# Session 11: Distances, SVD, and Principal Components Analysis

Levi Waldron

# Session 11 outline

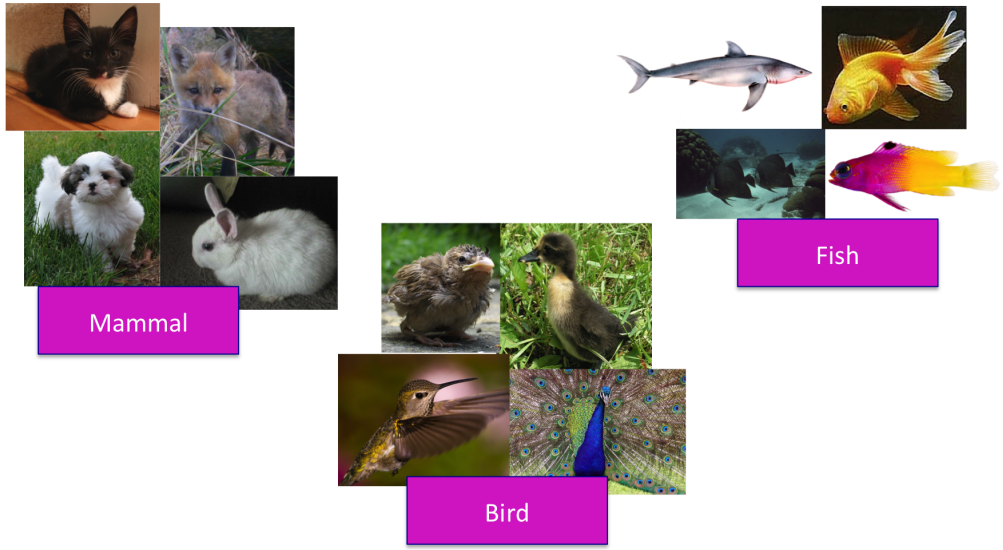
- Distances in high dimensions
- Singular Value Decomposition
- Principal Components Analysis

Extra reading: `http://genomicsclass.github.io/book/` (Chapter 8)

# Distances in high-dimensional data analysis

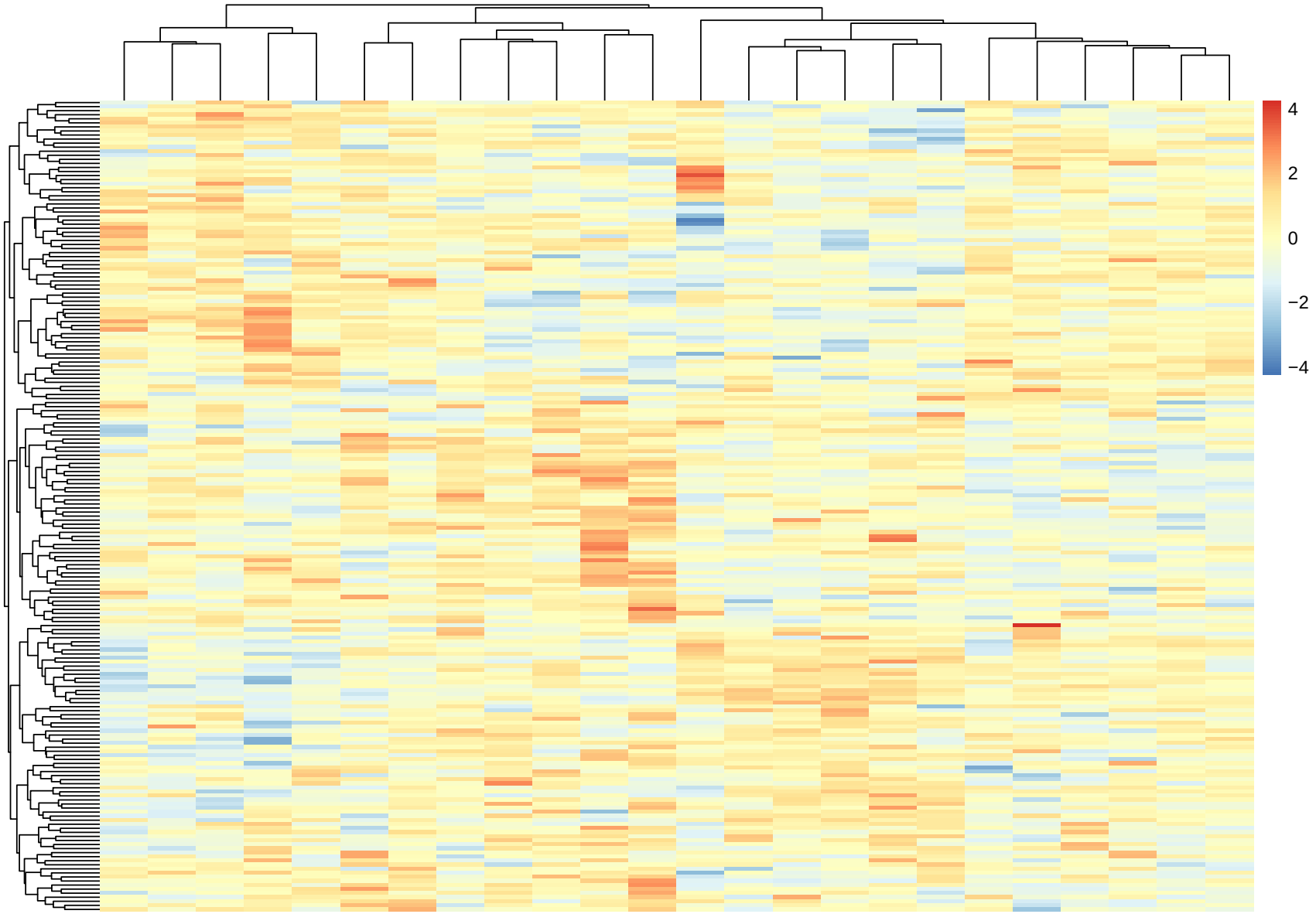
# The importance of distance

- High-dimensional data are complex and impossible to visualize in raw form
- We can only visualize 2-3 dimensions
- Distances can simplify thousands of dimensions



# The importance of distance (cont'd)

- Distances can help organize samples and variables



# The importance of distance (cont'd)

- Any clustering or classification of samples and/or genes involves combining or identifying objects that are close or similar.
- Distances or similarities are mathematical representations of what we mean by close or similar.
- The choice of distance is important and requires thought.
  - choice is subject-matter specific

