Beyond Euclidean Data Analysis: Riemannian PCA and Riemannian Regression

le cnam

Oldemar Rodríguez
CIMPA, University of Costa Rica



UNIVERSIDAD DE COSTA RICA

CIMPA Centro de Investigación en Matemática Pura y Aplicada

Outline

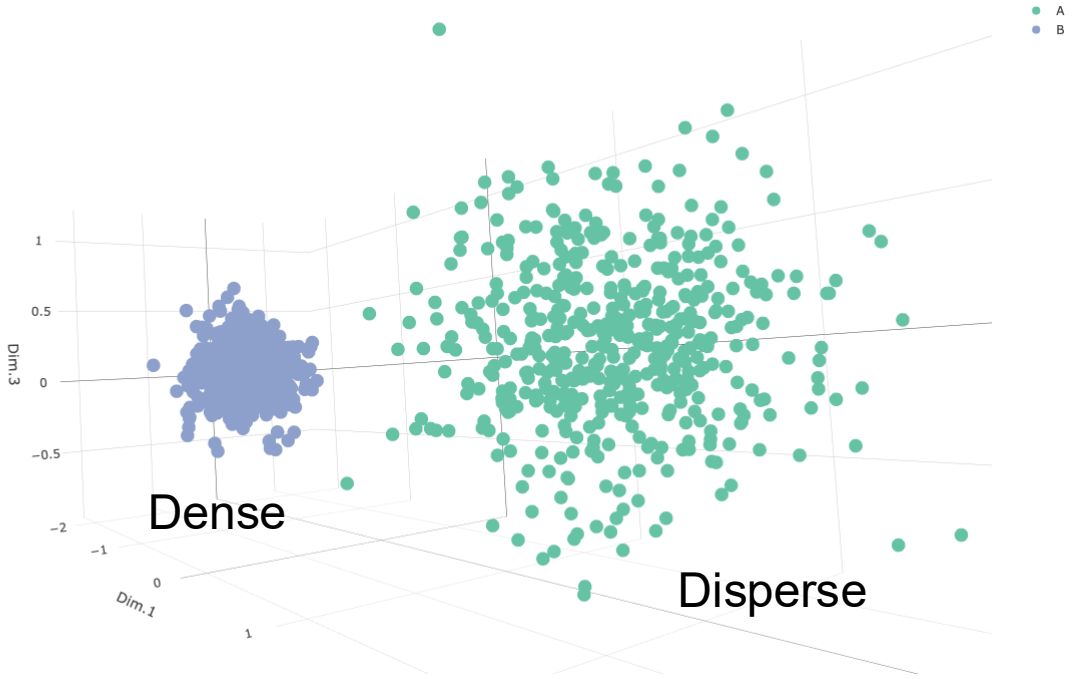
- Local distances
- Why Riemannian Statistics?
- Generating the Riemannian Manifold
- Riemannian Statistics for any type of data
- Riemannian PCA (R-PCA)
- Simulated and real data sets examples
- Riemannian Regression
- Simulated and real data sets examples
- R and Python Riemannian STATS Packages
- Some references

- Thinking that the rows of a data table are in Euclidean Space is as wrong as thinking that the earth is flat.



Local distances

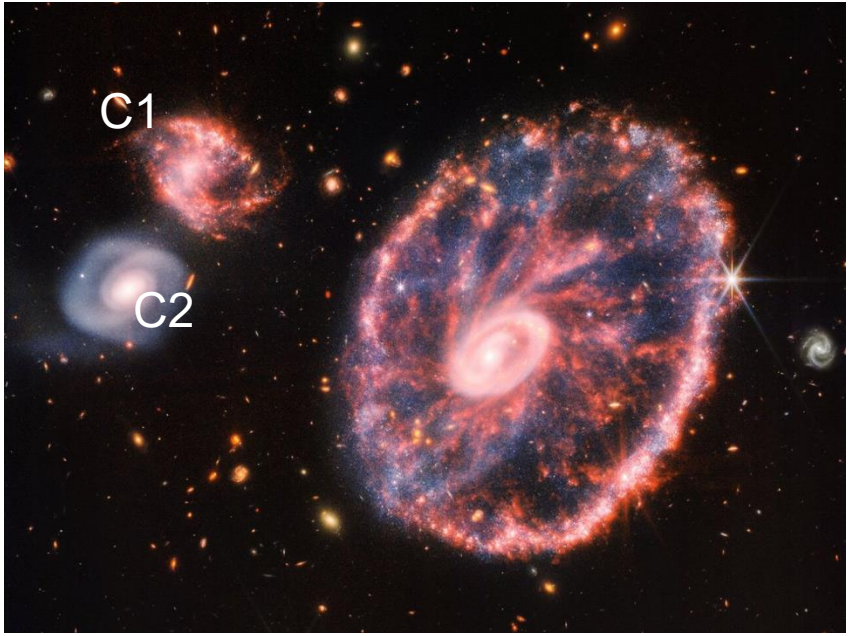
- The data tables have local distance notions.



The local metrics are very important

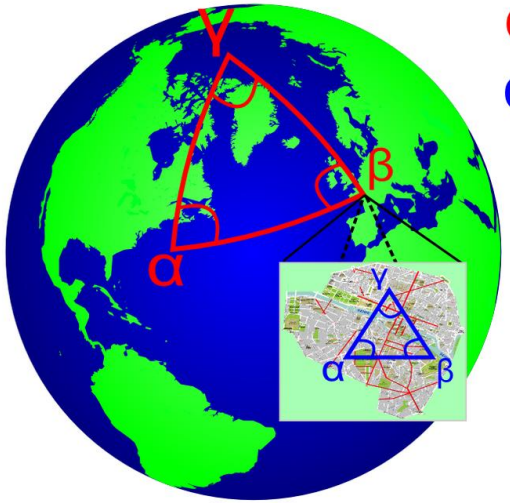
- For example, in this figure, Constellation 1 looks very similar to Constellation 2, but they are not, it is a matter of perspective.

