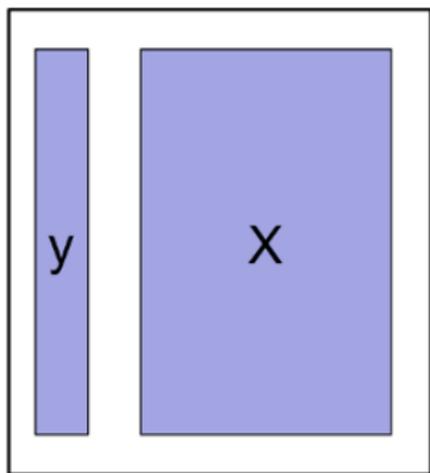# PQHS 471
Lecture 2: Unsupervised Learning (1)
Dimension Reduction
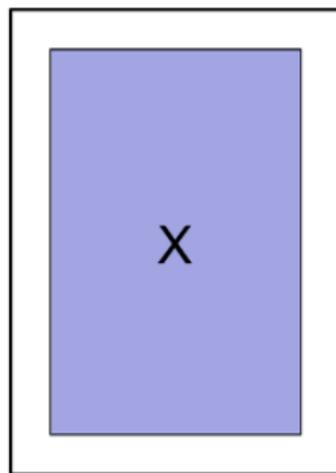
Recap: Supervised vs. Unsupervised

$\mathbf{X}$: independent variables, predictors, explanatory variables
$\mathbf{y}$: dependent variables, outcomes, response variables

Supervised Learning   Unsupervised Learning

Unsupervised Learning

- *unsupervised learning:* we observe on the features $X_1, X_2, ..., X_p$. We are not interested in prediction, because we do NOT have an associated response variable Y.

- *supervised learning:* we observe both a set of features $X_1, X_2, ..., X_p$ for each object, as well as a response or outcome varaible Y. The goal is then to predict Y using $X_1, X_2, ..., X_p$.

- In this semester, we will start with *unsupervised learning* and move on to *supervised learning* later.

The Goals of Unsupervised Learning

- Discover interesting things about the measurements: is there an informative way to visualize the data? Can we discover subgroups among the variables or among the observations?
- We will mainly discuss:
  - *Dimension Reduction:* a set of tools used for data visualization or data pre-processing (typically before supervised techniques are applied).
  - *Clustering:* a broad class of methods for discovering unknown subgroups in data.

The Challenge of Unsupervised Learning

- No simple goal (such as prediction of a response), more subjective

The Challenge of Unsupervised Learning

- No simple goal (such as prediction of a response), more subjective
- But techniques for unsupervised learning are of growing importance in a number of field:
  - personalized medicine: subgroups of colon cancer patients grouped by their gene expression values $\rightarrow$ targeted therapy.
  - marketing: groups of shoppers characterized by their browsing and purchase histories.
  - recommendation system: recommend movies to movie viewers grouped by their movie ratings/likes/dislikes

The Challenge of Unsupervised Learning

- No simple goal (such as prediction of a response), more subjective
- But techniques for unsupervised learning are of growing importance in a number of field:
  - personalized medicine: subgroups of colon cancer patients grouped by their gene expression values $\rightarrow$ targeted therapy.
  - marketing: groups of shoppers characterized by their browsing and purchase histories.
  - recommendation system: recommend movies to movie viewers grouped by their movie ratings/likes/dislikes
- One advantage: unlabeled data are easier to obtain than labeled data
  - unlabeled data: lab instrument, online via the cloud, internet of things(**IoT**), "big data"
  - labeled data: human intervention, disease diagnosis, cancer types, etc.

Dimension Reduction

The purpose of dimension reduction:

- data simplification
- data visualization
- reduce noise (if we can assume only the dominating dimensions are signal)
- variable selection for prediction

Principal Components Analysis (PCA)

- PCA is one of most classic algorithms for dimension reduction.
- PCA produces a **low-dimensional representation** of a dataset. It finds a sequence of linear combinations of the variables that have **maximal variance**, and are mutually uncorrelated.
- Apart from producing derived variables for use in supervised learning problems in potential later steps, PCA also serves as a tool for data visualization.

PCA — 1st principal component

- The first principal component of a set of features $X_1, X_2, ..., X_p$ is the normalized linear combination of the features

$$z_1 = \phi_{11}X_1 + \phi_{21}X_2 + ... + \phi_{p1}X_p$$

that has the largest variance. By normalized, we mean $||\boldsymbol{\phi_1}|| = 1$ (i.e. $\sum_{j=1}^{p} \phi_{j1}^2 = 1$)

- We refer to the elements $\phi_{11}, \phi_{21}, ..., \phi_{p1}$ as the loadings of the first principal component; together, the loadings make up the principal component loading vector

$$\boldsymbol{\phi_1} = (\phi_{11}, \phi_{21}, ..., \phi_{p1})^{\mathrm{T}}$$

- We constrain the loadings so that their sum of squares is equal to 1, since otherwise arbitrarily large loadings simply result in arbitrarily large variance.

PCA: example
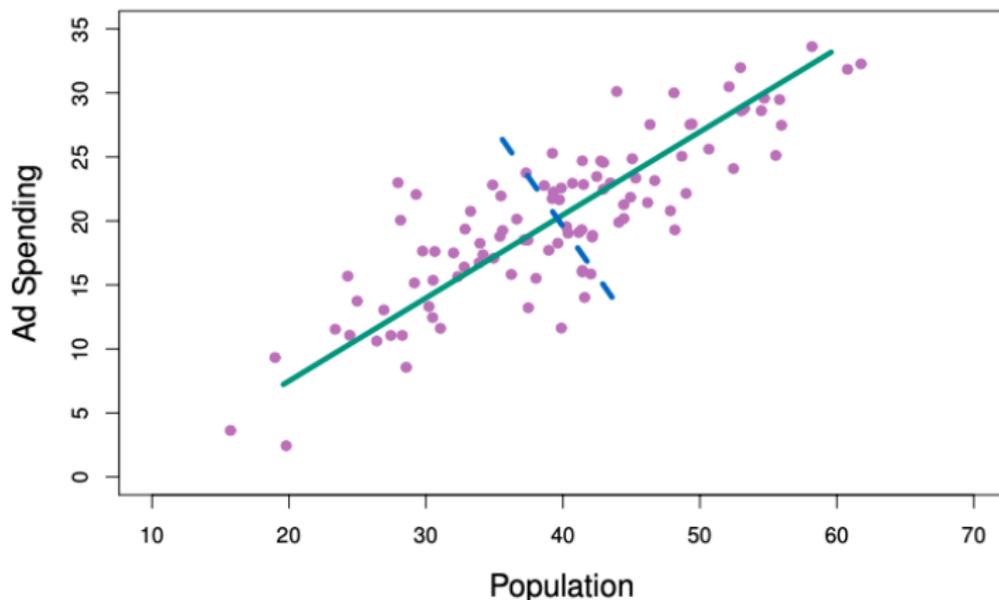
- x-axis: population, y-axis: ad spending
- "new x-axis": green solid line (i.e. 1st principal component)
- "new y-axis": blue dashed line (i.e. 2nd principal component)

