Data anaysis and Unsupervised Learning

Dimensionality Reduction: Beyond PCA and Non Linear Methods

MAP 573, 2020 – Julien Chiquet

École Polytechnique, Autumn semester, 2020

https://jchiquet.github.io/MAP573

Part I

Introduction

Packages required for reproducing the slides

```
library(tidyverse)  # opinionated collection of packages for data manipulation
library(FactoMineR) # PCA and oter linear method for dimension reduction
library(factoextra) # fancy plotting for FactoMineR output
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
```

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.

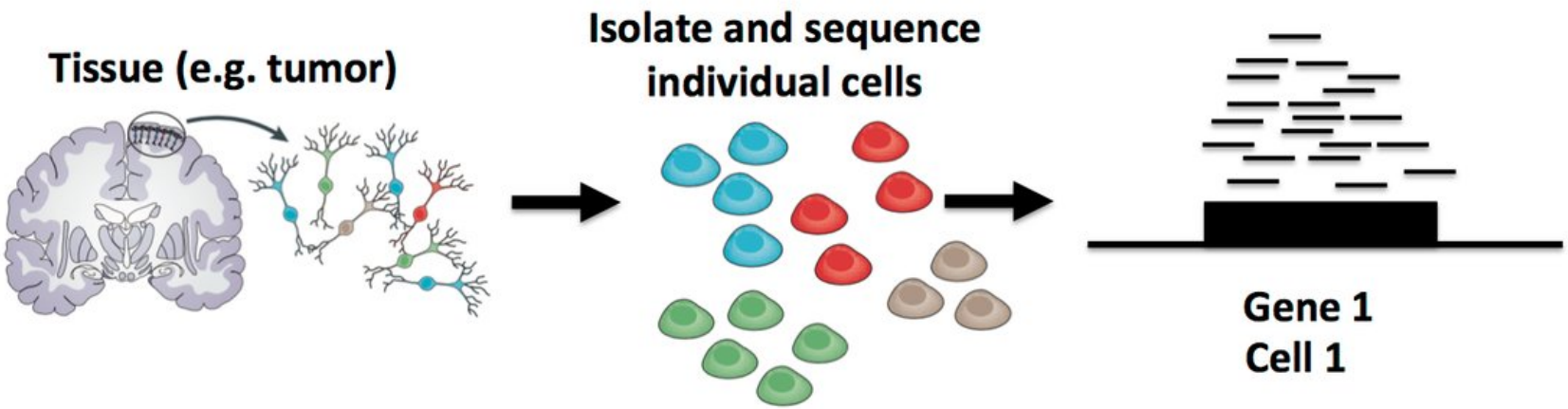


Figure: Single Cell RNA sequnencing data: general principle – source: Stephanie Hicks

Companion data set: 'scRNA'

Brief data summary I

Data manipulation

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

Cell types

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

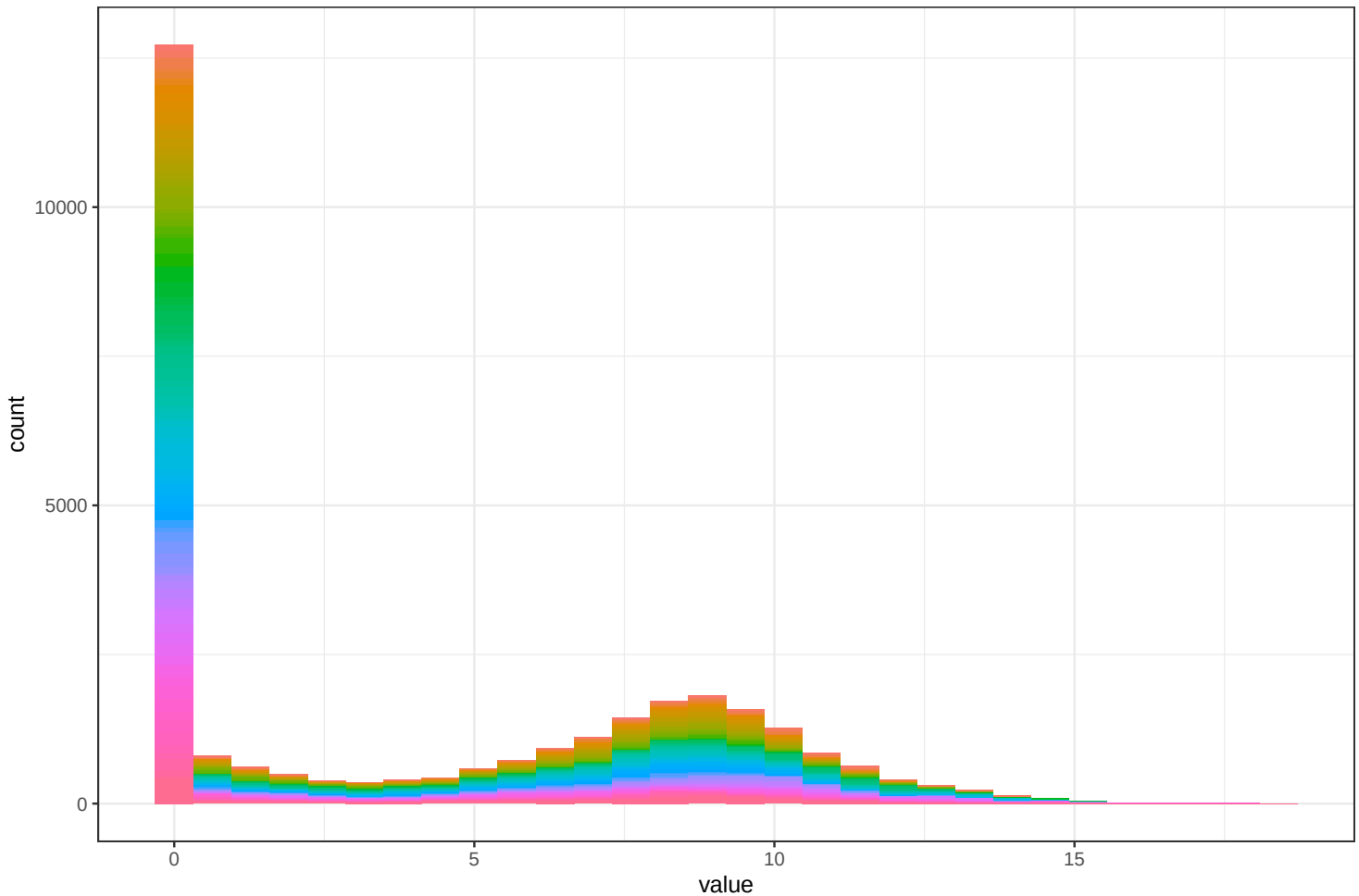
	cell_type
	HL60 :54
	K562 :42
	Kera :40
	BJ :37
	GW16 :26
	hiPSC :24
	(Other):78

Companion data set II: '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'

Principal Component Analysis

```
scRNA %>% PCA(graph = FALSE, quali.sup = which(colnames(scRNA) == "cell_type")) %>%
  fviz_pca_biplot(select.var = list(contrib = 30), habillage = "cell_type")
```

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

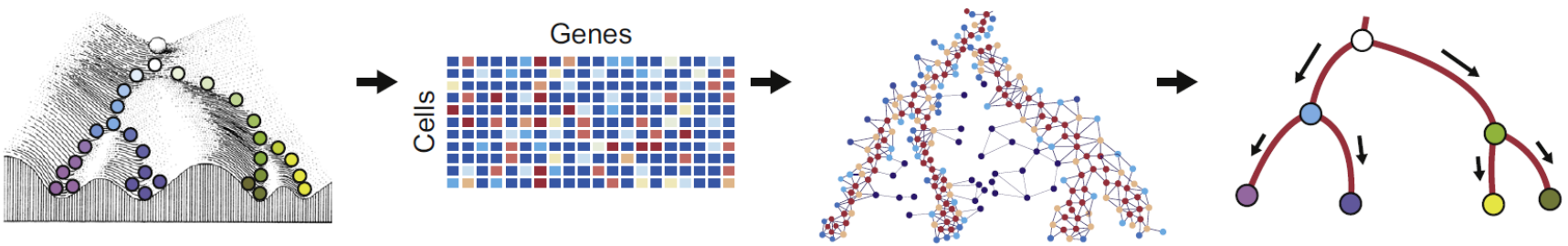


Figure: Intuition of manifolds and geometry underlying sc-data – source: F. Picard

Companion data set II: 'mollusk'

Abundance table (Species counts spread in various sites)

Description: *small size count data*

Abundance of 32 mollusk species in 163 samples. For each sample, 4 additional covariates are known.