- Constellation 2 could be light years away than Constellation 1 and the internal distances between planets are completely different.



A Manifold (Locally resembles euclidean space)

- A **manifold** is a collection of points that, locally but not globally, resembles Euclidean space.
- The earth is a manifold.



$\alpha + \beta + \gamma > 180°$

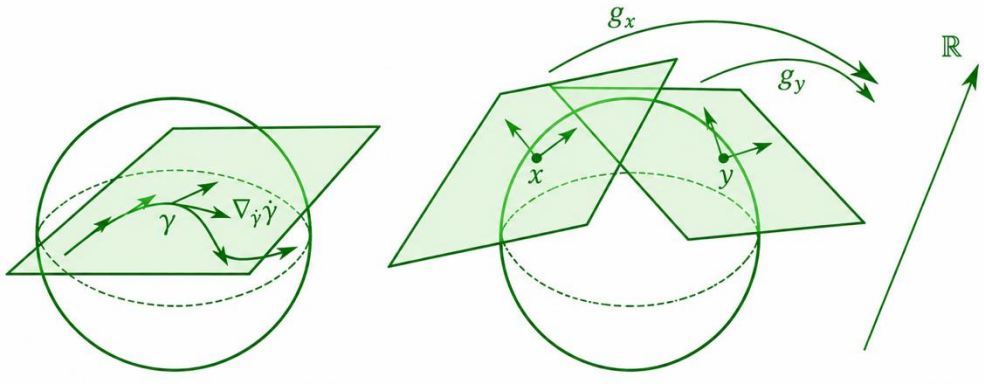
$\alpha + \beta + \gamma = 180°$

Riemannian Manifold (local distances)

Definition 3 A **Riemannian Manifold** is a differentiable manifold M such that, at each point $x \in M$, the tangent space T_xM is equipped with an inner product:

$$\langle \cdot, \cdot \rangle_x : T_xM \times T_xM \to \mathbb{R},$$

which depends smoothly on x.



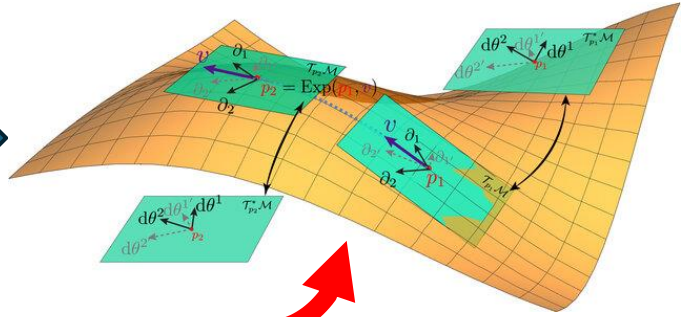
Riemannian Statistics Idea

Data Table
Euclidean Space



Some method or algorithm provides the data table with local metrics

Data Table
Riemannian Space



Classic Statistics,
Classic Machine Learning,
Classic AI….

Riemannian Statistics,
Riemannian Machine Learning,
Riemannian AI….

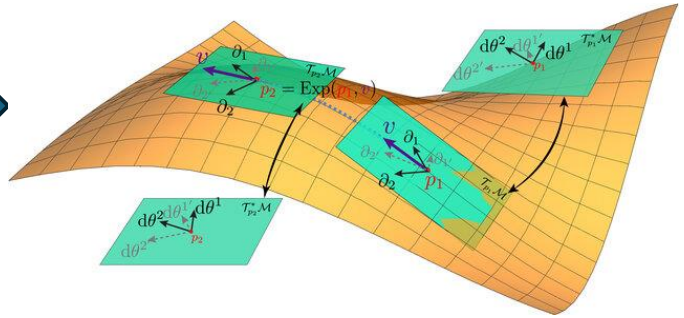
Riemannian Statistics Idea

Data Table
Euclidean Space



Optimization Process

Data Table
Riemannian Space



How to build the Tangent Spaces?

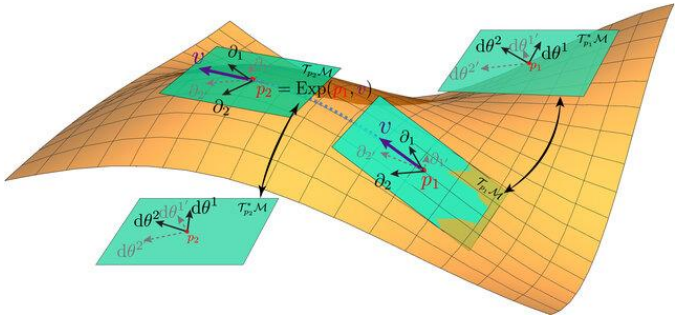
Riemannian Statistics Idea

Data Table
Euclidean Space



ISOMAP

Data Table
Riemannian Space



How to build the Tangent Spaces?

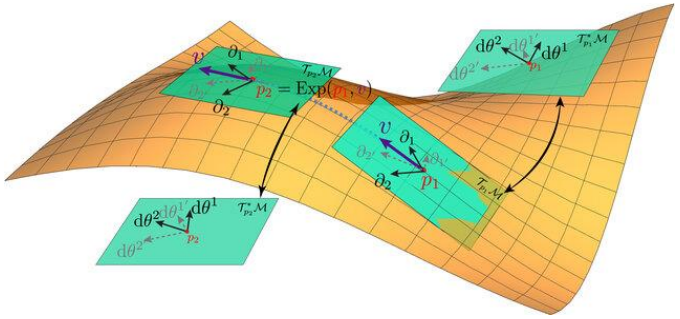
Riemannian Statistics Idea

Data Table
Euclidean Space



DBSCAN

Data Table
Riemannian Space



How to build the Tangent Spaces?

