Data Science for Manager Certificat

Dimensionality Reduction

Julien Chiquet

Spring 2021

https://jchiquet.github.io/DS4M

HEC PARIS

ACCÉLÉRATEUR & INCUBATEUR DE START-UP ÉCOLE POLYTECHNIQUE

Part I

Introduction

Packages used for data manipulation and representation

```
library(tidyverse)     # opinionated collection of packages for data manipulation
library(corrplot)      # (correlation) matrix plot
theme_set(theme_bw())
```

Exploratory analysis of (modern) data sets

Assume a table with n individuals described by p features/variables

Questions

Look for **patterns** or **structures** to summarize the data by

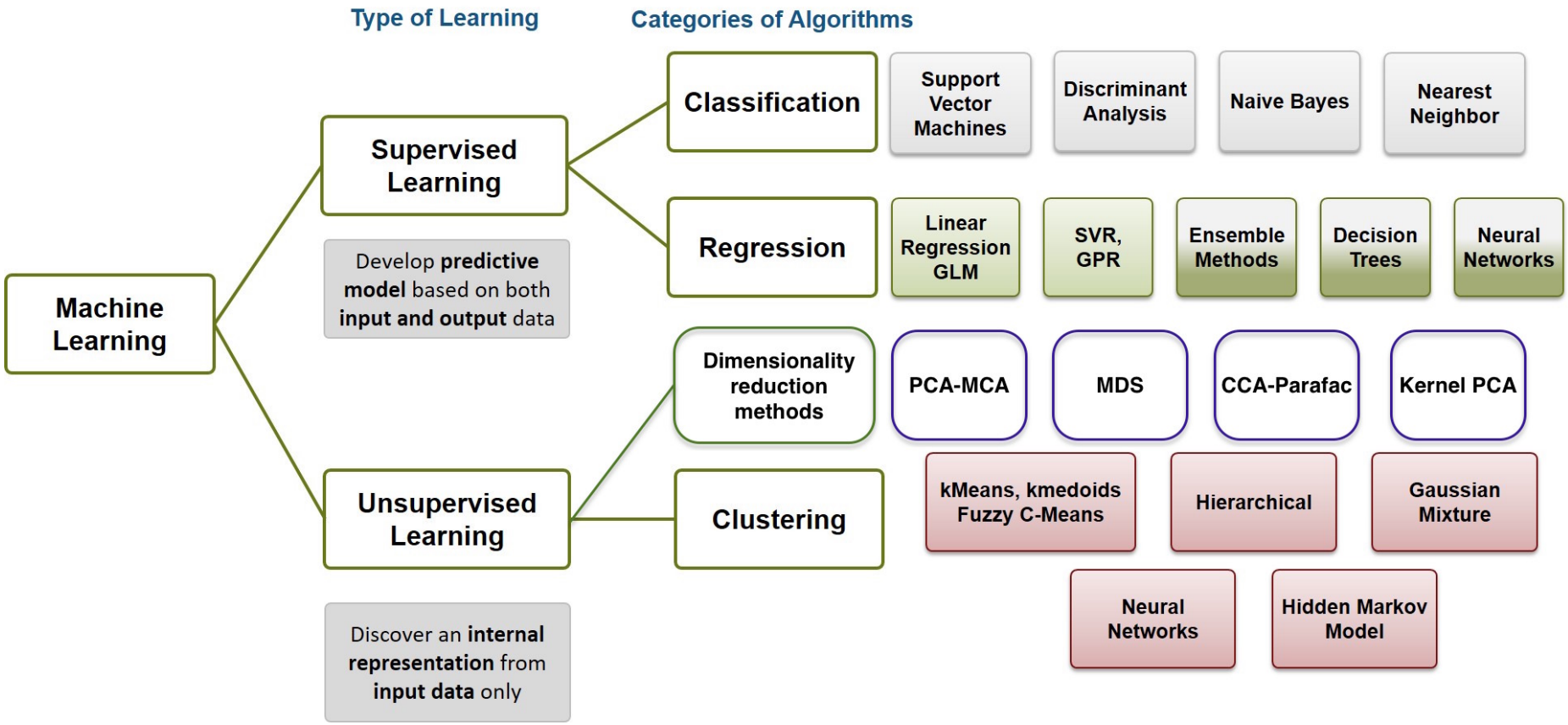
- Finding groups of "similar" individuals
- Finding variables important for these data
- Performing visualization

Challenges

Size	data may be large ("big data ": large n large p)
Dimension	data may be high dimensional (more variables than individual or $n \ll p$)
Redundancy	many variables may carry the same information
Unsupervised	we don't necessary know what we are looking after

Overview of Statistics & Machine Learning

Where is today's course in this big picture?



An example in genetics: 'snp'

Genetics variant in European population

Description: *medium/large data, high-dimensional*

500, 000 Genetics variants (SNP – Single Nucleotide Polymorphism) for 3000 individuals (1 meter $\times$ 166 meter (height $\times$ width)

- SNP : 90 % of human genetic variations
- coded as 0, 1 or 2 (10, 1 or 2 allel different against the population reference)

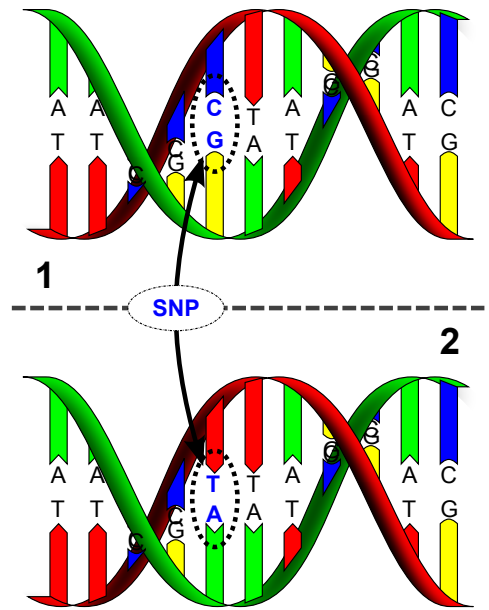


Figure: SNP (wikipedia)

Summarize 500,000 variables with 2 features

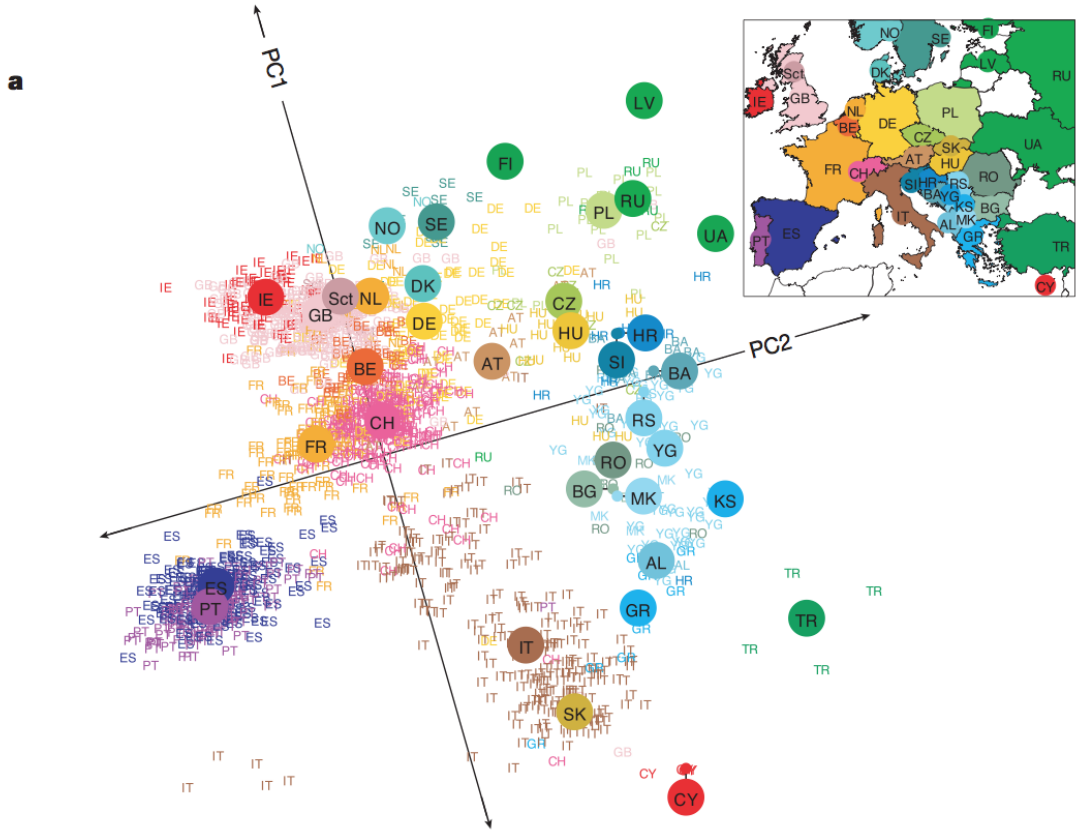


Figure: Dimension reduction + labels source: Nature "Gene Mirror Geography Within Europe", 2008

In the original messy $3,000 \times 500,000$ table, we may find

- an extremely strong structure between individuals (**"clustering"**)
- a very simple subspace where it is obvious (**"dimension reduction"**)

Dimension reduction: general goals

Main objective: find a **low-dimensional representation** that captures the "essence" of (high-dimensional) data

Application in Machine Learning

Preprocessing, Regularization

- Compression, denoising, anomaly detection
- Reduce overfitting in supervised learning

Application in Statistics/Data analysis

Better understanding of the data

- descriptive/exploratory methods
- visualization (difficult to plot and interpret $> 3D$!)

Dimension reduction: problem setup

Settings

- Training data : $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \in \mathbb{R}^p$, (i.i.d.)
- Space $\mathbb{R}^p$ of possibly high dimension $(n \ll p)$

Dimension Reduction Map

Construct a map Φ from the space $\mathbb{R}^p$ into a space $\mathbb{R}^q$ of smaller dimension:

$$\begin{aligned} \Phi: \quad & \mathbb{R}^p \to \mathbb{R}^q, q \ll p \\ & \mathbf{x} \mapsto \Phi(\mathbf{x}) \end{aligned}$$

How should we design/construct Φ?

Criterion

- Geometrical approach
- Reconstruction error
- Relationship preservation

Form of the map Φ

- Linear or non-linear ?
- tradeoff between interpretability and versatility ?
- tradeoff between high or low computational resource

Companion data set: 'scRNA'

Subsamples of normalized Single-Cell RNAseq

Description: *subsample of a large data set*

Gene-level expression of 100 representative genes for a collection of 301 cells spreaded in 11 cell-lines. Original transcription data are measured by counts obtained by *RNAseq* and normalized to be close to a Gaussian distribution.

Pollen, Alex A., et al. Low-coverage single-cell mRNA sequencing reveals cellular heterogeneity and activated signaling pathways in developing cerebral cortex.

Nature biotechnology 32.10 (2014): 1053.

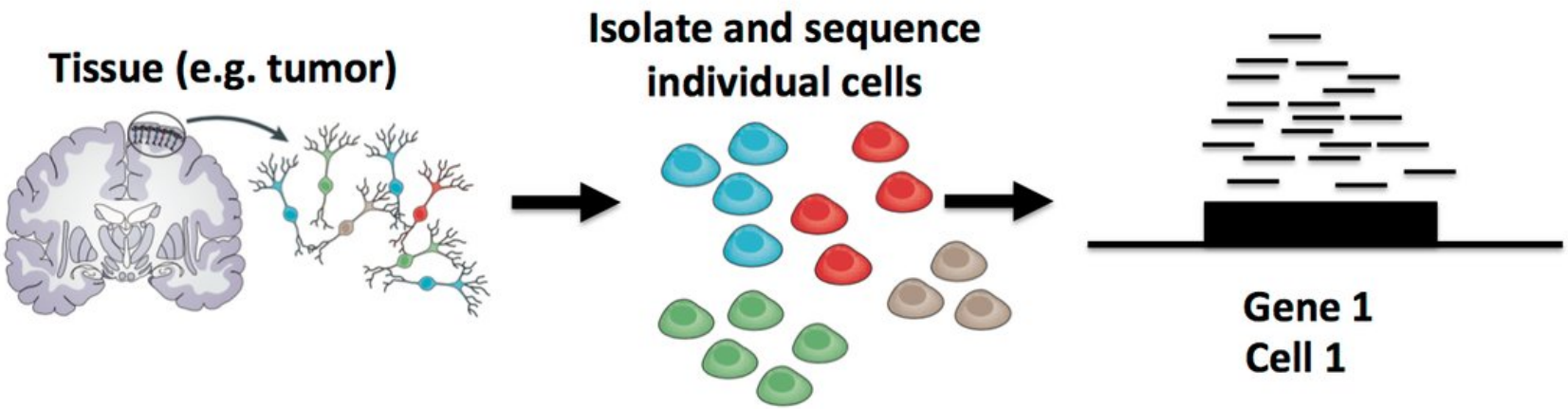


Figure: Single Cell RNA sequencing data: general principle – source: Stephanie Hicks

Companion data set: 'scRNA'

Brief data summary I

```
load("../../data/scRNA.RData")
scRNA <- as_tibble(t(pollen$data)) %>% add_column(cell_type = pollen$celltypes)
```

Data table

```
scRNA[, 1:6] %>% head(3) %>% knitr::kable("latex")
```

Spike1	MT2A	HBG2	PRG2	IFITM1	ANXA1
0	12.21149	0	0	11.96908	11.837198
0	11.30622	0	0	12.67121	8.098769
0	11.92623	0	0	12.35984	10.688626

Cell types

```
scRNA %>% dplyr::select(cell_type) %>% summary()  %>% knitr::kable()
```

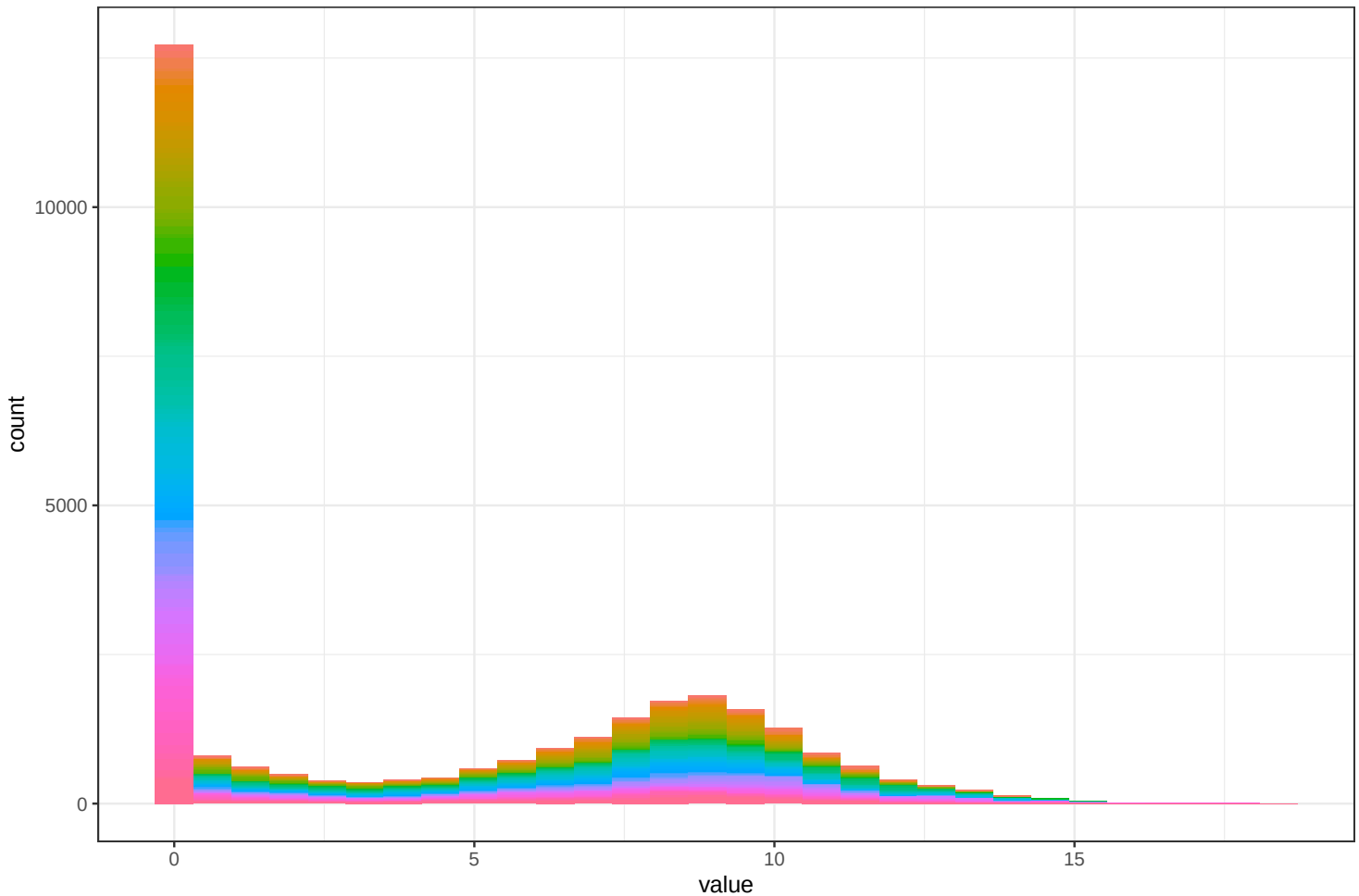
	cell_type
	HL60 :54
	K562 :42
	Kera :40
	BJ :37
	GW16 :26
	hiPSC :24
	(Other):78

Companion data set: 'scRNA'

Brief data summary II

Histogram of normalized expression

```
scRNA %>% dplyr::select(-cell_type) %>% pivot_longer(everything()) %>%
  ggplot() + aes(x = value, fill = name) + geom_histogram(show.legend = FALSE)
```

