How can we interpret these loadings?

Scaling matters

Essentially, if we do **not** scale the variables first, we are computing the eigenvectors of $X_c^\top X_c$, which is proportional to the sample covariance.

If we **do** scale the variables, we are computing the eigenvectors of the sample **correlation** matrix.

If our variables have differing sample variances, then the variables with larger variance will dominate the first principal components.

Application: NWSL Teams

```
biplot(pca)
```

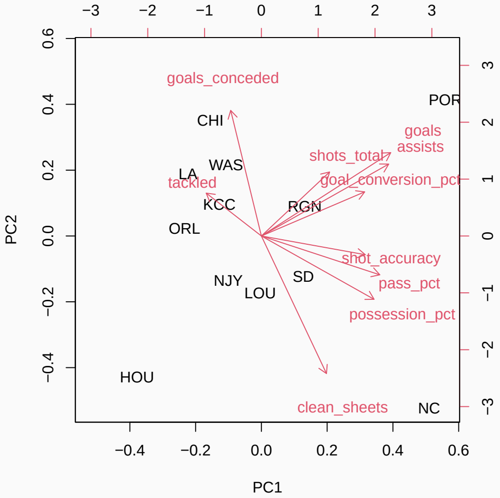


Figure 13: Biplot for NWSL 2023 team-level data

Application: Women's Soccer Teams

Biplots illustrate both:

- **principal components**: the positions of each observation in the rotated space (columns of U)
- **principal directions**: the columns of V contain **variable loadings** (the contribution of each variable to the PCs).

With which variables is the first principal component associated? What about the second principal component?

Application: NWSL Teams

```
plot(pca$rotation[,1], pca$rotation[,2], col = "white",
     xlim = c(-.75, .75), ylim = c(-.75, .75),
     xlab = "PC1", ylab = "PC2")
text(pca$rotation[,1], pca$rotation[,2], rownames(pca$rotation))
```

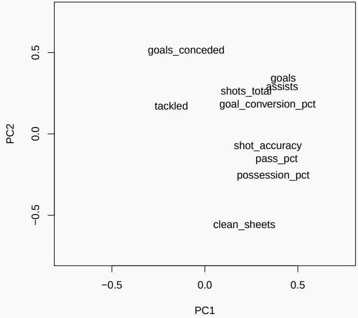


Figure 14: Variable loadings for first two PCs

Application: NWSL Teams

```
plot(pca$x[,1], pca$x[,2], col = "white",
     xlim = c(-3.75, 3.75), ylim = c(-3.75, 3.75),
     xlab = "PC1", ylab = "PC2")
text(
  pca$x[,1], pca$x[,2],
  teams$team_abbreviation[match(nwsl23$team_id, teams$team_id)],
)
```

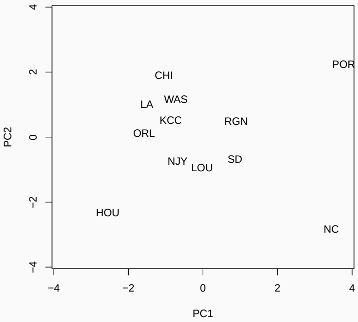


Figure 15: First two PCs for NWSL 2023 teams

Application: NWSL Teams

Pos	Team [v·t·e]	Pld	W	D	L	GF	GA	GD	Pts	Qualification
1	San Diego Wave FC	22	11	4	7	31	22	+9	**37**	NWSL Shield, playoffs – semifinals
2	Portland Thorns FC	22	10	5	7	42	32	+10	**35**	Playoffs – semifinals
3	North Carolina Courage	22	9	6	7	29	22	+7	**33**	Playoffs – quarterfinals
4	OL Reign	22	9	5	8	29	24	+5	**32**	
5	Angel City FC	22	8	7	7	31	30	+1	**31**	
6	NJ/NY Gotham FC	22	8	7	7	25	24	+1	**31**	
7	Orlando Pride	22	10	1	11	27	28	–1	**31**	
8	Washington Spirit	22	7	9	6	26	29	–3	**30**	
9	Racing Louisville FC	22	6	9	7	25	24	+1	**27**	
10	Houston Dash	22	6	8	8	16	18	–2	**26**	
11	Kansas City Current	22	8	2	12	30	36	–6	**26**	
12	Chicago Red Stars	22	7	3	12	28	50	–22	**24**	

Figure 16: Standings for NWSL 2023 from Wikipedia

Application: NWSL Teams

The first principal direction is associated with several variables about possession and scoring (goals and assists).

