STAT5630, Fall 2019

Dimension Reduction

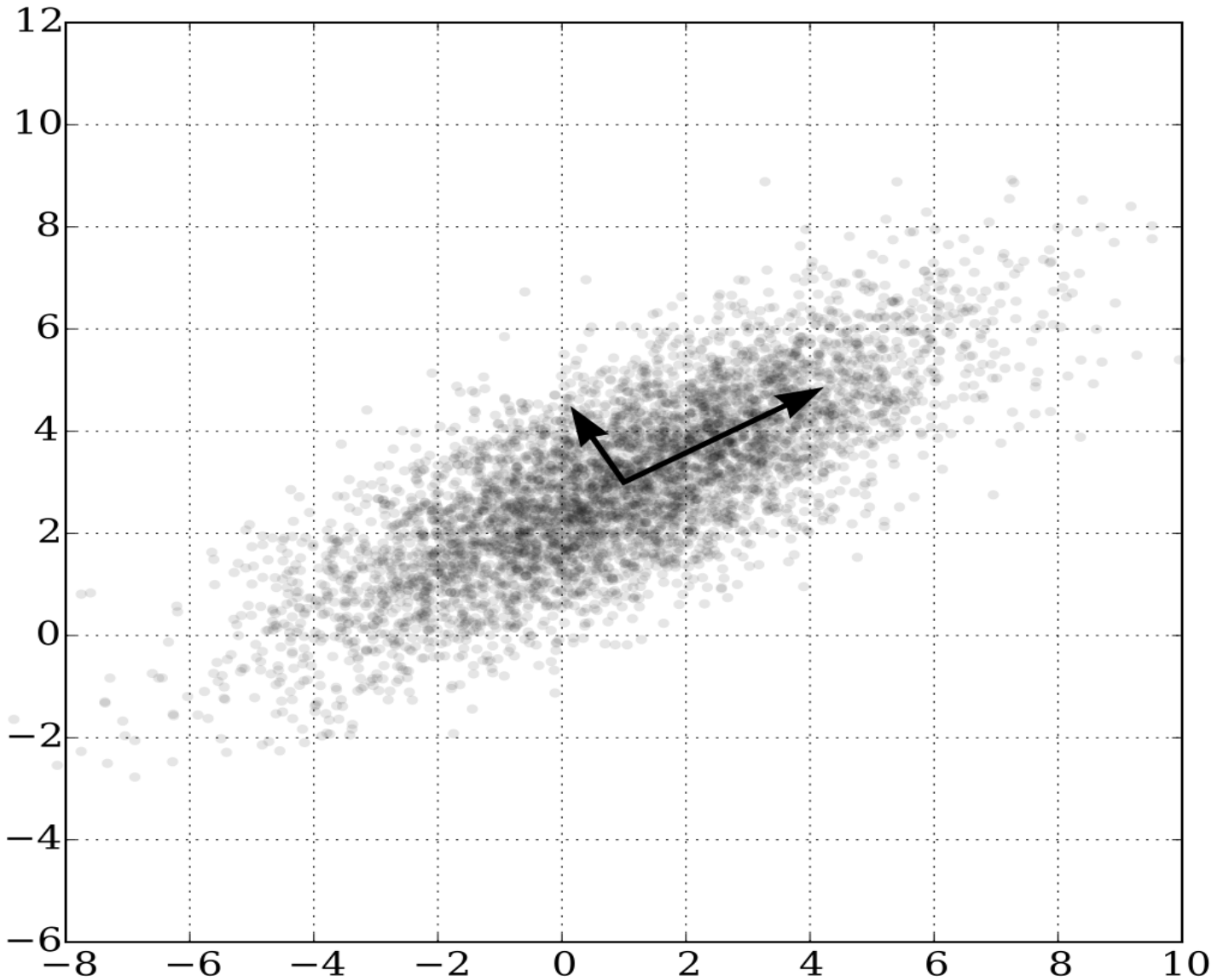
Xiwei Tang, Ph.D. <xt4yj@virginia.edu>

University of Virginia
November 7, 2019

Outline

- PCA and ICA
- Sufficient Dimension Reduction

Principal Component Analysis



Principle Component Analysis

- An old but very useful technique invented by Karl Pearson (1901)
- The main purpose is data visualization (in 2d or 3d plots)
- It also serves as a dimension reduction methods
- Unsupervised method, can be used for clustering, etc.