Source:
`http://master.bioconductor.org/help/course-materials/2002/Summer02Course/Distance/distance.pdf`

# Metrics and distances

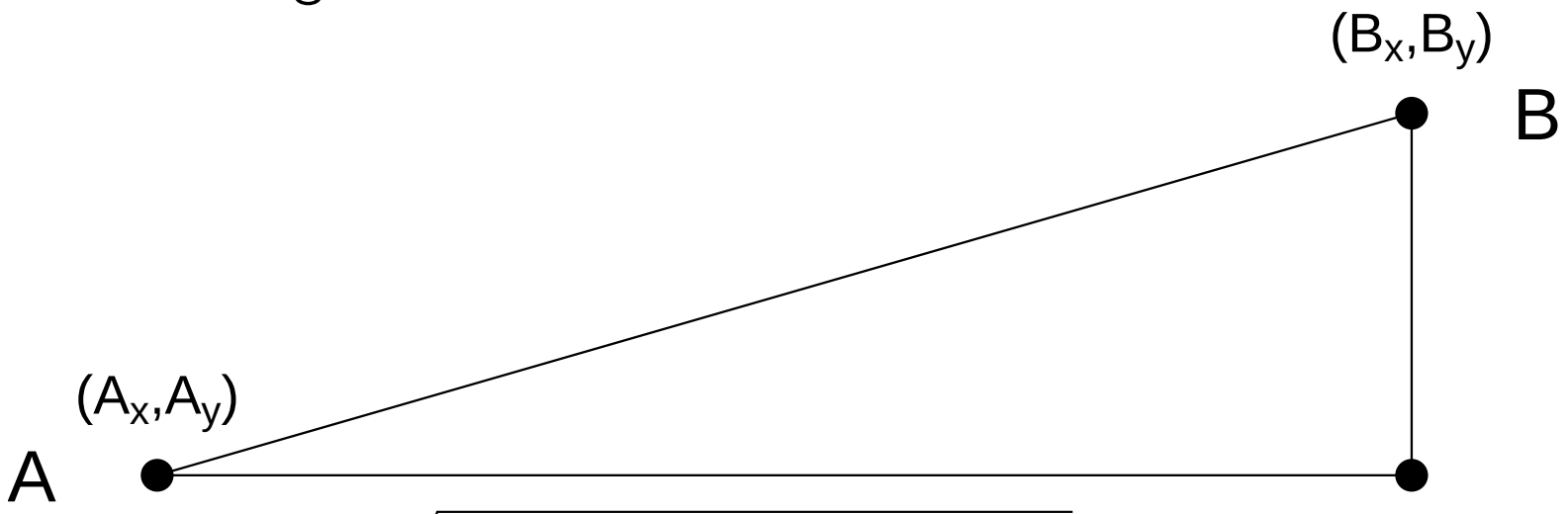
A **metric** satisfies the following five properties:

1. non-negativity $d(a, b) \geq 0$
2. symmetry $d(a, b) = d(b, a)$
3. identification mark $d(a, a) = 0$
4. definiteness $d(a, b) = 0$ if and only if $a = b$
5. triangle inequality $d(a, b) + d(b, c) \geq d(a, c)$

- A **distance** is only required to satisfy 1-3.
- A **similarity function** satisfies 1-2, and **increases** as $a$ and $b$ become more similar
- A **dissimilarity function** satisfies 1-2, and **decreases** as $a$ and $b$ become more similar

# Euclidian distance (metric)

- Remember grade school:




Euclidean d $= \sqrt{(A_x - B_x)^2 + (A_y - B_y)^2}$.

- **Side note**: also referred to as $L_2$ *norm*

# Euclidian distance in high dimensions

Consider the expression of thousands of genes in hundreds of tissues:

```
##biocLite("genomicsclass/tissuesGeneExpression")
library(tissuesGeneExpression)
data(tissuesGeneExpression)
dim(e) ##gene expression data
```

```
## [1] 22215   189
```

```
table(tissue) ##tissue[i] corresponds to e[,i]
```

```
## tissue
##  cerebellum       colon endometrium hippocampus      kidney        li
##          38          34          15          31          39
##    placenta
##           6
```

# Euclidian distance in high dimensions

- Points are no longer on the Cartesian plane,
- instead they are in higher dimensions. For example:
  - sample $i$ is defined by a point in 22,215 dimensional space: $(Y_{1,i}, \ldots, Y_{22215,i})^\top$.
  - feature $g$ is defined by a point in 189 dimensions $(Y_{g,189}, \ldots, Y_{g,189})^\top$

# Euclidian distance in high dimensions

Euclidean distance as for two dimensions. E.g., the distance between two samples $i$ and $j$ is:

$$\text{dist}(i,j) = \sqrt{\sum_{g=1}^{22215} (Y_{g,i} - Y_{g,j})^2}$$

and the distance between two features $h$ and $g$ is:

$$\text{dist}(h,g) = \sqrt{\sum_{i=1}^{189} (Y_{h,i} - Y_{g,i})^2}$$

# Matrix algebra notation

The distance between samples $i$ and $j$ can be written as:

$$\text{dist}(i,j) = \sqrt{(\mathbf{Y}_i - \mathbf{Y}_j)^\top (\mathbf{Y}_i - \mathbf{Y}_j)}$$

with $\mathbf{Y}_i$ and $\mathbf{Y}_j$ columns $i$ and $j$.

# Matrix algebra notation

```r
t(matrix(1:3, ncol=1))
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
```

```r
matrix(1:3, ncol=1)
```

```
##      [,1]
## [1,]    1
## [2,]    2
## [3,]    3
```

```r
t(matrix(1:3, ncol=1)) %*% matrix(1:3, ncol=1)
```

```
##      [,1]
## [1,]   14
```

# 3 sample example

```r
kidney1 <- e[, 1]
kidney2 <- e[, 2]
colon1 <- e[, 87]
sqrt(sum((kidney1 - kidney2)^2))
```

```
## [1] 85.8546
```

```r
sqrt(sum((kidney1 - colon1)^2))
```

```
## [1] 122.8919
```

# 3 sample example using dist()

```r
dim(e)
```

```
## [1] 22215   189
```

```r
(d <- dist(t(e[, c(1, 2, 87)])))
```

```
##                  GSM11805.CEL.gz GSM11814.CEL.gz
## GSM11814.CEL.gz          85.8546
## GSM92240.CEL.gz         122.8919        115.4773
```

```r
class(d)
```

```
## [1] "dist"
```

# The dist() function

Excerpt from ?dist:

```
dist(x, method = "euclidean", diag = FALSE,
     upper = FALSE, p = 2)
```

- **method:** the distance measure to be used.
  - This must be one of "euclidean", "maximum", "manhattan", "canberra", "binary" or "minkowski". Any unambiguous substring can be given.
- `dist` class output from `dist()` is used for many clustering algorithms and heatmap functions

*Caution*: `dist(e)` creates a 22215 x 22215 matrix that will probably crash your R session.