The second principal direction seems to have more to do with defense (goals conceded and clean sheets).

Application: NWSL Teams

```r
var_explained = pca$sdev^2 / sum(pca$sdev^2)
ggplot(data = data.frame(x = c(1:10), y = var_explained),
       mapping = aes(x = x, y = y)) + geom_line() +
  xlab("PCs") + ylab("Variance Explained") + ggtitle("Scree Plot") +
  ylim(0, 1)
```

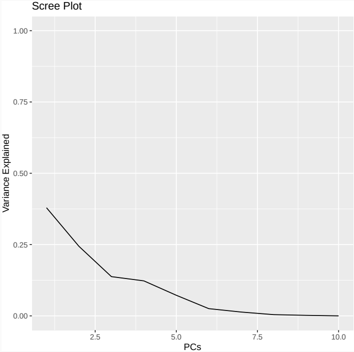


Figure 17: First two PCs for NWSL 2023 teams

Application: State-level characteristics

```r
head(state.x77)
```

```
           Population Income Illiteracy Life Exp Murder
Alabama          3615   3624        2.1    69.05   15.1
Alaska            365   6315        1.5    69.31   11.3
Arizona          2212   4530        1.8    70.55    7.8
Arkansas         2110   3378        1.9    70.66   10.1
California      21198   5114        1.1    71.71   10.3
Colorado         2541   4884        0.7    72.06    6.8
           HS Grad Frost   Area
Alabama       41.3    20  50708
Alaska        66.7   152 566432
Arizona       58.1    15 113417
Arkansas      39.9    65  51945
California    62.6    20 156361
Colorado      63.9   166 103766
```

Application: State-level characteristics

```
state_pca <- prcomp(state.x77, scale = T)
options(width = 60)
state_pca$rotation[, 1:3]
```

```
                   PC1         PC2         PC3
Population  0.12642809  0.41087417 -0.65632546
Income     -0.29882991  0.51897884 -0.10035919
Illiteracy  0.46766917  0.05296872  0.07089849
Life Exp   -0.41161037 -0.08165611 -0.35993297
Murder      0.44425672  0.30694934  0.10846751
HS Grad    -0.42468442  0.29876662  0.04970850
Frost      -0.35741244 -0.15358409  0.38711447
Area       -0.03338461  0.58762446  0.51038499
```

Application: State-level characteristics

```
biplot(state_pca)
```

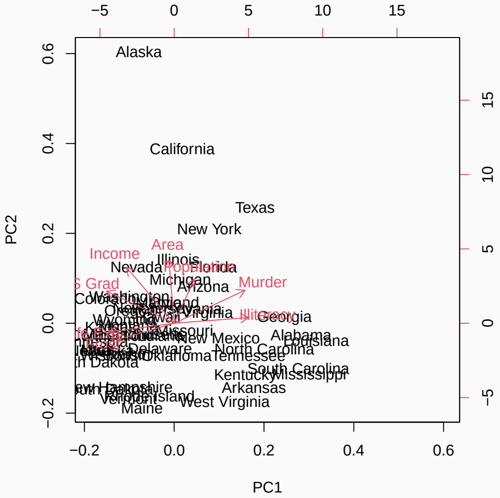


Figure 18: Biplot for state level characteristics, 1977

Application: State-level characteristics

```
plot(state_pca$rotation[,1], state_pca$rotation[,2], col = "white",
     xlim = c(-.75, .75), ylim = c(-.75, .75),
     xlab = "PC1", ylab = "PC2")
text(state_pca$rotation[,1], state_pca$rotation[,2],
     rownames(state_pca$rotation))
```

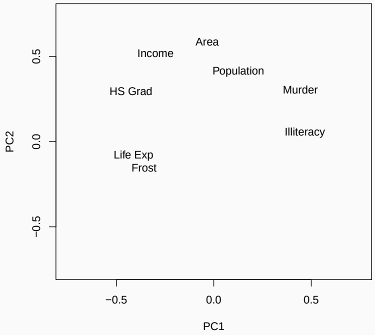


Figure 19: Variable loadings for PCA of state level characteristics, 1977

Application: State-level characteristics

```r
plot(state_pca$x[,1], state_pca$x[,2], col = "white",
     xlim = c(-4.5, 4.5), ylim = c(-2, 6),
     xlab = "PC1", ylab = "PC2")
text(state_pca$x[,1], state_pca$x[,2], rownames(state.x77))
```