Richardot-Coulet, M., Chessel D. and Bournaud M. Typological value of the benthos of old beds of a large river. Methodological approach. Archiv fùr Hydrobiologie, 107.

```
library(PLNmodels); data(mollusk)
mollusk <-
  prepare_data(mollusk$Abundance, mollusk$Covariate[c("season", "site")]) %>%
  dplyr::select(-Offset) %>% as_tibble() %>%  distinct() # remove duplicates
mollusk <- cbind(mollusk$Abundance, mollusk[c("season", "site")])
```

External Covariates

```
mollusk %>% dplyr::select(site, season) %>% summary() %>% t() %>% knitr::kable()
```

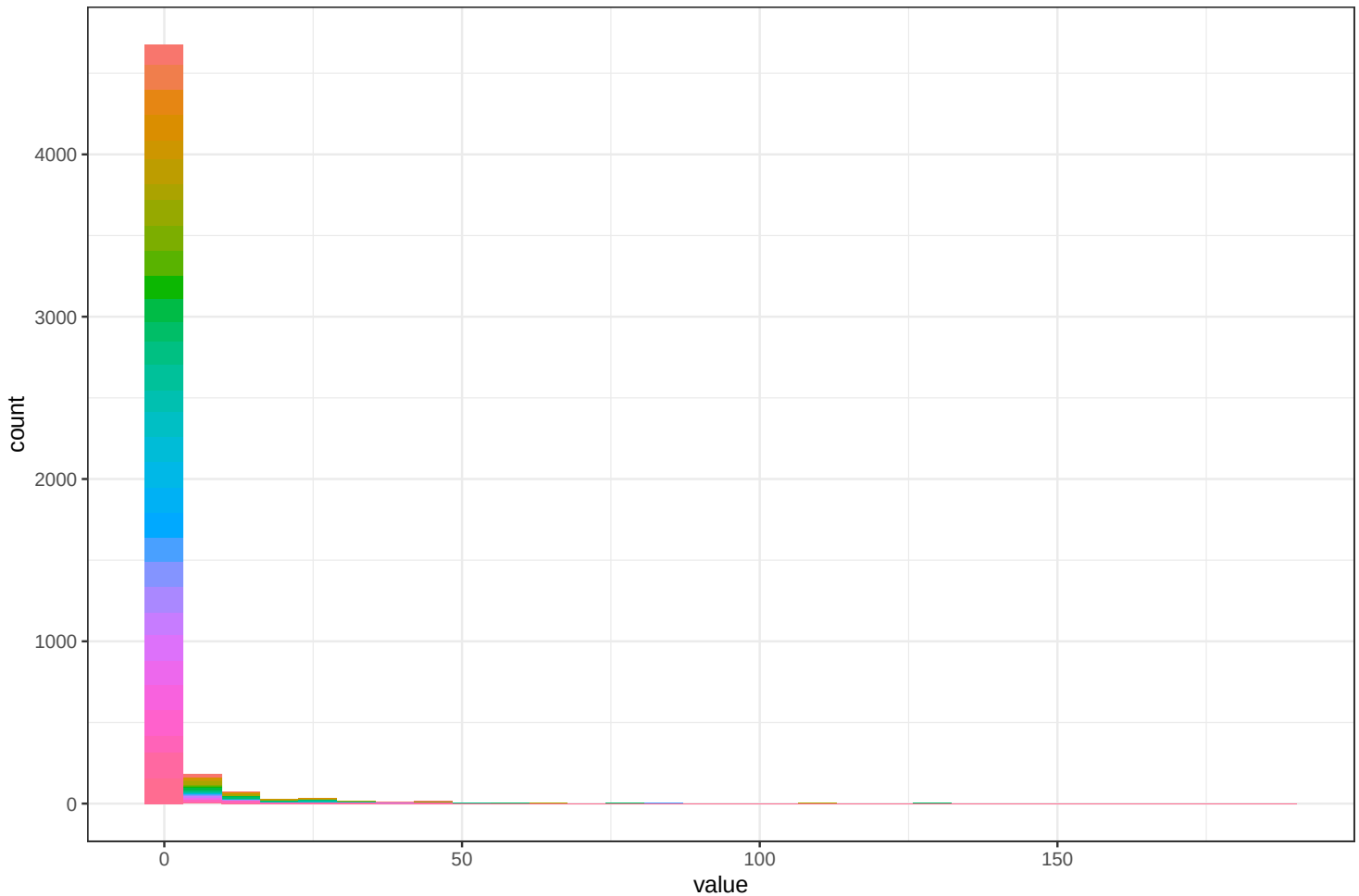
site	Negria1 :24	Negria2 :24	Pecheurs1:24	Pecheurs2:23	GGravier1:21	GGravie
season	automn:41	spring:43	summer:44	winter:30	NA	NA

Companion data set: 'mollusk'

Brief data summary II

Histogram of raw counts

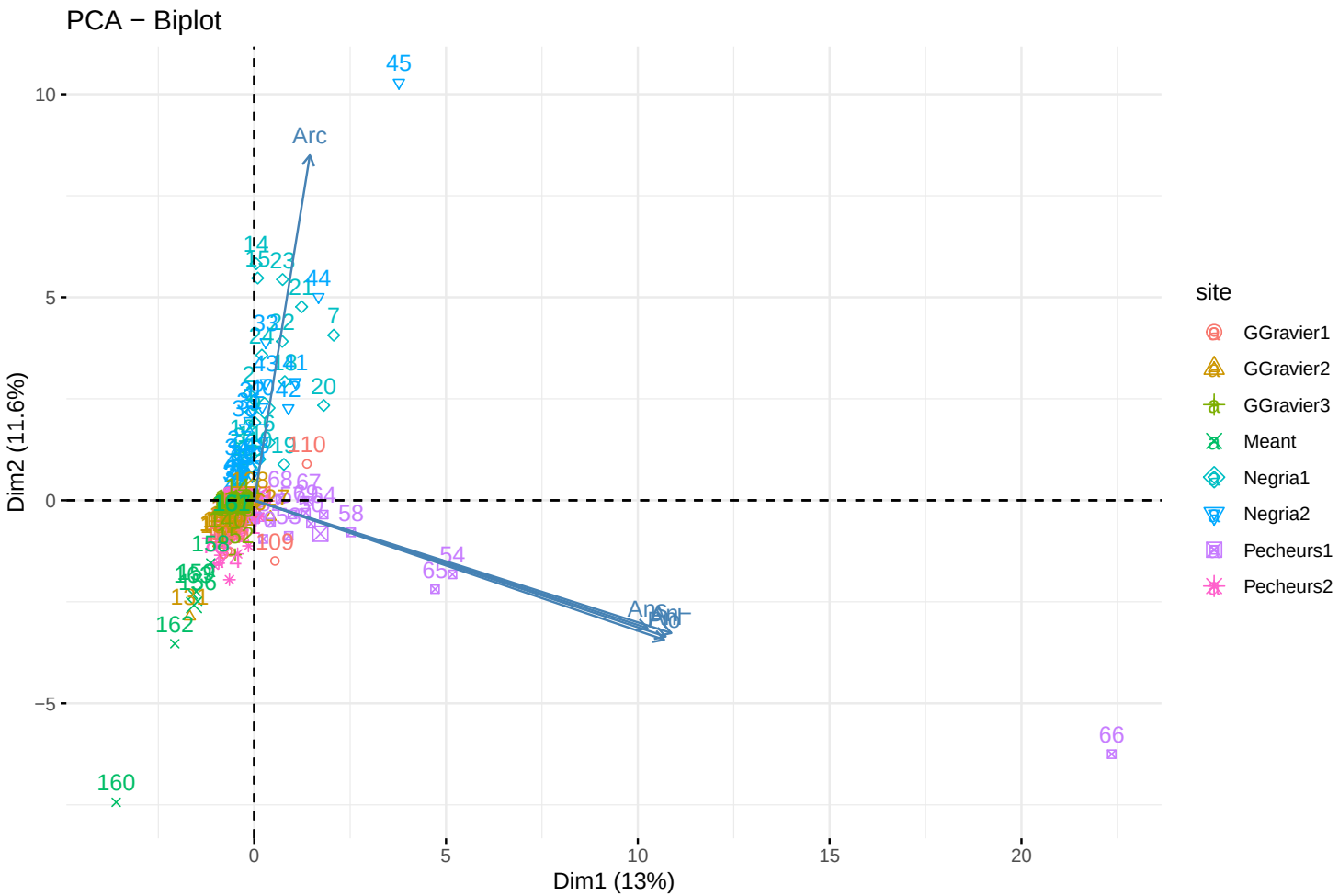
```
mollusk %>% dplyr::select(-site, -season) %>% pivot_longer(everything()) %>%
  ggplot() + aes(x = value, fill = name) + geom_histogram(show.legend = FALSE)
```