Computation of PCA

- Suppose we have a $n \times p$ dataset $\mathbf{X}$.
- PCA: Explain the variance-covariance structure among a set of random variables by a few linear combinations of the variables.
- Treating the p columns as random variables, we have a $p \times 1$ random vector $\mathbf{x}$
- $cov(\mathbf{x}) = E[(\mathbf{x} - E[\mathbf{x}])(\mathbf{x} - E[\mathbf{x}])^{\mathrm{T}}] = \Sigma_{p \times p}$
    - $\Sigma_{p \times p}$ is the variance-covariance matrix of $\mathbf{X}$
    - $\Sigma_{p \times p}$ is symmetric and positive-definite (p.d.)

- First PC:
  $\phi_1^{\mathrm{T}} \mathbf{X}^{\mathrm{T}}$ that maximize $var(\phi_1^{\mathrm{T}} \mathbf{X}^{\mathrm{T}})$, subject to $\phi_1^{\mathrm{T}} \phi_1 = 1$

Computation of PCA

- First PC:
  $\phi_1^{\mathrm{T}} \mathbf{X}^{\mathrm{T}}$ that maximize $var(\phi_1^{\mathrm{T}} \mathbf{X}^{\mathrm{T}})$, subject to $\phi_1^{\mathrm{T}} \phi_1 = 1$

- i^{th} PC:
  $\phi_i^{\mathrm{T}} \mathbf{X}^{\mathrm{T}}$ that maximize $var(\phi_i^{\mathrm{T}} \mathbf{X}^{\mathrm{T}})$, subject to $\phi_i^{\mathrm{T}} \phi_i = 1$ and
  $cov(\phi_i^{\mathrm{T}} \mathbf{X}^{\mathrm{T}}, \phi_k^{\mathrm{T}} \mathbf{X}^{\mathrm{T}}) = 0, \forall k < i$

Computation of PCA

- *The* solution: Σ's eigenvalue-eigenvector pairs (λ_i, ϕ_i), where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \geq 0$, $i = 1, 2, ..., p$

Computation of PCA

- *The* solution: Σ's eigenvalue-eigenvector pairs $(\lambda_i, \boldsymbol{\phi_i})$, where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \geq 0$, $i = 1, 2, ..., p$

- the ith PC:

$$\boldsymbol{z}_i = \boldsymbol{\phi_i}^{\mathrm{T}} \mathbf{X}^{\mathrm{T}}$$

$$Var(\boldsymbol{z}_i) = \boldsymbol{\phi_i}^{\mathrm{T}} \Sigma \boldsymbol{\phi_i} = \lambda_i$$

$$Cov(\boldsymbol{z}_i, \boldsymbol{z}_k) = \boldsymbol{\phi_i}^{\mathrm{T}} \Sigma \boldsymbol{\phi_k} = 0, i \neq k$$

PCA

- The eigenvalues are the variance components:

$$\sigma_{11} + \sigma_{22} + ... + \sigma_{pp} = \sum_{i=1}^{p} Var(\mathbf{x}_i)$$

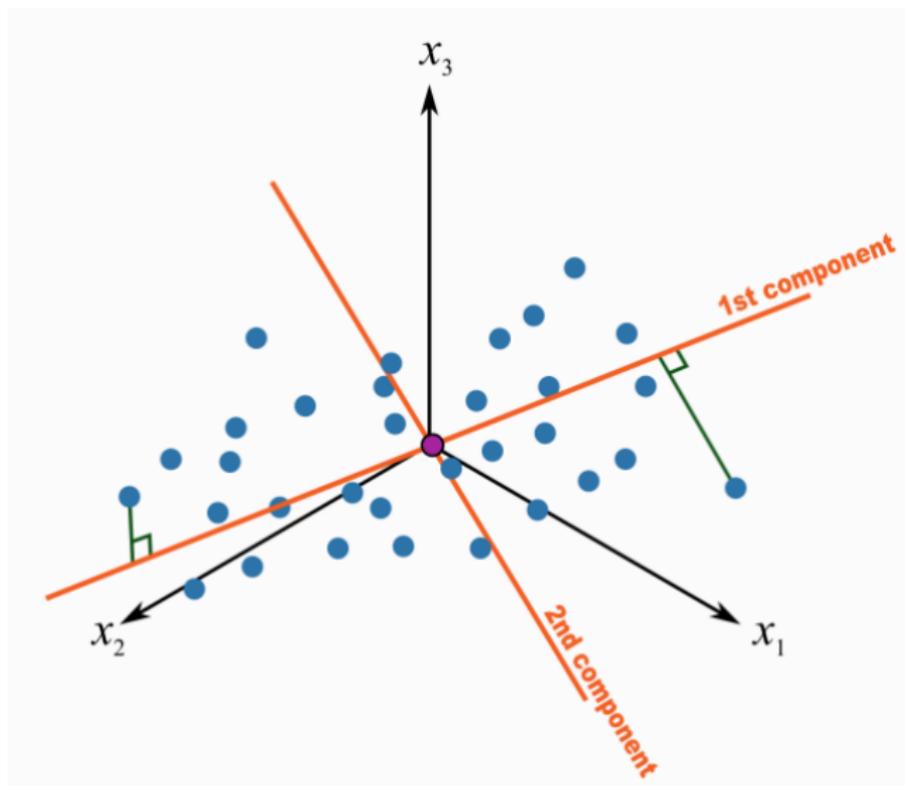$$= \lambda_1 + \lambda_2 + \cdots + \lambda_p = \sum_{i=1}^{p} Var(\boldsymbol{z}_i)$$

PCA

- The eigenvalues are the variance components:

$$\sigma_{11} + \sigma_{22} + ... + \sigma_{pp} = \sum_{i=1}^{p} Var(\mathbf{x}_i)$$

$$= \lambda_1 + \lambda_2 + \cdots + \lambda_p = \sum_{i=1}^{p} Var(\boldsymbol{z}_i)$$

- Proportion of total variance explained by the ith PC:

$$\frac{\lambda_i}{\lambda_1 + \lambda_2 + \cdots + \lambda_p}$$

PCA geometric explanation

PCA fun reading: big family dinner

Your great-grandmother: *"I heard you are studying 'Pee-See-Ay'. I wonder what that is..."*

1563

Imagine a big family dinner, where everybody starts asking you about PCA. First you explain it to your great-grandmother; then to you grandmother; then to your mother; then to your spouse; finally, to your daughter (who is a mathematician). Each time the next person is less of a layman. Here is how the conversation might go.