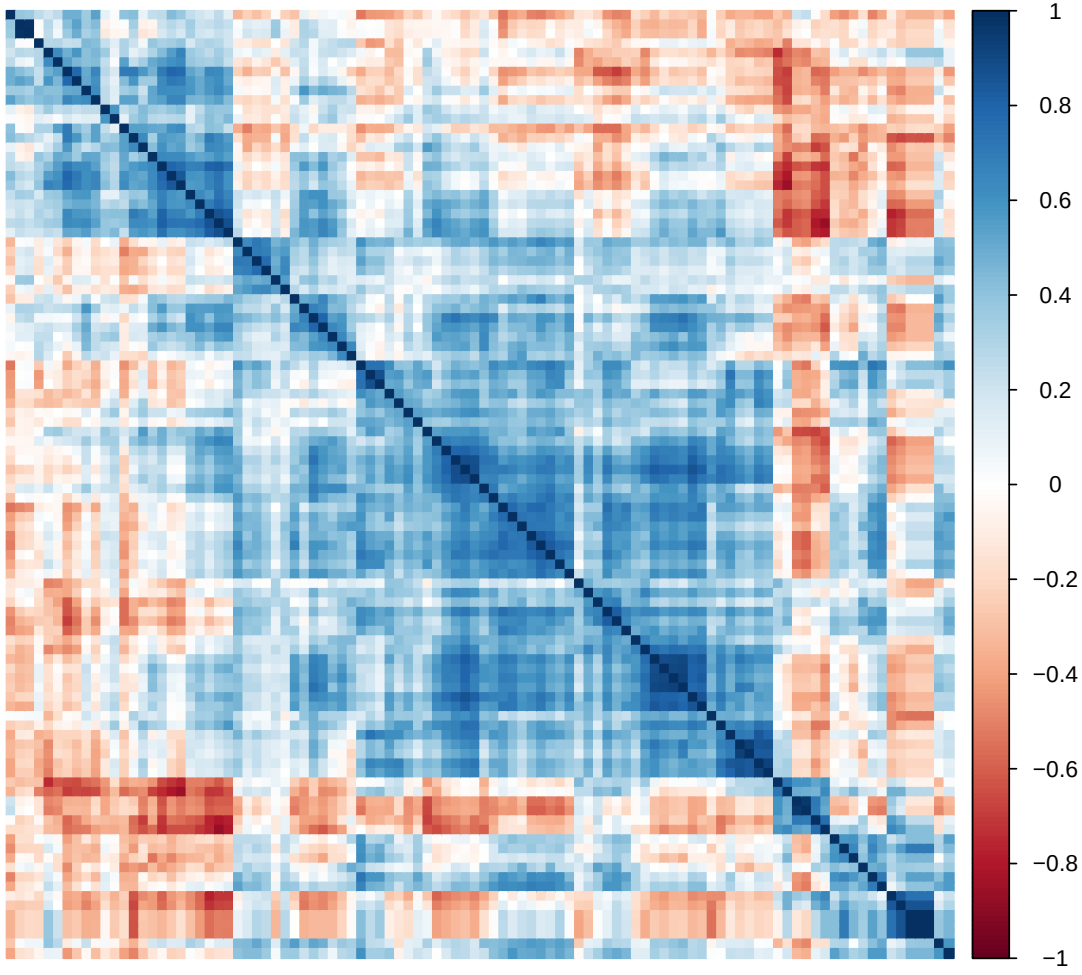


Companion data set: 'scRNA'

Brief data summary III

Correlation between gene expression

```
scRNA %>% dplyr::select(-cell_type) %>% cor() %>%
  corrplot(method = "color", tl.pos = "n", order = "hclust")
```



Part II

Linear Methods

Packages used to perform PCA

```
library(FactoMineR) # PCA and oter linear method for dimension reduction
library(factoextra) # fancy plotting for FactoMineR output
```

PCA and classical Linear methods

Principal component Analysis (PCA) is for continuous data

Non continuous data

- Correspondence analysis (CA): contingency table
- Multiple correspondence analysis (MCA): categorical data
- Multiple factor analysis (MFA): multi-table, array data

$\rightsquigarrow$ Basic adaptations that build on PCA to deal with non-continuous data

$\rightsquigarrow$ smart encoding of non-continuous data to continuous ones

We will focus on PCA, as the mother of most linear (and non-linear) methods.

Objectives

Individual/Observations

- similarity between observations with respect to all the variables
- Find pattern ($\sim$ partition) between individuals

Variables

- linear relationships between variables
- find synthetic variables

Link between the two

- characterization of the groups of individuals with variables
- specific observations to understand links between variables

Outline

Linear Methods

Vectors in $\mathbb{R}^n$

Definition and Basics

A vector $\mathbf{x} \in \mathbb{R}^d$ is defined by a d-uplet $(x_1, x_2, \ldots, x_d)$, *its coordinates*.

Elementary operations

- Addition of two vectors (define a parallelogram)

$$\mathbf{x} + \mathbf{y} = \begin{pmatrix} x_1 + y_1 \\ x_2 + y_2 \\ \vdots \\ x_d + y_d \end{pmatrix}$$

- Multiplication by a scalar (streching)

$$\lambda \mathbf{x} = \begin{pmatrix} \lambda x_1 \\ \lambda x_2 \\ \vdots \\ \lambda x_d \end{pmatrix}, \quad \lambda \in \mathbb{R}.$$

Properties

- associativity: $(\mathbf{x} + \mathbf{y}) + \mathbf{z} = \mathbf{x} + (\mathbf{y} + \mathbf{z})$
- commutativity: $\mathbf{x} + \mathbf{y} = \mathbf{y} + \mathbf{x}$
- linearity: $\lambda(\mathbf{x} + \mathbf{y}) = \lambda\mathbf{x} + \lambda\mathbf{y}$
- $(\lambda_1 + \lambda_2)\mathbf{x} = \lambda_1\mathbf{x} + \lambda_2\mathbf{x}$

Vectors in $\mathbb{R}^n$

Dot/Inner product and norm

Dot product of 2 vectors: sum of the products between each coordinate:

$$\langle \mathbf{x}, \mathbf{y} \rangle \equiv \mathbf{x} \cdot \mathbf{y} \equiv \mathbf{x}^\top \mathbf{y} \triangleq \sum_{i=1}^{d} x_i \, y_j.$$

- $\mathbf{x}^\top \mathbf{y} = \mathbf{y}^\top \mathbf{x}$
- $\mathbf{x}^\top (\mathbf{y} + \mathbf{z}) = \mathbf{x}^\top \mathbf{y} + \mathbf{x}^\top \mathbf{z}$
- $\lambda(\mathbf{x}^\top \mathbf{y}) = (\lambda(\mathbf{x})^\top \mathbf{y} = \mathbf{x}^\top (\lambda \mathbf{y})$
- if $\mathbf{x} = \mathbf{0}$, then $\mathbf{x}^\top \mathbf{x} = 0$.

(Euclidean) norm (a.k.a length, magnitude)

$$\|\mathbf{x}\| = \sqrt{\mathbf{x}^\top \mathbf{x}}. \quad \text{we have } \|\lambda \mathbf{x}\| = |\lambda| \|\mathbf{x}\|.$$

Vectors in $\mathbb{R}^n$

Distances and orthogonality

(Euclidean) distance between 2 vectors

$$\mathsf{dist}(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|.$$

Remark that when $\mathbf{x}$ and $\mathbf{y}$ are orthogonal and non zero, distances between $\mathbf{x}$ and $\mathbf{y}$ and $\mathbf{x}$ and $(-\mathbf{y})$ are the same. Then,

$$(\mathbf{x} - \mathbf{y})^\top (\mathbf{x} - \mathbf{y}) = (\mathbf{x} + \mathbf{y})^\top (\mathbf{x} + \mathbf{y}) \Leftrightarrow \mathbf{x}^\top \mathbf{y} = 0,$$

which motivates the following definition of orthornality:

Orthogonality

Two vectors $\mathbf{x}, \mathbf{y} \neq \mathbf{0}$ are orthogonal iff $\mathbf{x}^\top \mathbf{y} = 0$.

(Euclidean) distance between 2 vectors

$$\text{dist}(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|.$$

Remark that when $\mathbf{x}$ and $\mathbf{y}$ are orthogonal and non zero, distances between $\mathbf{x}$ and $\mathbf{y}$ and $\mathbf{x}$ and $(-\mathbf{y})$ are the same. Then,

$$(\mathbf{x} - \mathbf{y})^\top (\mathbf{x} - \mathbf{y}) = (\mathbf{x} + \mathbf{y})^\top (\mathbf{x} + \mathbf{y}) \Leftrightarrow \mathbf{x}^\top \mathbf{y} = 0,$$

which motivates the following definition of orthornality:

Orthogonality

Two vectors $\mathbf{x}, \mathbf{y} \neq \mathbf{0}$ are orthogonal iff $\mathbf{x}^\top \mathbf{y} = 0$.

Vectors in $\mathbb{R}^n$

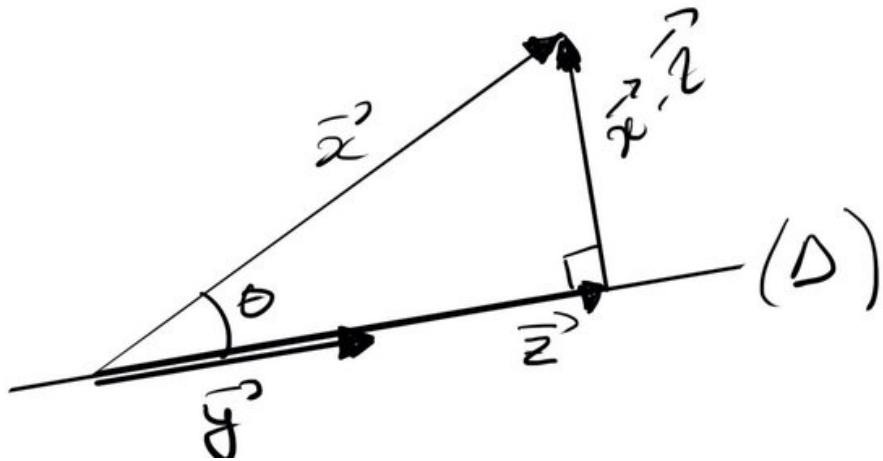
Orthogonal Projection and geometric definition of the dot product

Orthogonal projection of $\mathbf{x}$ onto $\mathbf{y}$

It is the vector $\mathbf{z}$ such that

1. $\mathbf{z} = \lambda \mathbf{y}$
2. $\mathbf{y}$ is orthogonal to $\mathbf{x} - \mathbf{z}$

We find $\lambda = \mathbf{x}^\top \mathbf{y} / \|\mathbf{y}\|^2$



Thanks to basic trigonometry theorem,

$$\cos(\theta) = \frac{\|\mathbf{z}\|}{\|\mathbf{x}\|} = \lambda \frac{\|\mathbf{y}\|}{\|\mathbf{x}\|}$$

and then we end with the following geometric definition of the dot product

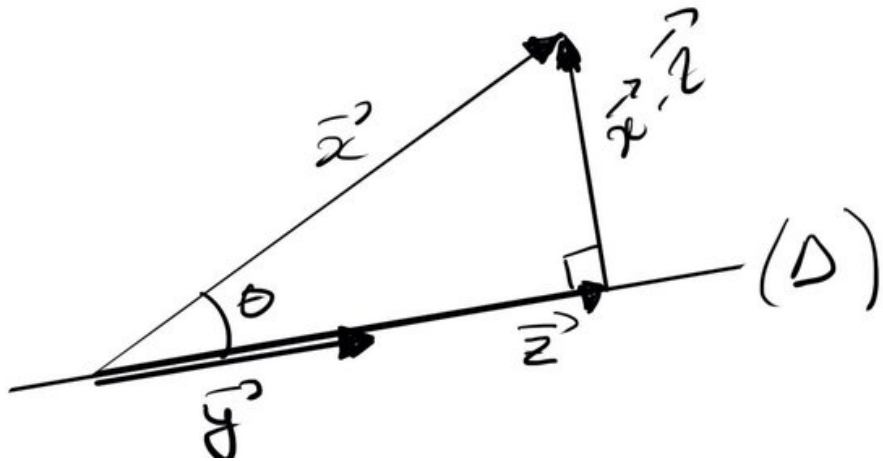
Dot product: geometric definition

$$\mathbf{x}^\top \mathbf{y} = \cos(\theta) \|\mathbf{x}\| \, \|\mathbf{y}\|$$

Vectors in $\mathbb{R}^n$

Orthogonal Projection and geometric definition of the dot product

Orthogonal projection of $\mathbf{x}$ onto $\mathbf{y}$



It is the vector $\mathbf{z}$ such that

1. $\mathbf{z} = \lambda \mathbf{y}$
2. $\mathbf{y}$ is orthogonal to $\mathbf{x} - \mathbf{z}$

We find $\lambda = \mathbf{x}^\top \mathbf{y} / \|\mathbf{y}\|^2$

Thanks to basic trigonometry theorem,

$$\cos(\theta) = \frac{\|\mathbf{z}\|}{\|\mathbf{x}\|} = \lambda \frac{\|\mathbf{y}\|}{\|\mathbf{x}\|}$$

and then we end with the following geometric definition of the dot product

Dot product: geometric definition

$$\mathbf{x}^\top \mathbf{y} = \cos(\theta) \|\mathbf{x}\| \, \|\mathbf{y}\|$$

Outline

Linear Methods

The data matrix

The data set is a $n \times p$ matrix $\mathbf{X} = (x_{ij})$ with values in $\mathbb{R}$:

- each row $\mathbf{x}_i$ represents an individual/observation
- each col $\mathbf{x}^j$ represents a variable/attribute

$$\mathbf{X} = \begin{array}{c} \\ \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_i \\ \vdots \\ \mathbf{x}_n \end{array} \begin{array}{c} \begin{array}{cccccc} \mathbf{x}^1 & \mathbf{x}^2 & \dots & \mathbf{x}^j & \dots & \mathbf{x}^p \end{array} \\ \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1j} & \dots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{2j} & \dots & x_{2p} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{i1} & x_{i2} & \dots & x_{ij} & \dots & x_{ip} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nj} & \dots & x_{np} \end{pmatrix} \end{array}$$

```
scRNA[, 1:8] %>% head(3) %>% knitr::kable("latex")
```

Spike1	MT2A	HBG2	PRG2	IFITM1	ANXA1	HBG1	MPO
0	12.21149	0	0	11.96908	11.837198	0	0
0	11.30622	0	0	12.67121	8.098769	0	0
0	11.92623	0	0	12.35984	10.688626	0	0

Cloud of observation in $\mathbb{R}^p$

Individuals can be represented in the variable space $\mathbb{R}^p$ as a point cloud

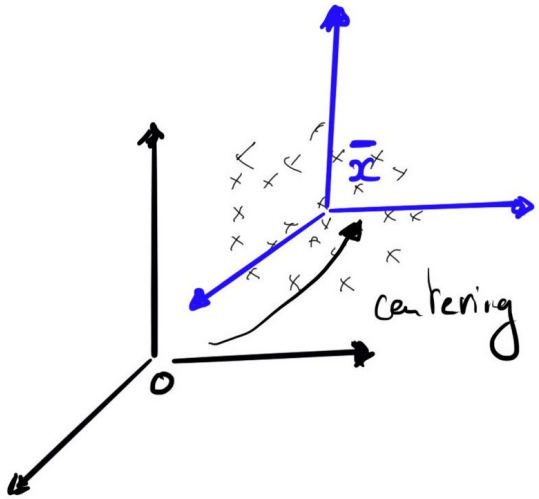


Figure: Example in $\mathbb{R}^3$

Center of Inertia

(or barycentrum, or empirical mean)

$$\bar{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i = \begin{pmatrix} \sum_{i=1}^{n} x_{i1}/n \\ \sum_{i=1}^{n} x_{i2}/n \\ \vdots \\ \sum_{i=1}^{n} x_{ip}/n \end{pmatrix}$$

We center the cloud $\mathbf{X}$ around $\mathbf{x}$ denote this by $\mathbf{X}^c$

$$\mathbf{X}^c = \begin{pmatrix} x_{11} - \bar{x}_1 & \dots & x_{1j} - \bar{x}_j & \dots & x_{1p} - \bar{x}_p \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{i1} - \bar{x}_1 & \dots & x_{ij} - \bar{x}_j & \dots & x_{ip} - \bar{x}_p \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{n1} - \bar{x}_1 & \dots & x_{nj} - \bar{x}_j & \dots & x_{np} - \bar{x}_p \end{pmatrix}$$

Inertia and Variance

Total Inertia: distance of the individuals to the center of the cloud

$$I_T = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{p}(x_{ij} - \bar{x}_j)^2 = \frac{1}{n}\sum_{i=1}^{n}\|\mathbf{x}_i - \bar{\mathbf{x}}\|^2 = \frac{1}{n}\sum_{i=1}^{n}\text{dist}^2(\mathbf{x}_i, \bar{\mathbf{x}})$$

I_T is proportional to the total variance

Let $\hat{\mathbf{\Sigma}}$ be the empirical variance-covariance matrix

$$I_T = \frac{1}{n}\sum_{j=1}^{p}\sum_{i=1}^{n}(x_{ij} - \bar{x}_j)^2 = \sum_{j=1}^{p}\frac{1}{n}\|\mathbf{x}^j - \bar{x}_j\|^2 = \sum_{j=1}^{p}\mathbb{V}(\mathbf{x}^j) = \text{trace}\left(\hat{\mathbf{\Sigma}}\right)$$

$\rightsquigarrow$ Good representation has large inertia (much variability)

$\rightsquigarrow$ Large dispertion $\sim$ Large distances between points

Inertia with respect to an axis

The Inertia of the cloud wrt axe Δ is the sum of the distances between all points and their orthogonal projection on Δ.