Companion data set: 'mollusk'

Principal Component Analysis

```
mollusk %>% PCA(graph = FALSE, quali.sup = which(map_lgl(mollusk, is.factor))) %>%
  fviz_pca_biplot(select.var = list(contrib = 5), habillage = "site")
```



PCA (and linear methods) limitations

Do not account for 'complex' data distribution

- PCA is tied to a hidden **Gaussian assumption**
- Fails with **Count data**
- Fails with **Skew data**

Possible solutions

- Probabilistic (non Gaussian) models
- Need transformed (non-linear) input space

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

Part II

Non-linear methods

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

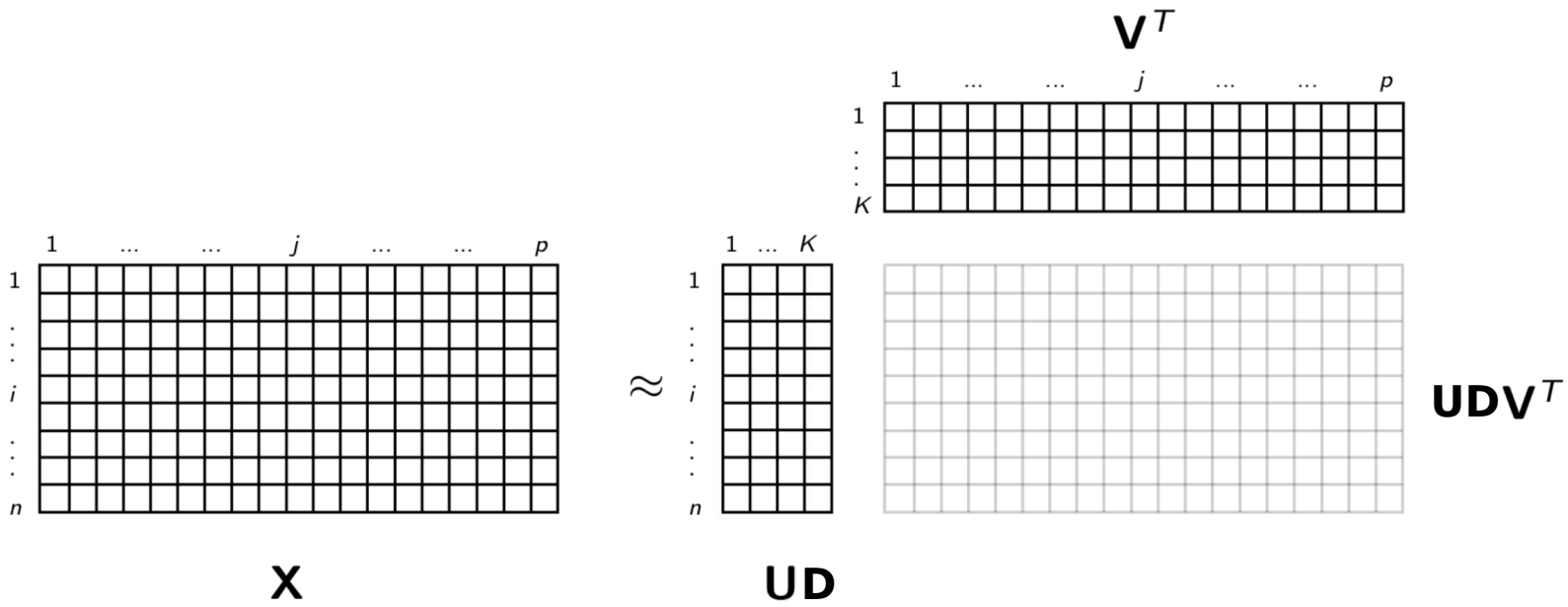
- $\mathbf{D}_{r\times r} = \text{diag}(\delta_1, ...\delta_r)$ is the diagonal matrix of singular values.
- $\mathbf{U}$ is orthonormal, whose columns are eigen vectors of $(\mathbf{M}\mathbf{M}^T)$
- $\mathbf{V}$ is orthonormal whose columns are eigen vectors of $(\mathbf{M}^T\mathbf{M})$

$\rightsquigarrow$ 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

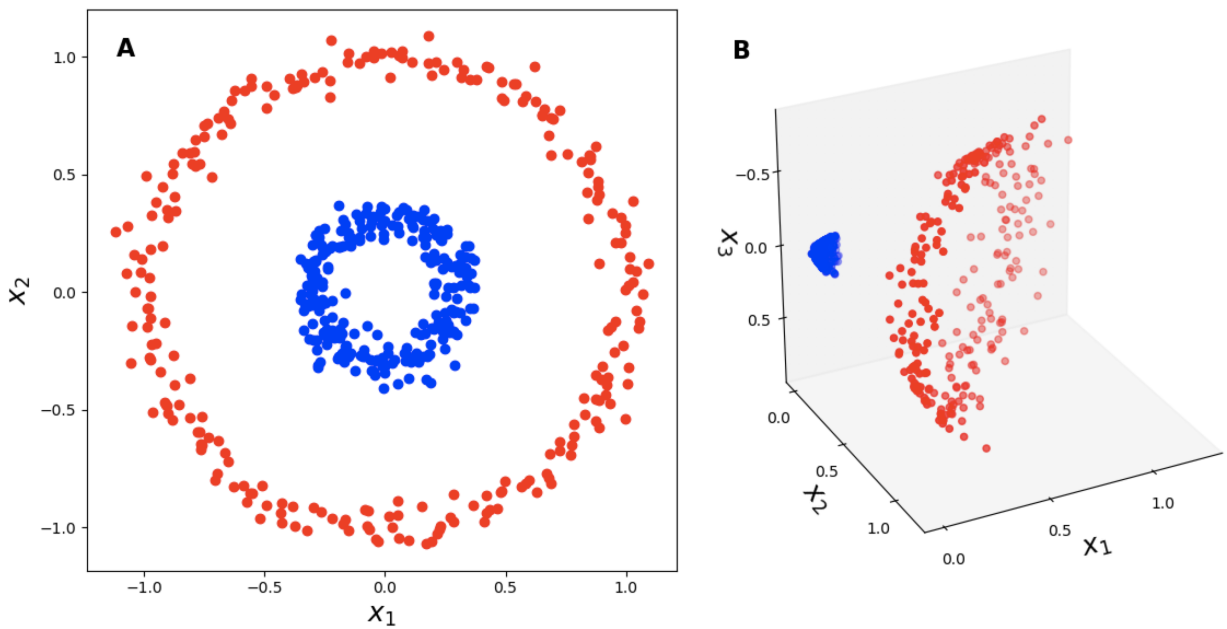


Figure: Transformation $\Psi : \mathbf{x} \rightarrow \Psi(\mathbf{x})$ (illustration in presence of existing labels)