Great-grandmother: I heard you are studying "Pee-See-Ay". I wonder what that is...

+50

You: Ah, it's just a method of summarizing some data. Look, we have some wine bottles standing here on the table. We can describe each wine by its colour, by how strong it is, by how old it is, and so on (see this very nice visualization of wine properties taken from here). We can compose a whole list of different characteristics of each wine in our cellar. But many of them will measure related properties and so will be redundant. If so, we should be able to summarize each wine with fewer characteristics! This is what PCA does.

Pale Salmon
Provence Rosé, White Zinfandel

Medium Salmon
Sangiovese Rosé

Deep Salmon
Syrah Rosé

Pale Pink
Mourvèdre Rosé

Medium Pink
Grenache Rosé

Pale Ruby
Pinot Noir

Medium Ruby
Tempranillo, GSM Blend

Deep Ruby
Merlot, Cabernet Sauvignon, Tannat

Pale Purple
Beaujolais Nouveau, Valpolicella

Medium Purple
Malbec, Syrah, Teroldego

Picture taken from: ▶ Click this stackexchange post link

PCA data scaling

- Using correlation matrix instead of covariance matrix?
  - This is equivalent to first standardizing all x vectors

$$x^{'} = \frac{x - \mu_i}{\sqrt{\sigma_{ii}}}$$

- Using the correlation matrix avoids the domination from one x variable due to scaling (unit changes), for example using inch instead of foot.

PCA data scaling example

Data scaling is recommended (and often necessary) to avoid the domination of certain variables. Example (inch vs foot):

$$\Sigma_1 = \left[\begin{array}{cc} 1 & 4 \\ 4 & 100 \end{array} \right], \Sigma_2 = \left[\begin{array}{cc} 1 & 0.4 \\ 0.4 & 1 \end{array} \right]$$

PCA from Σ_1:

$$\lambda_1 = 100.16, \boldsymbol{\phi_1}^{\mathrm{T}} = (0.040, 0.999)$$
$$\lambda_2 = 0.84, \boldsymbol{\phi_2}^{\mathrm{T}} = (0.999, -0.040)$$
$$\frac{\lambda_1}{\lambda_1 + \lambda_2} = 0.99$$

PCA from Σ_2:

$$\lambda_1 = 1.4, \boldsymbol{\phi_1}^{\mathrm{T}} = (0.707, 0.707)$$
$$\lambda_2 = 0.6, \boldsymbol{\phi_2}^{\mathrm{T}} = (0.707, -0.707)$$
$$\frac{\lambda_1}{\lambda_1 + \lambda_2} = 0.7$$

PCA example

- USAarrests data: For each of the 50 states in the US, the dataset contains the number of arrests per 100,000 residents for each of the three crimes: Assault, Murder, and Rape. It also has the variable UrbanPop, the percent of the population in each state living in urban areas.
- principal component loading vectors have length $p = 4$
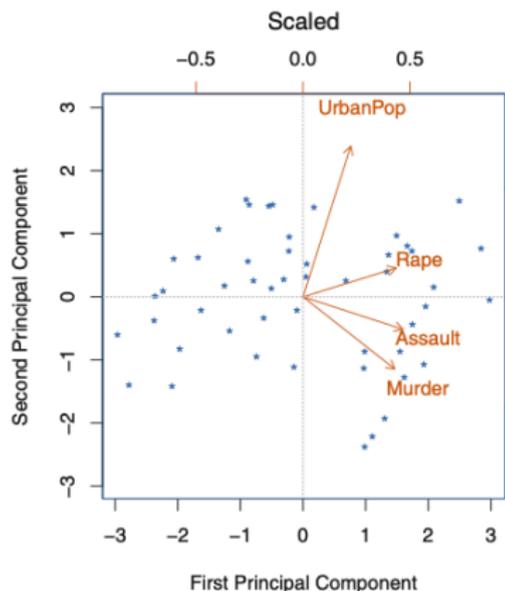- PCA was performed after standardizing each variable to have $mean = 0$ and $s.d. = 1$

USAarrests data: PCA plot
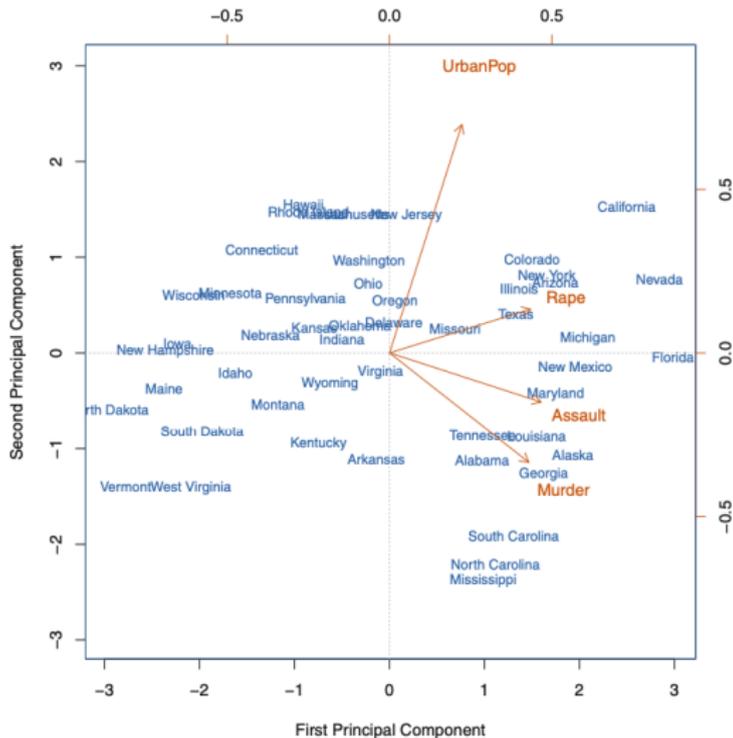
	PC1	PC2
Murder	0.5358995	−0.4181809
Assault	0.5831836	−0.1879856
UrbanPop	0.2781909	0.8728062
Rape	0.5434321	0.1673186