$$I_\Delta = \frac{1}{n}\sum_{i=1}^{n} \text{dist}^2(\mathbf{x}_i, \Delta)$$

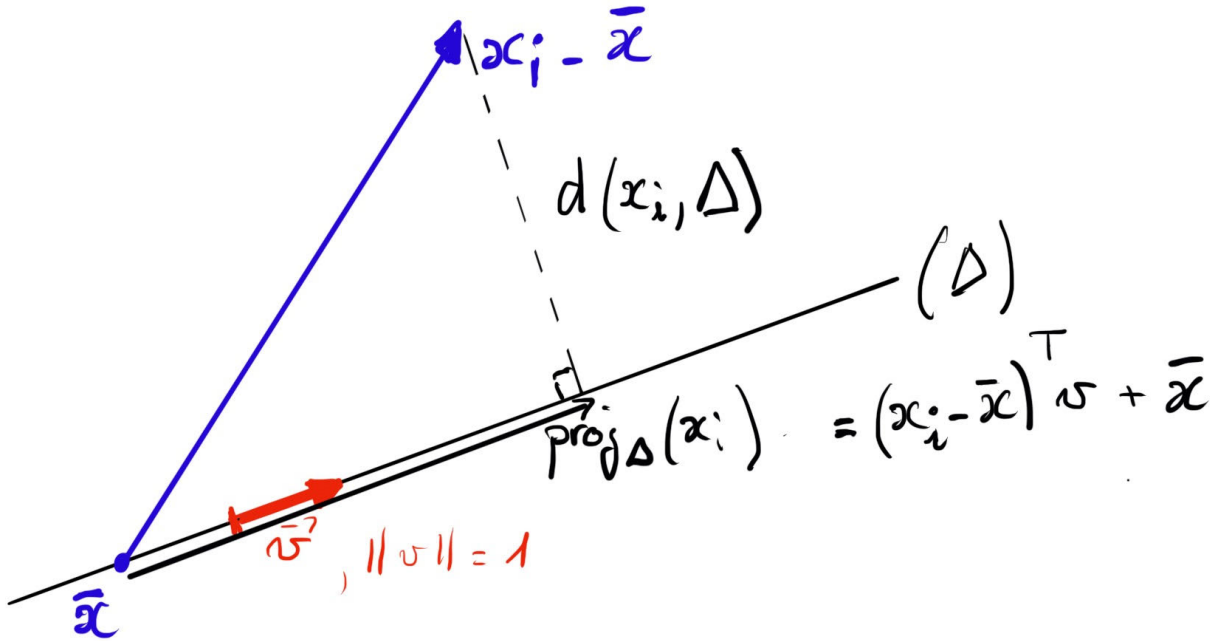
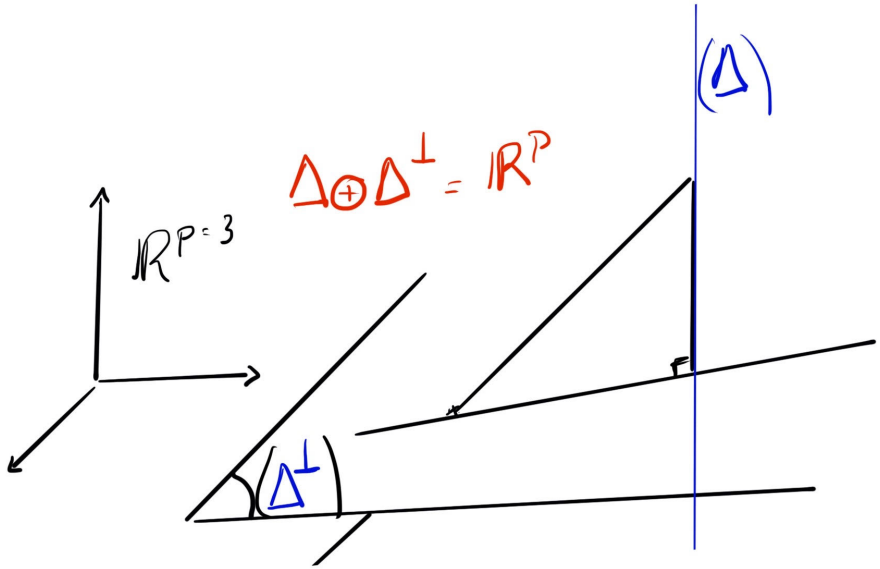


Figure: Projection of $\mathbf{x}_i$ onto a line Δ passing through $\bar{\mathbf{x}}$

Decomposition of total Inertia (1)

Let $\Delta^{\perp}$ be the orthogonal subspace of Δ in $\mathbb{R}^p$



Theorem (Huygens)

A consequence of the above (Pythagoras Theorem) is the decomposition of the following total inertia:

$$I_T = I_\Delta + I_{\Delta^\perp}$$

By projecting the cloud $\mathbf{X}$ onto Δ, with loss the inertia measured by $\Delta^{\perp}$

Decomposition of total Inertia (2)

Consider only subspaces with dimension 1 (that is, lines or axes). We can decompose $\mathbb{R}^p$ as the sum of p othogonal axis.

$$\mathbb{R}^p = \Delta_1 \oplus \Delta_2 \oplus \cdots \oplus \Delta_p$$

$\rightsquigarrow$ These axes form a new basis for representing the point cloud.

Theorem (Huygens)

$$I_T = I_{\Delta_1} + I_{\Delta_2} + \cdots + I_{\Delta_p}$$

Outline

Linear Methods

Finding the best axis (1)

Definition of the problem

- The best axis Δ_1 is the "closest" to the point cloud
- Inertia of Δ_1 measures the distance between the data and Δ_1
- Δ_1 is defined by the director vector $\mathbf{v}_1$, such as $\|\mathbf{v}_1\| = 1$
- $\Delta_1^{\perp}$ is defined by the normal vector $\mathbf{v}_1$, such as $\|\mathbf{v}_1\| = 1$

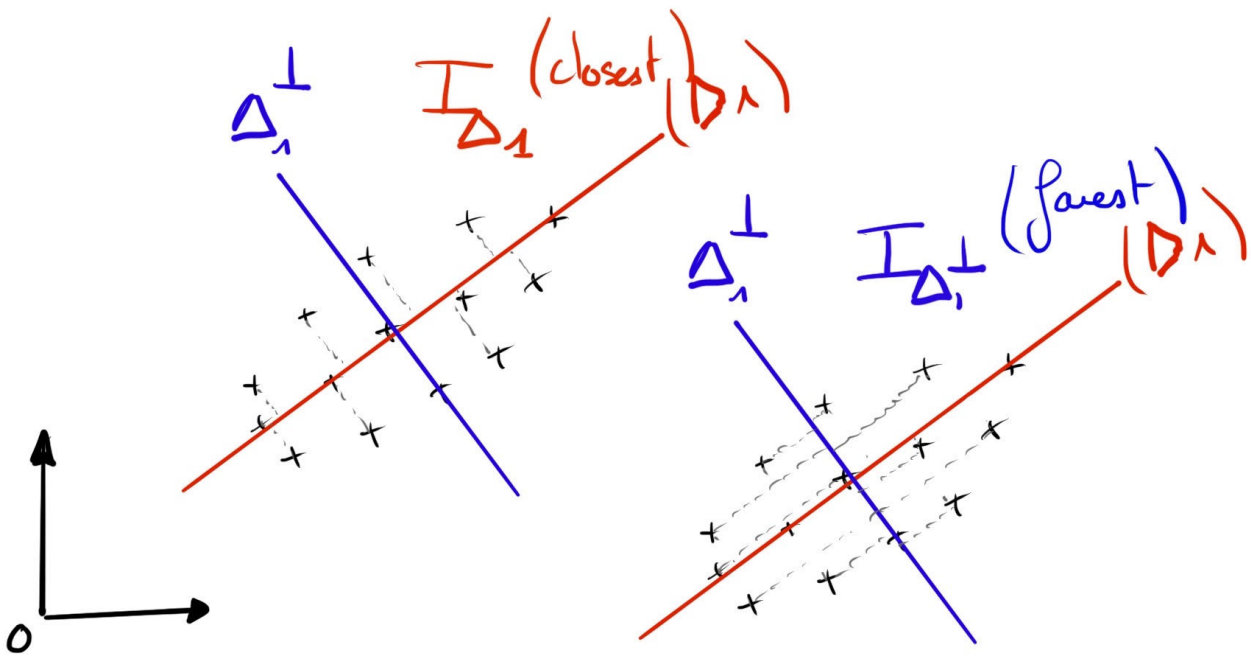
$\rightsquigarrow$ The best axis Δ_1 is the one with the minimal Inertia.

Finding the best axis (2)

Stating the optimization problem

Since $\Delta_1 \oplus \Delta_1^\perp = \mathbb{R}^p$ and $I_T = I_{\Delta_1} + I_{\Delta_1^\perp}$, then

$$\underset{\mathbf{v}\in\mathbb{R}^p:\|\mathbf{v}\|=1}{\text{minimize}}\ I_{\Delta_1} \Leftrightarrow \underset{\mathbf{v}\in\mathbb{R}^p:\|\mathbf{v}\|=1}{\text{maximize}}\ I_{\Delta_1^\perp}$$



Finding the best axis (3)

Stating the problem (algebraically)

Find $\mathbf{v}_1; \|\mathbf{v}_1\| = 1$ that maximizes

$$
\begin{aligned}
I_{\Delta_1^\perp} &= \frac{1}{n}\sum_{i=1}^{n} \text{dist}(\mathbf{x}_i, \Delta_1^\perp)^2 \\
&= \frac{1}{n}\sum_{i=1}^{n} \mathbf{v}_1^\top (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\top \mathbf{v}_1 \\
&= \mathbf{v}_1^\top \left(\sum_{i=1}^{n} \frac{1}{n} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\top \right) \mathbf{v}_1 \\
&= \mathbf{v}_1^\top \hat{\mathbf{\Sigma}} \mathbf{v}_1
\end{aligned}
$$

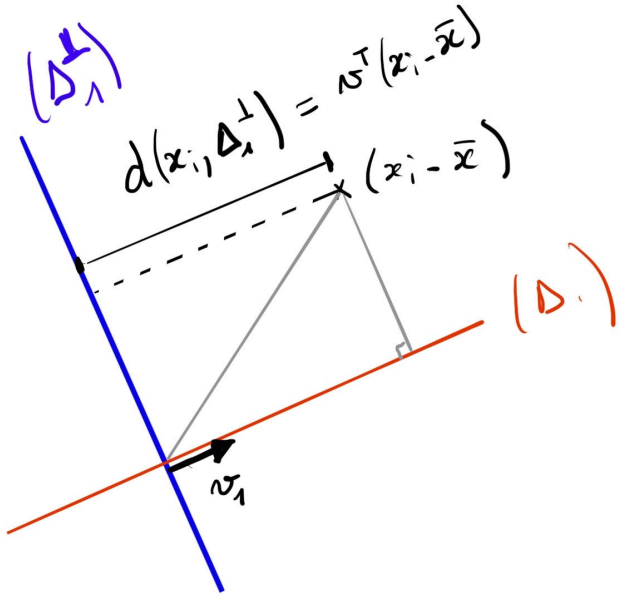


Figure: Geometrical insight

Finding the best axis (4)

We solve a simple constraint maximization problem with the method of Lagrange multipliers:

$$\underset{\mathbf{v}_1:\|\mathbf{v}_1\|=1}{\text{maximize}}\, \mathbf{v}_1^\top \hat{\mathbf{\Sigma}} \mathbf{v}_1 \Leftrightarrow \underset{\mathbf{v}_1 \in \mathbb{R}^p, \lambda_1 > 0}{\text{maximize}}\, \mathbf{v}_1^\top \hat{\mathbf{\Sigma}} \mathbf{v}_1 - \lambda_1 (\|\mathbf{v}_1\|^2 - 1)$$

By straightforward (vector) differentiation, an using that $\mathbf{v}_1^\top \mathbf{v}_1 = 1$

$$\begin{cases} 2\hat{\mathbf{\Sigma}}\mathbf{v}_1 - 2\lambda_1 \mathbf{v}_1 = 0 \\ \mathbf{v}_1^\top \mathbf{v}_1 - 1 = 0 \end{cases} \Leftrightarrow \begin{cases} \hat{\mathbf{\Sigma}}\mathbf{v}_1 = \lambda_1 \mathbf{v}_1 \\ \mathbf{v}_1^\top \hat{\mathbf{\Sigma}} \mathbf{v}_1 = \lambda_1 \mathbf{v}_1^\top \mathbf{v}_1 = \lambda_1 = I_{\Delta_1}^\perp \end{cases}$$

- $\mathbf{v}_1$ is the first (normalized) eigen vector of $\hat{\mathbf{\Sigma}}$
- λ_1 is the first eigen value of $\hat{\mathbf{\Sigma}}$

$\rightsquigarrow$ **Δ_1 is defined by the first eigen vector of $\hat{\Sigma}$**

$\rightsquigarrow$ **Variance "carried" by Δ_1 is equal to the largest eigen value of $\hat{\Sigma}$**

Finding the following axes

Second best axis

Find Δ_2 with dimension 1, director vector $\mathbf{v}_2$ orthogonal to Δ_1 solving

$$\underset{\mathbf{v}_2 \in \mathbb{R}^p}{\text{maximize}}\, I_{\Delta_2^\perp} = \mathbf{v}_2^\top \hat{\mathbf{\Sigma}} \mathbf{v}_2, \quad \text{with } \|\mathbf{v}_2\| = 1, \mathbf{v}_1^\top \mathbf{v}_2 = 0.$$

$\rightsquigarrow$ $\mathbf{v}_2$ is the second eigen vector of $\hat{\mathbf{\Sigma}}$ with eigen value λ_2

And so on!

PCA is roughly a matrix factorisation problem

$$\hat{\mathbf{\Sigma}} = \mathbf{V} \mathbf{\Lambda} \mathbf{V}^\top, \quad \mathbf{V} = \begin{pmatrix} \mathbf{v}_1 & \mathbf{v}_2, & \dots & \mathbf{v}_p \end{pmatrix}, \quad \mathbf{\Lambda} = \text{diag}(\lambda_1, \dots, \lambda_p)$$

- $\mathbf{V}$ is an orthogonal matrix of normalized eigen vectors.
- $\mathbf{\Lambda}$ is diagonal matrix of ordered eigen values.

Finding the following axes

Second best axis

Find Δ_2 with dimension 1, director vector $\mathbf{v}_2$ orthogonal to Δ_1 solving

$$\underset{\mathbf{v}_2 \in \mathbb{R}^p}{\text{maximize}}\, I_{\Delta_2^\perp} = \mathbf{v}_2^\top \hat{\mathbf{\Sigma}} \mathbf{v}_2, \quad \text{with } \|\mathbf{v}_2\| = 1, \mathbf{v}_1^\top \mathbf{v}_2 = 0.$$

$\rightsquigarrow$ $\mathbf{v}_2$ is the second eigen vector of $\hat{\mathbf{\Sigma}}$ with eigen value λ_2

And so on!

PCA is roughly a matrix factorisation problem

$$\hat{\mathbf{\Sigma}} = \mathbf{V} \mathbf{\Lambda} \mathbf{V}^\top, \quad \mathbf{V} = \begin{pmatrix} \mathbf{v}_1 & \mathbf{v}_2, & \dots & \mathbf{v}_p \end{pmatrix}, \quad \mathbf{\Lambda} = \text{diag}(\lambda_1, \dots, \lambda_p)$$

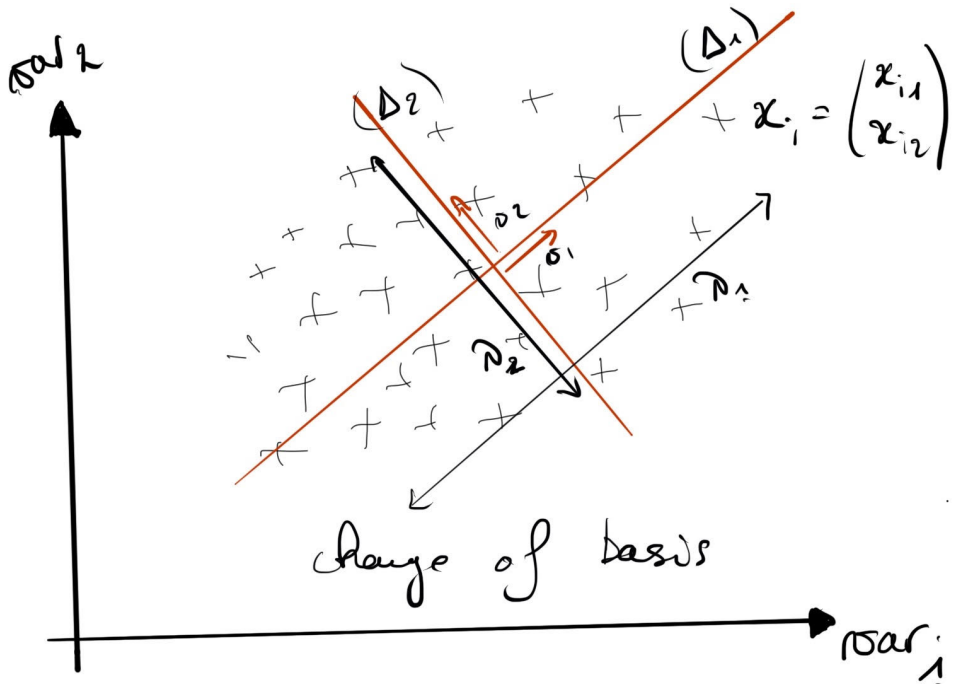
- $\mathbf{V}$ is an orthogonal matrix of normalized eigen vectors.
- $\mathbf{\Lambda}$ is diagonal matrix of ordered eigen values.