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 $\mathbf{U}$ a $n \times q$ orthonormal matrix

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

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 $\mathbf{K} = k(\mathbf{x}_i, \mathbf{x}_{i'})$

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

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

- **Polynormial 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

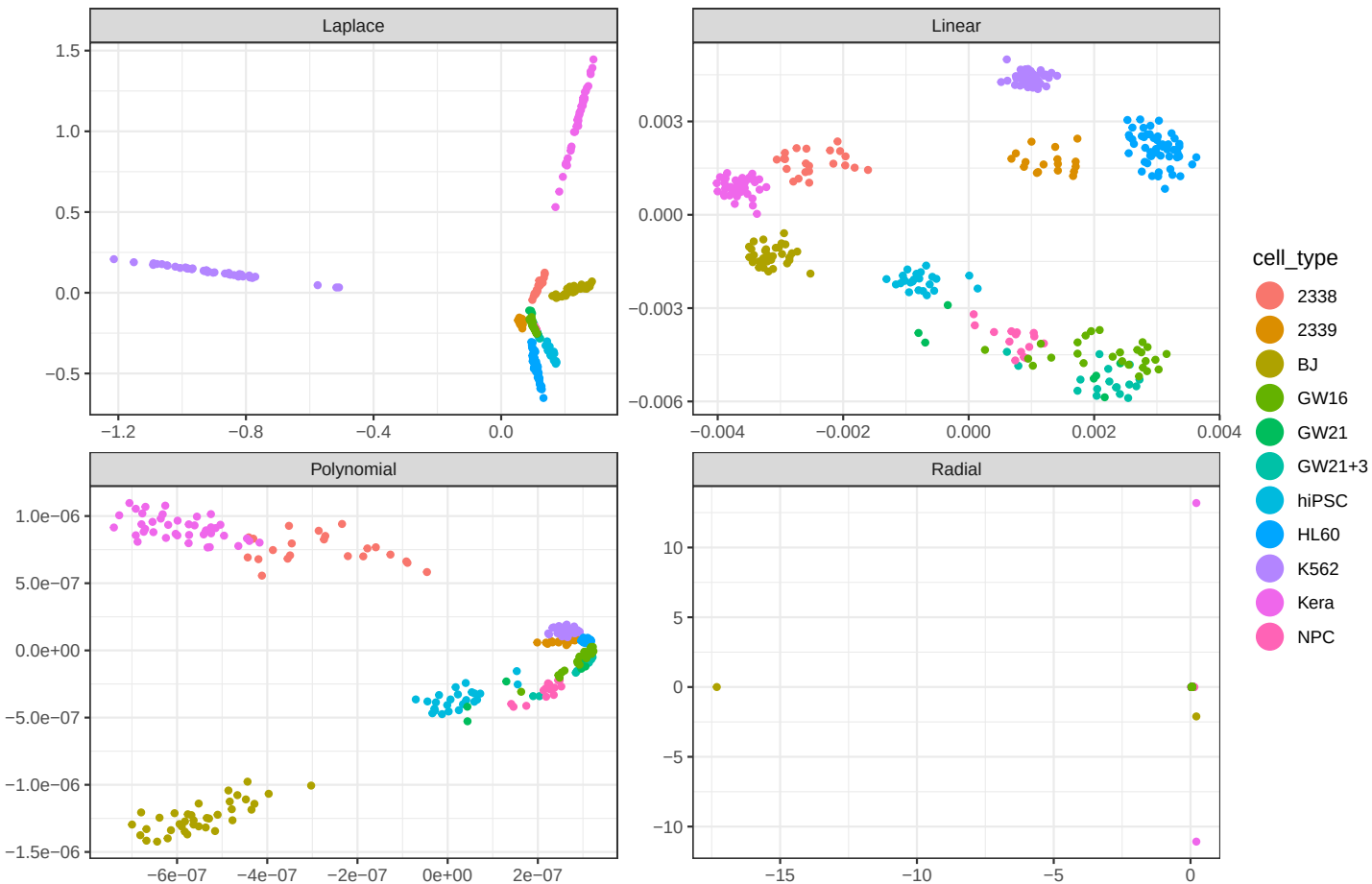
Run the fit

```
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

Non-negative Matrix Factorization – NMF

Setup

Assume that $\mathbf{X}$ contains only non-negative entries (i.e. ≥ 0).

Model

Linear assumption + non-negativity constraints on both $\mathbf{V}$ and $\tilde{\mathbf{x}}$

$$\Phi(\mathbf{x}) = \mathbf{V}^\top \mathbf{x} = \tilde{\mathbf{x}}$$
$$\mathbf{x} \simeq \tilde{\Phi}(\tilde{\mathbf{x}}) = \mathbf{V}\tilde{\mathbf{x}}$$

For the whole data matrix $\mathbf{X}$,

$$\hat{\mathbf{X}} = \underbrace{\tilde{\mathbf{X}}}_{\mathbf{F}\text{, the factors}} \mathbf{V}^\top$$

NMF reconstruction errors

Build $\hat{\mathbf{X}} = \mathbf{F}\mathbf{V}^\top$ to minimize a distance $D(\hat{\mathbf{X}}, \mathbf{X})$. Several choice, e.g:

- Least-square loss (distance measured by Frobenius norm)

$$\hat{\mathbf{X}}^{\text{ls}} = \underset{\substack{\mathbf{F}\in\mathcal{M}(\mathbb{R}_+)_{n,q} \\ \mathbf{V}\in\mathcal{M}(\mathbb{R}_+)_{p,q}}}{\arg\min} \left\| \mathbf{X} - \mathbf{F}\mathbf{V}^\top \right\|_F^2,$$

- Generalized Kullback-Leibler divergence ("distance" for distributions)

$$\begin{aligned}\hat{\mathbf{X}}^{\text{kl}} &= \underset{\substack{\mathbf{F}\in\mathcal{M}(\mathbb{R}_+)_{n,q} \\ \mathbf{V}\in\mathcal{M}(\mathbb{R}_+)_{p,q}}}{\arg\min} \sum_{i,j} x_{ij} \log\left(\frac{x_{ij}}{(\mathbf{F}\mathbf{V}^\top)_{ij}}\right) + (\mathbf{F}\mathbf{V}^\top)_{ij} \\ &= \underset{\substack{\mathbf{F}\in\mathcal{M}(\mathbb{R}_+)_{n,q} \\ \mathbf{V}\in\mathcal{M}(\mathbb{R}_+)_{p,q}}}{\arg\max} \sum_{i,j} x_{ij} \log((\mathbf{F}\mathbf{V}^\top)_{ij}) - (\mathbf{F}\mathbf{V}^\top)_{ij},\end{aligned}$$

$\rightsquigarrow$ log-likelihood of a Poisson distribution with mean $(\mathbf{F}\mathbf{V}^\top)_{ij}$.

Example on 'mollusk' I

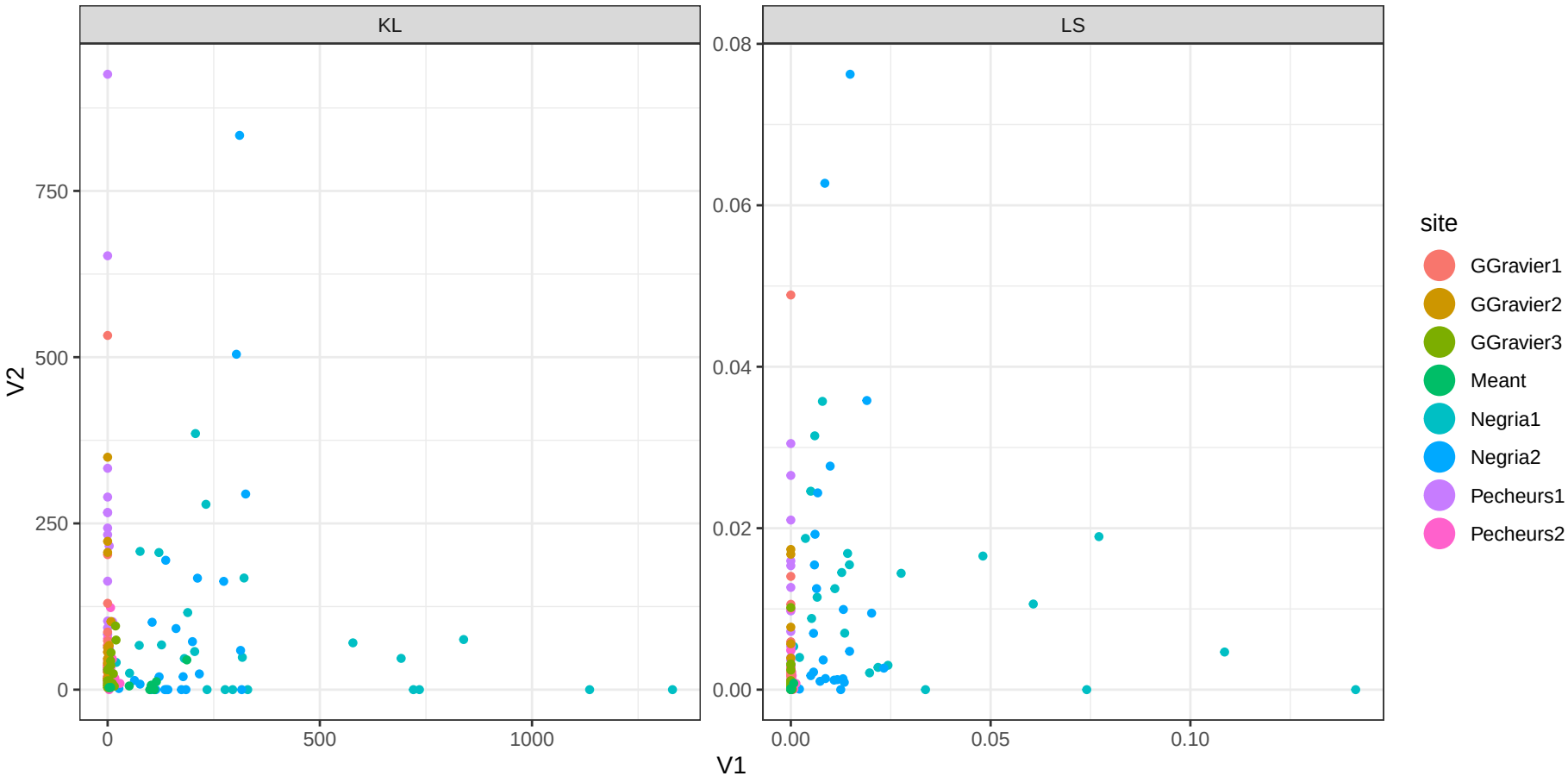
Run the fit