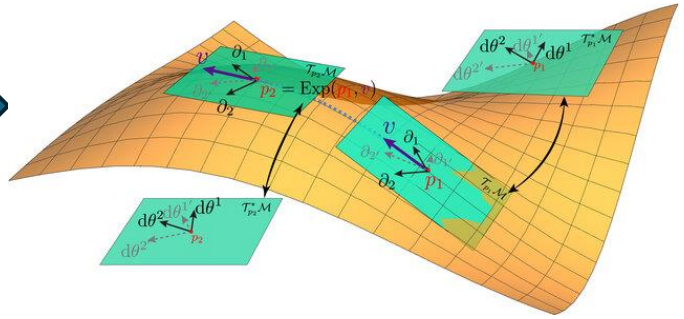
Riemannian Statistics Idea

Data Table
Euclidean Space



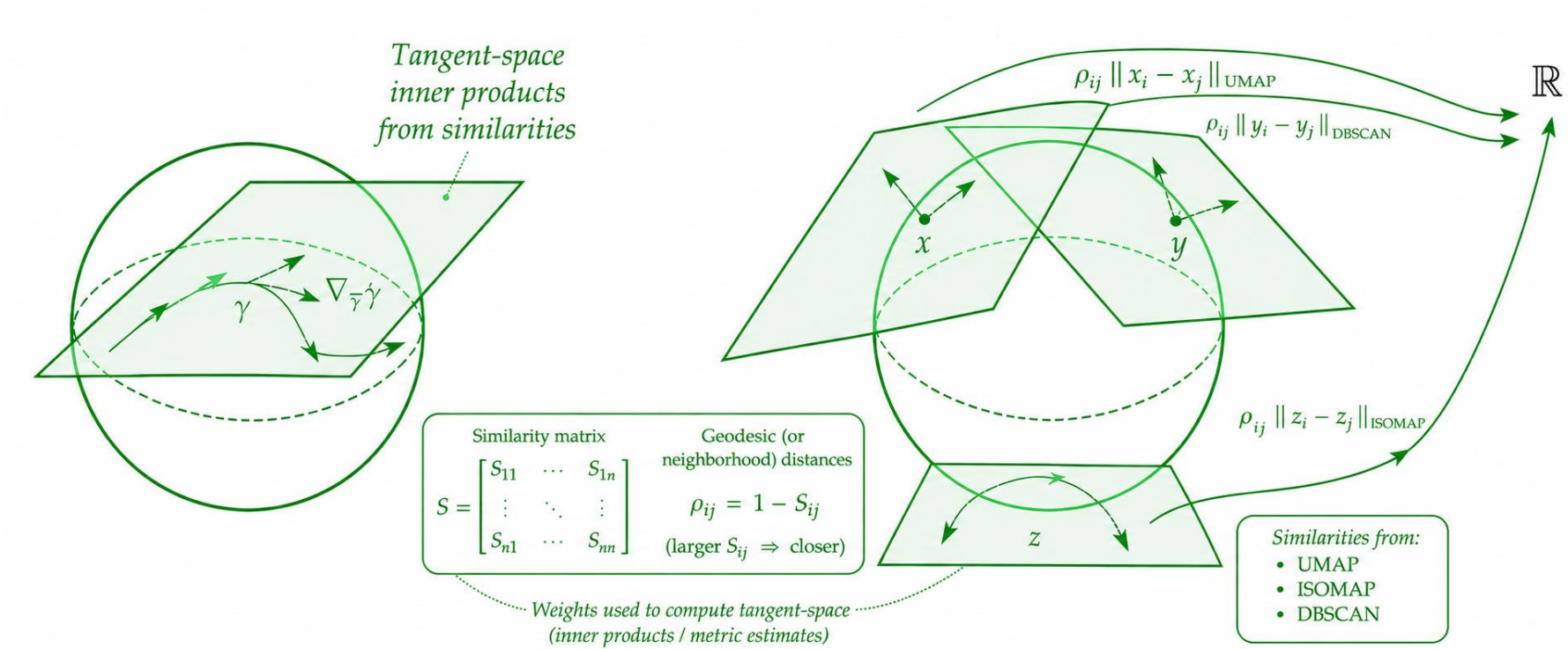
UMAP

Data Table
Riemannian Space



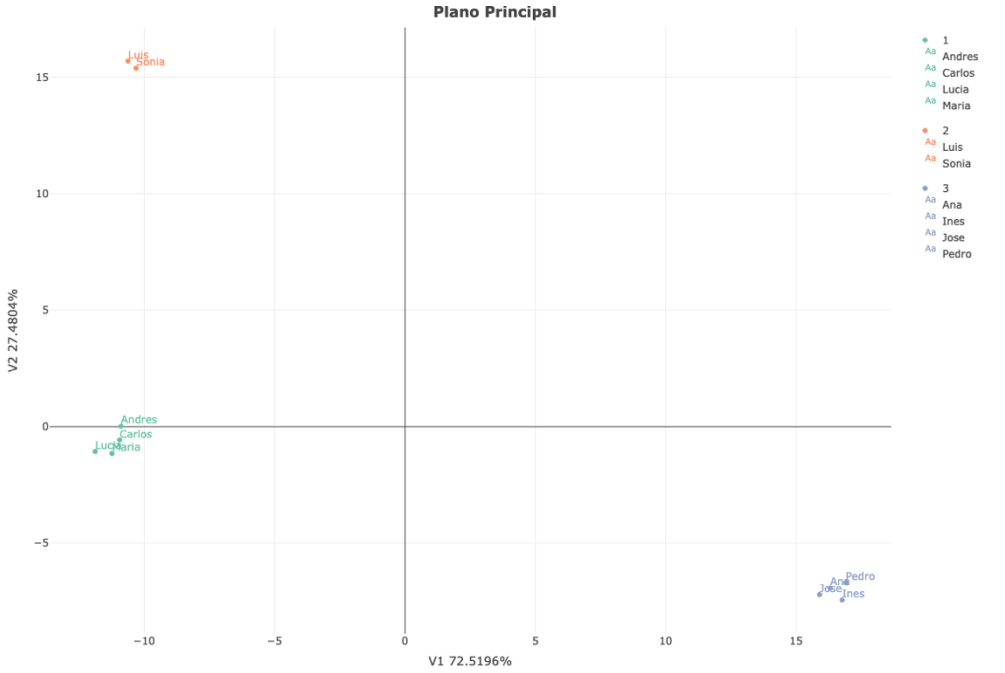
How to build the Tangent Spaces?

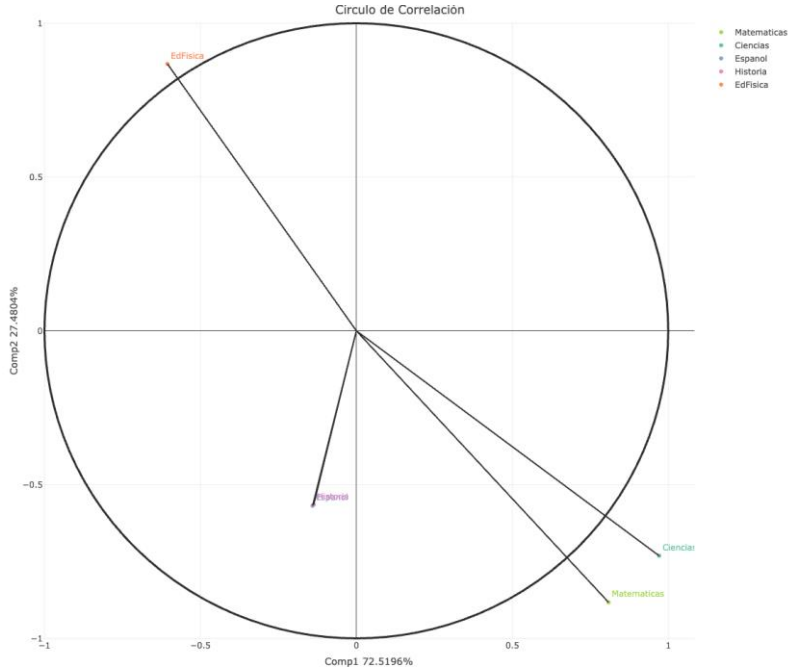
Riemannian PCA general idea



Why not just use ISOMAP, DBSCAN or UMAP?

Because these methods generate statistical errors, since the principal components are not linear combinations of the original variables.





Why not just use ISOMAP, DBSCAN or UMAP?

- We don’t have the property:

$$R^2(X^j, C^s) + R^2(X^j, C^r) \leq 1.$$

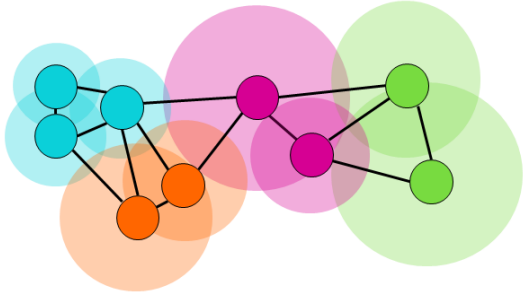
- This is because we are computing correlations as the data were in a Euclidean space.

- But the data is on a Riemannian manifold, with local distances.

- ***So, we need something like a Riemannian correlation, we need a Riemannian mean, in general we need a "Riemannian Statistics".***

Finding the Tangent Spaces with UMAP

How does UMAP construct a high-dimensional fuzzy graph?



2. Draw a radius from each point and connect points where the radii overlap