Interpretation in $\mathbb{R}^p$

$\mathbf{V}$ describes a new orthogonal basis and a rotation of data in this basis

$\rightsquigarrow$ PCA is an appropriate rotation on axes that maximizes the variance

$$\begin{cases} \Delta_1 & \oplus & \dots & \oplus & \Delta_p \\ \mathbf{v}_1 & \perp & \dots & \perp & \mathbf{v}_p \\ \lambda_1 & > & \dots & > & \lambda_p \\ I_{\Delta_1^\perp} & > & \dots & > & I_{\Delta_p^\perp} \end{cases}$$



Outline

Linear Methods

Outline

Linear Methods

Contribution of each axis and quality of the representation

Δ_k is carrying inertia/variance defined by its orthogonal, thus

$$I_T = I_{\Delta_1^\perp} + \cdots + I_{\Delta_p^\perp} = \lambda_1 + \cdots + \lambda_p$$

Relative contribution of axis k

$$\text{contrib}(\Delta_k) = \frac{\lambda_k}{\sum_{k=1}^{p} \lambda_j} = \frac{\lambda_k}{\text{trace}\left(\hat{\boldsymbol{\Sigma}}\right)} \times 100$$

$\rightsquigarrow$ Percentage of explained inertia/variance explained

Global quality of the representation on the first k axes

$$\text{contrib}(\Delta_1, \ldots, \Delta_k) = \frac{\lambda_1 + \cdots + \lambda_k}{\text{trace}\left(\hat{\boldsymbol{\Sigma}}\right)} \times 100$$

A few axes may explain a large proportion of the total variance.
$\rightsquigarrow$ This paves the way for dimension reduction

Contribution of each axis and quality of the representation

Δ_k is carrying inertia/variance defined by its orthogonal, thus

$$I_T = I_{\Delta_1^\perp} + \cdots + I_{\Delta_p^\perp} = \lambda_1 + \cdots + \lambda_p$$

Relative contribution of axis k

$$\text{contrib}(\Delta_k) = \frac{\lambda_k}{\sum_{k=1}^{p} \lambda_j} = \frac{\lambda_k}{\text{trace}\left(\hat{\mathbf{\Sigma}}\right)} \times 100$$

$\rightsquigarrow$ Percentage of explained inertia/variance explained

Global quality of the representation on the first k axes

$$\text{contrib}(\Delta_1, \ldots, \Delta_k) = \frac{\lambda_1 + \cdots + \lambda_k}{\text{trace}\left(\hat{\mathbf{\Sigma}}\right)} \times 100$$

A few axes may explain a large proportion of the total variance.
$\rightsquigarrow$ This paves the way for dimension reduction

Contribution of each axis and quality of the representation

Δ_k is carrying inertia/variance defined by its orthogonal, thus

$$I_T = I_{\Delta_1^\perp} + \cdots + I_{\Delta_p^\perp} = \lambda_1 + \cdots + \lambda_p$$

Relative contribution of axis k

$$\text{contrib}(\Delta_k) = \frac{\lambda_k}{\sum_{k=1}^{p} \lambda_j} = \frac{\lambda_k}{\text{trace}\left(\hat{\mathbf{\Sigma}}\right)} \times 100$$

$\rightsquigarrow$ Percentage of explained inertia/variance explained

Global quality of the representation on the first k axes

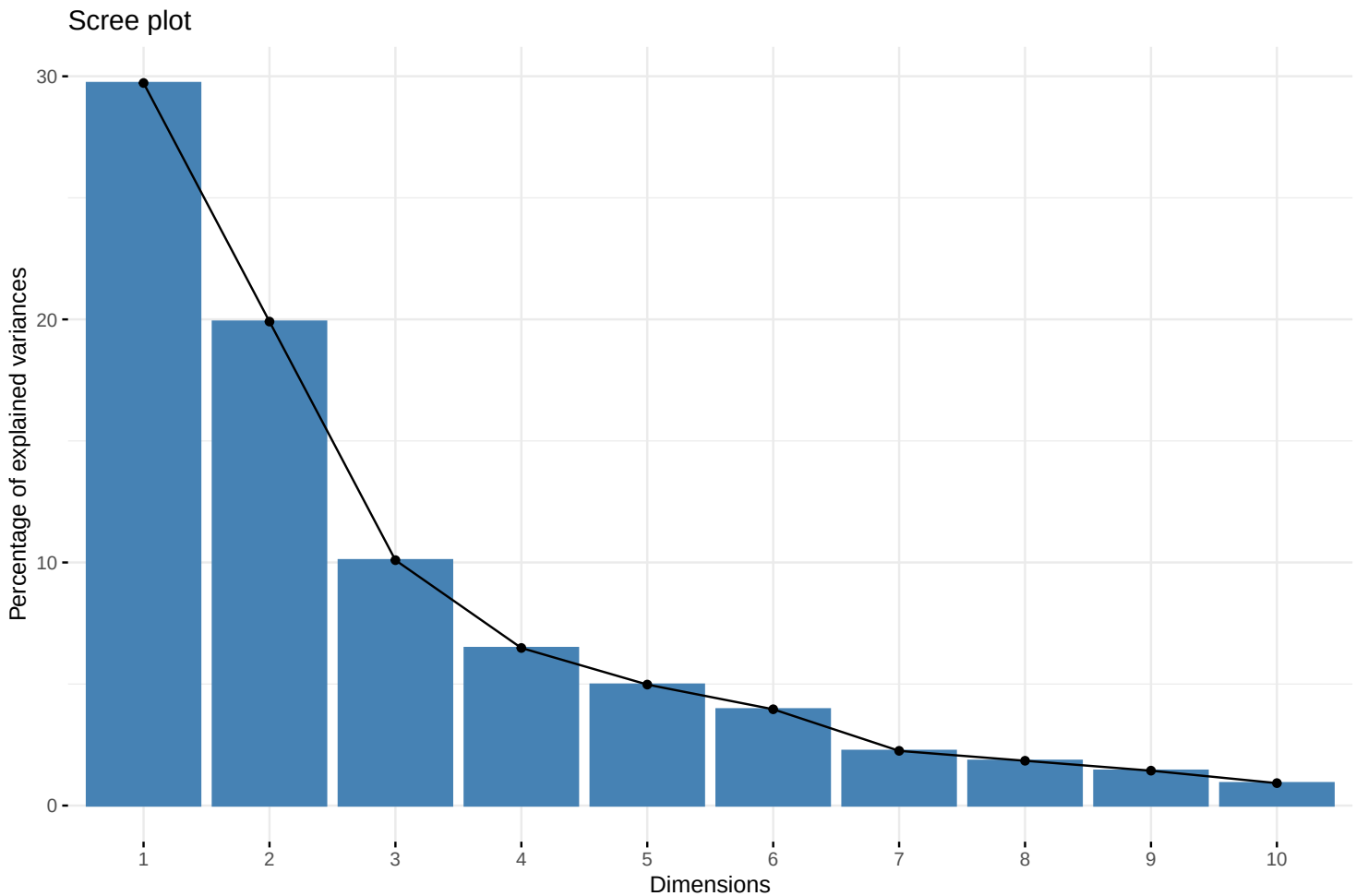
$$\text{contrib}(\Delta_1, \ldots, \Delta_k) = \frac{\lambda_1 + \cdots + \lambda_k}{\text{trace}\left(\hat{\mathbf{\Sigma}}\right)} \times 100$$

A few axes may explain a large proportion of the total variance.

$\rightsquigarrow$ This paves the way for dimension reduction

Scree plot: 'crabs'

```
scRNA_pca <- scRNA %>%
  PCA(graph = FALSE, quali.sup = which(colnames(scRNA) == "cell_type"))
fviz_eig(scRNA_pca)
```



Outline

Linear Methods

Individuals: representation in the new basis

Projection of point $\mathbf{x}_i$ axis k

The projection of $\mathbf{x}_i$ onto axis Δ_k is $c_{ik}\mathbf{v}_k$, with

$$c_{ik} = \mathbf{v}_k^\top (\mathbf{x}_i - \bar{\mathbf{x}}),$$

the coordinate of i in the basis $\mathbf{v}_k$ (along axis Δ_k).

Coordinates of i in the new basis

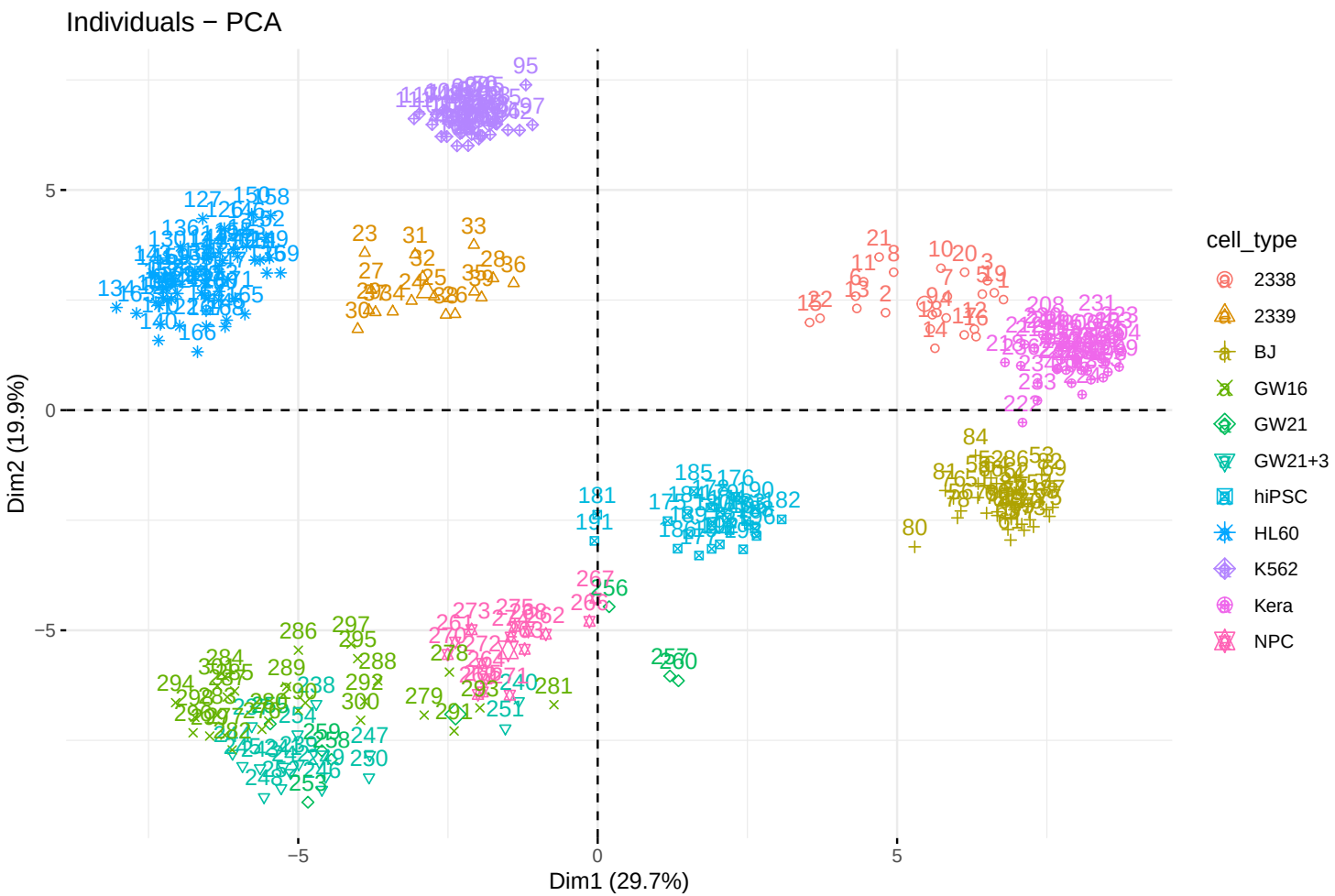
Coordinates of i in the new basis $\{\mathbf{v}_1, \ldots, \mathbf{v}_p\}$ is thus

$$\mathbf{c}_i = (\mathbf{V}^\top (\mathbf{x}_i - \bar{\mathbf{x}}))^\top = (\mathbf{x}_i - \bar{\mathbf{x}})^\top \mathbf{V} = \mathbf{X}_i^c \mathbf{V}, \quad \mathbf{c}_i \in \mathbb{R}^p.$$

- $\mathbf{V}$ are often the called the **loadings**, or **weights**
- $\mathbf{c}_i$ are the **scores** or **coordinates** in the new space for the individuals

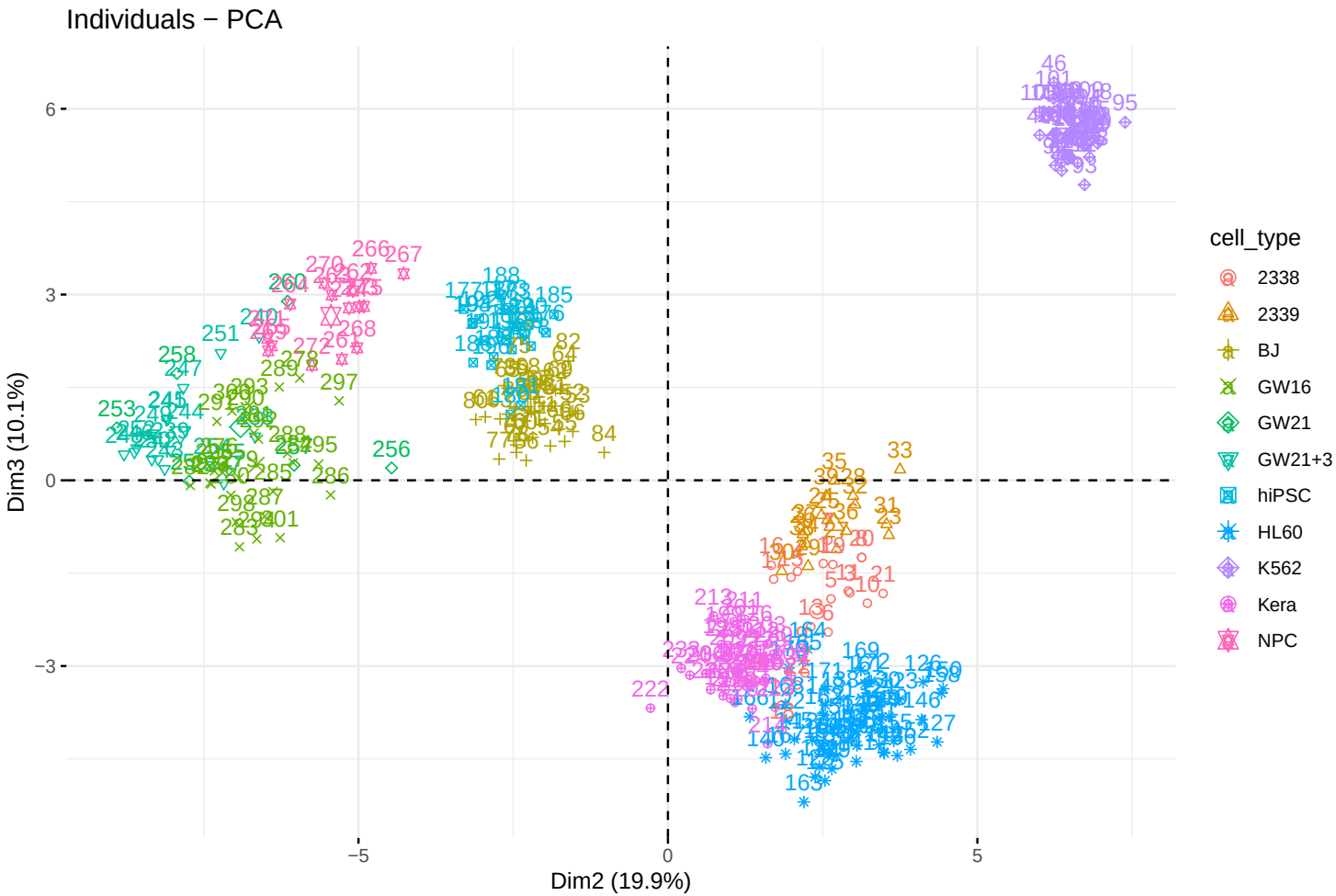
Individual visualization: projection in the new basis (1)

```
fviz_pca_ind(scRNA_pca, habillage = "cell_type")
```



Individual visualization: projection in the new basis (2)

```
fviz_pca_ind(scRNA_pca, axes = c(2,3), habillage = "cell_type")
```



Warning: about distances after projection

Close projection doesn't mean close individuals!

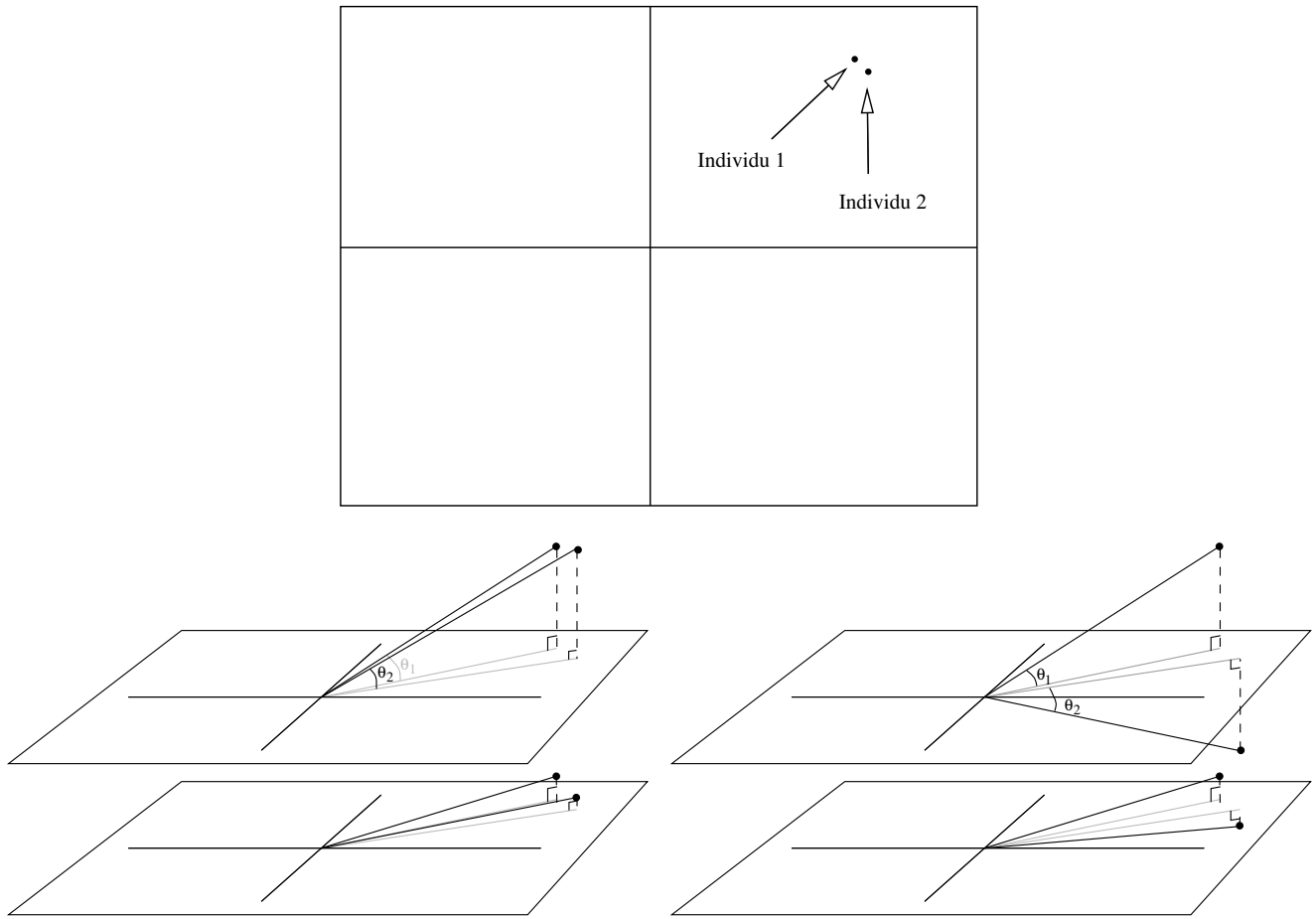


Figure: Same projections but different situations (source: E. Matzner)

$\rightsquigarrow$ Only work when individuals are well represented in the lower space