Principle Component Analysis Revisited

- Given that we have a $n \times p$ design matrix $\mathbf{X}$, there are many equivalent approaches (motivations):
  - Explain the most variation: Produce a derived set of uncorrelated variables $\mathbf{Z}_k = \mathbf{X}\boldsymbol{\alpha}_k$, $k = 1, \ldots, q < p$ that are linear combinations of the original variables, and that explain most of the variation in the original set, $\boldsymbol{\alpha}_k$ is a p-by-1 loading vector
  - Approximate the design matrix: Approximate the design matrix $\mathbf{X}$ by the best (using Frobenius norm) rank-q matrix, which can be performed through SVD

Eigen-Decomposition and Singular Value Decomposition

- Sample covariance matrix $\widehat{\Sigma} = \mathbf{X}^\mathsf{T}\mathbf{X}/(n-1)$ can be diagonalize

$$\widehat{\Sigma} = \mathbf{V}\mathbf{D}^*\mathbf{V}^\mathsf{T},$$

  where orthonormal columns of $\mathbf{V}$ are principle directions (loadings) and projecting $\mathbf{X}$ on these loadings gives the principal components

- On the other hand, if we decompose $\mathbf{X}$ into

$$\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^\mathsf{T},$$

  we can rewrite $\widehat{\Sigma}$ as

$$\widehat{\Sigma} = \mathbf{V}\mathbf{D}\mathbf{U}^\mathsf{T}\mathbf{U}\mathbf{D}\mathbf{V}^\mathsf{T}/(n-1) = \mathbf{V}\frac{\mathbf{D}^2}{n-1}\mathbf{V}^\mathsf{T}$$

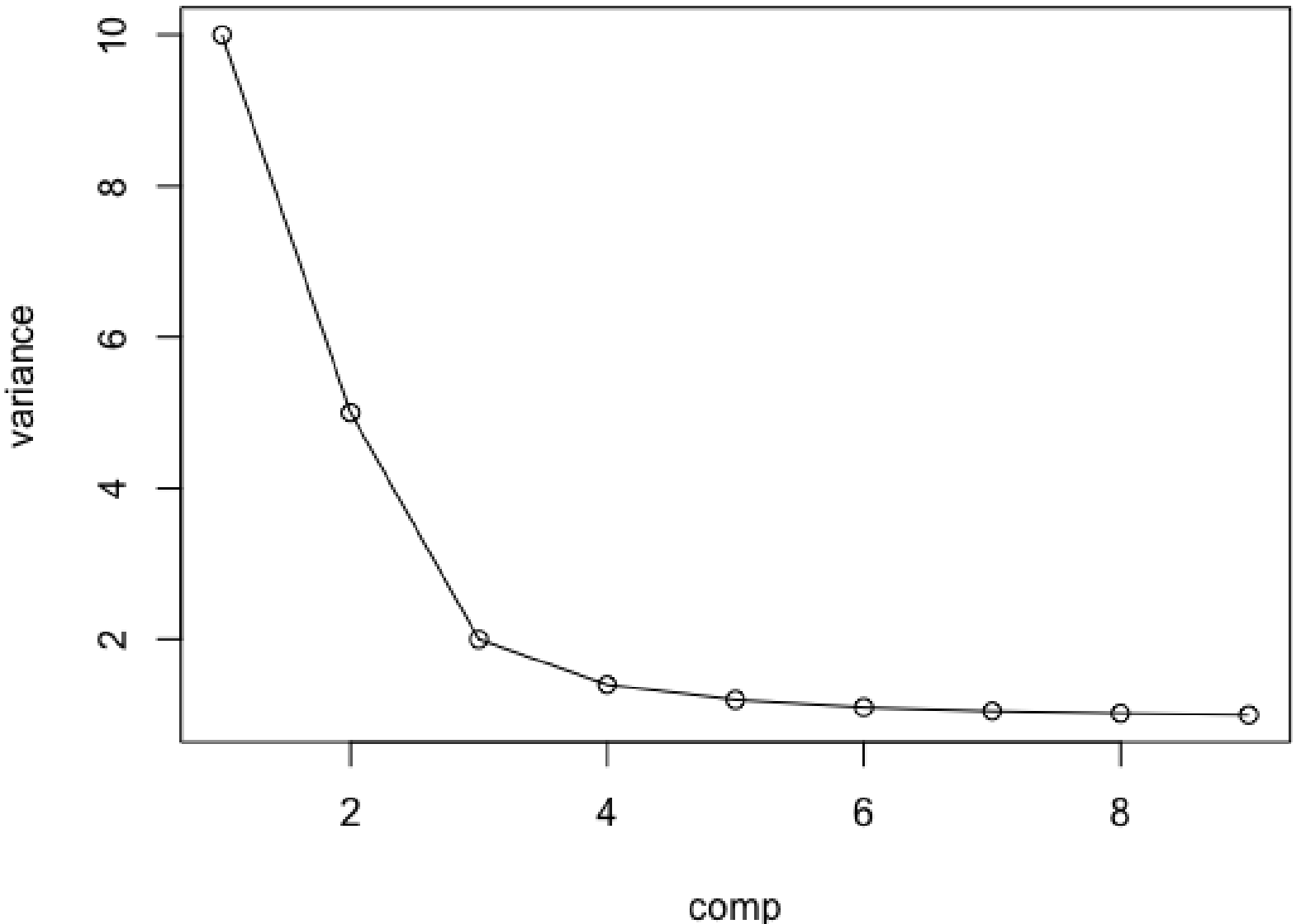
- So the right singular vectors $\mathbf{V}$ of $\mathbf{X}$ are just the principle directions, and the principal components are