# Note on standardization

- In practice, variables are typically "standardized", *i.e.* converted to z-score
  - This is done to equalize the contributions of each variable to distance

$$x_{gi} \leftarrow \frac{(x_{gi} - \bar{x}_g)}{s_g}$$

- Note: Euclidian distance and Pearson correlation ($r$) are related:
  - $\frac{d_E(x,y)^2}{2m} = 1 - r_{xy}$

# Dimension reduction and PCA

# Motivation for dimension reduction

Simulate the heights of twin pairs:

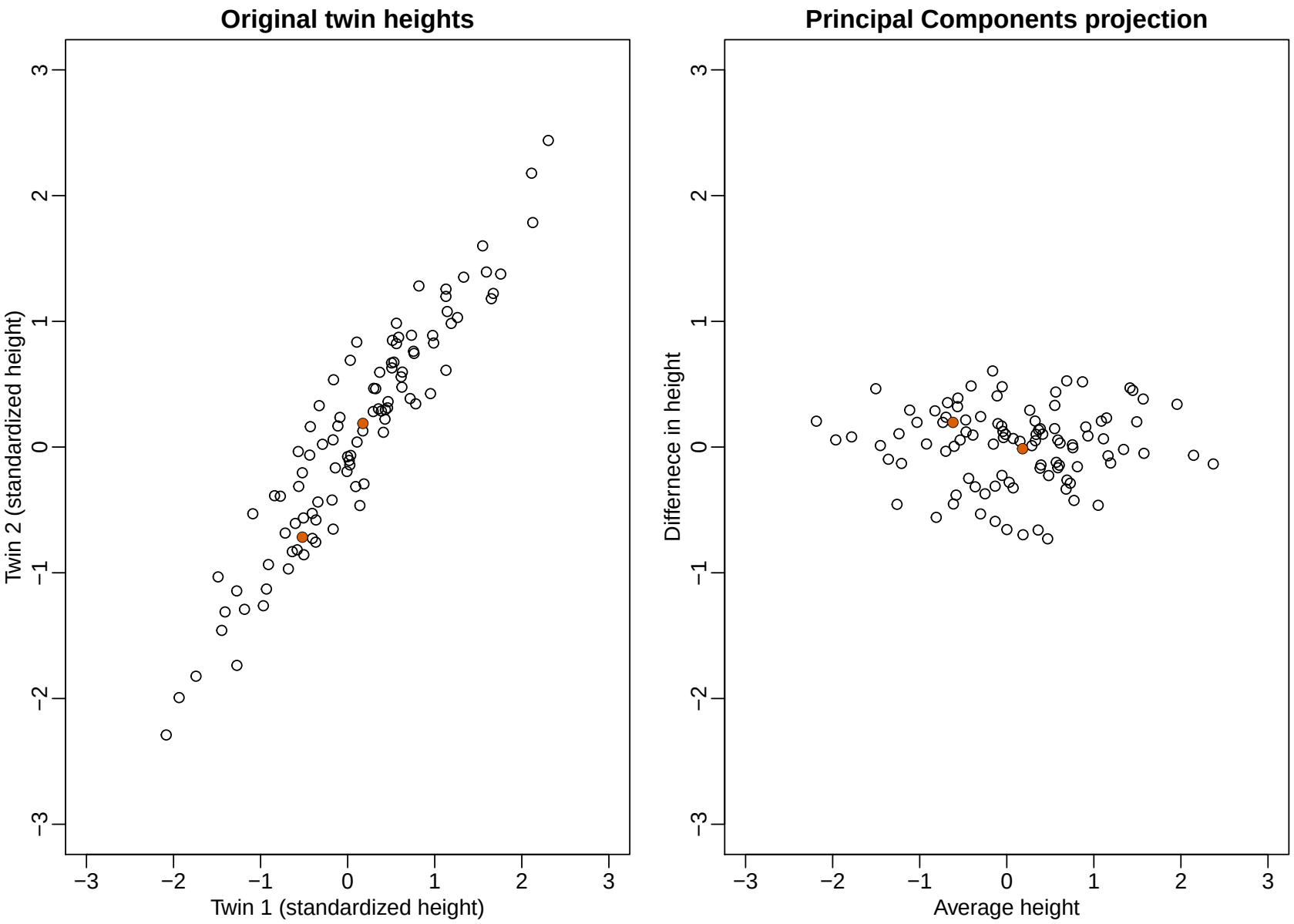
```r
dim(y)
```

```
## [1]   2 100
```

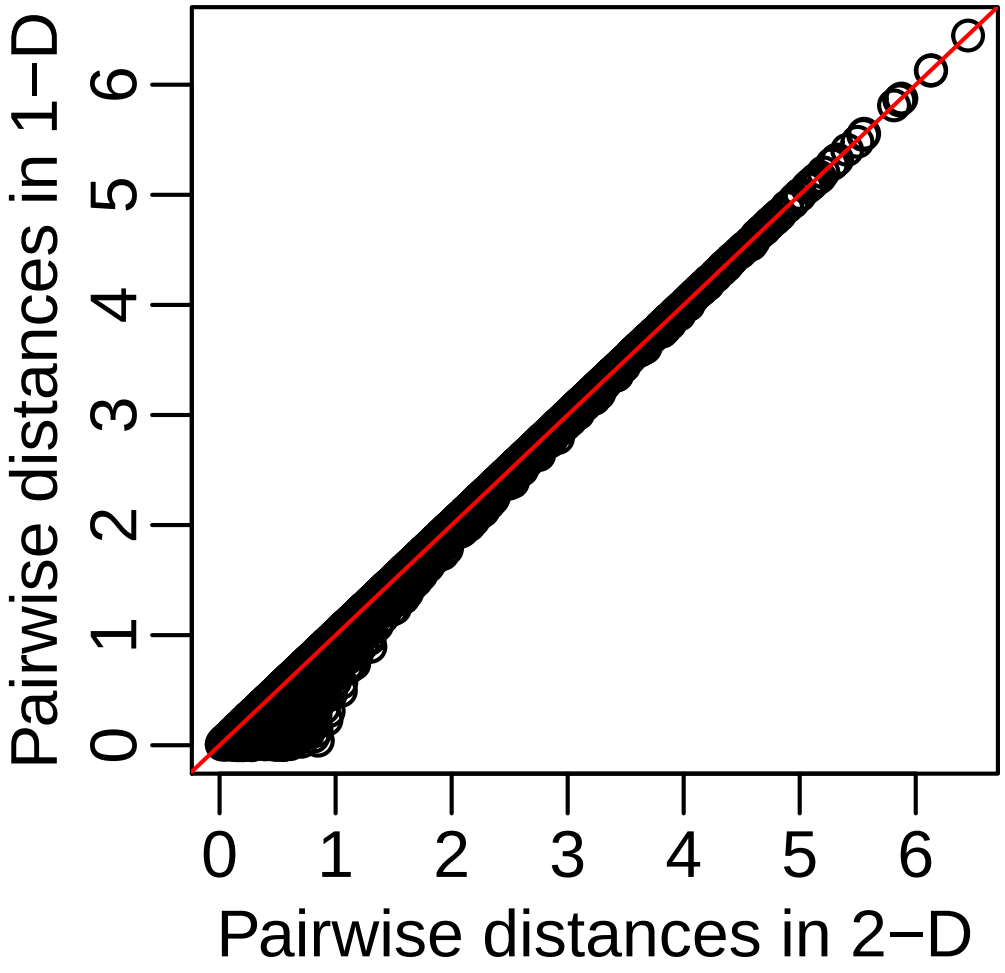
```r
cor(t(y))
```

```
##           [,1]      [,2]
## [1,] 1.0000000 0.9433295
## [2,] 0.9433295 1.0000000
```

# Motivation for dimension reduction



# Motivation for dimension reduction

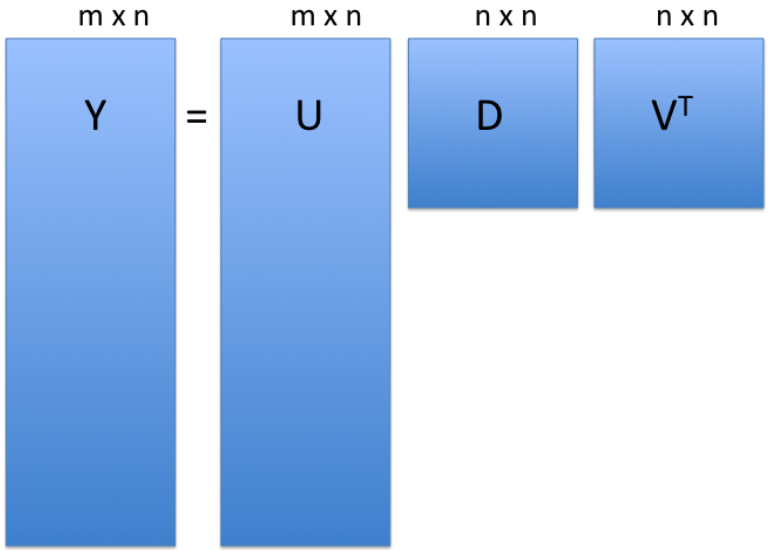



- Not much loss of height differences when just using average heights of twin pairs.
  - because twin heights are highly correlated