Individual: quality of the representation

Property

- An individual i is well represented by Δ_k if it is close to this axis.
- In other word, vector $\mathbf{x}_i - \bar{\mathbf{x}}$ and $\mathbf{v}_k$ are close to collinear

We use the cosine of the angle θ_{ik} between $\mathbf{x}_i - \bar{\mathbf{x}}$ and $\mathbf{v}_k$ to measure the degree of co-linearity:

$$\cos^2(\theta_{ik}) = \frac{\left(\mathbf{v}_k^\top (\mathbf{x}_i - \bar{\mathbf{x}})\right)^2}{\|\mathbf{x}_i - \bar{\mathbf{x}}\|^2 \cancel{\|\mathbf{v}_k\|^2}}$$

```
factoextra::get_pca_ind(scRNA_pca)$cos2 %>% head(3) %>% kable("latex")
```

Dim.1	Dim.2	Dim.3	Dim.4	Dim.5
0.3976361	0.0545911	0.0156510	0.0949606	0.0040849
0.1946920	0.0412816	0.0815729	0.2278256	0.0000568
0.4160489	0.0849204	0.0324573	0.0912393	0.0327544

Individual: contribution to an axis

Property

- Inertia "explained" by Δ_k is inertia of $\Delta_k^\perp$
- $I_{\Delta_k^\perp} = n^{-1} \sum_{i=1}^n \text{dist}^2(\Delta_k^\perp, \mathbf{x}_i)$

Contribution of $\mathbf{x}_i$ to axis Δ_k is the proportion of variance/inertia carried by individual i:

$$\text{contr}(\mathbf{x}_i) = \frac{n^{-1}\text{dist}^2(\Delta_k^\perp, \mathbf{x}_i)}{I_{\Delta_k^\perp}} = \frac{\left(\mathbf{v}_k^\top(\mathbf{x}_i - \bar{\mathbf{x}})\right)^2}{n\lambda_k}$$

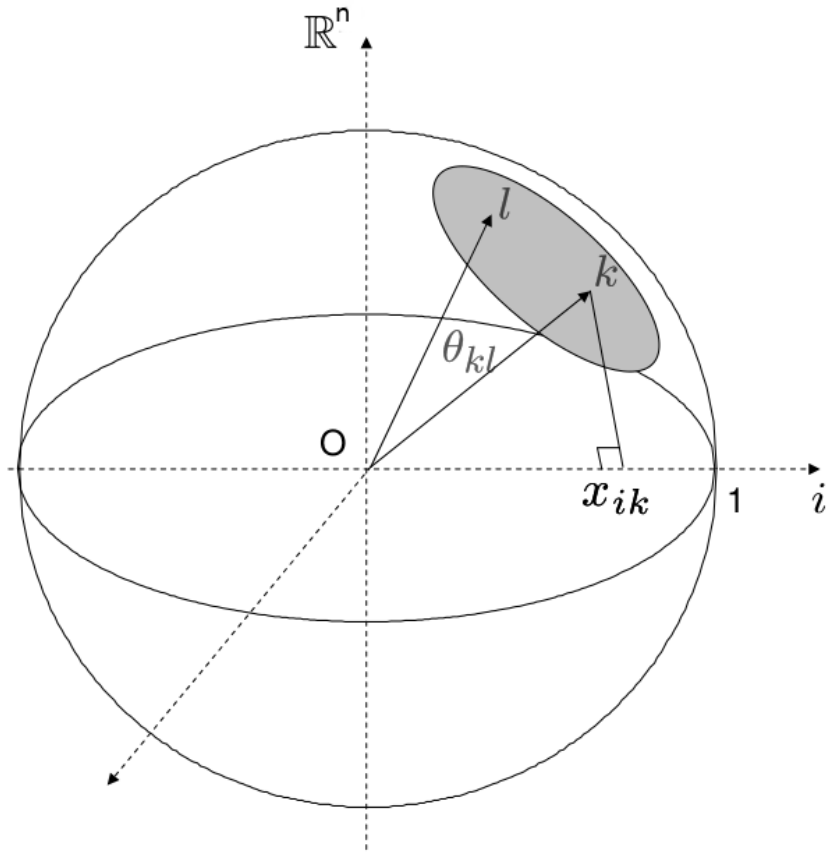
```
factoextra::get_pca_ind(scRNA_pca)$contr %>% head(3) %>% kable("latex")
```

Dim.1	Dim.2	Dim.3	Dim.4	Dim.5
0.5131474	0.1051793	0.0594716	0.5619077	0.0314858
0.2582327	0.0817469	0.3185806	1.3855779	0.0004498
0.4731939	0.1441978	0.1086970	0.4758193	0.2225046

Outline

Linear Methods

Cloud of variables in $\mathbb{R}^n$



Direct equivalence between geometry and statistics (collinearity $\equiv$ correlation)

$$\cos(\theta_{kl}) = \frac{\langle \mathbf{x}^k, \mathbf{x}^\ell \rangle}{\|\mathbf{x}^k\| \|\mathbf{x}^\ell\|} = \rho(\mathbf{x}^k, \mathbf{x}^\ell)$$

Principal Components

Dual representation

A symmetric reasoning can be made in $\mathbb{R}^n$ for the variables, like with the individuals in $\mathbb{R}^p$.
$\rightsquigarrow$ New axes are linear combinaison of the original variables, which can be seen as **new variables** in the new latent space

Principal component

It is the linear combinasion formed by the orginal variables with weights given by the loadings $\mathbf{v}_k = (v_{k1}, \ldots, v_{kj}, \ldots, v_{kp})$

$$\mathbf{f}_k = \sum_{j=1}^{p} v_{kj}(\mathbf{x}^j - \bar{x}_j) = \mathbf{X}^c \mathbf{v}_k, \quad \mathbf{f}_k \in \mathbb{R}^n$$

Sometimes called "factors" in factor analysis, as latent (hidden) variables.

Variable representation in the new space

Connection with original variables

- essential for interpretation
- answer to the question: how to read the axes of the individual map
- use correlation to measure connection to original variable

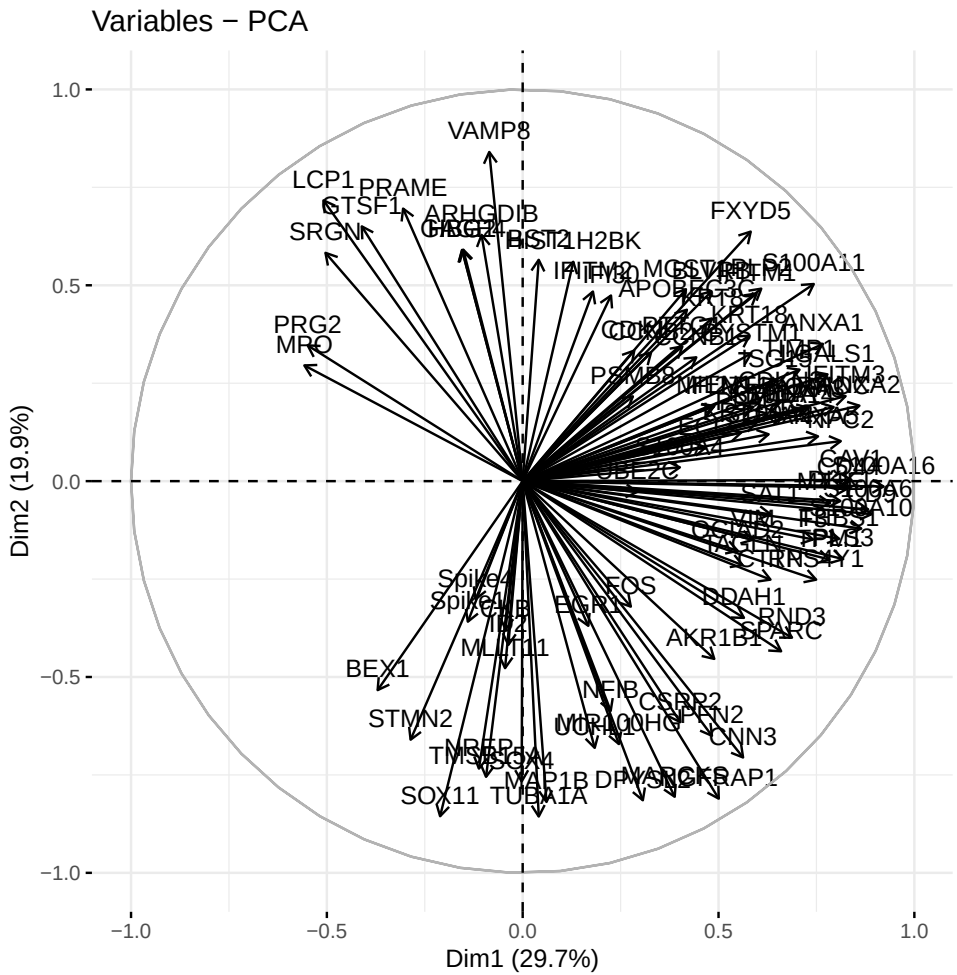
$$\mathbb{V}(\mathbf{f}_k) = \mathbf{v}_k^\top \frac{1}{n} (\mathbf{X}^c)^\top \mathbf{X}^c \mathbf{v}_k = \mathbf{v}_k^\top \hat{\mathbf{\Sigma}} \mathbf{v}_k = \lambda_k$$

$$\text{cov}(\mathbf{f}_k, (\mathbf{x}^j - \bar{x}_j)) = \frac{1}{n} \mathbf{v}_k \top \mathbf{X}^{c\top} \mathbf{X}^c e_j = \lambda_k \mathbf{v}_k^\top e_j = \lambda_k \, v_{kj}$$

$$\text{cor}(\mathbf{f}_k, (\mathbf{x}^j - \bar{x}_j)) = \sqrt{\frac{\lambda_k}{\mathbb{V}(\mathbf{x}^j)}} v_{kj}$$

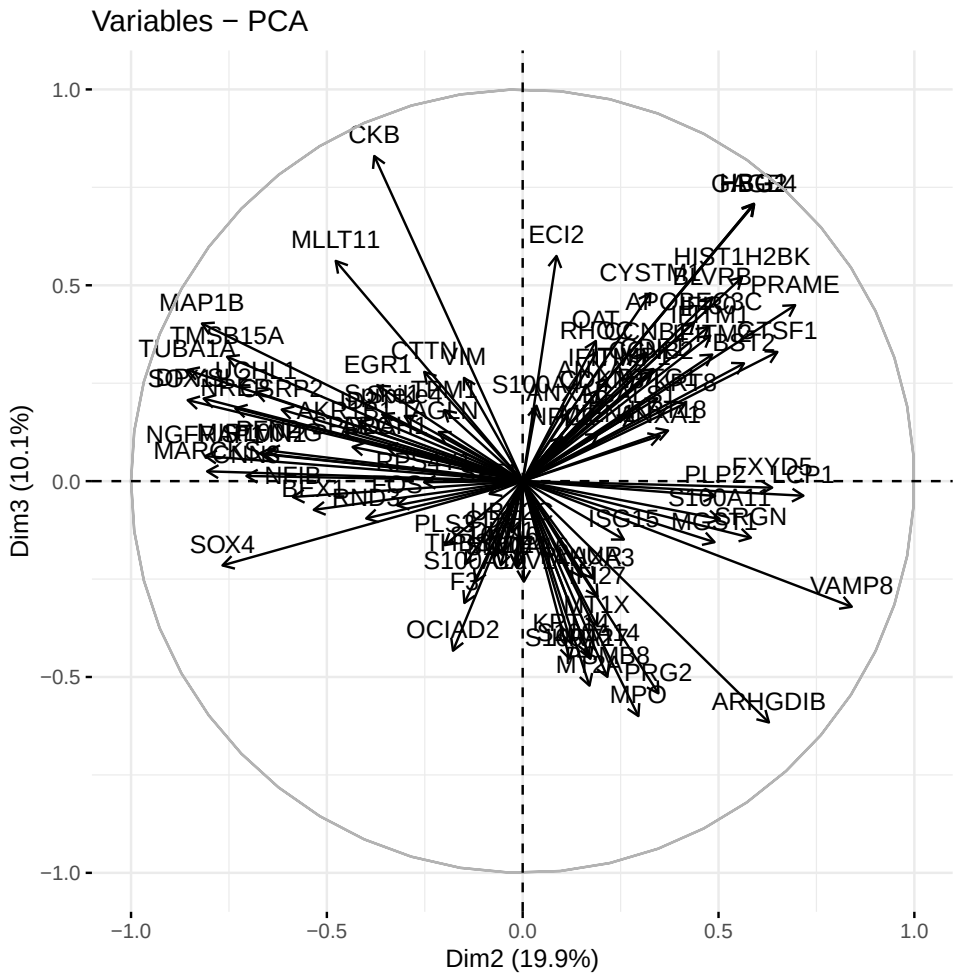
Variable vizualisation: correlation circle (1)

```
fviz_pca_var(scRNA_pca)
```



Variable vizualisation: correlation circle (2)

```
fviz_pca_var(scRNA_pca, axes = c(2,3))
```



Warning: about angle after projection

Close projection doesn't mean close variable!

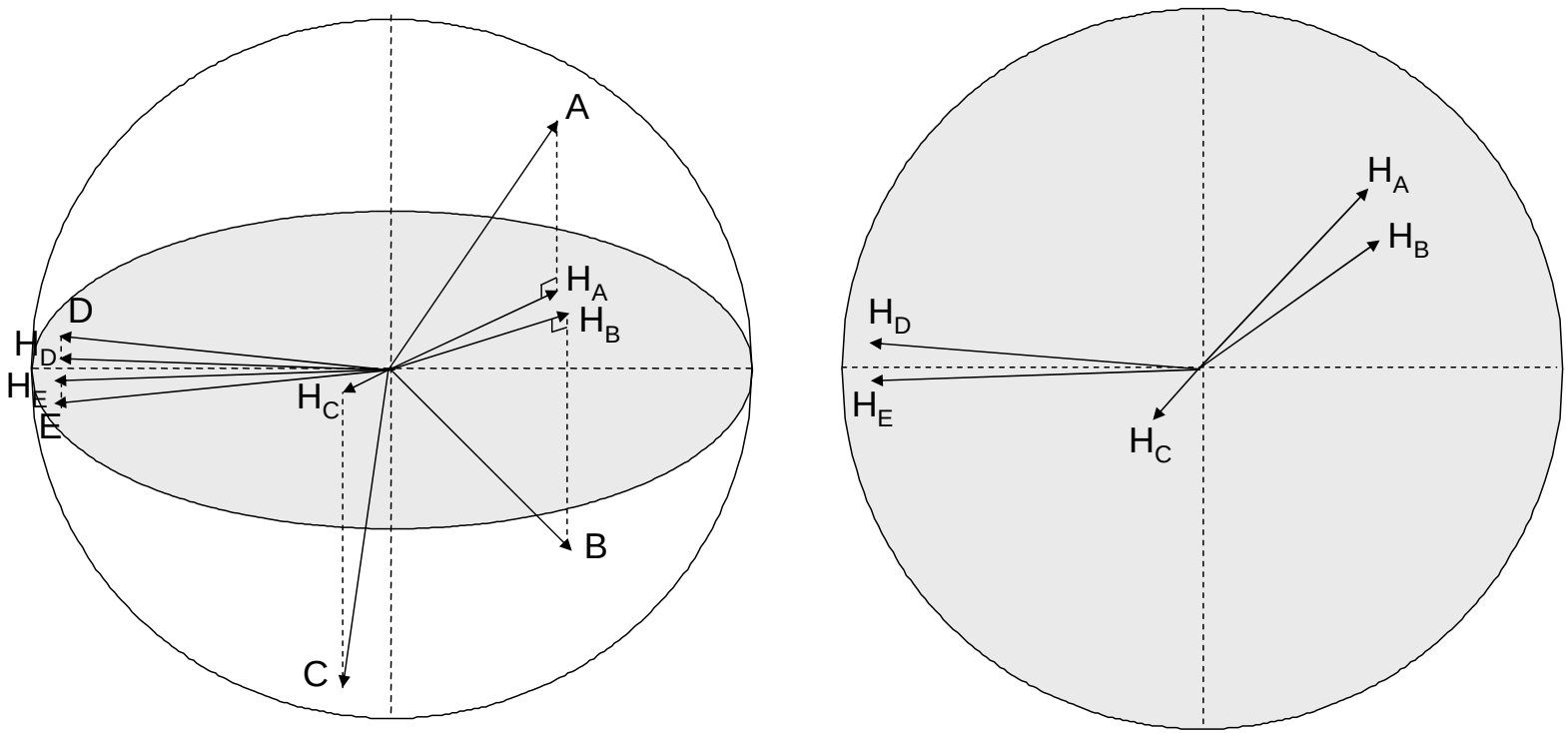


Figure: Same angle but different situations (source: J. Josse)

$\rightsquigarrow$ Only work when variables are well represented in the latent space

Variable: quality of the representation

Same story as for individuals

Property

- An variable j is well represented by Δ_k if its projection is close to $\mathbf{f}_k$.
- High collinearity means high absolute correlation and high cosine.
- use cosine to the square of the angle between the original and new variables.