$$\mathbf{X}\mathbf{V} = \mathbf{U}\mathbf{D}\mathbf{V}^\mathsf{T}\mathbf{V} = \mathbf{U}\mathbf{D}$$

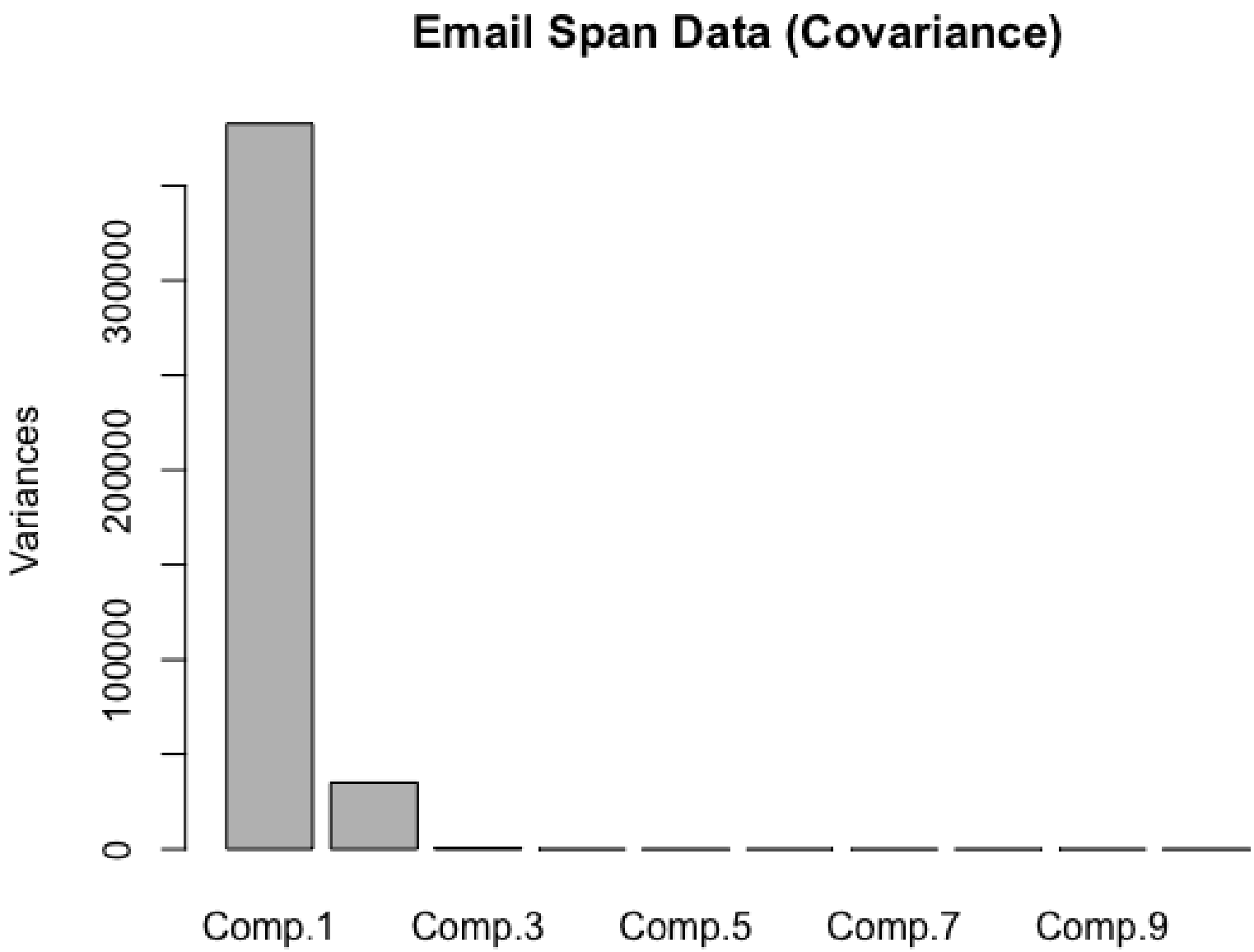
Principle Component Analysis Review

- Dimension Reduction: If we are searching for one linear combination, say α such that $\mathbf{X}\alpha$ has the largest variance, we will choose the first column of $\mathbf{V}$ as α
- The columns of $\mathbf{XV}$ are corresponding principle components, and are orthogonal with each other
- PCA is usually performed by centering $\mathbf{X}$ first (column-wise, i.e., by each variable).
- Be careful to select either covariance matrix or correlation matrix in PCA
- However, PCA is unsupervised, i.e., the directions does not reflect the relationship with the response.