If the variables are in different units(scales), mean-centering ($mean = 0$) and scaling ($sd = 1$) is recommended.
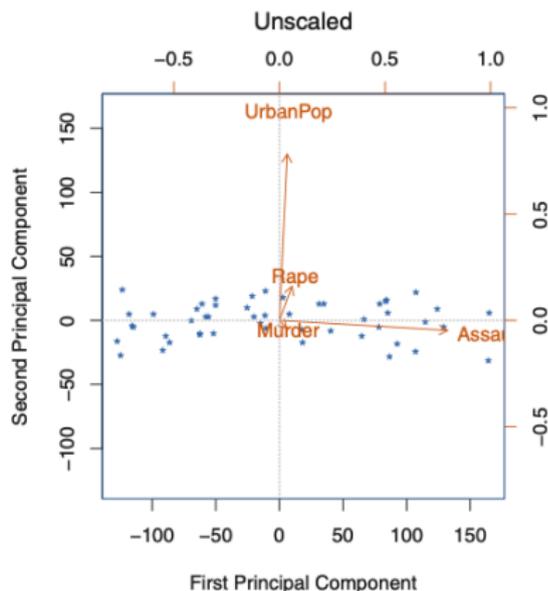
PCA finds the hyperplane closest to the observations

It defines a hyperplane in the p-dimensional space that is closest to the n observations.
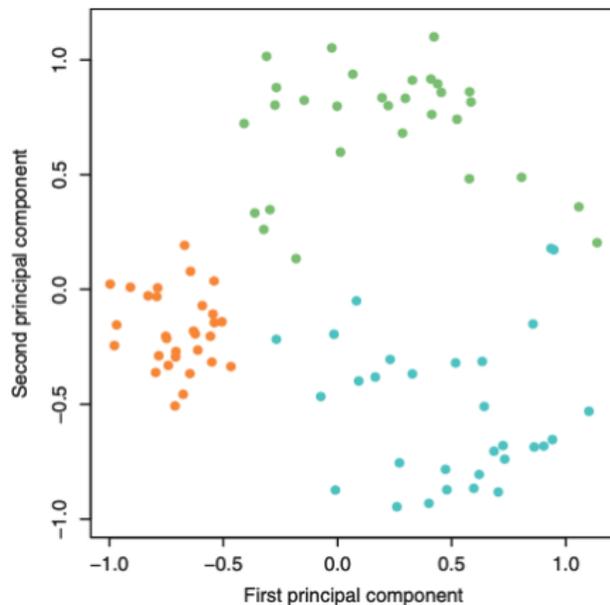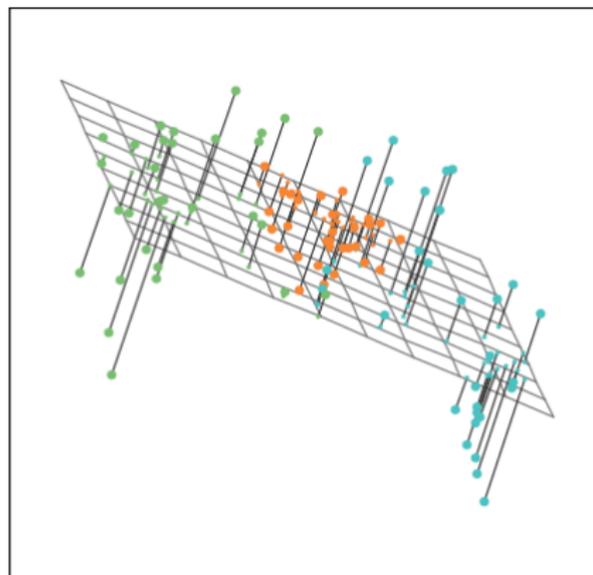
Proportion Variance Explained

The proportion of variance explained (PVE) by each principal component:
$\frac{\lambda_k}{\lambda_1 + \lambda_2 + \cdots + \lambda_p}$, where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \geq 0$, $i = 1, 2, ..., p$

Then how many PCs to retain? Look for "elbow" and "plateau"!
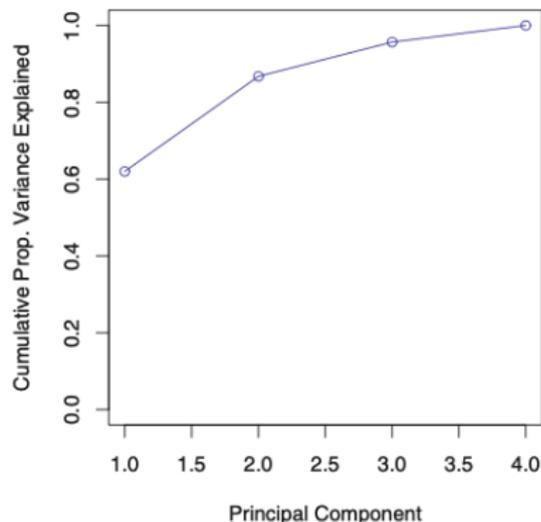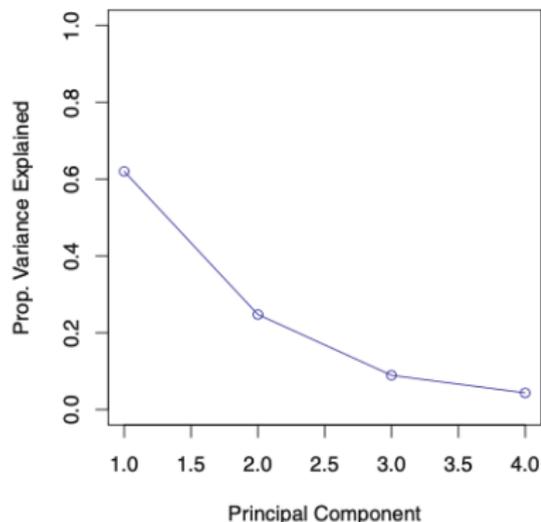
Assumption: the small amount of variation explained by low-rank PCs is noise.

Sparse PCA

In high-dimensional genomics data, loadings of a single PC on 10,000 genes doesn't make much sense. To obtain "sparse" loadings, and make the interpretation easier, and the model more robust:

SCoTLASS

$N \times p$ data matrix $\mathbf{X}$. $\qquad\qquad a_k^T(\mathbf{X}^T\mathbf{X})a_k,$

subject to

$$a_k^T a_k = 1 \quad \text{and (for} \quad k \geq 2) \quad a_h^T a_k = 0, \quad h < k;$$

and the extra constraints

$$\sum_{j=1}^{p} |a_{kj}| \leq t$$

Projection pursuit

- A very broad term: finding the most "interesting" direction of projection. How the projection is done depends on the definition of "interesting". If it is maximal variation, then PP leads to PCA.
- In a narrower sense:
  - Finding non-Gaussian projections. Why?
  - Because: For most high-dimensional clouds, most low-dimensional projections are close to Gaussian, but important information in the data is in the directions for which the projected data could be far from Gaussian.
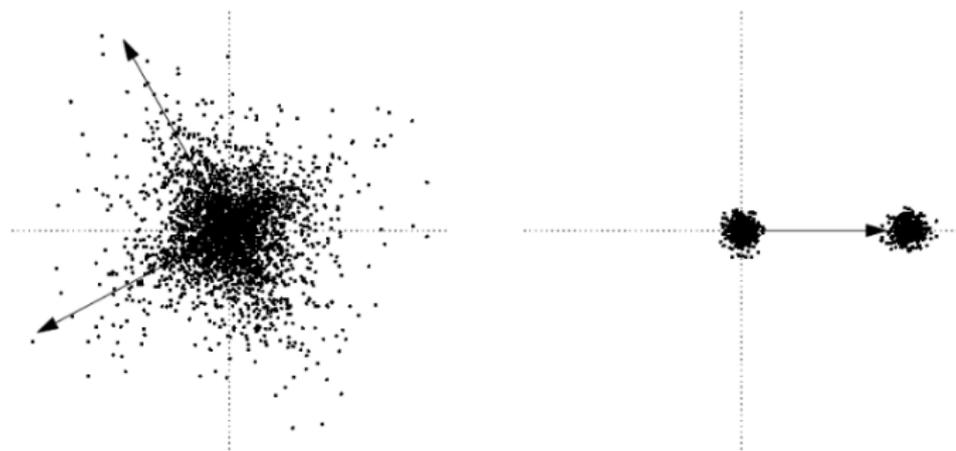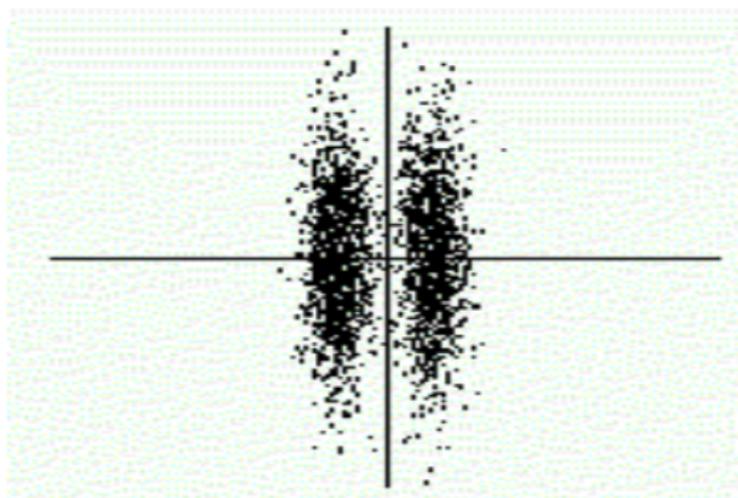
Figure 3.1: Example directions of significant structure. Directions of high kurtosis (left) and multi-modality (right) are two examples of "interesting" directions in the space.

It boils down to objective functions – each kind of "interesting" has its own objective function to maximize.

Projection pursuit with **multi-modality** as objective.

Projection pursuit

- One objective function to measure **multi-modality**:

$$R_w(\boldsymbol{x}) = \frac{1}{3}E[(\boldsymbol{x}\boldsymbol{w})^3] - \frac{1}{4}E^2[(\boldsymbol{x}\boldsymbol{w})^2]$$

- It uses the first three moments of the distribution.
- It can help finding clusters through visualization.
- Can think of PCA as a special case of PP.

Independent Component Analysis (ICA)

Statistical Independence

For any functions $g_1()$ and $g_2()$,
$E[g_1(\boldsymbol{x_i})g_2(\boldsymbol{x_j})] - E[g_1(\boldsymbol{x_i})]E[g_2(\boldsymbol{x_j})] = 0$, for $i \neq j$

- In PCA, lack of correlation does NOT indicate statsitical independence.
- Lack of correlation only determines the 2nd-degree cross-moment.
- Compare with PCA, ICA finds the factors(directions) that are statistically independent, rather than just uncorrelated.

ICA

- In ICA, the latent variables are assumed to be independent and non-Gaussian.
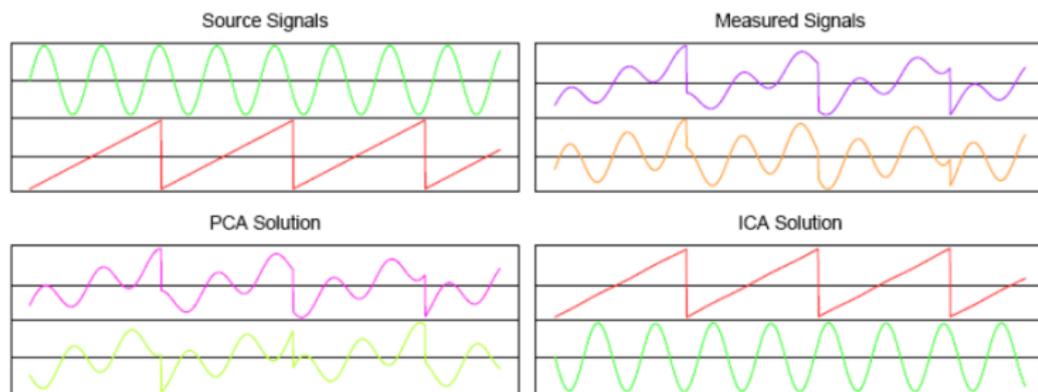
- Since for multivariate Gaussian, uncorrelatedness = independence, if the true hidden factors are Gaussian, they can still be determined only up to a rotation.

Non-negative Matrix Factorization

Non-negative Matrix Factorization (NMF)

Problem setup:
Given a non-negative matrix V, find non-negative matrices W and H such that:

$$V \approx WH$$

Non-negative Matrix Factorization (NMF)

Problem setup:
Given a non-negative matrix V, find non-negative matrices W and H such that:

$$V \approx WH$$

History:

1. Originally proposed in chemometrics to extract information from chemical systems in the 1970s.

2. Later became well-known after Lee & Seung (Nature, 1999; NIPS, 2001) proposed algorithms to conduct NMF and studied their statistical properties.

3. Gained popularity recently among computer vision, audio signal processing, bioinformatics, etc.

Factorization details

Factorization:

- Factorize $m \times n$ matrix V into an $m \times r$ matrix W and $r \times n$ matrix H.
- Usually r is chosen to be smaller than n, such that W and H are smaller than original matrix V.

**Interpretation**:

- NMF can be rewritten by column as:

$$v \approx Wh$$

 where v and h are the corresponding columns of V and H.
- Each data vector v is approximated by a linear combination of the columns of W, weighted by the components of h.
    - W: Optimized basis vectors for the linear approximation of the data in V – columns of W spans the low dimensional space.
    - H: Weights for the linear combination on basis W – the coordinates of the projection in the low dimensional space.