$\rightsquigarrow$ The projection of j must be close to the boundady of the correlation circle

```
factoextra::get_pca_var(scRNA_pca)$cos2 %>% head(3) %>% kable("latex")
```

	Dim.1	Dim.2	Dim.3	Dim.4	Dim.5
Spike1	0.0196220	0.1287491	0.0292639	0.0206783	0.6007645
MT2A	0.4428833	0.0290404	0.2725646	0.0640107	0.0344313
HBG2	0.0238491	0.3478273	0.4996552	0.0329798	0.0343303

Variable: contribution to an axis

Similarly to individuals, we can measure the contribution of the original variables to the construction of the new ones.

```
factoextra::get_pca_var(scRNA_pca)$contr %>% kable("latex")
```

	Dim.1	Dim.2	Dim.3	Dim.4	Dim.5
Spike1	0.0660146	0.6466842	0.2898943	0.3189899	12.0718795
MT2A	1.4899972	0.1458647	2.7000781	0.9874472	0.6918700
HBG2	0.0802359	1.7470759	4.9496810	0.5087554	0.6898405
PRG2	1.0139009	0.6028304	2.9068121	1.8379608	0.0213405
IFITM1	1.2007133	1.1528680	1.3666546	0.7653552	1.4680454
ANXA1	1.9780804	0.6164234	0.1320548	0.0856922	2.6505138
HBG1	0.0807819	1.7503356	4.9690534	0.5020974	0.6898666
MPO	1.0457392	0.4382956	3.5630063	1.8958006	0.0677635
S100A6	2.6183716	0.0261683	0.3634093	0.1959209	0.4435930
TUBA1A	0.0056589	3.6825240	0.7969656	0.0007246	0.1601781
ARHGDIB	0.0371295	1.9854261	3.7599628	0.0232048	0.0015973
ANXA2	2.4874475	0.1862547	0.5291199	0.0009373	0.0726448
LGALS1	2.0381581	0.3730531	0.2802331	0.4490980	0.6752508
RPS4Y1	1.8891255	0.3179103	0.0000777	1.7898114	1.1765074
S100A11	1.8583855	1.2721098	0.0928818	0.1401025	0.0000681
IFITM3	2.2872679	0.2365565	0.6775251	0.0247328	0.2177227
S100A16	2.8571375	0.0009453	0.4875158	0.0048417	0.0865240
NGFRAP1	0.8430747	3.3003985	0.0404199	0.1031052	0.1506221

Outline

Linear Methods

Unifying view of variables and individuals

Principal components

The full matrix of principal component connects individual coordinates to latent factors:

$$\mathrm{PC} = \mathbf{X}^c\mathbf{V} = \begin{pmatrix} \mathbf{f}_1 & \mathbf{f}_2 & \dots & \mathbf{f}_p \end{pmatrix} = \begin{pmatrix} \mathbf{c}_1^\top \\ \mathbf{c}_2^\top \\ \dots \\ \mathbf{c}_n^\top \end{pmatrix}$$

- new variables (latent factor) are seen column-wise
- new coordinates are seen row-wise

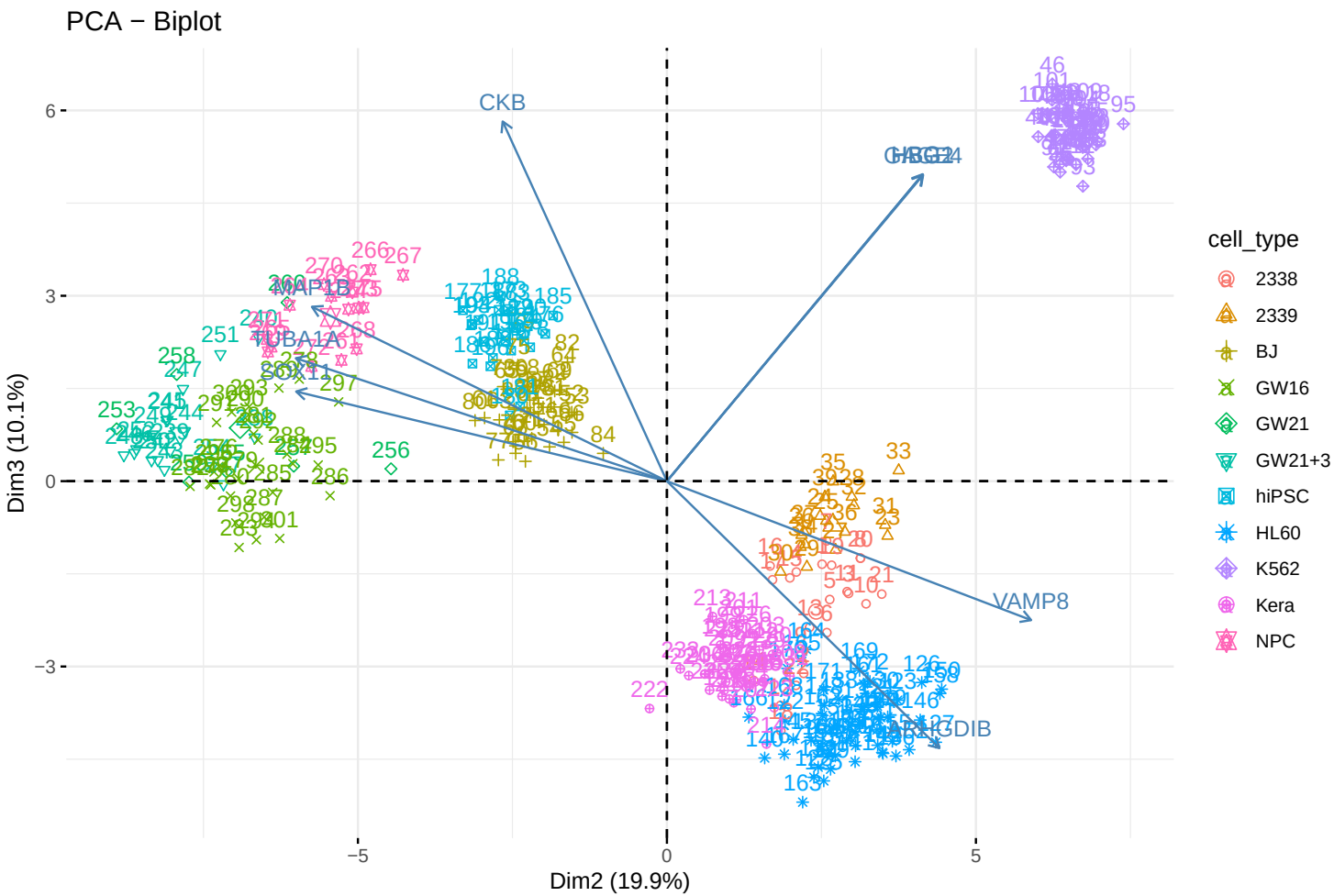
$\rightsquigarrow$ Everything can be interpreted on a single plot, called the biplot

Biplot (1)

```
factoextra::fviz_pca_biplot(scRNA_pca,
  axes = c(1,2), habillage = "cell_type",
  select.var = list(contrib = 30)
)
```

Biplot (2)

```
factoextra::fviz_pca_biplot(scRNA_pca,
  axes = c(2,3), habillage = "cell_type",
  select.var = list(cos2 = .75)
)
```



Reconstruction formula

Recall that $\mathbf{F} = (\mathbf{f}_1, \ldots, \mathbf{f}_p)$ is the matrix of Principal components. Then,

- $\mathbf{f}_k = \mathbf{X}^c \mathbf{v}_k$ for projection on axis k
- $\mathbf{F} = \mathbf{X}^c \mathbf{V}$ for all axis.

Using orthogonality of $\mathbf{V}$, we get back the original data as follows, without loss ($\mathbf{V}^T$ performs the inverse rotation of $\mathbf{V}$):

$$\mathbf{X}^c = \mathbf{F}\mathbf{V}^\top$$

We obtain an approximation $\tilde{\mathbf{X}}^c$ (compression) of the data $\mathbf{X}^c$ by considering a subset $\mathcal{S}$ of PC, typically $\mathcal{S} = 1, \ldots, q$ with $q \ll p$.

$$\tilde{\mathbf{X}}^c = \mathbf{F}_{\mathcal{S}} \mathbf{V}_{\mathcal{S}}^\top = \mathbf{X}^c \mathbf{V}_{\mathcal{S}} \mathbf{V}_{\mathcal{S}}^\top$$

$\rightsquigarrow$ This is a rank-q approximation of $\mathbf{X}$ (information captured by the first q axes).

Reconstruction formula

Recall that $\mathbf{F} = (\mathbf{f}_1, \ldots, \mathbf{f}_p)$ is the matrix of Principal components. Then,

- $\mathbf{f}_k = \mathbf{X}^c \mathbf{v}_k$ for projection on axis k
- $\mathbf{F} = \mathbf{X}^c \mathbf{V}$ for all axis.

Using orthogonality of $\mathbf{V}$, we get back the original data as follows, without loss ($\mathbf{V}^T$ performs the inverse rotation of $\mathbf{V}$):

$$\mathbf{X}^c = \mathbf{F}\mathbf{V}^\top$$

We obtain an approximation $\tilde{\mathbf{X}}^c$ (compression) of the data $\mathbf{X}^c$ by considering a subset $\mathcal{S}$ of PC, typically $\mathcal{S} = 1, \ldots, q$ with $q \ll p$.

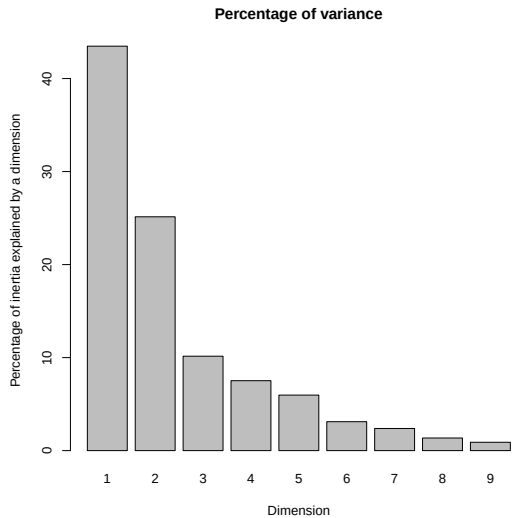
$$\tilde{\mathbf{X}}^c = \mathbf{F}_{\mathcal{S}} \mathbf{V}_{\mathcal{S}}^\top = \mathbf{X}^c \mathbf{V}_{\mathcal{S}} \mathbf{V}_{\mathcal{S}}^\top$$

$\rightsquigarrow$ This is a rank-q approximation of $\mathbf{X}$ (information captured by the first q axes).

Choosing the number of components

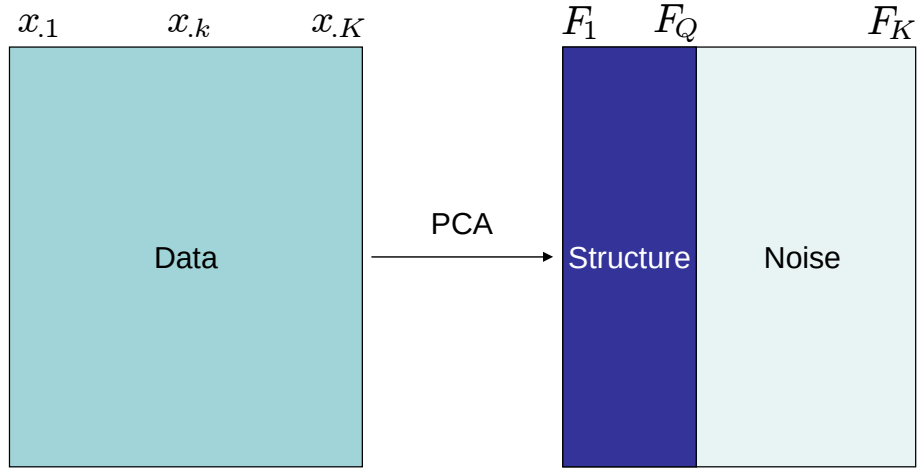
Various solutions, open question

Scree plot, test on eigenvalues, confidence interval, cross-validation, generalized cross-validation, etc.



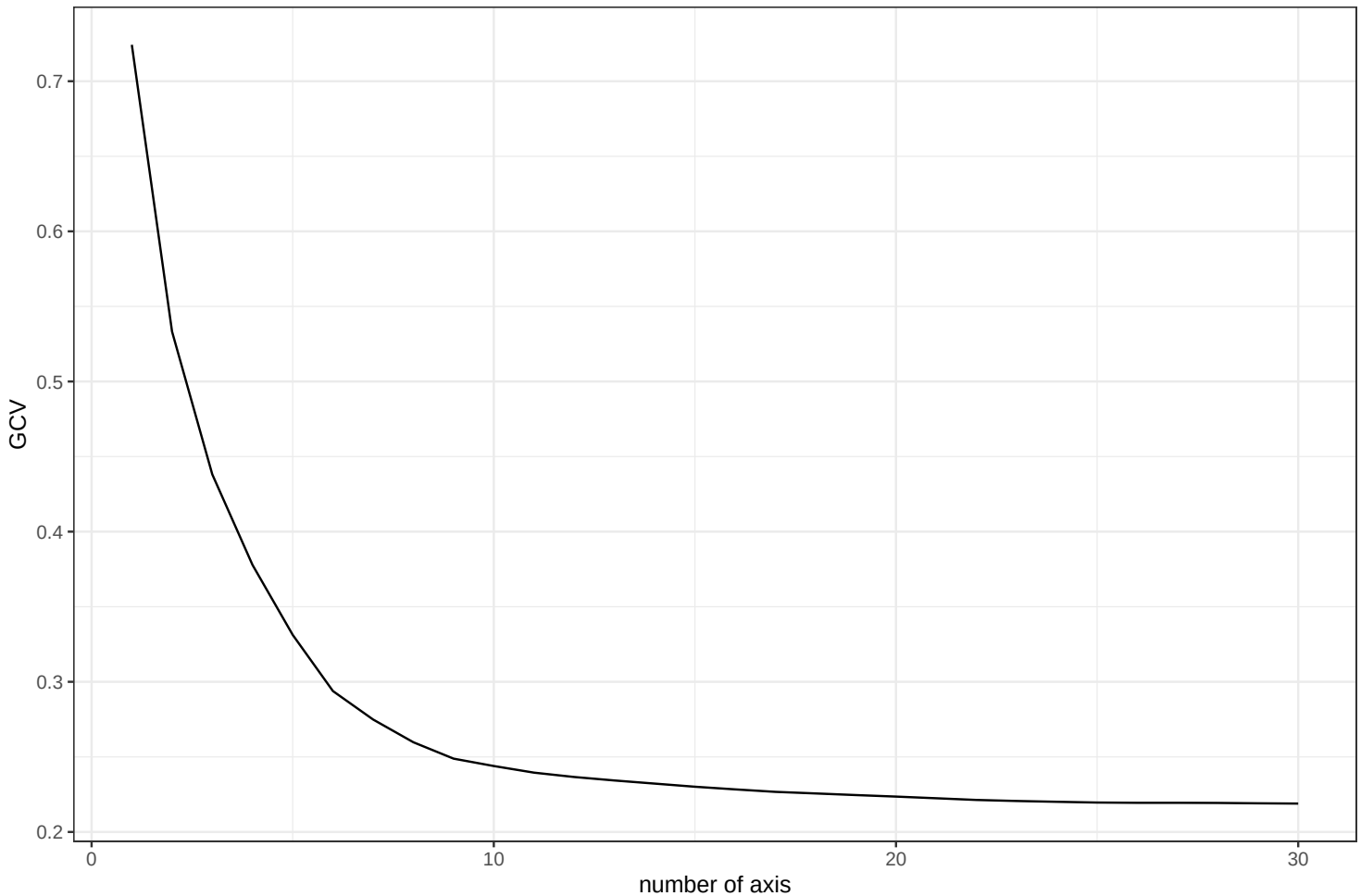
Objectives

- Interpretation
- Separate structure and noise
- Data compression



Example: Generalized Cross Validation

```
GCV <- dplyr::select(scRNA, -cell_type) %>% as.matrix() %>%
  FactoMineR::estim_ncp(ncp.min = 1, ncp.max = 30)
qplot(1:length(GCV$criterion), GCV$criterion, geom = "line", xlab = "number of axis
```



Part III

Non-linear Methods

Packages required for reproducing the slides

```
library(NMF)          # Non-Negative Matrix factorisation
library(kernlab)      # Kernel-based methods, among which kernel-PCA
library(MASS)         # Various statistical tools, including metric MDS
library(Rtsne)        # tSNE implementation in R
library(umap)         # Uniform Manifold Approximation and Projection

theme_set(theme_bw()) # my default theme for ggplot2
```

PCA (and linear methods) limitations

Do not account for complex pattern

- Linear methods are powerful for **planar structures**
- May fail at describing **manifolds**

Fail at preserving local geometry

- High dimensional data are characterized by **multiscale properties** (local / global structures)
- Non Linear projection helps at preserving **local characteristics** of distances

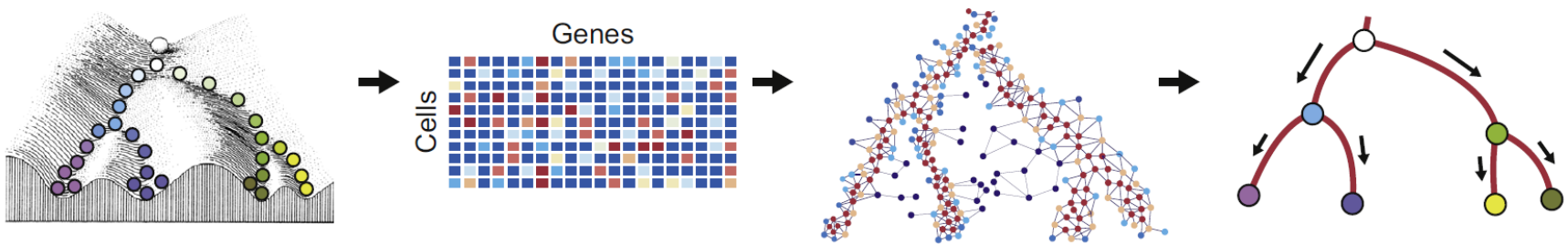


Figure: Intuition of manifolds and geometry underlying sc-data – source: F. Picard

Dimension reduction: revisiting the problem setup

Settings

- Training data : $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \in \mathbb{R}^p$, (i.i.d.)
- Space $\mathbb{R}^p$ of possibly high dimension $(n \ll p)$

Dimension Reduction Map

Construct a map Φ from the space $\mathbb{R}^p$ into a space $\mathbb{R}^q$ of smaller dimension:

$$\begin{aligned} \Phi : \quad & \mathbb{R}^p \to \mathbb{R}^q, q \ll p \\ & \mathbf{x} \mapsto \Phi(\mathbf{x}) \end{aligned}$$

How should we design/construct Φ?