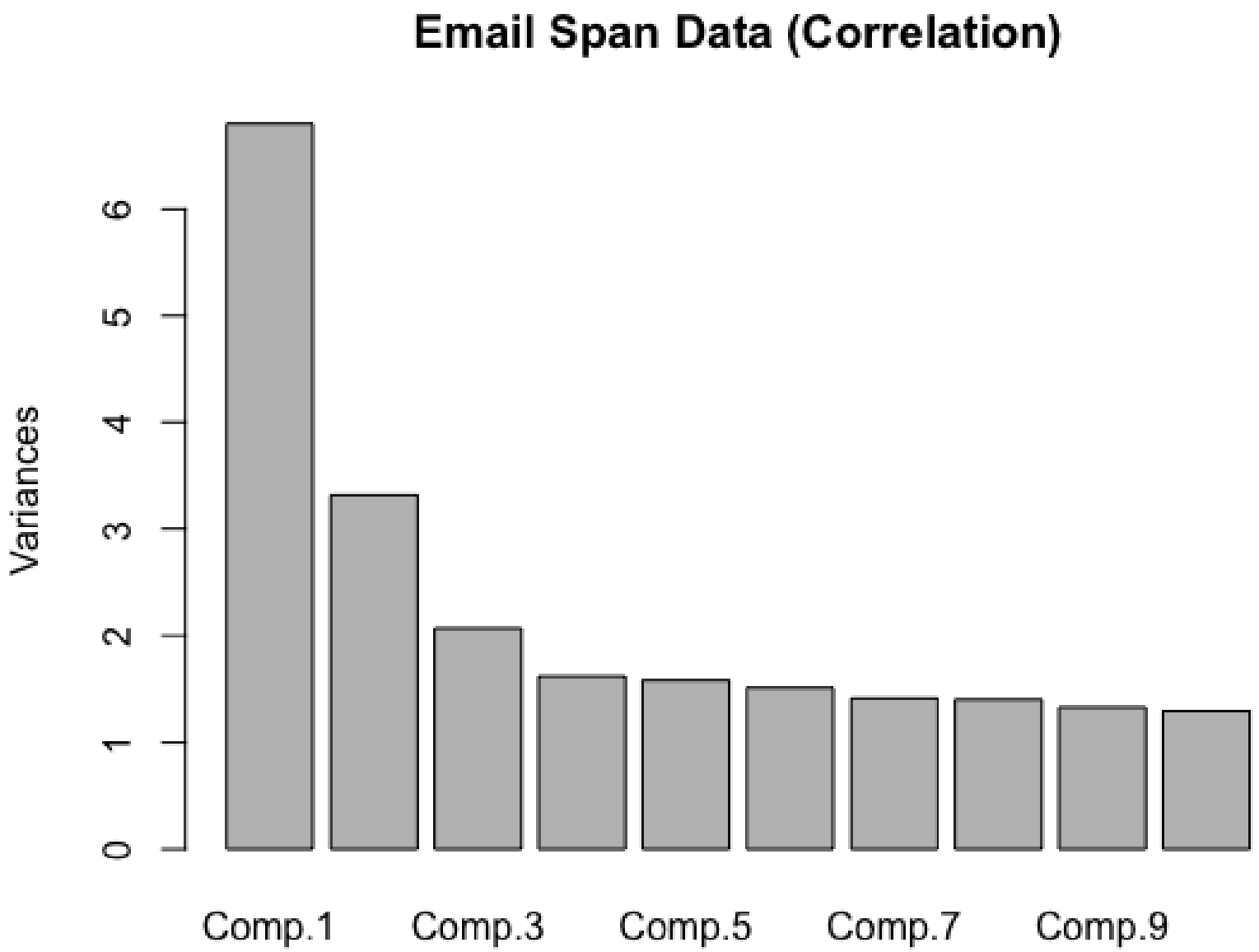
How to Select Number of Components



Covariance vs Correlation



Covariance vs Correlation



Denoising

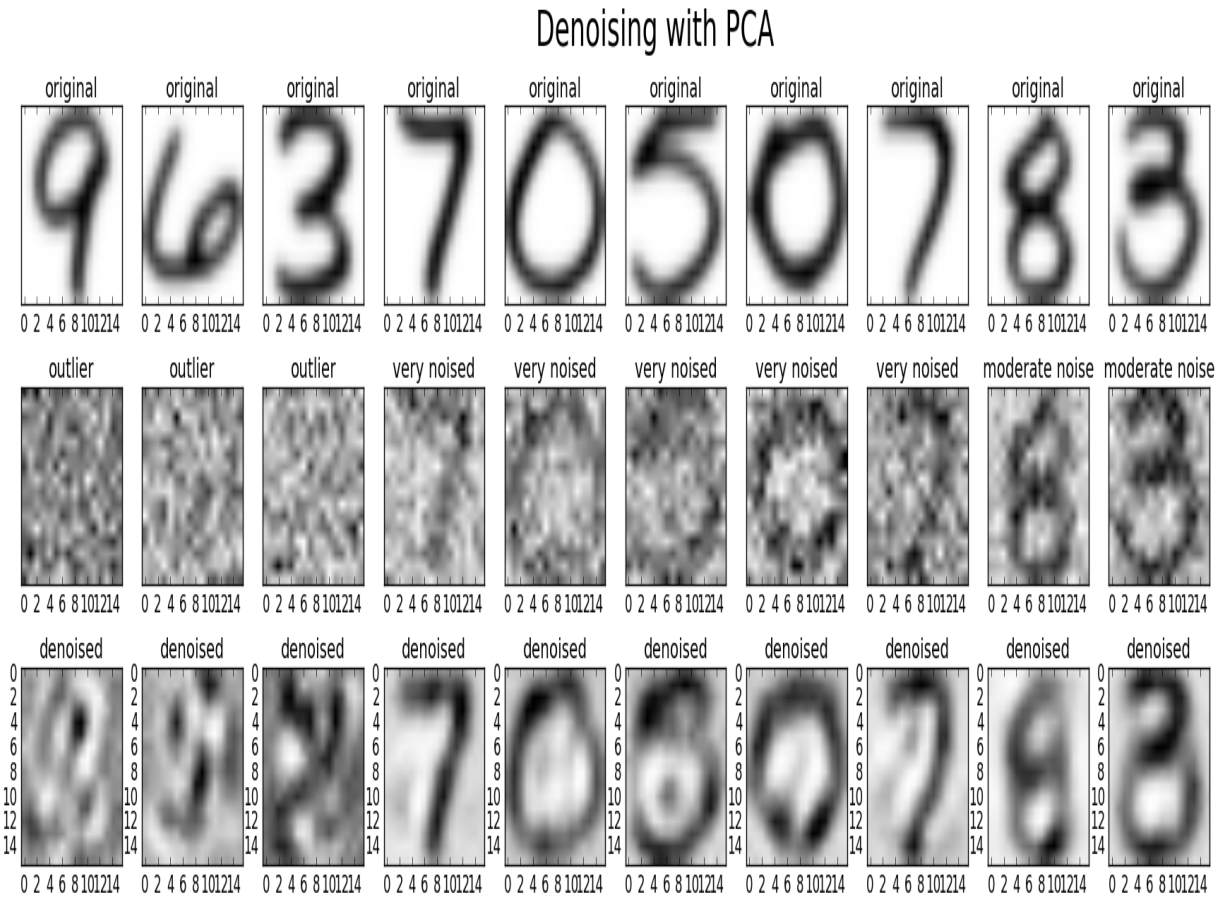