Heterogeneous mixture

- Human tissues have diverse cell types/states.
- Traditional RNA-seq ("bulk" RNA-seq) can measure **averaged signal** across millions of cells.
- Single-cell RNA-seq (scRNA-seq) give us the first data-driven approach to study the **heterogeneous tissue** at single-cell level.

Lawson et al. Nature. https://www.nature.com/articles/s41556-018-0236-7

Bulk RNA-seq

Single-cell RNA-seq
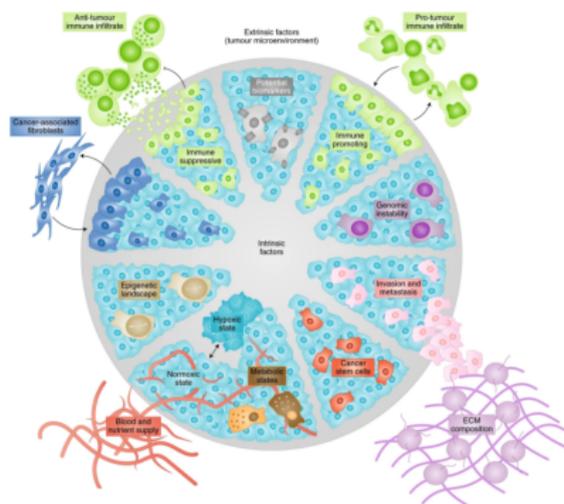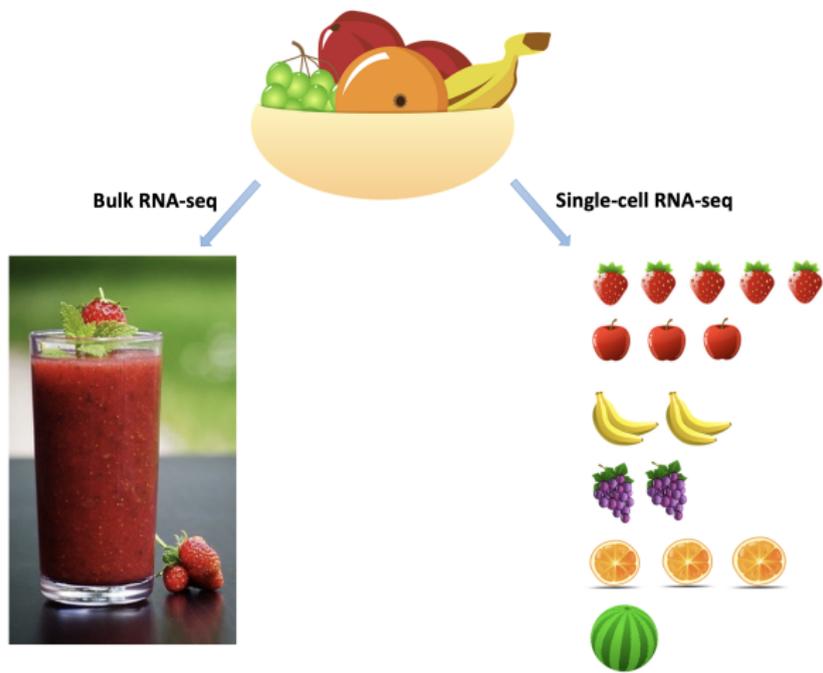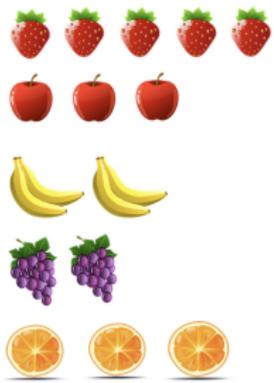
NMF cost functions

- To find an approximate factorization $V \approx WH$, one need to define the cost function to quantify the quality of approximation.

- The measure of distance between two non-negative matrices A and B can be useful.

- Commonly, the square of the Euclidean distance between A and B is adopted:

$$||A - B||^2 = \sum_{ij} (A_{ij} - B_{ij})^2$$

Therefore, we can formulate NMF as the following **optimization problem**:

Problem:

- Minimize $||V - WH||^2$ with respect to W and H, subject to the constraints $W, H \geq 0$

Properties

- Although $||V - WH||^2$ is convex in W only or H only, it is not convex in both variables together.
- Therefore, one can not expect an algorithm finding the global minima.
- Gradient descent is the simple technique to find at least the local minima.

General NMF algorithm

Input Non-negative matrix $V \in \mathbb{R}_+^{m \times n}$ and factorization rank r.
Output W, $H \geq 0$ s.t. $V \approx WH$

Algorithm:

1. **(Starting point)** Generate some initial matrices $W^{(0)} \geq 0$ and $H^{(0)} \geq 0$;

2. **(Iteration)** For $t = 1, 2, \dots$ do

$$W_{ik}^{(t)} = W_{ik}^{(t-1)} \frac{(VH^{(t-1)T})_{ik}}{(W^{(t-1)}H^{(t-1)}H^{(t-1)T})_{ik}}$$

$$H_{kj}^{(t)} = H_{kj}^{(t-1)} \frac{(W^{(t-1)T}V)_{kj}}{(W^{(t-1)T}W^{(t-1)}H^{(t-1)})_{kj}}$$

for all $i \in \{1, 2, ..., m\}$, $j \in \{1, 2, ..., n\}$, $k \in \{1, 2, ..., r\}$.

3. **(Stop criteria)** Stop iteration if W and H are at a stationary point.

NMF in R programming

The *NMF* package provide functions (*nmf*) to solve NMF problems.

```
R Documentation
nmf {NMF}
Running NMF algorithms

Description

The function nmf is a S4 generic defines the main interface to run
NMF algorithms within the framework defined in package NMF.
It has many methods that facilitates applying, developing and testing
NMF algorithms.

The package vignette vignette('NMF') contains an introduction to the
interface, through a sample data analysis.

Usage

  nmf(x, rank, method, ...)
```

Example: NMF in R

```
> library(NMF)

# random data
x <- rmatrix(20,10)

# run default algorithm with rank 2
res <- nmf(x, 2)

# The result is an object of classNMFfit
> fit(res)
<Object of class:NMFstd>
features: 20
basis/rank: 2
samples: 10
```

Example: NMF in R

```
> library(NMF)

# random data
x <- rmatrix(20,10)

# run default algorithm with rank 2
res <- nmf(x, 2)

# The result is an object of classNMFfit
> fit(res)
<Object of class:NMFstd>
features: 20
basis/rank: 2
samples: 10
```