Geometrical approach (**see slides on PCA**)

Idea to go beyond linear approaches

- Modify the model by amending the **reconstruction error**
- Focus on **Relationship preservation**

Form of the map Φ

- Linear or **non-linear ?**
- tradeoff between interpretability and **versatility ?**
- tradeoff between **high** or low computational resource

Outline

Non-linear Methods

Outline

Non-linear Methods

Reconstruction error approach

1. Construct a map Φ from the space $\mathbb{R}^p$ into a space $\mathbb{R}^q$ of smaller dimension:

$$\begin{aligned} \Phi: \quad & \mathbb{R}^p \to \mathbb{R}^q, q \ll p \\ & \mathbf{x} \mapsto \Phi(\mathbf{x}) = \tilde{\mathbf{x}} \end{aligned}$$

2. Construct $\tilde{\Phi}$ from $\mathbb{R}^q$ to $\mathbb{R}^p$ (reconstruction formula)
3. Control an error ϵ between $\mathbf{x}$ and its reconstruction $\hat{\mathbf{x}} = \tilde{\Phi}(\Phi(\mathbf{x}))$

For instance, the error measured with the Frobenius between the original data matrix $\mathbf{X}$ and its approximation:

$$\epsilon(\mathbf{X}, \hat{\mathbf{X}}) = \left\| \mathbf{X} - \hat{\mathbf{X}} \right\|_F^2 = \sum_{i=1}^{n} \left\| \mathbf{x}_i - \tilde{\Phi}(\Phi(\mathbf{x}_i)) \right\|^2$$

Reconstruction error approach

1. Construct a map Φ from the space $\mathbb{R}^p$ into a space $\mathbb{R}^q$ of smaller dimension:

$$\begin{aligned} \Phi: \quad & \mathbb{R}^p \to \mathbb{R}^q, q \ll p \\ & \mathbf{x} \mapsto \Phi(\mathbf{x}) = \tilde{\mathbf{x}} \end{aligned}$$

2. Construct $\tilde{\Phi}$ from $\mathbb{R}^q$ to $\mathbb{R}^p$ (reconstruction formula)
3. Control an error ϵ between $\mathbf{x}$ and its reconstruction $\hat{\mathbf{x}} = \tilde{\Phi}(\Phi(\mathbf{x}))$

For instance, the error measured with the Frobenius between the original data matrix $\mathbf{X}$ and its approximation:

$$\epsilon(\mathbf{X}, \hat{\mathbf{X}}) = \left\| \mathbf{X} - \hat{\mathbf{X}} \right\|_F^2 = \sum_{i=1}^{n} \left\| \mathbf{x}_i - \tilde{\Phi}(\Phi(\mathbf{x}_i)) \right\|^2$$

Reinterpretation of PCA

PCA model

Let $\mathbf{V}$ be a $p \times q$ matrix whose columns are of q orthonormal vectors.

$$\Phi(\mathbf{x}) = \mathbf{V}^\top(\mathbf{x} - \boldsymbol{\mu}) = \tilde{\mathbf{x}}$$
$$\mathbf{x} \simeq \tilde{\Phi}(\tilde{\mathbf{x}}) = \boldsymbol{\mu} + \mathbf{V}\tilde{\mathbf{x}}$$

$\rightsquigarrow$ Model with **Linear assumption + ortho-normality constraints**

PCA reconstruction error

$$\underset{\boldsymbol{\mu} \in \mathbb{R}^p, \mathbf{V} \in \mathcal{O}_{p,q}}{\text{minimize}} \sum_{i=1}^{n} \left\| (\mathbf{x}_i - \boldsymbol{\mu}) - \mathbf{V}\mathbf{V}^\top(\mathbf{x}_i - \boldsymbol{\mu}) \right\|^2$$

Solution (explicit)

- $\boldsymbol{\mu} = \bar{\mathbf{x}}$ the empirical mean
- $\mathbf{V}$ an orthonormal basis of the space spanned by the q first eigenvectors of the empirical covariance matrix

Reinterpretation of PCA

PCA model

Let $\mathbf{V}$ be a $p \times q$ matrix whose columns are of q orthonormal vectors.

$$\Phi(\mathbf{x}) = \mathbf{V}^\top(\mathbf{x} - \boldsymbol{\mu}) = \tilde{\mathbf{x}}$$
$$\mathbf{x} \simeq \tilde{\Phi}(\tilde{\mathbf{x}}) = \boldsymbol{\mu} + \mathbf{V}\tilde{\mathbf{x}}$$

$\rightsquigarrow$ Model with **Linear assumption + ortho-normality constraints**

PCA reconstruction error

$$\underset{\boldsymbol{\mu}\in\mathbb{R}^p, \mathbf{V}\in\mathcal{O}_{p,q}}{\text{minimize}} \sum_{i=1}^{n} \left\| (\mathbf{x}_i - \boldsymbol{\mu}) - \mathbf{V}\mathbf{V}^\top(\mathbf{x}_i - \boldsymbol{\mu}) \right\|^2$$

Solution (explicit)

- $\boldsymbol{\mu} = \bar{\mathbf{x}}$ the empirical mean
- $\mathbf{V}$ an orthonormal basis of the space spanned by the q first eigenvectors of the empirical covariance matrix

Important digression: SVD

Singular Value Decomposition (SVD)

The SVD of $\mathbf{M}$ a $n \times p$ matrix is the factorization given by

$$\mathbf{M} = \mathbf{U}\mathbf{D}\mathbf{V}^\top,$$

where $r = \min(n, p)$ and

- $\mathbf{D}_{r\times r} = \text{diag}(\delta_1, ...\delta_r)$ is the diagonal matrix of singular values.
- $\mathbf{U}$ is orthonormal, whose columns are eigen vectors of $(\mathbf{M}\mathbf{M}^T)$
- $\mathbf{V}$ is orthonormal whose columns are eigen vectors of $(\mathbf{M}^T\mathbf{M})$

$\rightsquigarrow$ Time complexity in $\mathcal{O}(npqr)$ (less when $k \ll r$ components are required)

Connection with eigen decomposition of the covariance matrix

$$\mathbf{M}^\top\mathbf{M} = \mathbf{V}\mathbf{D}\mathbf{U}^\top\mathbf{U}\mathbf{D}\mathbf{V}^\top$$
$$= \mathbf{V}\mathbf{D}^2\mathbf{V}^\top = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$$

Important digression: SVD

Singular Value Decomposition (SVD)

The SVD of $\mathbf{M}$ a $n \times p$ matrix is the factorization given by

$$\mathbf{M} = \mathbf{U}\mathbf{D}\mathbf{V}^\top,$$

where $r = \min(n, p)$ and

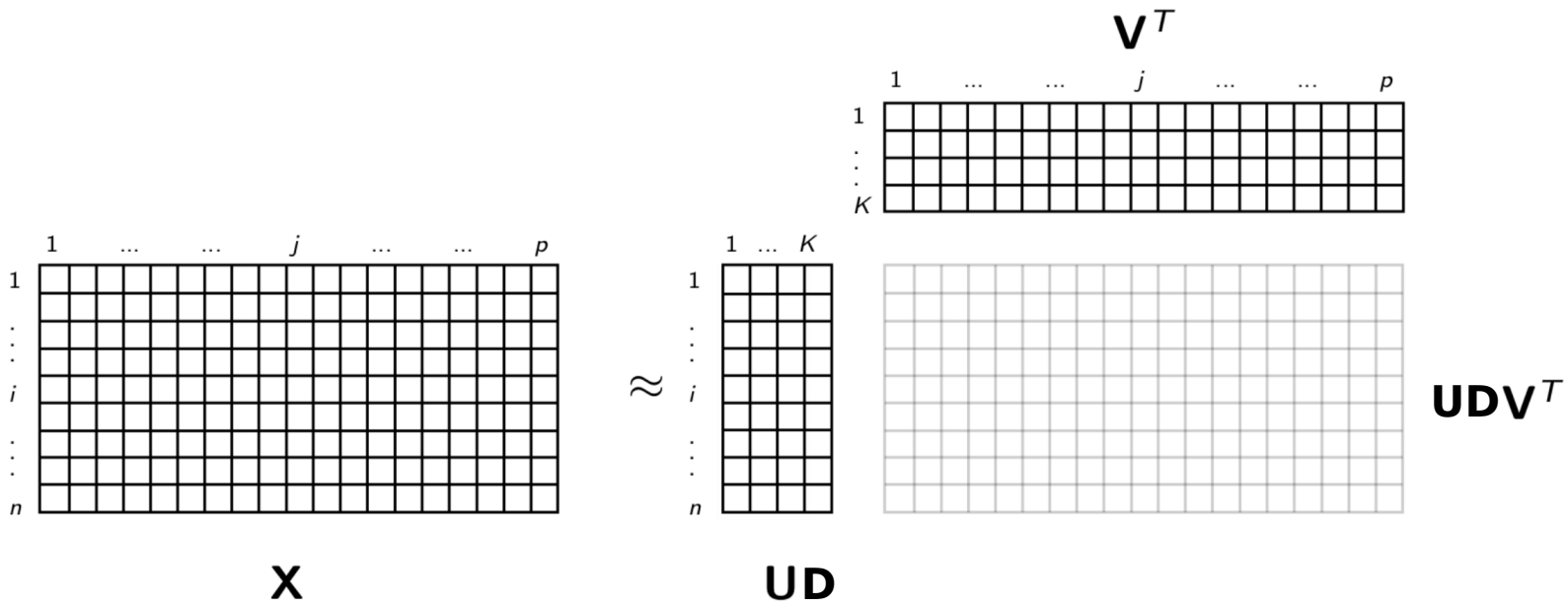
- $\mathbf{D}_{r\times r} = \text{diag}(\delta_1, ...\delta_r)$ is the diagonal matrix of singular values.
- $\mathbf{U}$ is orthonormal, whose columns are eigen vectors of $(\mathbf{M}\mathbf{M}^T)$
- $\mathbf{V}$ is orthonormal whose columns are eigen vectors of $(\mathbf{M}^T\mathbf{M})$

$\rightsquigarrow$ Time complexity in $\mathcal{O}(npqr)$ (less when $k \ll r$ components are required)

Connection with eigen decomposition of the covariance matrix

$$\begin{aligned}\mathbf{M}^\top\mathbf{M} &= \mathbf{V}\mathbf{D}\mathbf{U}^\top\mathbf{U}\mathbf{D}\mathbf{V}^\top \\ &= \mathbf{V}\mathbf{D}^2\mathbf{V}^\top = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top\end{aligned}$$

PCA solution is given by SVD of the centered data matrix



Since $\tilde{\mathbf{X}} = \mathbf{X}^c\mathbf{V} = \mathbf{U}\mathbf{D}\mathbf{V}^\top\mathbf{V} = \mathbf{U}\mathbf{D}$, PCA can be rephrased as

$$\hat{\mathbf{X}^c} = \mathbf{F}\mathbf{V}^\top = \underset{\mathbf{F}\in\mathcal{M}_{n,q},\mathbf{V}\in\mathcal{O}_{p,q}}{\arg\min} \left\|\mathbf{X}^c - \mathbf{F}\mathbf{V}^\top\right\|_F^2 \text{ with } \|\mathbf{A}\|_F^2 = \sum_{ij} a_{ij}^2,$$

$$\left.\tilde{\mathbf{X}} \in \mathbb{R}^{n\times q}, \mathbf{V} \in \mathbb{R}^{p\times q}\right\} \text{ Best linear low-rank representation of } \mathbf{X}$$

Outline

Non-linear Methods

Kernel-PCA

Principle: non linear transformation of $\mathbf{x}$ prior to linear PCA

1. Project the data into a higher space where it is linearly separable
2. Apply PCA to the transformed data

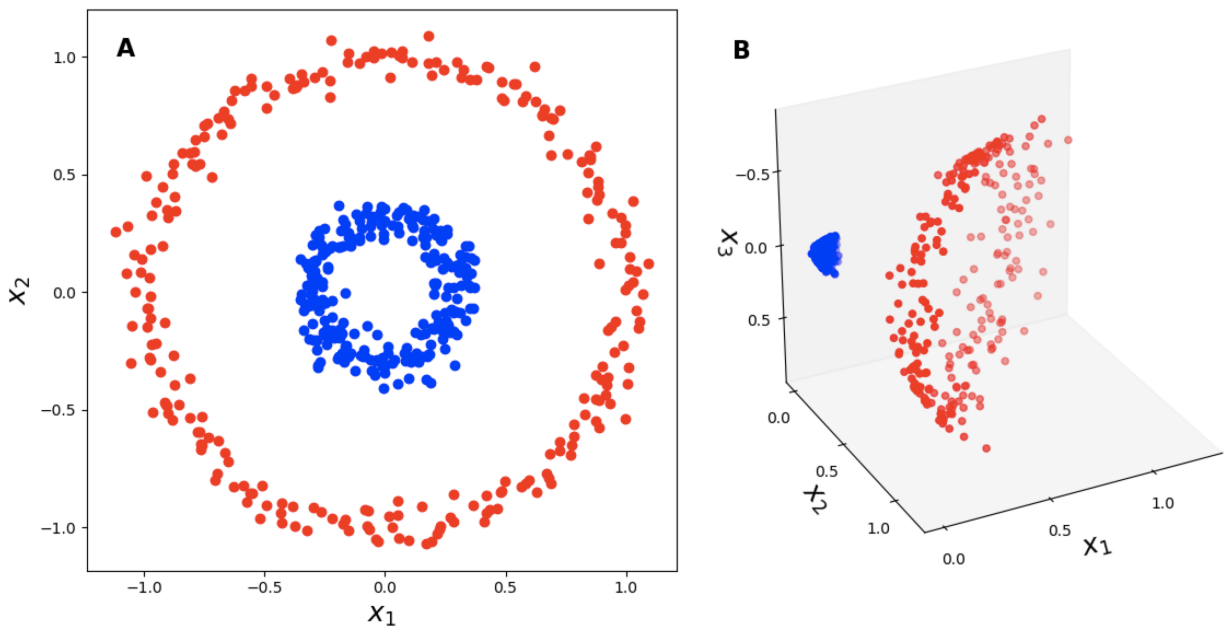


Figure: Transformation $\Psi : \mathbf{x} \rightarrow \Psi(\mathbf{x})$ (illustration in presence of existing labels)

Kernel-PCA

Kernel PCA Model

Assume a non linear transformation $\Psi(\mathbf{x}_i)$ where $\Psi : \mathbb{R}^p \to \mathbb{R}^n$, then perform linear PCA, with $\mathbf{V}$ a $n \times q$ orthonormal matrix

$$\Phi(\mathbf{x}) = \mathbf{V}^\top \Psi(\mathbf{x} - \boldsymbol{\mu}) = \tilde{\mathbf{x}}$$

Kernel trick

Never calculate $\Psi(\mathbf{x}_i)$ thanks to the kernel trick:

$$K = k(\mathbf{x}, \mathbf{y}) = (\Psi(\mathbf{x}), \Psi(\mathbf{y})) = \Psi(\mathbf{x})^T \Psi(\mathbf{y})$$

Solution

Eigen-decomposition of the doubly centered kernel matrix $\mathbf{K} = k(\mathbf{x}_i, \mathbf{x}_{i'})$

$$\tilde{\mathbf{K}} = (\mathbf{I} - \mathbf{1}\mathbf{1}^\top/n)\mathbf{K}(\mathbf{I} - \mathbf{1}\mathbf{1}^\top/n) = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^\top$$

Choice of a kernel

A symmetric positive definite function $k(\mathbf{x}, \mathbf{y}) \in \mathbb{R}$, which depends on the kind of **similarity** assumed

Some common kernels

- **Polynomial Kernel**

$$k(\mathbf{x}_i, \mathbf{x}_{i'}) = (\mathbf{x}_i^\top \mathbf{x}_{i'} + c)^d$$

- **Gaussian (radial) kernel**

$$k(\mathbf{x}_i, \mathbf{x}_{i'}) = \exp \frac{-\|\mathbf{x}_i - \mathbf{x}_{i'}\|^2}{2\sigma^2}$$

- **Laplacian kernel**

$$k(\mathbf{x}_i, \mathbf{x}_{i'}) = \exp \frac{-\|\mathbf{x}_i - \mathbf{x}_{i'}\|}{\sigma}$$

$\rightsquigarrow$ Kernel PCA suffers from the choice of the Kernel

Example on scRNA

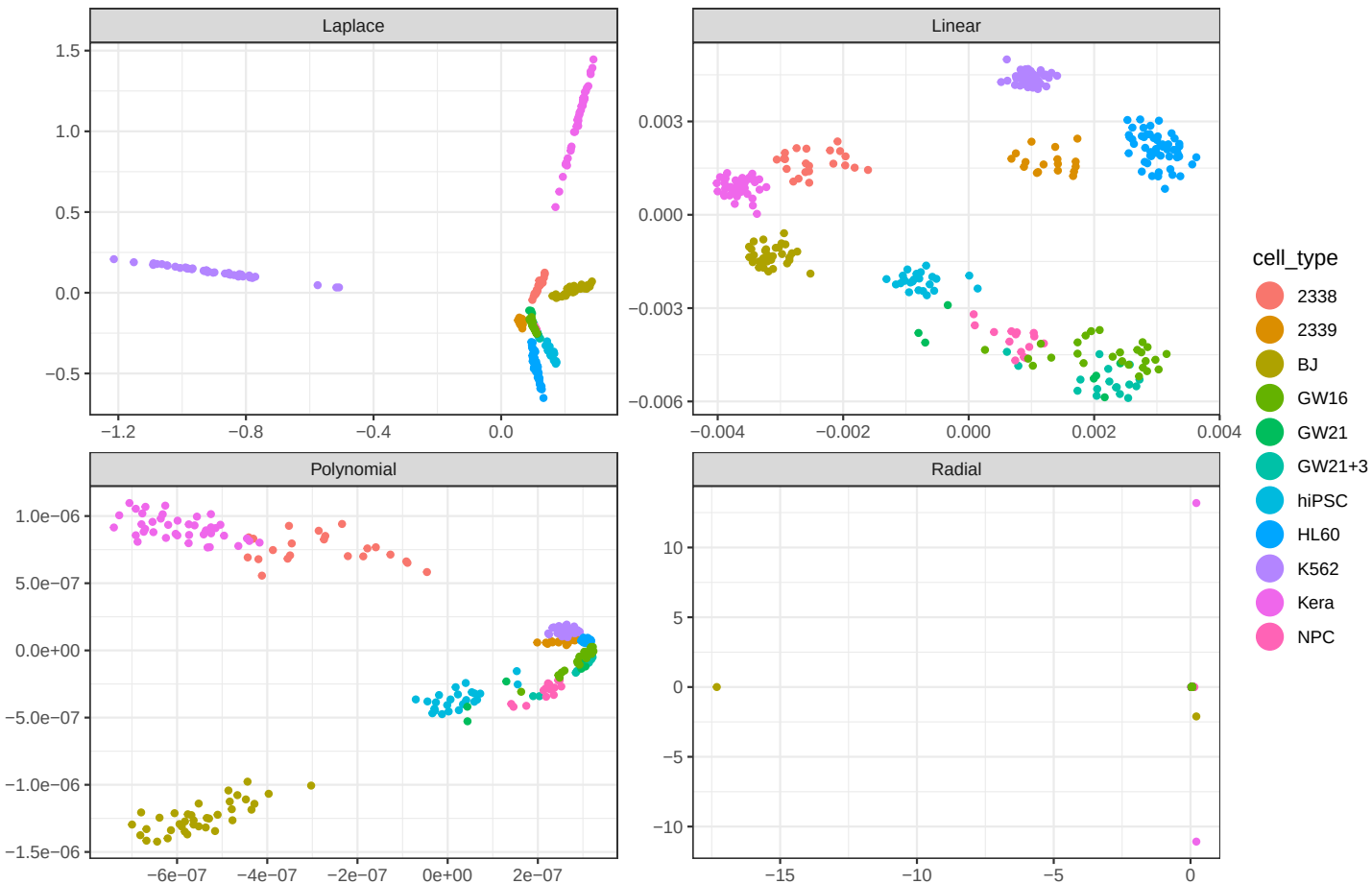
Run the fit

```
scRNA_expr <- scRNA %>% dplyr::select(-cell_type) %>% as.matrix()

kPCA_radial <-
  kpca(scRNA_expr, kernel = "rbfdot", features = 2, kpar = list(sigma = 0.5)) %>%
  pcv() %>% as.data.frame() %>%
  add_column(kernel = "Radial") %>%
  add_column(cell_type = scRNA$cell_type)

kPCA_linear <-
  kpca(scRNA_expr, kernel = "vanilladot", features = 2, kpar = list()) %>%
  pcv() %>% as.data.frame() %>%
  add_column(kernel = "Linear") %>%
  add_column(cell_type = scRNA$cell_type)

kPCA_polydot <- kpca(scRNA_expr, kernel = "polydot", features = 2, kpar = list(degr
  pcv() %>% as.data.frame() %>%
  add_column(kernel = "Polynomial") %>%
  add_column(cell_type = scRNA$cell_type)

kPCA_laplacedot <- kpca(scRNA_expr, kernel = "laplacedot", features = 2) %>%
  pcv() %>% as.data.frame() %>%
  add_column(kernel = "Laplace") %>%
  add_column(cell_type = scRNA$cell_type)
```