- Then the **Riemannian Dissimilarity Coefficient** is computed as:

$$\rho_{ij} = 1 - S_{\text{UMAP}}(x_i, x_j)$$

Finding the Tangent Spaces with UMAP

The algorithm begins by constructing a weighted k-nearest neighbor graph from the dataset $X = \{x_1, \ldots, x_n\}$. The process can be described as follows:

1. The algorithm identifies the k-nearest neighbors using a metric d. This step produces a simplicial complex (graph) where each point has outgoing edges to its k-nearest neighbors.

2. The local parameters, ρ_i and σ_i, are defined as follows:

   - ρ_i: **Local Connectivity**. For each x_i, ρ_i is the minimum distance to its nearest neighbor that is greater than zero:

$$\rho_i = \min\{d(x_i, x_{i_j}) \mid 1 \leq j \leq k,\ d(x_i, x_{i_j}) > 0\}.$$

     This ensures that x_i connects to at least one other point with an edge of weight 1.

   - σ_i: **Local Scaling**. σ_i normalizes the distances of all neighbors, ensuring consistency in the local metric. It is computed by solving:

$$\sum_{j=1}^{k} \exp\left(-\frac{\max(0, d(x_i, x_{i_j}) - \rho_i)}{\sigma_i}\right) = \log_2(k).$$

     Then σ_i local scale adjusted using the k-nearest neighbors.