# Singular Value Decomposition (SVD)

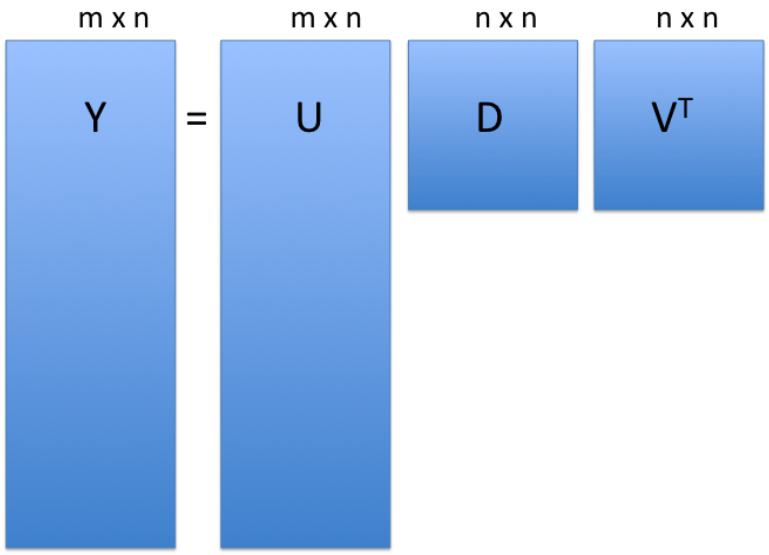
SVD generalizes the example rotation we looked at:

$$\mathbf{Y} = \mathbf{U}\mathbf{D}\mathbf{V}^\top$$




- **note**: the above formulation is for $m > n$

# Singular Value Decomposition (SVD)




- **Y**: the m rows x n cols matrix of measurements
- **U**: m x n *orthogonal* matrix (**scores**)
  - orthogonal = unit length and "perpendicular" columns
- **D**: n x n diagonal matrix (**eigenvalues**)
- **V**: n $\times$ n orthogonal matrix (**eigenvectors or loadings**)

# SVD of gene expression dataset

```r
e.standardize.fast <- t(scale(t(e), scale=FALSE))
s <- svd(e.standardize.fast)
names(s)
```

```
## [1] "d" "u" "v"
```

# SVD of gene expression dataset

```r
dim(s$u)      # loadings
```

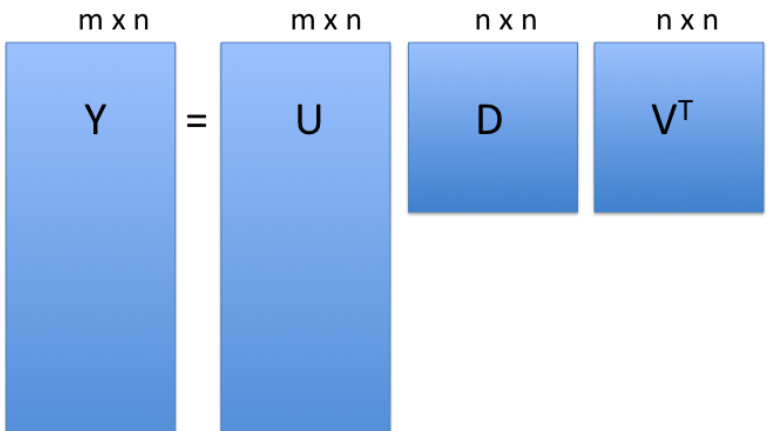
```
## [1] 22215   189
```

```r
length(s$d)  # eigenvalues
```

```
## [1] 189
```