Example on scRNA

Compare the projections

```
rbind(kPCA_linear, kPCA_polydot, kPCA_radial, kPCA_laplacedot) %>%
  ggplot(aes(x = V1, y = V2, color = cell_type)) +
  geom_point(size=1.25) + guides(colour = guide_legend(override.aes = list(size=6))) +
  facet_wrap(.~kernel, scales = 'free') + labs(x = '', y = '')
```



Outline

Non-linear Methods

Other approaches

Linear model with other constraints

Let $\mathbf{V}$ be a $p \times q$ matrix and $\tilde{\mathbf{x}} \in \mathbb{R}^q$

$$\mathbf{x} \simeq \boldsymbol{\mu} + \sum_{j=1}^{q} \tilde{x}^j \mathbf{V}^j = \boldsymbol{\mu} + \mathbf{V}\tilde{\mathbf{x}}$$

Apply other constraints on $\mathbf{V}$ and or the factor/representation $\tilde{\mathbf{x}}$

- $\mathbf{V}$ and $\tilde{\mathbf{x}}$ non-negative: **Non-negative Matrix Factorization**

```
library(NMF)
```

- $\mathbf{V}$ sparse, possibly orthogonal: **sparse PCA**

```
library(sparsepca)
```

- $\tilde{\mathbf{x}}$ sparse : **Dictionary learning**

```
library(SPAMS)
```

- $(\tilde{X}^j, \tilde{X}^\ell)$ independent : **Independent Component Anaysis**

```
library(fastICA)
```

Auto-encoders

Highly non-linear model

Find Φ and $\tilde{\Phi}$ with **two** neural-networks, controlling the error.

$$\epsilon(\mathbf{X}, \hat{\mathbf{X}}) = \sum_{i=1}^{n} \left\| \mathbf{x}_i - \tilde{\Phi}(\Phi(\mathbf{x}_i)) \right\|^2 + \text{regularization}(\mathbf{\Phi}, \tilde{\mathbf{\Phi}})$$

- # layers and neurons determine the **model complexity**
- Need regularization to avoid **overfitting**
- Fitted with optimization tools like stochastic gradient descent
- Require much more data and more computational resources
- **Interpretation questionable**

Some Python equivalents of (torch, pytorch, tensorflow):

```
library(keras)
library(torch)
```

Outline

Non-linear Methods

Pairwise Relation

Focus on pairwise relation $\mathcal{R}(\mathbf{x}_i, \mathbf{x}_{i'})$.

Distance Preservation

- Construct a map Φ from the space $\mathbb{R}^p$ into a space $\mathbb{R}^q$ of smaller dimension:

$$\begin{aligned} \Phi: \quad & \mathbb{R}^p \to \mathbb{R}^q, q \ll p \\ & \mathbf{x} \mapsto \Phi(\mathbf{x}) \end{aligned}$$

$$\text{such that} \quad \mathcal{R}(\mathbf{x}_i, \mathbf{x}_{i'}) \sim \mathcal{R}'(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_{i'})$$

Multidimensional scaling

Try to preserve inner product related to the distance (e.g. Euclidean)

t-SNE – Stochastic Neighborhood Embedding

Try to preserve relations with close neighbors with Gaussian kernel

Outline

Non-linear Methods

Stochastic Neighbor Embedding (SNE)

Let $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ be the original points in $\mathbb{R}^p$, and measure similarities by

$$p_{ij} = (p_{j|i} + p_{i|j})/2n$$

where

$$\begin{aligned} p_{j|i} &= \frac{\exp(-\|\mathbf{x}_j - \mathbf{x}_i\|^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|\mathbf{x}_k - \mathbf{x}_i\|^2/2\sigma_i^2)}, \\ &= \frac{\exp(-d_{ij}^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-d_{ki}^2/2\sigma_i^2)} \end{aligned}$$

- $\rightsquigarrow$ SNE preserves relations with **close neighbors** with Gaussian kernels
- $\rightsquigarrow$ σ smooths the data (linked to the regularity of the target manifold)

The perplexity parameter

The variance σ_i^2 should adjust to local densities (neighborhood of point i)

Perplexity: a smoothed effective number of neighbors

The perplexity is defined by

$$Perp(p_i) = 2^{H(p_i)}, \qquad H(p_i) = -\sum_{j=1}^{n} p_{j|i} \log_2 p_{j|i}$$

where H is the Shannon entropy of $p_i = (p_{1|i}, \ldots, p_{n|i})$.

$\rightsquigarrow$ SNE performs a binary search for the value of σ_i that produces a p_i with a fixed perplexity that is specified by the user.

tSNE and Student / Cauchy kernels

Consider $(\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_n)$ are points in the low dimensional space $\mathbb{R}^{q=2}$

- Consider a similarity between points in the new representation:

$$q_{i|j} = \frac{\exp(-\|\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_j\|^2)}{\sum_{k \neq i} \exp(-\|\tilde{\mathbf{x}}_k - \tilde{\mathbf{x}}_j\|^2)}$$

- Robustify this kernel by using Student(1) kernels (ie Cauchy)

$$q_{i|j} = \frac{(1 + \|\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_j\|^2)^{-1}}{\sum_{k \neq i} (1 + \|\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_k\|^2)^{-1}}$$

t-SNE: pros/cons

Properties

- good at preserving local distances (intra-cluster variance)
- not so good for global representation (inter-cluster variance)
- good at creating clusters of close points, bad at positioning clusters wrt each other

Limitations

- importance of preprocessing: initialize with PCA and feature selection plus log transform (non linear transform)
- percent of explained variance ? interpretation of the q distribution ?

Example on scRNA I

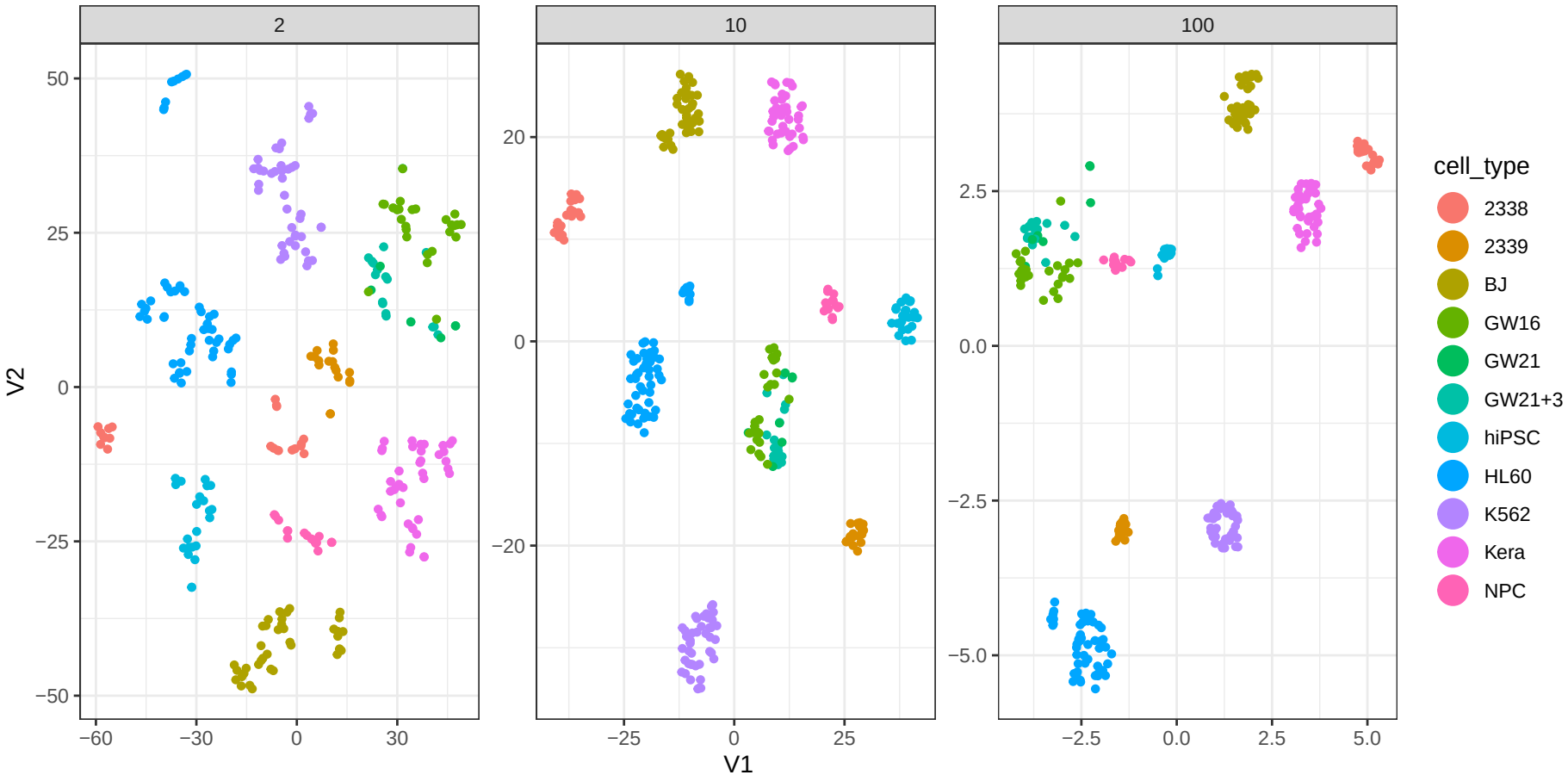
Run the fit

```
scRNA_expr <- scRNA %>% dplyr::select(-cell_type) %>% as.matrix()

tSNE_perp2   <- Rtsne(scRNA_expr, perplexity =   2)$Y %>%
  as.data.frame() %>% add_column(perplexity = 2) %>% add_column(cell_type = scRNA$c

tSNE_perp10  <- Rtsne(scRNA_expr, perplexity =  10)$Y %>%
  as.data.frame() %>% add_column(perplexity = 10) %>% add_column(cell_type = scRNA$

tSNE_perp100 <- Rtsne(scRNA_expr, perplexity = 100)$Y %>%
  as.data.frame() %>% add_column(perplexity = 100) %>% add_column(cell_type = scRNA
```

Compare perplexity

```
rbind(tSNE_perp2,tSNE_perp10,tSNE_perp100) %>%
  ggplot(aes(x = V1, y = V2, color = cell_type)) +
     geom_point(size=1.25) +
     guides(colour = guide_legend(override.aes = list(size=6))) +
  facet_wrap(.~perplexity, scales = 'free')
```

Example on scRNA II



Outline

Non-linear Methods

Multidimensional scaling

a.k.a Principale Coordinates Analysis

Problem setup

Consider a collection of points $\mathbf{x}_i \in \mathbb{R}^p$ and assume either

- $D = d_{ii'}$ a $n \times n$ dissimilarity matrix, or
- $S = s_{ii'}$ a $n \times n$ similarity matrix, or

Goal: find $\tilde{\mathbf{x}}_i \in \mathbb{R}^q$ while preserving S/D in the latent space

$\rightsquigarrow$ Don't need access to the position in $\mathbb{R}^p$ (only D or S $\rightsquigarrow$ 'kernel').

Classical MDS model

Measure similarities with the (centered) **inner product** and minimize

$$\sum_{i \neq i'} \left((\mathbf{x}_i - \boldsymbol{\mu})^\top (\mathbf{x}_i - \boldsymbol{\mu}) - \tilde{\mathbf{x}}_i^\top \tilde{\mathbf{x}}_{i'} \right)^2,$$

assuming a linear model $\tilde{\mathbf{x}} = \mathbf{V}^\top (\mathbf{x}_i - \boldsymbol{\mu})$, with $\mathbf{V} \in \mathcal{O}_{p \times q}$.

Isomap

Basic idea

- Metric MDS performs embedding based on pairwise Euclidean-based distance
- Isomap embeds a distance induced by a neighborhood graph

Formally, consider a neighborhood $\mathcal{N}_i$ for each point, then

$$d_{ii'} = \begin{cases} +\infty & \text{if } j \notin \mathcal{N}_i \\ \|\mathbf{x}_i - \mathbf{x}_{i'}\| & \end{cases},$$

and compute the shortest path distance for each pair prior to MDS.

```
library(vegan)
```

Uniform Manifold Approximation and Projection I

- Use another distance based of $k-$neighborhood graph
- tends to preserve both local and glocal

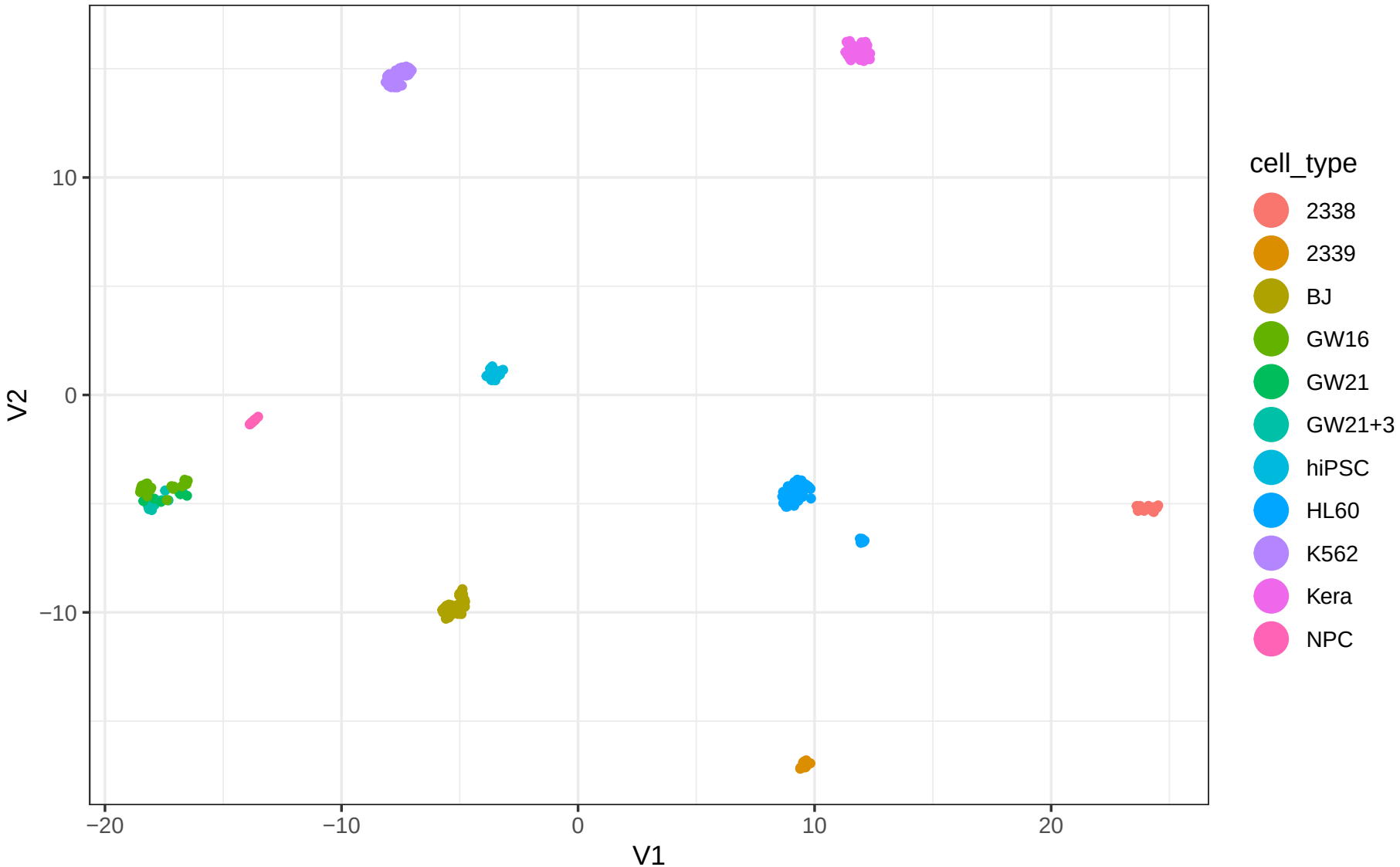
Run the fit on scRNA

```
scRNA_expr <- scRNA %>% dplyr::select(-cell_type) %>% as.matrix()
umap_fit   <- umap(scRNA_expr)$layout %>%
  as.data.frame() %>% add_column(cell_type = scRNA$cell_type)
```

Visualization

```
umap_fit %>%
  ggplot(aes(x = V1, y = V2, color = cell_type)) +
     geom_point(size=1.25) +
     guides(colour = guide_legend(override.aes = list(size=6)))
```

Uniform Manifold Approximation and Projection II



Uniform Manifold Approximation and Projection III

To conclude

You can play online on https://projector.tensorflow.org/