```
nmf_KL <- mollusk %>% dplyr::select(-site, -season) %>%
  nmf(rank = 2, method = 'brunet') %>% basis() %>%
  as.data.frame() %>% add_column(algo = "KL") %>% add_column(site = mollusk$site)
nmf_LS <- mollusk %>% dplyr::select(-site, -season) %>%
  nmf(rank = 2, method = 'lee') %>% basis() %>%
  as.data.frame() %>% add_column(algo = "LS") %>% add_column(site = mollusk$site)
```

Compare algorithms

```
rbind(nmf_KL, nmf_LS) %>%
  ggplot(aes(x = V1, y = V2, color = site)) +
     geom_point(size=1.25) +
     guides(colour = guide_legend(override.aes = list(size=6))) +
  facet_wrap(.~algo, scales = 'free')
```

Example on 'mollusk' II



NMF: limitations

Caveats

- Basis $\mathbf{V}$ formed by standard NMF is not orthogonal!
- Visualization is questionable . . .
- Used to performed matrix factorization rather than exploratory analysis

Other model-based approaches

Use a probabilistic-based model to better described non-negative data

$\rightsquigarrow$ Look for models handling**surdispersion**
multivariate Poisson-lognormal model, Poisson-Gamma, etc.

$\rightsquigarrow$ look for **zero-inflated** distributions

$$\mathbb{P}(\mathbf{x}_i) = \pi_0 \ \delta_0 + (1 - \pi_0) f(\mathbf{x}_i)$$

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 $\tilde{\mathbf{x}}$

- $\mathbf{V}$ 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.

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

- # 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)
```

$\rightsquigarrow$ First rudimentary steps with auto-encoders during next homework

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

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 $\mathbf{V} \in \mathcal{O}_{p \times q}$.

Classical MDS: solution

With the linear model $\tilde{\mathbf{x}} = \Phi(\mathbf{x}) = \mathbf{V}^\top(\mathbf{x}_i - \boldsymbol{\mu})$, we aim at minimizing

$$\begin{aligned}\text{Stress}^{cMDS} &= \sum_{i \neq i'} \left((\mathbf{x}_i - \boldsymbol{\mu})^\top (\mathbf{x}_{i'} - \boldsymbol{\mu}) - \tilde{\mathbf{x}}_i^\top \tilde{\mathbf{x}}_{i'} \right)^2, \\ &= \sum_{i \neq i'} \left((\mathbf{x}_i - \boldsymbol{\mu})^\top (\mathbf{x}_{i'} - \boldsymbol{\mu}) - (\mathbf{x}_i - \boldsymbol{\mu})^\top \mathbf{V}\mathbf{V}^\top (\mathbf{x}_{i'} - \boldsymbol{\mu}) \right)^2,\end{aligned}$$

It can be showed that $\underset{\boldsymbol{\mu} \in \mathbb{R}^p, \mathbf{V} \in \mathcal{O}_{pq}}{\text{minimize}}$ $\text{Stress}^{cMDS}(\tilde{\mathbf{x}}_i)$ is dual to principal component analysis and leads to

$$\tilde{\mathbf{X}} = \mathbf{X}^c\mathbf{V} = \mathbf{U}\mathbf{D}\mathbf{V}^\top\mathbf{V} = \mathbf{U}\mathbf{D}.$$

$\rightsquigarrow$ The principal coordinates in $\mathbb{R}^q$ correspond to the scores of the n individuals projected on the first q principal components.

Metric Multidimensional Scalings

Idea to generalize classical MDS: preserving similarities in term of **inner product** amounts to preserve dissimilarity in terms of Euclidean distance

Least-squares/Kruskal-Shephard scaling

Use a distance base formulation with the following loss (Stress) function:

$$\text{Stress}^{SK} = \sum_{i \neq i'} \left(d_{ii'} - \|\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_{i'}\|\right)^2,$$

$\rightsquigarrow$ Almost equivalent to classical MDS when d is the Euclidean distance

$\rightsquigarrow$ Generalize to any **quantitative** dissimilarity/distance d

Sammong mapping - Variant of the loss (Stress) function

$$\text{Stress}^{SM} = \sum_{i \neq i'} \frac{\left(d_{ii'} - \|\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_{i'}\|\right)^2}{d_{ii'}}.$$

Metric Multidimensional Scalings

Idea to generalize classical MDS: preserving similarities in term of **inner product** amounts to preserve dissimilarity in terms of Euclidean distance

Least-squares/Kruskal-Shephard scaling

Use a distance base formulation with the following loss (Stress) function:

$$\text{Stress}^{SK} = \sum_{i \neq i'} \left(d_{ii'} - \|\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_{i'}\|\right)^2,$$

- $\rightsquigarrow$ Almost equivalent to classical MDS when d is the Euclidean distance
- $\rightsquigarrow$ Generalize to any **quantitative** dissimilarity/distance d

Sammong mapping - Variant of the loss (Stress) function

$$\text{Stress}^{SM} = \sum_{i \neq i'} \frac{\left(d_{ii'} - \|\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_{i'}\|\right)^2}{d_{ii'}}.$$

Metric Multidimensional Scalings

Idea to generalize classical MDS: preserving similarities in term of **inner product** amounts to preserve dissimilarity in terms of Euclidean distance

Least-squares/Kruskal-Shephard scaling

Use a distance base formulation with the following loss (Stress) function:

$$\text{Stress}^{SK} = \sum_{i \neq i'} \left(d_{ii'} - \|\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_{i'}\|\right)^2,$$

- $\rightsquigarrow$ Almost equivalent to classical MDS when d is the Euclidean distance
- $\rightsquigarrow$ Generalize to any **quantitative** dissimilarity/distance d

Sammong mapping - Variant of the loss (Stress) function

$$\text{Stress}^{SM} = \sum_{i \neq i'} \frac{\left(d_{ii'} - \|\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_{i'}\|\right)^2}{d_{ii'}}.$$

Non-Metric Dimension Scaling

Idea: dissimilarities are often only known by their rank order

Shephard-Kruskal non-metric scaling

$$\text{Stress}^{NM} = \sum_{i \neq i'} \frac{(d_{ii'} - f(d_{ii'}))^2}{\sum_{i \neq i'} d_{ii'}^2},$$

where f is an arbitrary increasing function preserving the order

- $\rightsquigarrow$ Only the order it required
- $\rightsquigarrow$ f acts as an isotonic regression curve for the $d_{ii'}$.