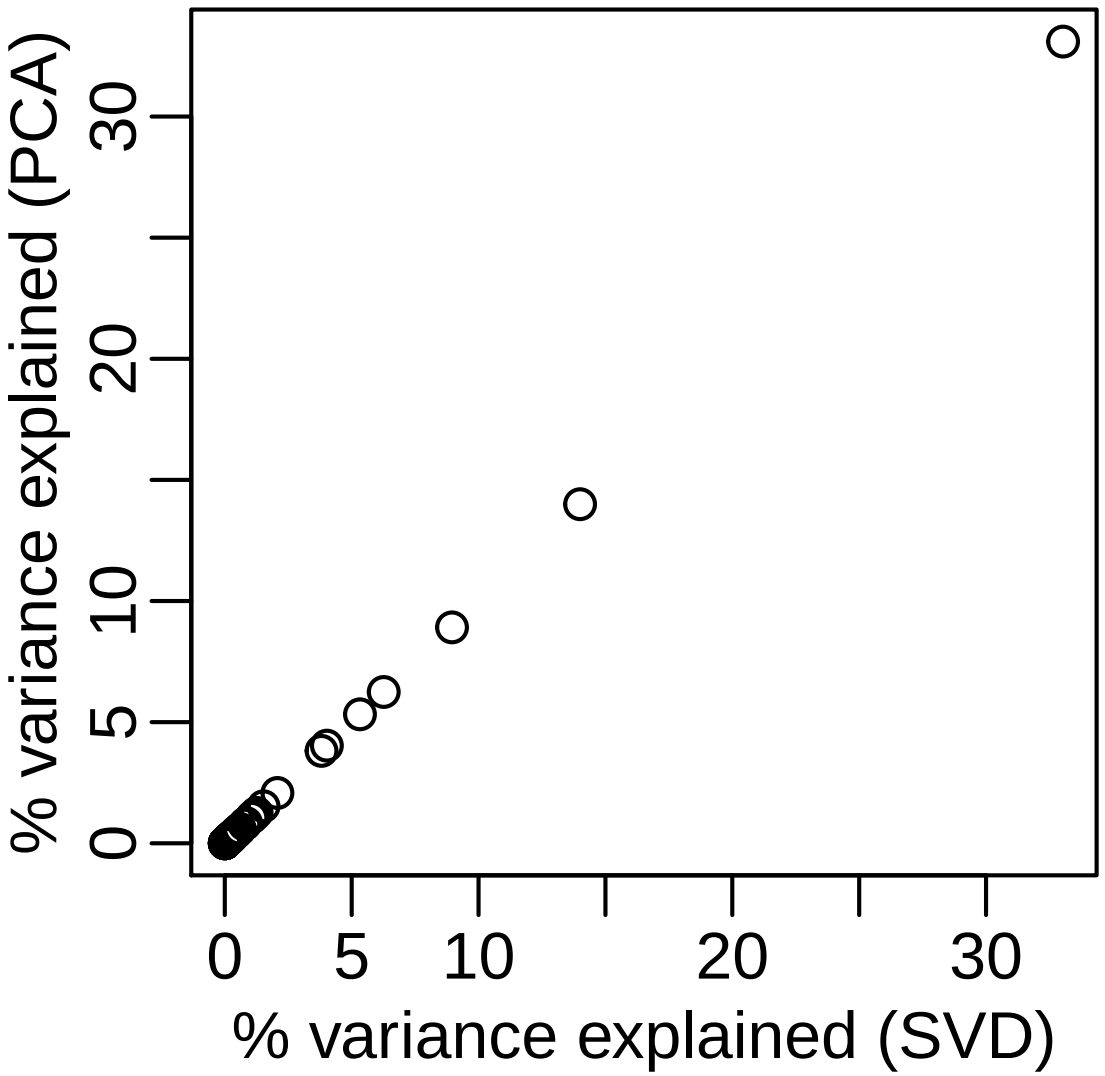
```r
dim(s$v)      # d %*% vT = scores
```

```
## [1] 189 189
```