Denoising with PCA
original
outlier
very noised
moderate noise
denoised

Image Compression

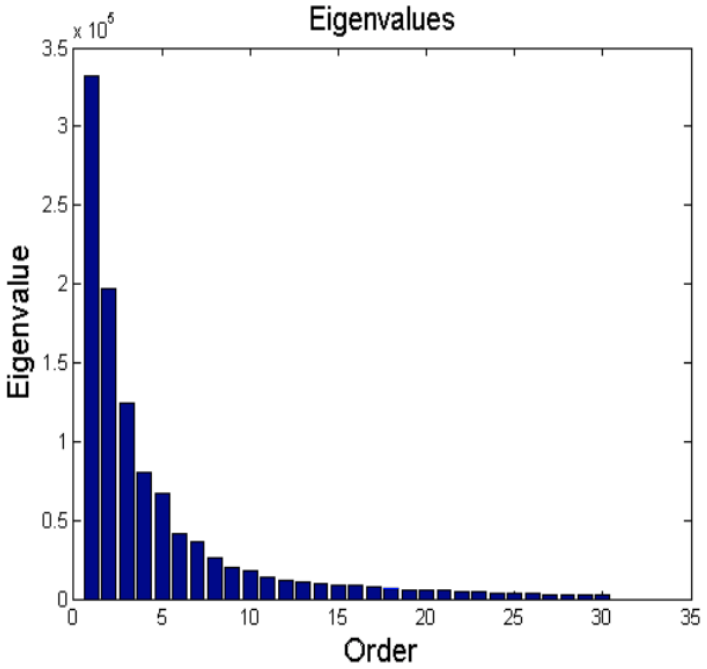
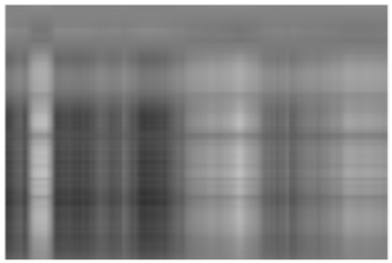