3. To construct the weighted directed graph, we use ρ_i and σ_i, and the weight of each directed edge (x_i, x_{i_j}) is computed as:

$$w_{ij} = w((x_i, x_{i_j})) = \exp\left(-\frac{\text{máx}(0, d(x_i, x_{i_j}) - \rho_i)}{\sigma_i}\right).$$

A similarity threshold $\tau \in (0, 1)$ is fixed, then $w_{ij} \geq \tau$ induces an effective local radius:

$$\varepsilon_i(\tau) = \rho_i + \sigma_i \log\left(\frac{1}{\tau}\right).$$

4. To symmetrized the graph we convert the directed graph into an undirected graph, the adjacency matrix A is symmetrized using:

$$B = A + A^t - A \circ A^t,$$

where A is the adjacency matrix, $\circ$ denotes the Hadamard product, and t represents the transpose matrix.

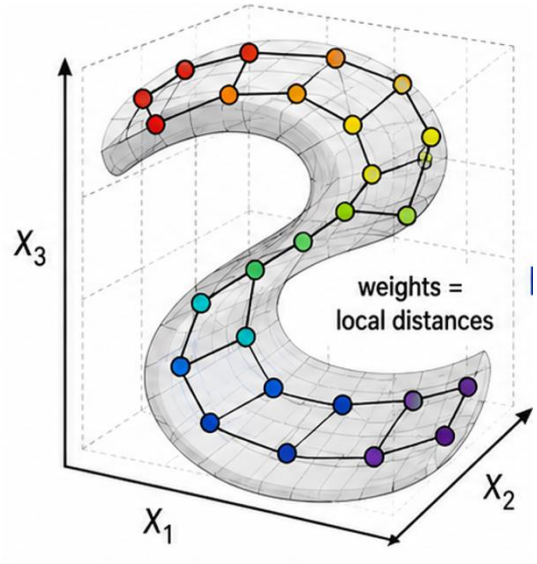
5. The symmetrized graph G forms a **simplicial set** that captures both the local and global topological structure of the data.

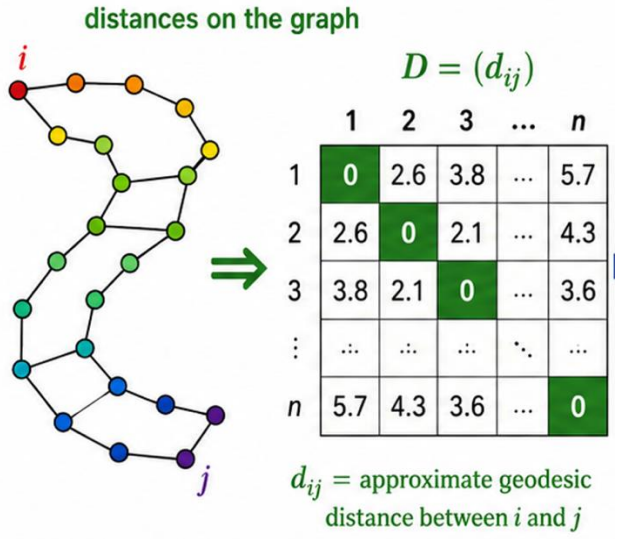
   Also, this symmetrized graph G allows us to define a local similarity as $S_{\mathsf{UMAP}}(x_i, x_j) = B_{ij}$.

Riemannian Dissimilarity Coefficient

$$\rho_{ij} = 1 - S_{\mathsf{UMAP}}(x_i, x_j)$$

Finding the Tangent Spaces with ISOMAP





- Then the **Riemannian Dissimilarity Coefficient** is computed as:

$$\rho_{ij} = 1 - S_{ij}^{\text{ISOMAP}}.$$

Finding the Tangent Spaces with ISOMAP

- Let be the data table:

$$X = \begin{pmatrix} x_{11} & \cdots & x_{1p} \\ x_{21} & \cdots & x_{2p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{pmatrix} \in M_{n\times p}.$$

- First, we compute the usual Euclidean distance matrix $D = d_{ij}$ where $d_{ij} = \|x_i - x_j\|$.
- Using these distances, ISOMAP constructs a weighted k-nearest-neighbor graph:

$$G_k = (V, E_k), \qquad V = \{x_1, \ldots, x_n\}.$$

- The edge weights are local Euclidean distances:

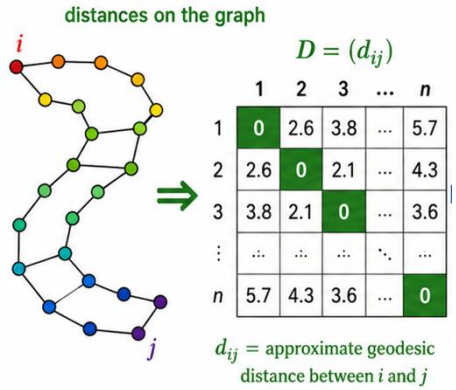
$$w_{ij} = \begin{cases} d_{ij}, & \text{if } x_i \text{ and } x_j \text{ are neighbors,} \\ +\infty, & \text{otherwise,} \\ 0, & \text{if } i = j. \end{cases}$$

- If the graph is fragmented, the number of neighbors is increased until the first connected graph is obtained:

$$k_* = \min\{k : \ G_k \text{ is connected}\}.$$

- Once the graph is connected, ISOMAP replaces Euclidean distances by shortest-path distances on the graph:

$$d_{ij}^{\text{ISOMAP}} = \min_{\gamma: i \rightsquigarrow j} \sum_{(a,b)\in\gamma} d_{ab}.$$