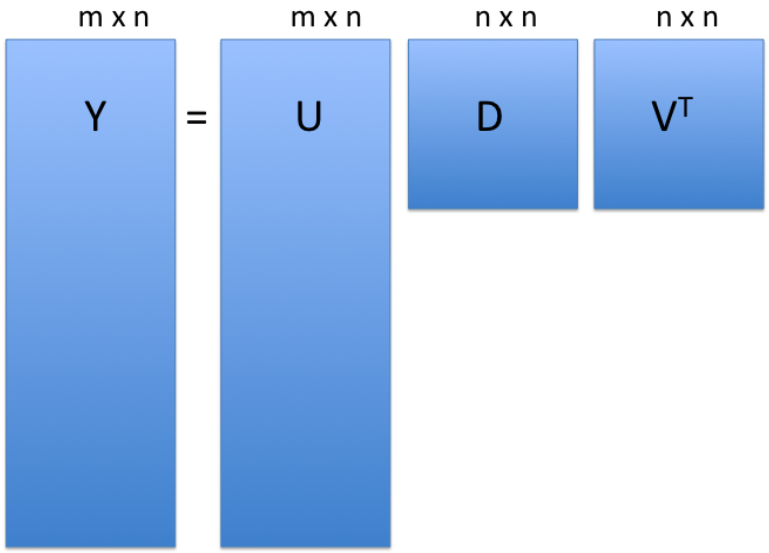


# PCA of gene expression dataset

```
p <- princomp(e.standardize.fast)
```

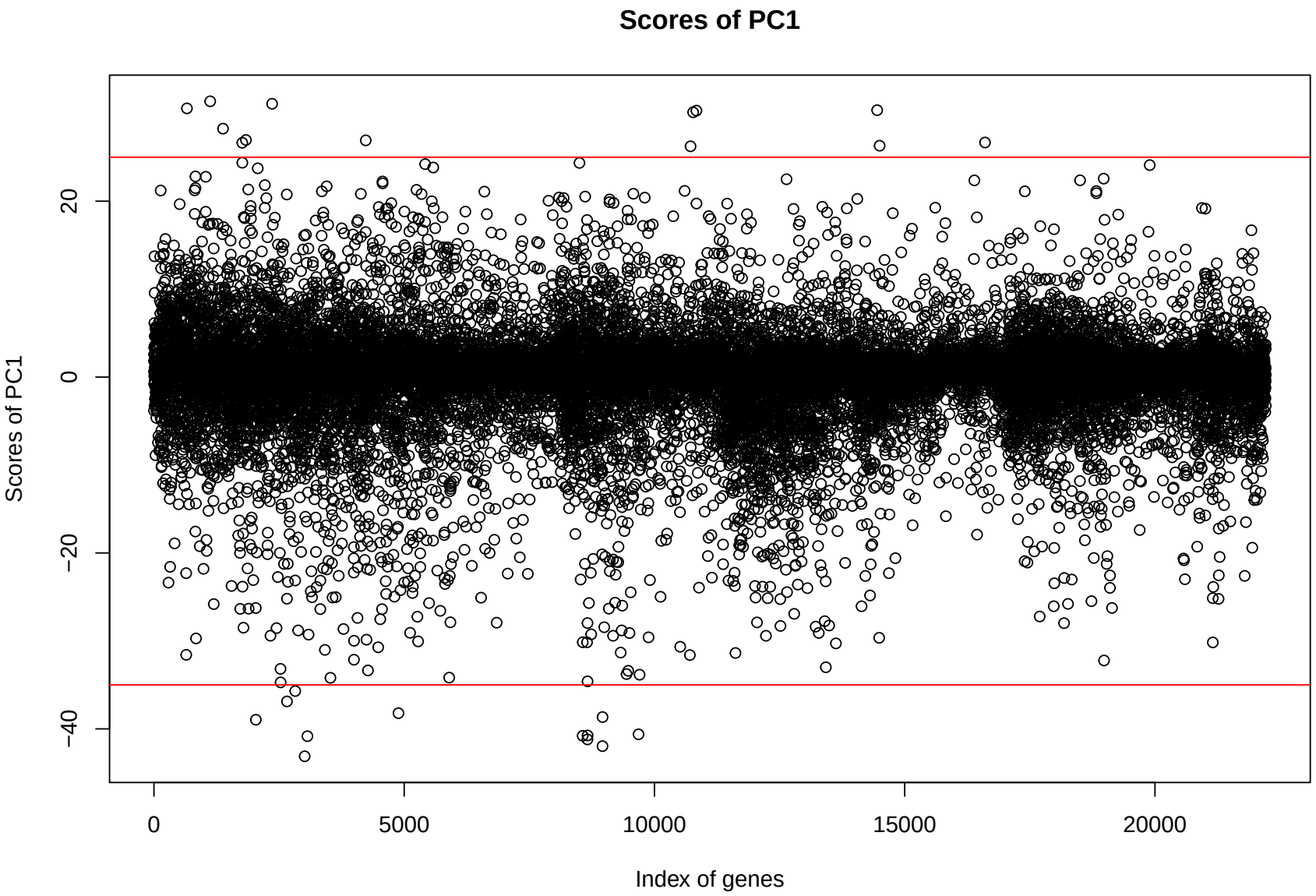


# PCA interpretation: scores



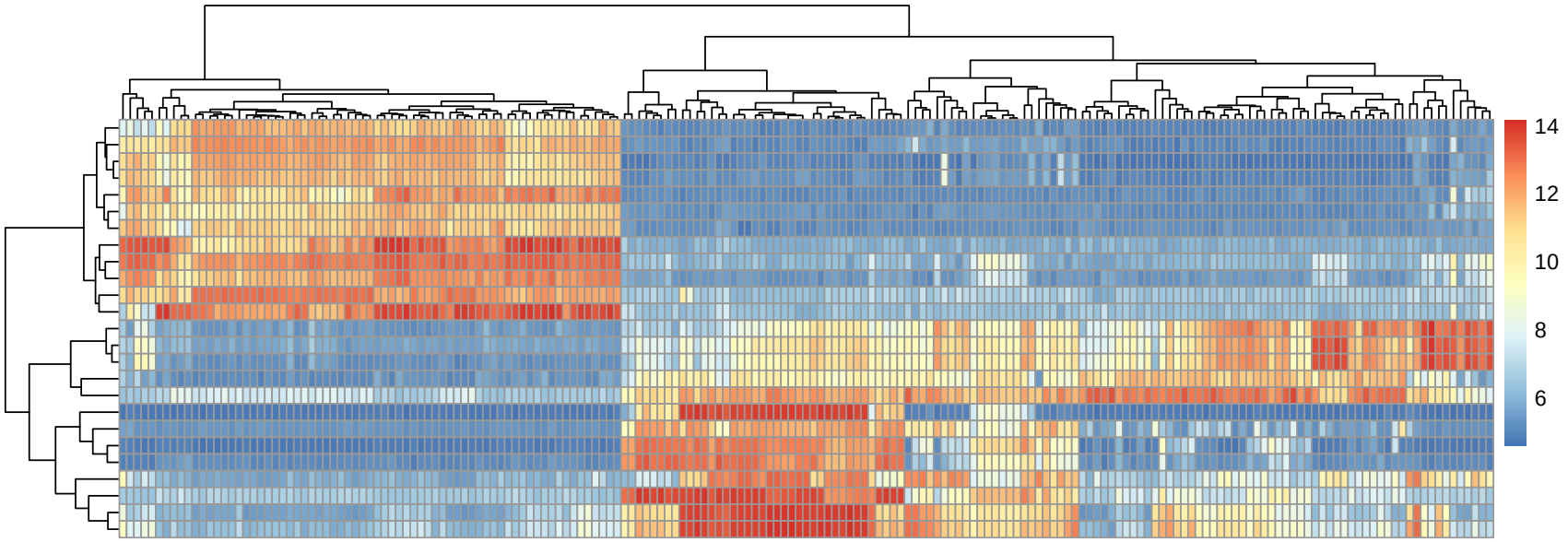

- **U** (**scores**): relate the *PCA* axes to original variables
  - think of principal component axes as a weighted combination of original axes