Example on 'mollusk' I

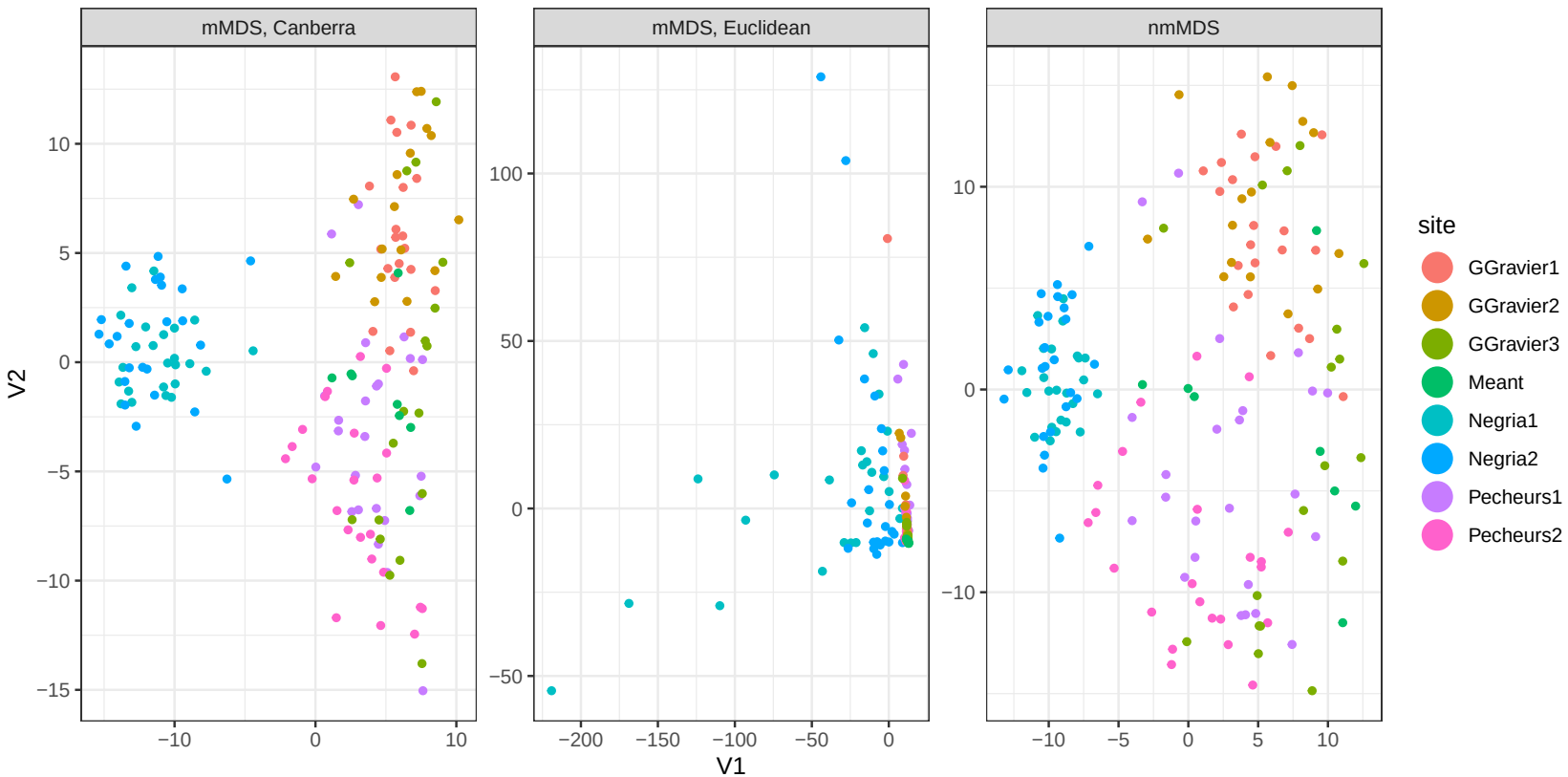
Run the fit

```
mollusk_ab <- mollusk %>% dplyr::select(-site, -season) %>%  as.matrix()

mmds_euclidean <- cmdscale(dist(mollusk_ab)) %>%
  as.data.frame() %>% add_column(type = "mMDS, Euclidean") %>% add_column(site = mo

mmds_canberra <- cmdscale(dist(mollusk_ab, method = "canberra")) %>%
  as.data.frame() %>% add_column(type = "mMDS, Canberra") %>% add_column(site = mol

nmds <- MASS::isoMDS(dist(mollusk_ab, "canberra"))$points %>%
  as.data.frame() %>% add_column(type = "nmMDS") %>% add_column(site = mollusk$site

## initial  value 39.689470
## iter   5 value 32.736128
## final  value 32.587709
## converged
```

Compare type of MDS

Example on 'mollusk' II

```
rbind(mmds_euclidean, mmds_canberra, nmds) %>%
  ggplot(aes(x = V1, y = V2, color = site)) +
     geom_point(size=1.25) +
     guides(colour = guide_legend(override.aes = list(size=6))) +
  facet_wrap(.~type, scales = 'free')
```



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)

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

Optimizing tSNE

- Minimize the KL between p and q so that the data representation minimizes:

$$C(y) = \sum_{ij} KL(p_{ij}, q_{ij})$$

- The cost function is not convex

$$\left[\frac{\partial C(y)}{\partial y}\right]_i = \sum_j (p_{ij} - q_{ij})(y_i - y_j)$$

- Interpreted as the resultant force created by a set of springs between the map point y_i and all other map points $(y_j)_j$. All springs exert a force along the direction $(y_i - y_j)$.
- $(p_{ij} - q_{ij})$ is viewed as a stiffness of the force exerted by the spring between y_i and y_j.