- Thus, the method produces an approximate distance matrix:

$$D^{\text{ISOMAP}} = \left(d_{ij}^{\text{ISOMAP}}\right)_{n\times n}.$$

- From the ISOMAP distance matrix, compute:

$$d_{\max} = \max_{1\le i,j\le n} d_{ij}^{\text{ISOMAP}}.$$

- Then normalize the distances:

$$\widetilde{d}_{ij}^{\text{ISOMAP}} = \frac{d_{ij}^{\text{ISOMAP}}}{d_{\max}}, \qquad 0 \le \widetilde{d}_{ij}^{\text{ISOMAP}} \le 1.$$

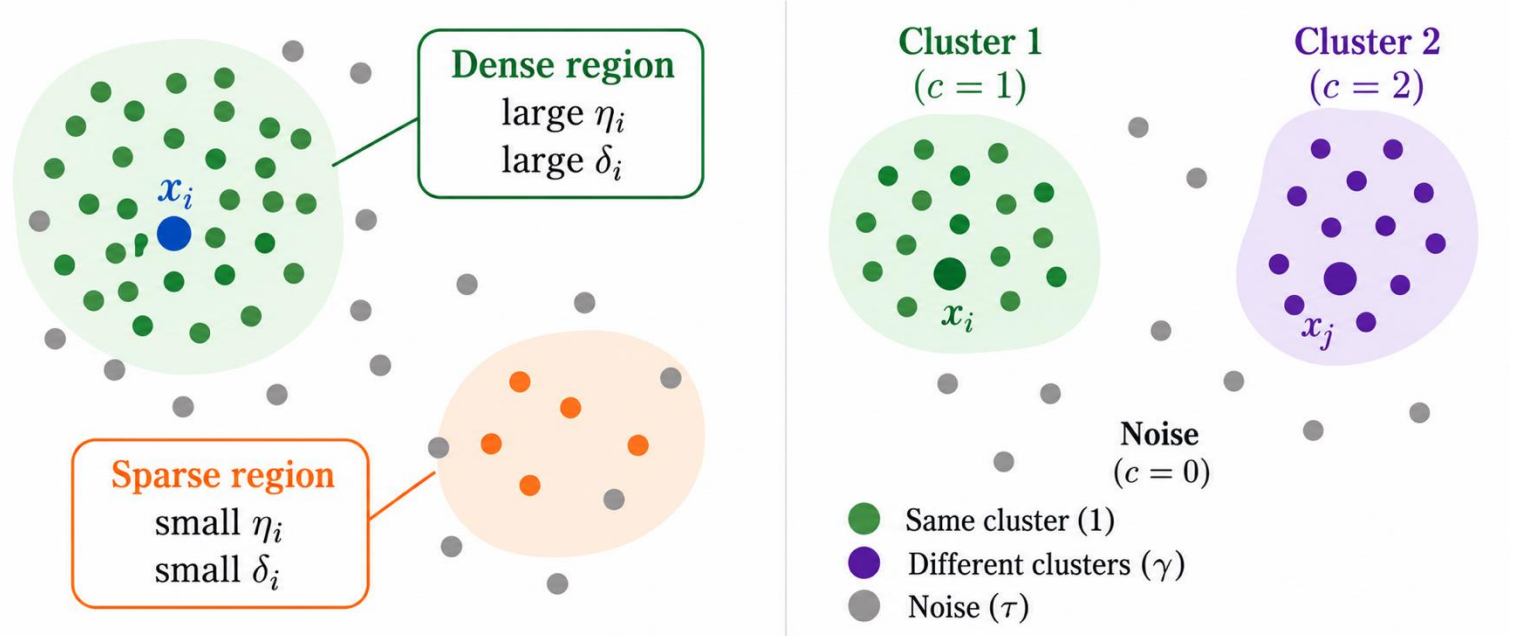
- We define the ISOMAP-induced similarity matrix by:

$$S_{ij}^{\text{ISOMAP}} = 1 - \widetilde{d}_{ij}^{\text{ISOMAP}} = 1 - \frac{d_{ij}^{\text{ISOMAP}}}{d_{\max}}, \qquad i \ne j.$$

- Then the **Riemannian Dissimilarity Coefficient** is computed as:

$$\rho_{ij} = 1 - S_{ij}^{\text{ISOMAP}}.$$

Finding the Tangent Spaces with DBSCAN



- Then the **Riemannian Dissimilarity Coefficient** is computed as:

$$\rho_{ij} = 1 - S_{ij}^{\text{DBSCAN}}.$$

Finding the Tangent Spaces with DBSCAN

- DBSCAN detects local density through ε-neighborhoods:

$$N_\varepsilon(x_i) = \{x_j : \|x_i - x_j\| \leq \varepsilon\}.$$

- The local density of x_i is defined as:

$$\eta_i = 1 + \text{card}\left(\{x_j : j \neq i,\ \|x_i - x_j\| \leq \varepsilon\}\right),$$

and normalized by:

$$\delta_i = \frac{\eta_i}{\max_r \eta_r}.$$

- The similarity between two observations combines three components:

$$S_{ij}^{\text{DBSCAN}} = \exp\left(-\frac{\|x_i - x_j\|}{\sigma}\right)\left(\sqrt{\delta_i \delta_j}\right)^{\alpha} c_{ij}.$$