# PCA interpretation: scores



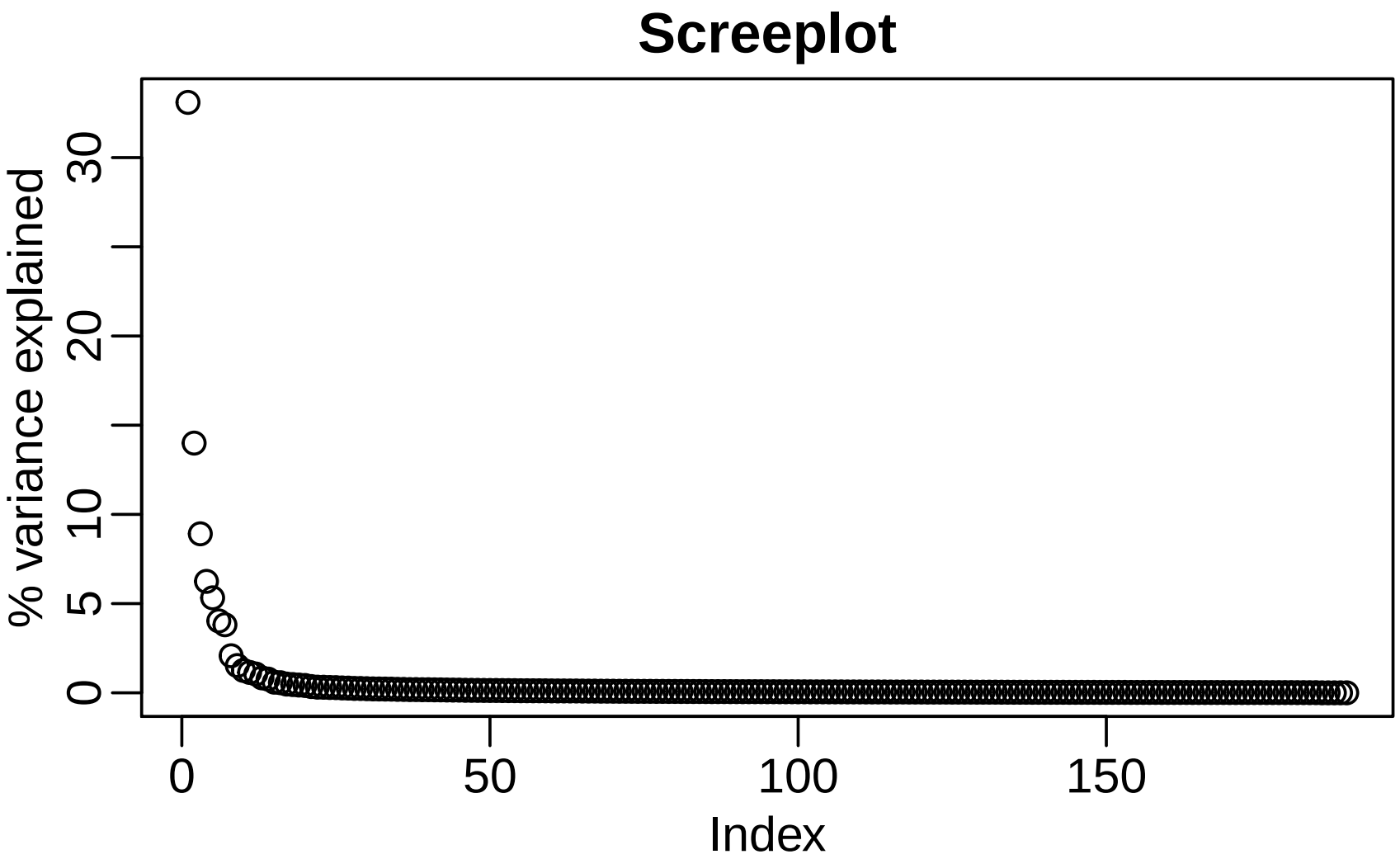
# PCA interpretation: scores

Genes with high PC1 scores:



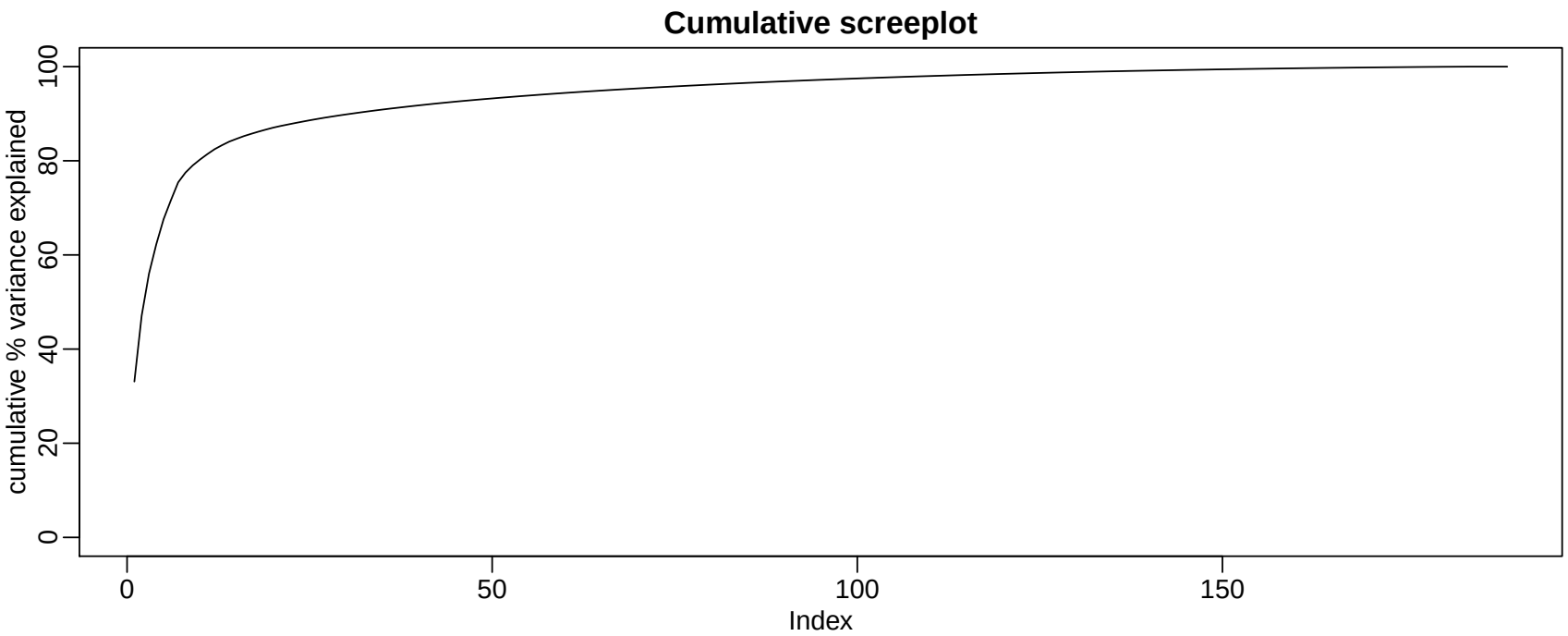
# PCA interpretation: eigenvalues

- **D** (**eigenvalues**): standard deviation scaling factor that each decomposed variable is multiplied by.