Image Compression



(a) 1 principal component

(b) 5 principal component

(c) 9 principal component

(d) 13 principal component

(e) 17 principal component

(f) 21 principal component

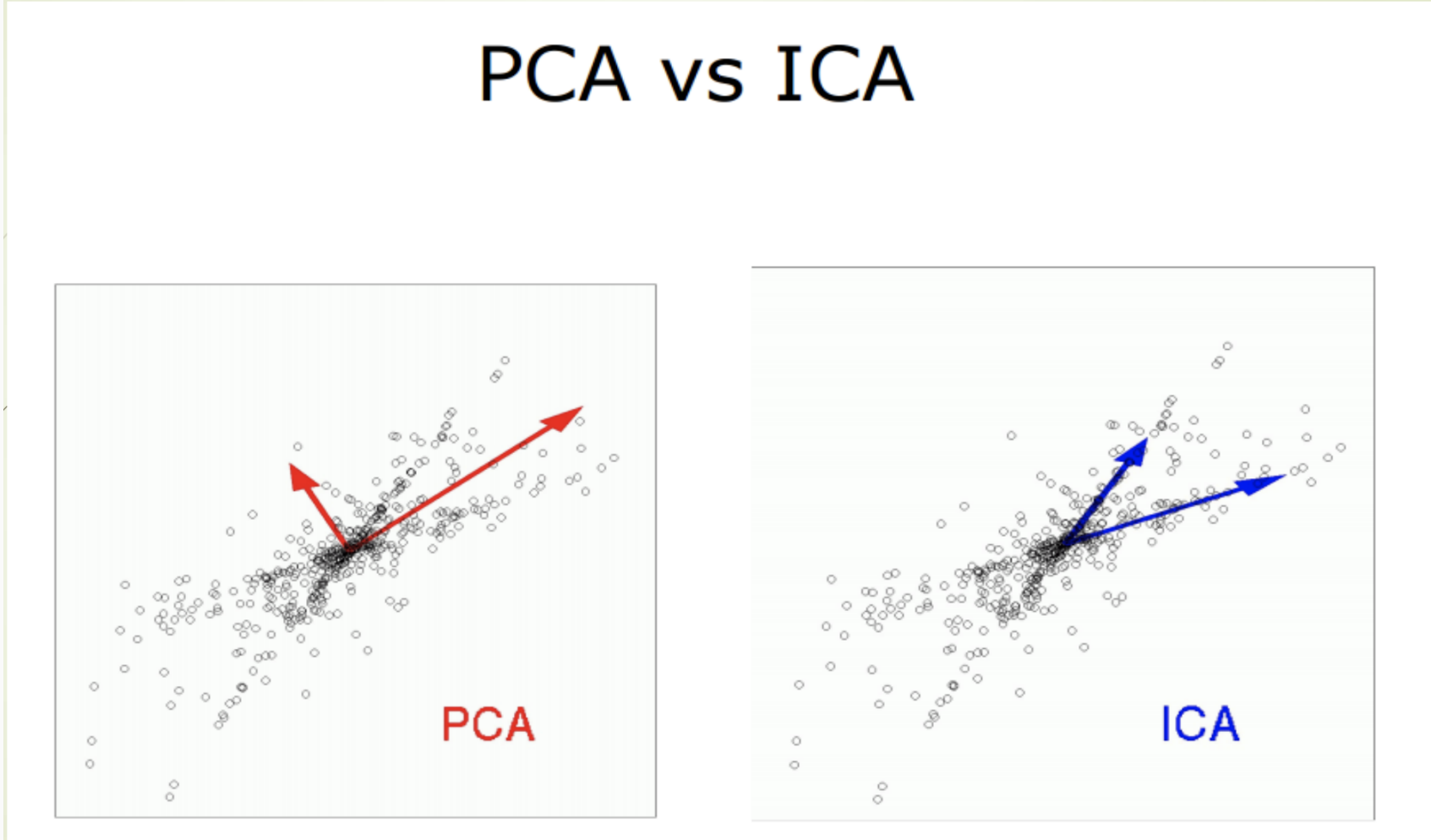
(g) 25 principal component

(h) 29 principal component

Independent Component Analysis (ICA)

- Let $\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_d$ denote the projection directions of independent components
- ICA: find these directions such that data projected onto these directions have maximum statistical independence
- How to maximize independence ?
  - Minimize the mutual information
  - Maximize the non-Gaussianity

ICA vs PCA



Sufficient Dimension Reduction

A more flexible model

- Consider the model structure:

$$Y = m(X^\mathsf{T}\boldsymbol{\beta}_1, \ldots, X^\mathsf{T}\boldsymbol{\beta}_K, \epsilon) \tag{1}$$

where

- $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_K$ are unknown projection vectors,
- K is (usually) unknown and assumed to be much less than p.
- m is an unknown function $\mathbb{R}^{K+1} \to \mathbb{R}$,
- ϵ is a noise random variable with $E(\epsilon|X) = 0$.