Example: NMF in R

```
# get matrix W using basis() function
w <- basis(res)
dim(w)
[1] 20  2

# get matrix H using coef() function
h <- coef(res)
dim(h)
[1]  2 10

# Additionally, several build-in algorithms are available to choose fro
# use (method) argument to specify algorithm
> nmfAlgorithm()
 [1] "brunet"    "KL"        "lee"       "Frobenius" "offset"    "nsNMF
 [7] "ls-nmf"    "pe-nmf"    "siNMF"     "snmf/r"    "snmf/l"

# for example
res <- nmf(x, 2, method = "lee")
```

NMF Application 1: Image processing

Guillamet et al. (Artificial Intelligence, 2002) conducted facial feature extraction. Each column of data matrix $X \in \mathbb{R}_+^{p \times n}$ is a vectorized gray-level image of a face, with (i,j)th entry of X being the intensity of the ith pixel in the jth face.

$$\underbrace{X(:,j)}_{j\text{th facial image}} \approx \sum_{k=1}^{r} \underbrace{W(:,k)}_{\text{facial features}} \underbrace{H(k,j)}_{\substack{\text{importance of features} \\ \text{in } j\text{th image}}} = \underbrace{WH(:,j)}_{\substack{\text{approximation} \\ \text{of } j\text{th image}}}$$
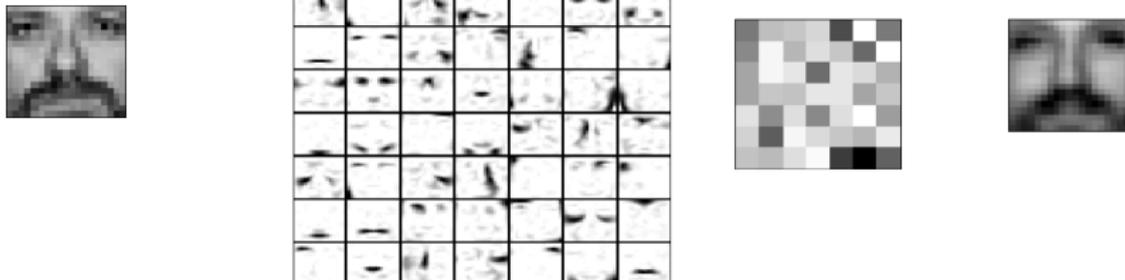
Columns in W: Basis images (mouth, nose, mustache etc.), after convert back to matrix of the same size as face pictures.
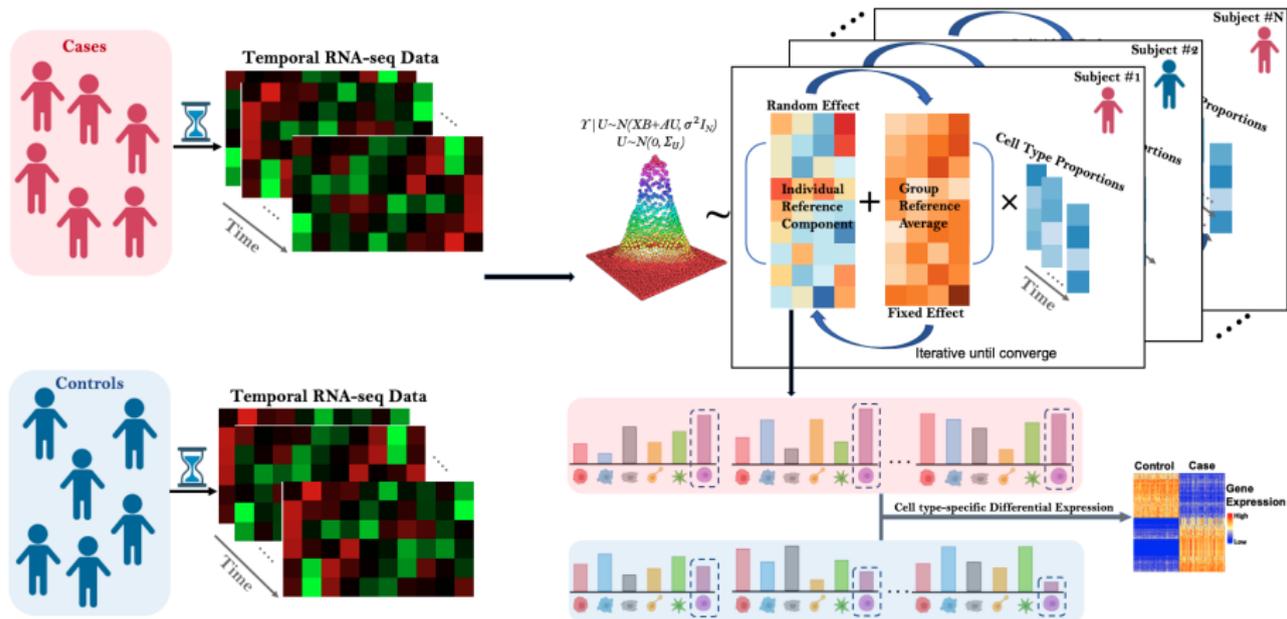
H: Weights of basis images.
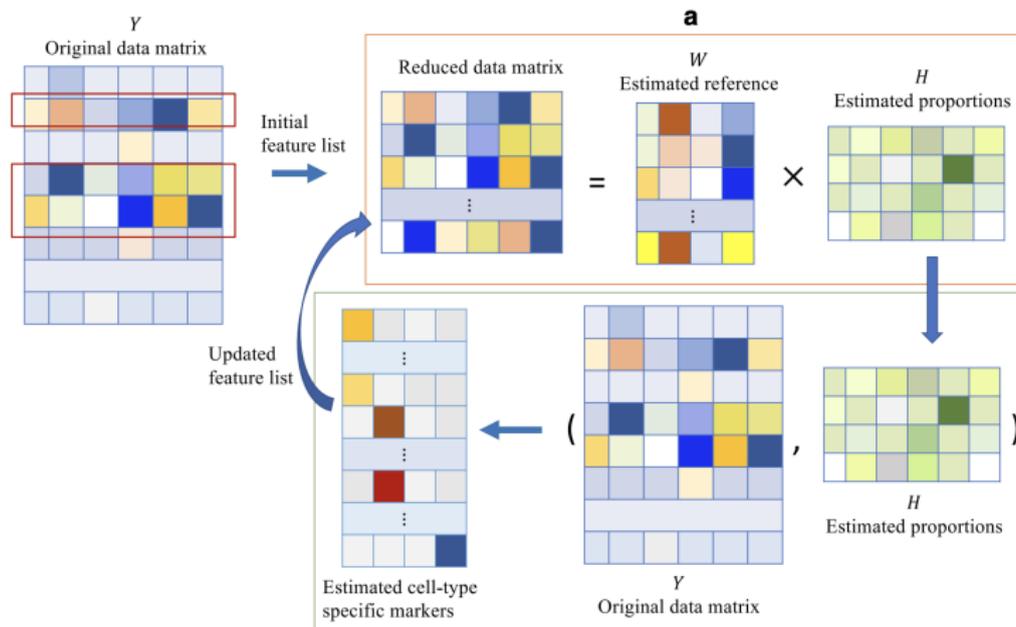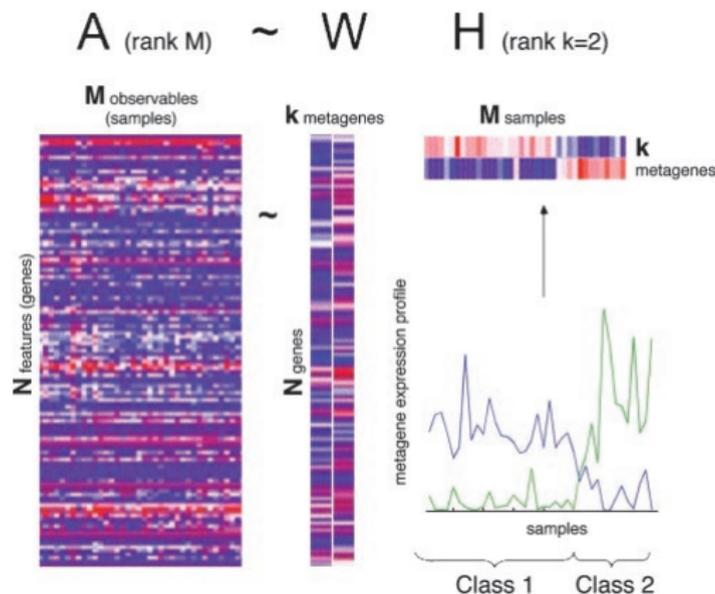
Li et al. Genome Biology 2019