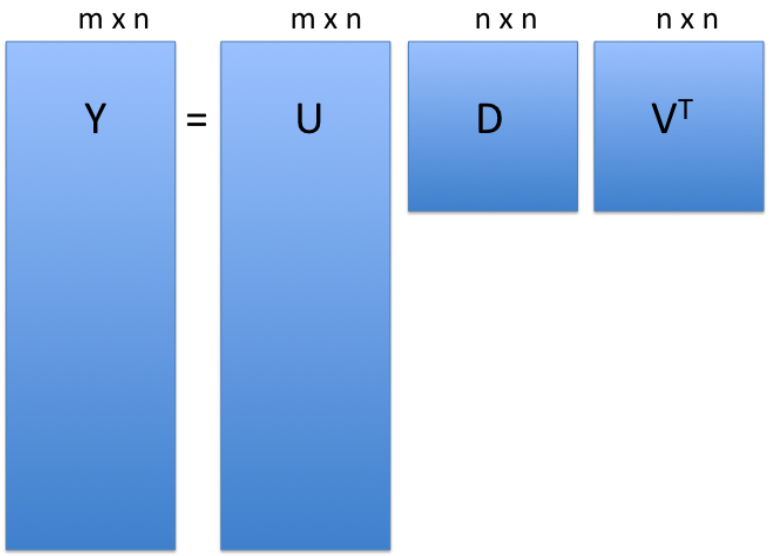


# PCA interpretation: eigenvalues

Alternatively as cumulative % variance explained (using `cumsum()` function):

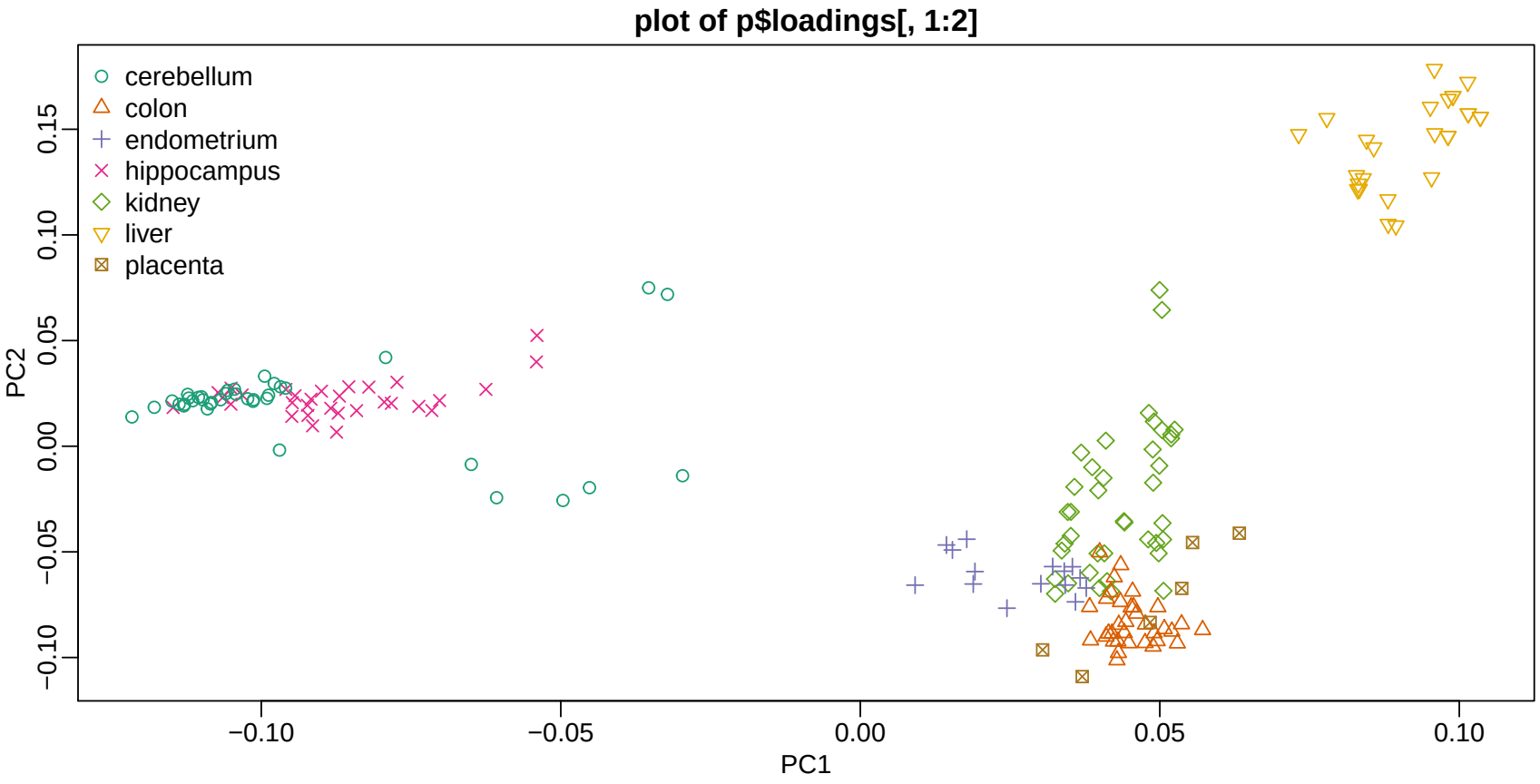


# PCA interpretation: loadings




- **V** (**loadings**): The "datapoints" in the reduced prinipal component space

# PCA interpretation: loadings



# Conclusions

- **Note**: signs of eigenvalues (square to get variances) and eigenvectors (loadings) can be arbitrarily flipped
- PCA is useful for dimension reduction when you have *correlated* variables
- Variables are always centered.
- Variables are also scaled unless you know they have the same scale in the population
- PCA projection can be applied to new datasets if you know the matrix calculations
- PCA is subject to over-fitting, screeplot can be tested by cross-validation