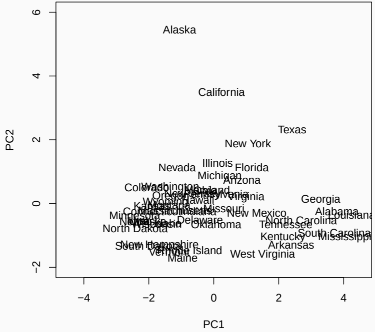


Figure 20: PCs for state level characteristics, 1977

Application: *The New York Times*

One way to turn documents into numerical data is to represent each document as a **bag of words**: a vector where each component represents the count of a particular word. These vectors are typically quite long and often **sparse**: many values are zero.

We can download a toy dataset from *the New York Times* here Shalizi (n.d.).

This dataset has 102 rows, and 4432 columns including the class label and and the rest representing the counts for every distinct word that appears in at least one of the stories.

Application: *The New York Times*

```r
head(nyt.frame)[, 1:6]
```

```
  class.labels          X.         X.d X.nd        X.s
1          art 0.008706748 0.00000000    0 0.00000000
2          art 0.005848328 0.00000000    0 0.00000000
3          art 0.016035669 0.00000000    0 0.01140303
4          art 0.026414939 0.00000000    0 0.00000000
5          art 0.007285014 0.00000000    0 0.01100835
6          art 0.002158439 0.03363435    0 0.03913930
         X.th
1 0.009251444
2 0.000000000
3 0.000000000
4 0.000000000
5 0.000000000
6 0.000000000
```

Application: *The New York Times*

```
# drop class labels
nyt.pca <- prcomp(nyt.frame[, -1])
nyt.latent.sem <- nyt.pca$rotation
head(nyt.pca$rotation[, 1:3])
```

```
                PC1          PC2          PC3
X.       0.027008304 -0.005153922 -0.044675585
X.d      0.040733115  0.002834693 -0.026691733
X.nd    -0.006573117  0.004266679 -0.007905854
X.s     -0.022760270  0.036805485  0.038662704
X.th     0.035216761  0.014050485 -0.030373464
X.this  -0.004997333  0.013267267  0.005533896
```

Application: *The New York Times*

```r
# Largest positive coordinates for the second PC
signif(sort(nyt.latent.sem[, 2], decreasing = TRUE)[1:30], 2)
```

```
      art      museum      images     artists   donations
    0.150       0.120       0.095       0.092       0.075
  museums    painting         tax   paintings   sculpture
    0.073       0.073       0.070       0.065       0.060
  gallery sculptures     painted       white    patterns
    0.055       0.051       0.050       0.050       0.047
   artist      nature     service  decorative        feet
    0.047       0.046       0.046       0.043       0.043
  digital      statue       color    computer       paris
    0.043       0.042       0.042       0.041       0.041
      war collections     diamond       stone     dealers
    0.041       0.041       0.041       0.041       0.040
```

Application: *The New York Times*

```r
# Largest negative coordinates for the second PC
signif(sort(nyt.latent.sem[, 2], decreasing = FALSE)[1:30], 2)
```

```
       her         she     theater       opera
    -0.220      -0.220      -0.160      -0.130
        ms           i        hour  production
    -0.130      -0.083      -0.081      -0.075
      sang    festival       music     musical
    -0.075      -0.074      -0.070      -0.070
     songs       vocal   orchestra          la
    -0.068      -0.067      -0.067      -0.065
   singing     matinee performance        band
    -0.065      -0.061      -0.061      -0.060
    awards   composers        says          my
    -0.058      -0.058      -0.058      -0.056
        im        play    broadway      singer
    -0.056      -0.056      -0.055      -0.052
    cooper performances
    -0.051      -0.051
```

Application: *The New York Times*

```
plot(
  nyt.pca$x[, 1:2],
  pch = ifelse(nyt.frame[, "class.labels"] == "music", "m", "a"),
  col = ifelse(nyt.frame[, "class.labels"] == "music", "blue", "red")
)
```

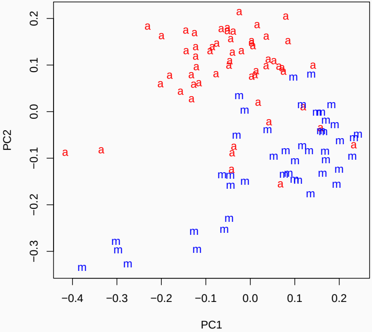


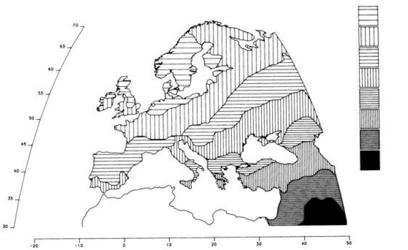
Figure 21: Projection of articles on the first two PCs.