A more flexible model

- This is much more flexible than many models that we have considered so far. Some special cases:
  - Linear model: $K = 1$, $m(a, \epsilon) = a + \epsilon$
  - Single index model: $Y = g(X^\mathsf{T}\boldsymbol{\beta}) + \epsilon$
  - Heteroscedastic linear model: $Y = X^\mathsf{T}\boldsymbol{\beta}_1 + g(X^\mathsf{T}\boldsymbol{\beta}_2)\epsilon$
  - Neural networks, projection pursuit: $Y = \sum_{k=1}^{K} \alpha_k f_k(X^\mathsf{T}\boldsymbol{\beta}_k) + \epsilon$
- Even in the most naive case, we can use $K = p$.

Sufficient Dimension Reduction: Inverse Regression

- When the structural dimension K is much less than p, there is a benefit.
  - We could first estimate the space $\mathbf{S}_{Y|X}$ without estimating the regression function $m(\cdot)$
  - Then perform a K dimensional nonparametric regression on the reduced dimension.
- Instead of regressioning Y on X, lets try a new strategy, regressing X on Y, and see how the curve of $E(X|Y = y)$ (a p-dimensional curve) moves in a p-dimensional space.

Sliced Inverse Regression Algorithm

1. Obtain the centered and standardized variable

$$Z = \Sigma_x^{-1/2}(X - \mathsf{E}(X))$$

2. Sort the dataset $\{Z_i, y_i\}_{i=1}^n$ by the y_i values. Divide the dataset into H slices (according to the sorted values of y_i') as equally as possible.
3. Within each slice h, compute the slice sample mean of Z, i.e.,

$$\overline{z}_h = n_h^{-1} \sum_{i \in \text{slice } h} z_i,$$

where n_h is the number of observations in slice h.

Sliced Inverse Regression Algorithm

4. Estimate Cov(E($Z|Y$)): Compute the estimation matrix using the slice means of Z, weighted by the slice sizes:

$$\mathbf{M} = n^{-1} \sum_{h=1}^{H} n_h \overline{z}_h \overline{z}_h^{\mathsf{T}}$$

5. Perform eigen-decomposition on $\mathbf{M}$ and obtain the K largest eigenvectors $\widehat{\boldsymbol{\alpha}}_1, \ldots, \widehat{\boldsymbol{\alpha}}_K$, then transform them back to $\widehat{\Sigma}_x^{-1/2}\widehat{\boldsymbol{\alpha}}_1, \ldots, \widehat{\Sigma}_x^{-1/2}\widehat{\boldsymbol{\alpha}}_K$.

The Inverse Regression Curve

```
> library(rgl)
> library(akima)
> library(dr)
> # generate some data with two directions
> n = 500; p = 10
> x = matrix(rnorm(n*p), n, p)
> b = matrix(c(1, 1, rep(0, p-2)))
> y = 0.125*(x %*% b)^3 + 0.5*rnorm(n)
> # visualize the data
> plot3d(x[,1], x[, 2], y, col="red", size=5)
> s = interp(x[,1], x[, 2], 0.125*(x %*% b)^3)
> surface3d(s$x, s$y, s$z, col = 'gray', alpha = 0.4, add = T)
> fit.sir = dr(y~., data = data.frame(x, y), method = "sir")
```

$E(Y|X) = 0.125(X_1 + X_2)^3$ is a nonlinear function, direct modeling of this conditional mean function is difficult. Lets consider the inverse curve, $E(X|Y = y)$, as a function of y.

The Inverse Regression Curve