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

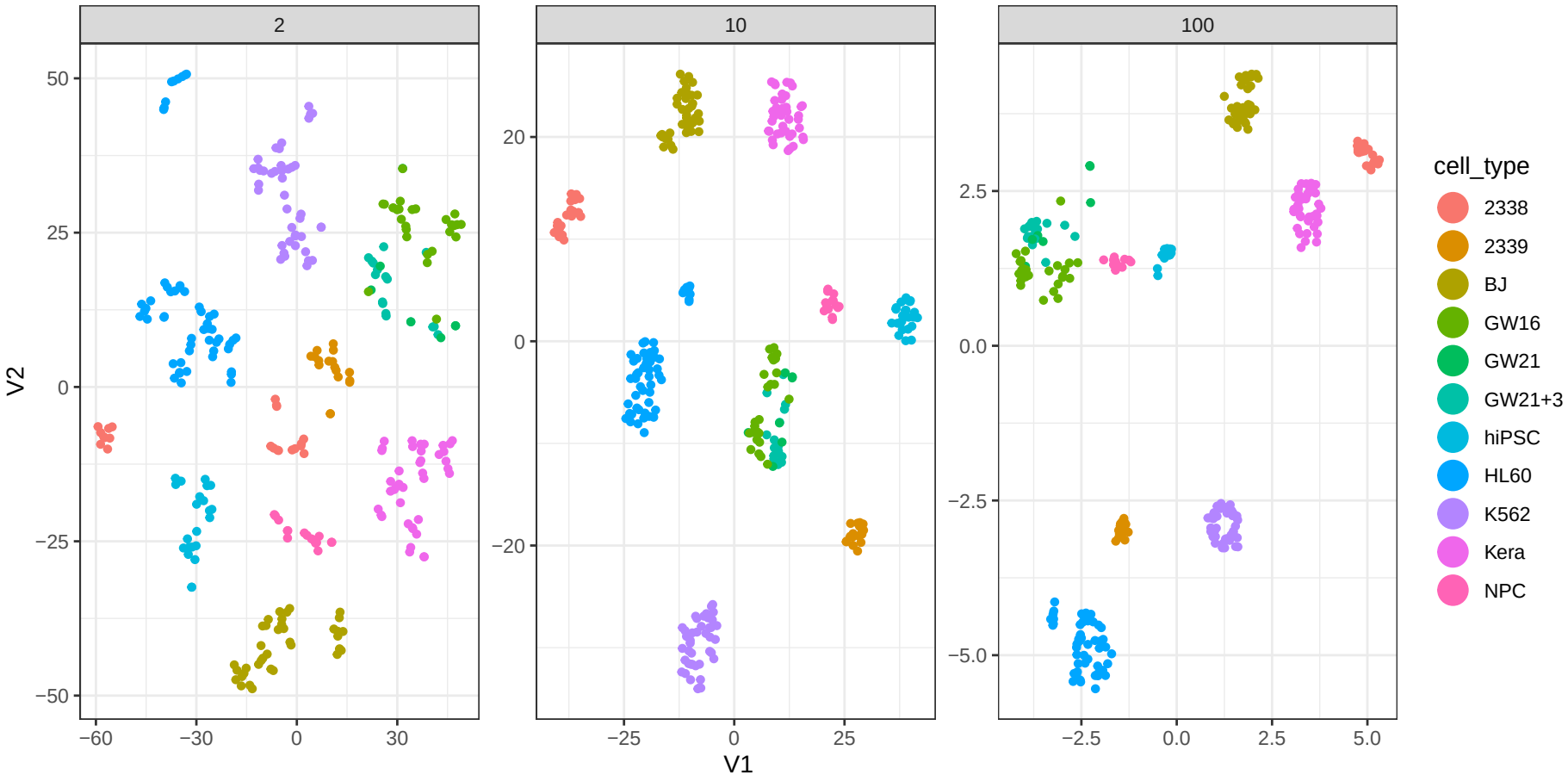
Run the fit

```
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



Example on 'mollusk' I

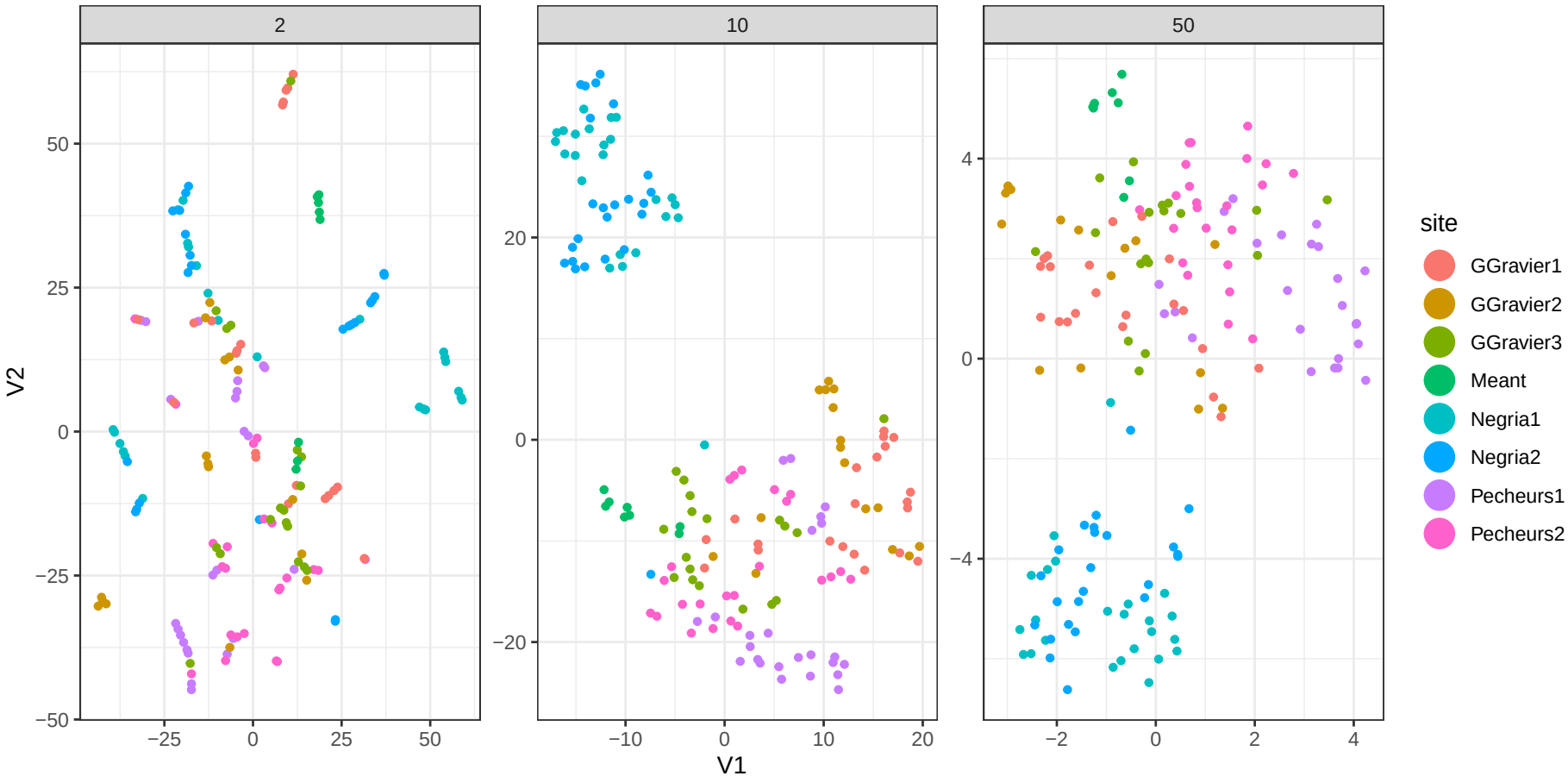
Run the fit

```
mollusk_ab <- mollusk %>% dplyr::select(-site, -season) %>%  as.matrix()

tSNE_perp2   <- Rtsne(mollusk_ab, perplexity =   2)$Y %>%
  as.data.frame() %>% add_column(perplexity = 2) %>% add_column(site = mollusk$site

tSNE_perp10  <- Rtsne(log(1 + mollusk_ab), perplexity =  10)$Y %>%
  as.data.frame() %>% add_column(perplexity = 10) %>% add_column(site = mollusk$sit

tSNE_perp50 <- Rtsne(log(1 + mollusk_ab), perplexity = 50)$Y %>%
  as.data.frame() %>% add_column(perplexity = 50) %>% add_column(site = mollusk$sit
```

Compare perplexity