NMF Application 3: Individual-level signal decomposition
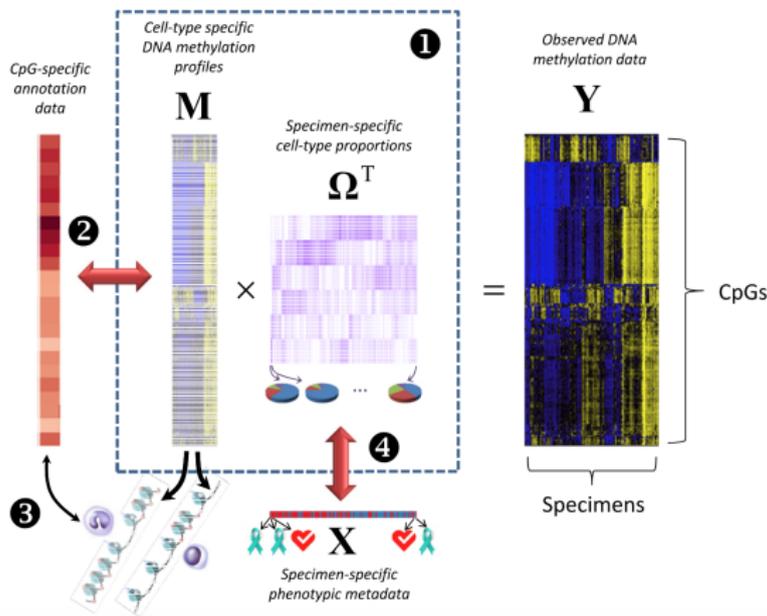
NMF Application 4: Gene expression

Brunet *et al.* (PNAS, 2004) *Metagenes and molecular pattern discovery using matrix factorization.*

Deconvolute gene expression data (A) into metagene profiles (W) and proportions (H).

Houseman et al. (BMC Bioinformatics, 2016): deconvolutes DNA methylation data (Y) into cell-type specific profile (M) and cell-type proportions (Ω^T).

Oliver Stegle (Genome Biology, 2020): Multi-Omics Factor Analysis (MOFA): reconstructs a low-dimensional representation of single-cell multi-modal data.

NMF readings
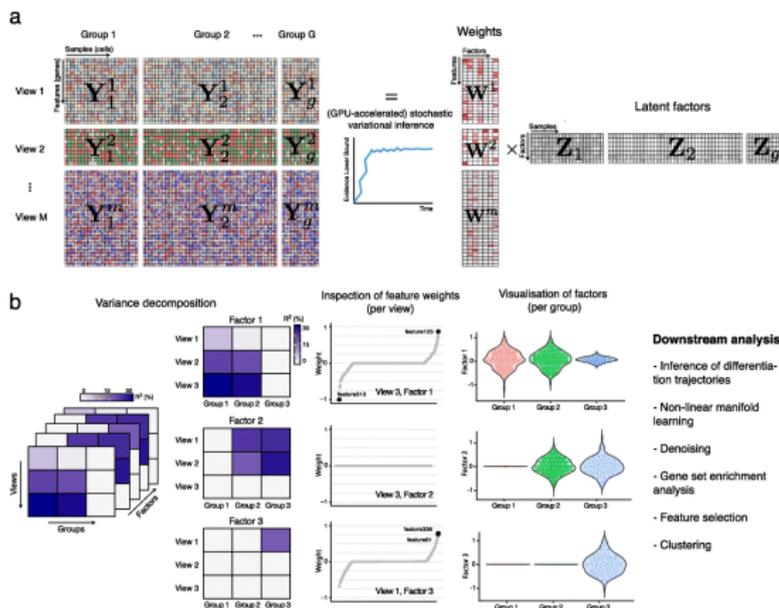
- Lee, D. D., Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. **Nature**, 401(6755), 788.
- Lee, D. D., Seung, H. S. (2001). Algorithms for non-negative matrix factorization. **Advances in neural information processing systems** (pp. 556-562).
- Brunet et al. (2004) Metagenes and molecular pattern discovery using matrix factorization. **PNAS**, 101 (12) 4164-4169;
- Houseman, E. A. et al. (2016). Reference-free deconvolution of DNA methylation data and mediation by cell composition effects. **BMC bioinformatics**, 17(1), 259.
- Stein-O'Brien G. L. et al. (2018). Enter the matrix: factorization uncovers knowledge from omics. **Trends in Genetics**.

Textbook chapters

Textbook:
ISLR: An Introduction to Statistical Learning: with applications in R

- ISLR chapter 12: 12.1, 12.2

Additional readings (optional):
ESL: The Elements of Statistical Learning

- ESL chapter 14: 14.1, 14.5 - 14.7