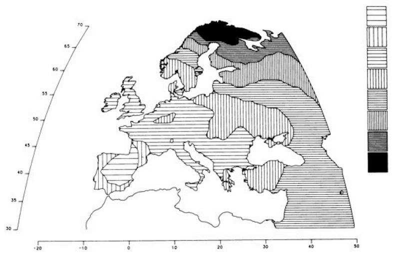
Pitfalls of PCA

Overinterpretation

- It is common to try to interpret the principal components, but it's important to be cautious. We should be wary of ”reifying” concepts.
- A key example comes from Cavalli-Sforza (1997), who describes a PCA with a data matrix where the rows represent locations and columns represent frequency of gene variants.

Cavalli-Sforza et al. (1997): Population migration from PCs?





Cavalli-Sforza et al. (1997): Population migration from PCs?

”Hidden patterns in the geography of Europe shown by the first five principal components, explaining respectively 28%, 22%, 11%, 7%, and 5% of the total genetic variation for 95 classical polymorphisms. ***The first component is almost superimposable to the archaeological dates of the spread of farming from the Middle East between 10,000 and 6,000 years ago.*** *The second principal component parallels a probable spread of Uralic people and/or languages to the northeast of Europe… ”*

Overinterpretation

Shalizi (n.d.) reviews a paper by Novembre and Stephens (2008) that points out that these kinds of patterns are expected when carrying out PCA with **any** spatially correlated data.

Novembre and Stephens simulated data based on genetic diffusion processes, without any migration/population expansion and produced similar maps.

Overinterpretation

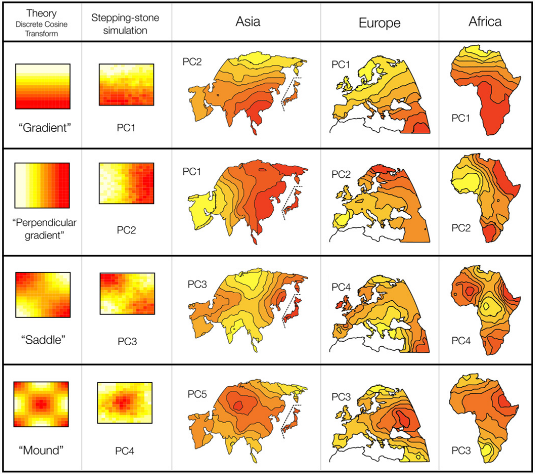


Figure 22: PCs based on simulated data with no migration

Overinterpretation

In other words, Novembre and Stephens do not disprove that migration happened, but they show that PCA of Cavalli-Sforza et al. doesn't provide strong evidence of the migration.

PCs must thus be interpreted with caution.

Linear projections

```r
theta <- runif(100, min = 0, max = 2 * pi)
x <- cos(theta)
y <- sin(theta)
plot(x, y, pch = 16)
```

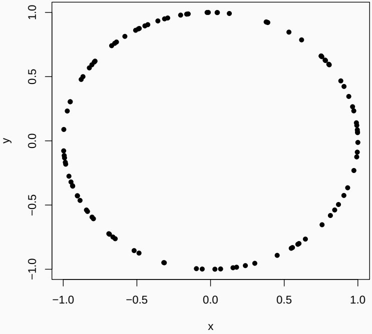


Figure 23: Circular data

Linear projections

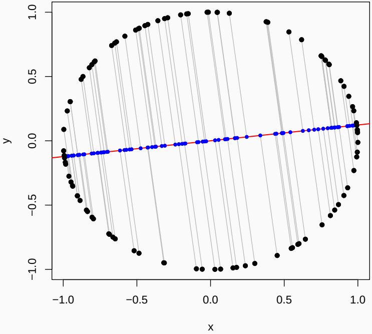


Figure 24: First PC and projection of circular data