```
rbind(tSNE_perp2,tSNE_perp10,tSNE_perp50) %>%
  ggplot(aes(x = V1, y = V2, color = site)) +
     geom_point(size=1.25) +
     guides(colour = guide_legend(override.aes = list(size=6))) +
  facet_wrap(.~perplexity, scales = 'free')
```

Example on 'mollusk' II



Outline

Non-linear methods

Isomap

Basic idea

- MMDS performs embedding based on the pairwise Euclidean-based distance distance
- Isomap uses a distance induced by a neighborhood graph embedded

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

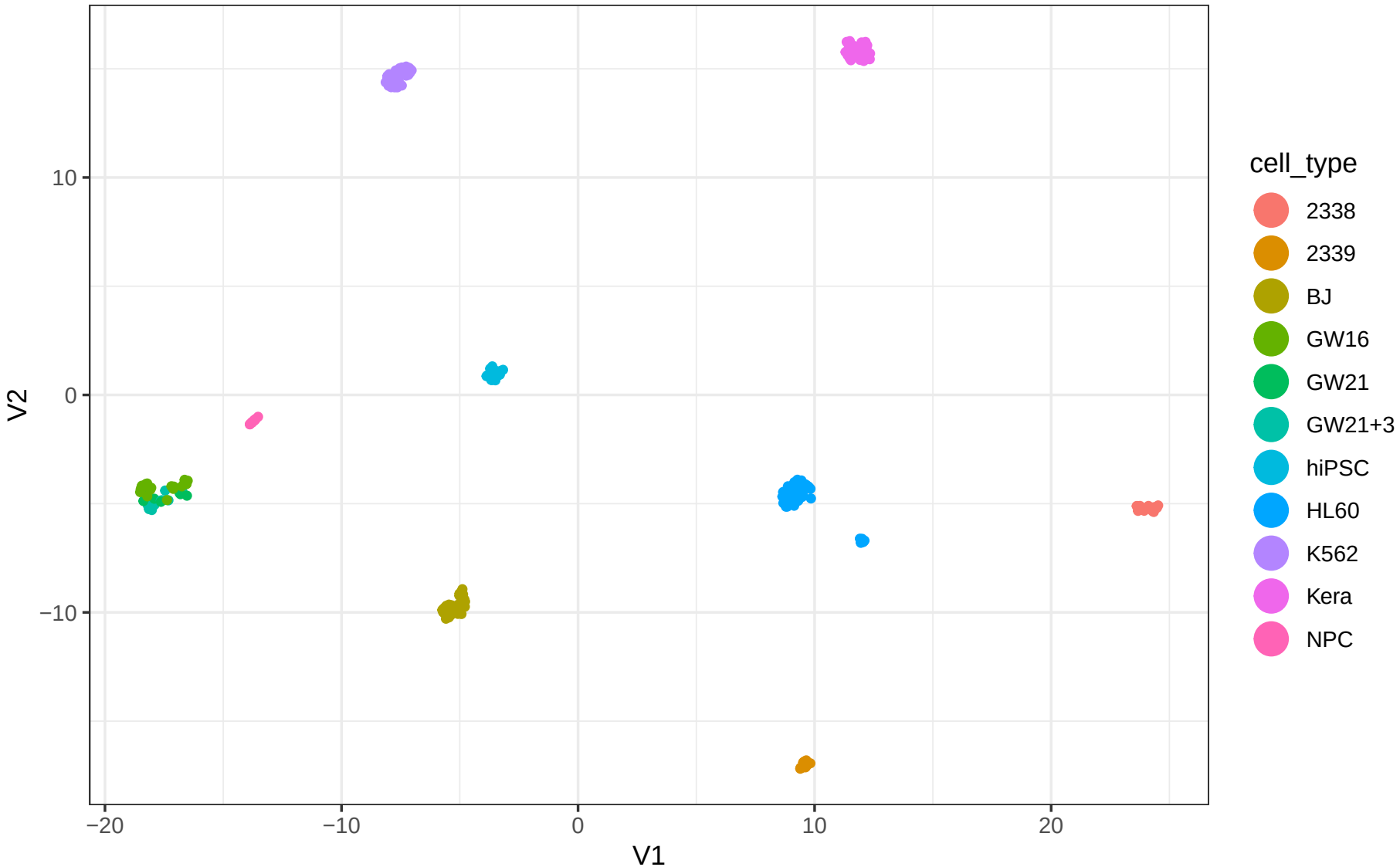
Run the fit on scRNA

```
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



Example on 'mollusk' I

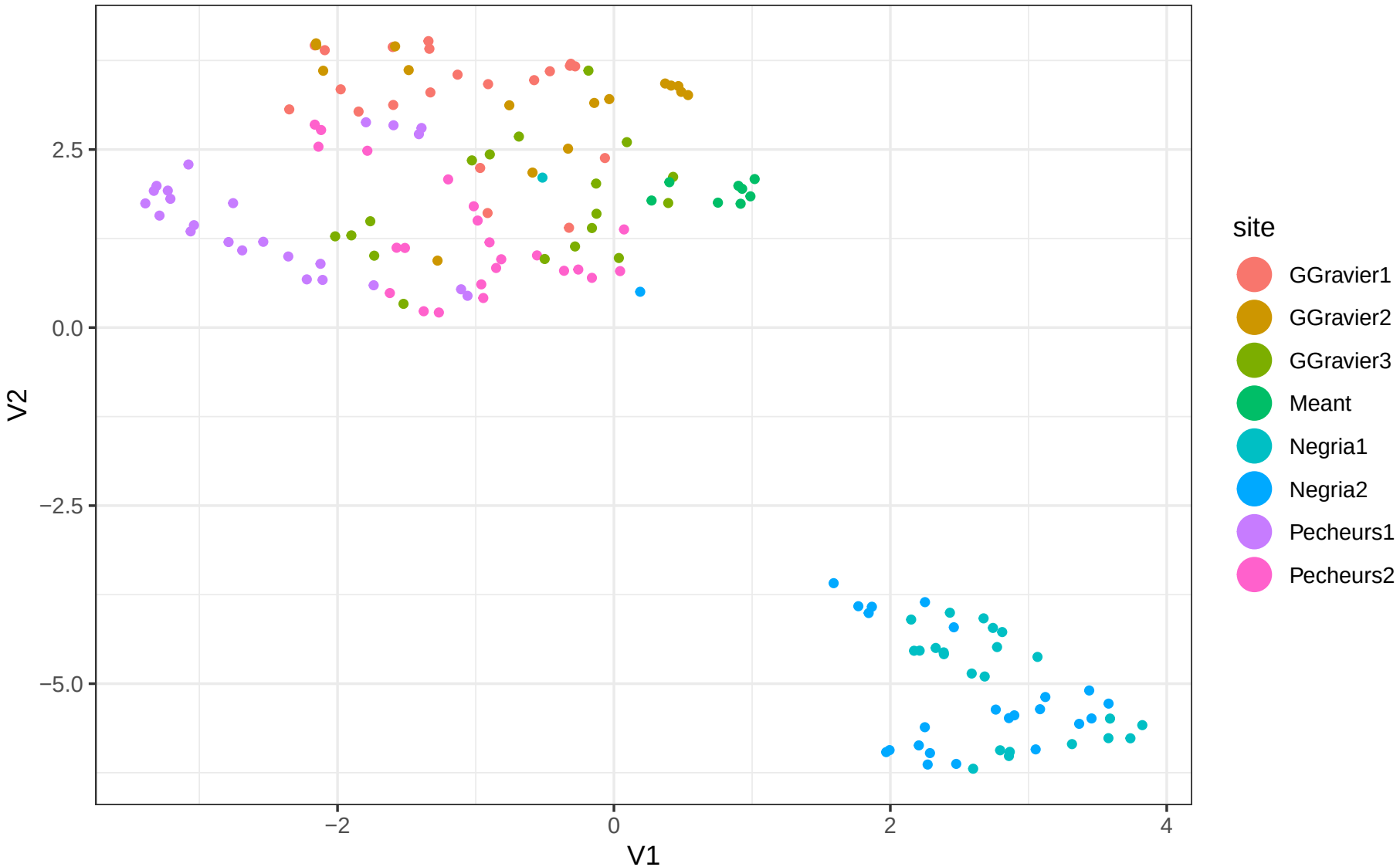
Run the fit

```
mollusk_ab <- mollusk %>% dplyr::select(-site, -season) %>%  as.matrix()
umap_fit   <- umap(log(1 + mollusk_ab))$layout %>%
  as.data.frame() %>% add_column(site = mollusk$site)
```

Visualization

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

Example on 'mollusk' II



To conclude

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