- Here, σ controls distance decay, α controls the density effect, and

$$c_{ij} = \begin{cases} 1, & c_i = c_j,\ c_i, c_j > 0, \\ \gamma, & c_i \neq c_j,\ c_i, c_j > 0, \\ \tau, & c_i = 0 \text{ or } c_j = 0. \end{cases}$$

  where $c_i, c_j \in \{0, 1, \ldots, k\}$ are the DBSCAN labels of x_i and x_j, with $c_i = 0$ denoting noise and $c_i > 0$ denoting membership in a dense cluster. A point x_i is a non-noise point when it belongs to some dense region detected by DBSCAN. Here $0 \leq \tau < \gamma < 1$: γ penalizes different non-noise clusters, while τ penalizes pairs involving noise.
- Thus, close points in dense regions and in the same non-noise cluster have high similarity. As in the UMAP and ISOMAP versions, DBSCAN similarity matrix is transformed into the **Riemannian Dissimilarity Coefficient**:

$$\rho_{ij} = 1 - S_{ij}^{\text{DBSCAN}}.$$

Definition

Let x_α and x_β rows of X, we define the **Riemannian Subtraction** as:

$$x_\alpha \ominus x_\beta = \rho_{\alpha\beta}(x_\alpha - x_\beta),$$

where:

$$\rho_{\alpha\beta} = 1 - S(x_\alpha, x_\beta).$$

Then, local distances can be defined as:

$$d(x_\alpha, x_\beta) = \|x_\alpha \ominus x_\beta\| = \sqrt{\langle x_\alpha \ominus x_\beta, x_\alpha \ominus x_\beta \rangle},$$

where $\langle \cdot, \cdot \rangle$ is an inner product on the vector space $\mathbb{R}^p$. We also define the the addition induced by the UMAP algorithm as $x_\alpha = x_\beta \oplus x_\gamma$ if $x_\gamma = x_\alpha \ominus x_\beta$.

The Riemannian Mean

In the subsequent definition, we generalize Fréchet mean:

Definition

Let $X \in M_{n\times p}$ the data table. We denote by $x_1, \ldots, x_n \in \mathbb{R}^p$ the rows of X and by $y_1, \ldots, y_p \in \mathbb{R}^n$ the columns of X. Each vector x_i can be also considered a point in the Riemannian manifold M. The **Riemannian Mean** is the minimizer of the sum-of-squared distances to the data:

$$g = \arg\min_{x\in M} \sum_{i=1}^{n} d\left(x, x_i\right)^2 .$$

Definition

We defined the **Riemannian variance-covariance** matrix $S \in M_{p\times p}$ of X as:

$$S = \frac{1}{n}\sum_{i=1}^{n}(x_i \ominus g)(x_i \ominus g)^t,$$

where:

$$(x_i \ominus g) = \rho_{i\lambda}\begin{bmatrix} x_{i1} - g_1 \\ \vdots \\ x_{ip} - g_p \end{bmatrix}.$$

Note that g must be equal to x_λ for some λ. Then, we can define **Riemannian correlation** between y_i and y_j columns of X, that are in $\mathbb{R}^n$, as follows $R(y_i, y_j) = \frac{S_{ij}}{\sqrt{S_{ii}S_{jj}}}$.

Algorithm 1 Riemannian Principal Component Analysis (R-PCA)

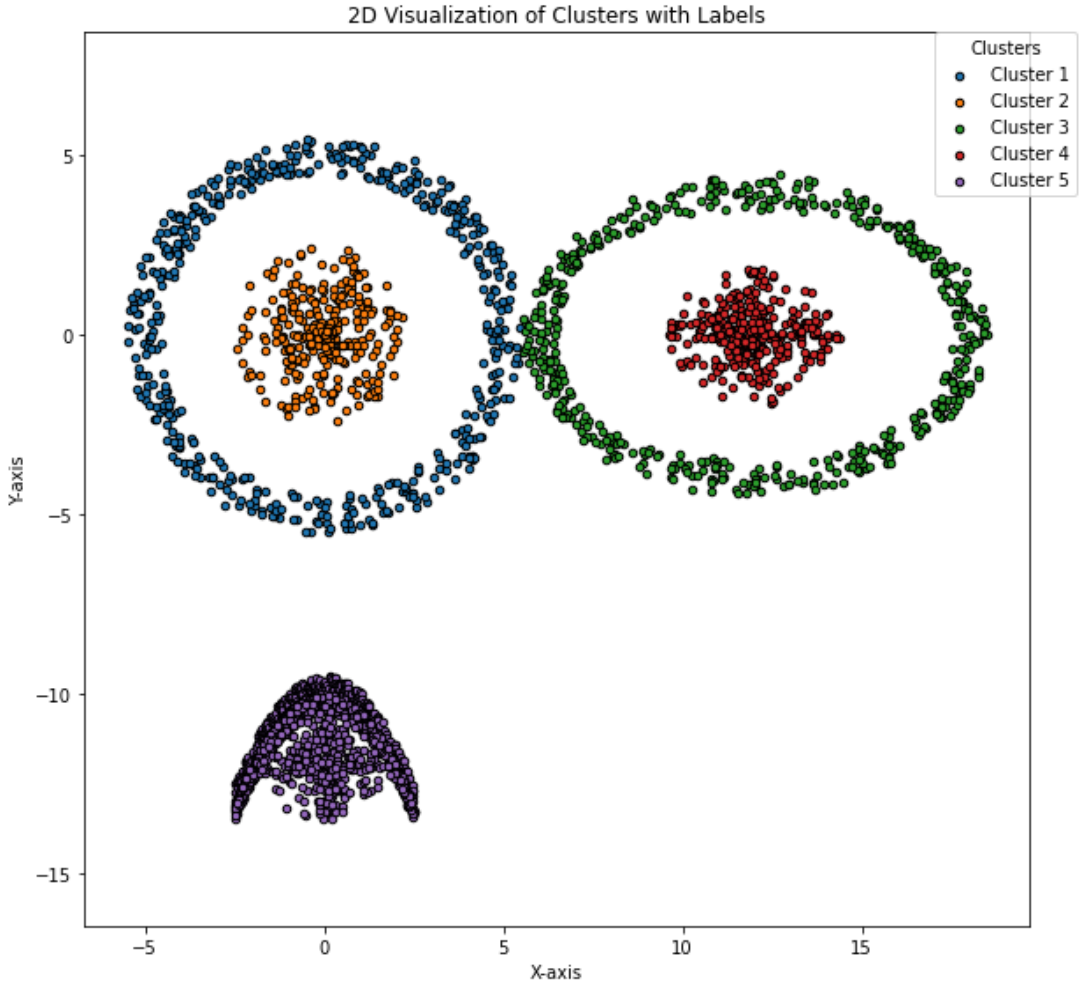
Input: $x_1, \ldots, x_n \in \mathbb{R}^p$ and the number k of nearest neighbors.
Output: Principal directions $v_s \in S_i$, variances $\lambda_s \in \mathbb{R}$.
Compute the matrix similarities $S(x_i, x_j)$ with $i, j = 1, \ldots, n$.
Compute the matrix $P = \rho_{ij}$ for $i, j = 1, \ldots, n$.
Compute the Riemannian distance matrix D.
Compute the Riemannian mean g using D.
Compute the Riemannian variance-covariance matrix $S \in M_{p \times p}$ as:

$$S = \frac{1}{n} \sum_{i=1}^{n} (x_i \ominus g)(x_i \ominus g)^t.$$

Compute the Riemannian correlation matrix $R \in M_{p \times p}$ where $R_{ij} = \frac{S_{ij}}{\sqrt{S_{ii} S_{jj}}}$.
Extract eigenvectors and eigenvalues of R: $\{v_s, \lambda_s\}$.
Compute the Riemannian Principal Components.
For the correlation circle, compute the correlations between the original variables and the principal components.

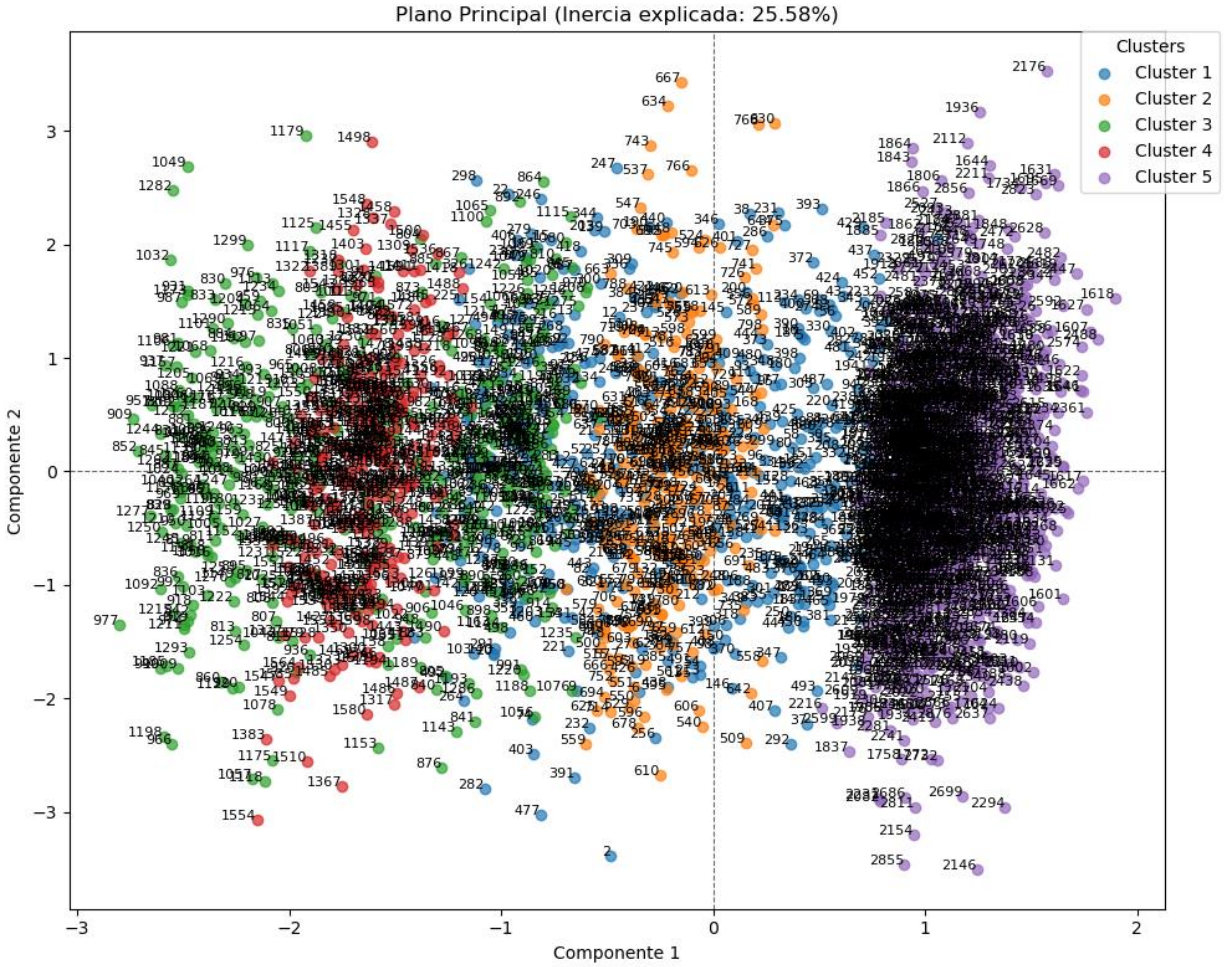
Example 1. Applications with simulated data:

- The **Data10D.csv** file contains data structured with 2900 rows and 10 variables, such that the first two columns determine five clusters within the dataset.
- The other 8 variables are random noise.



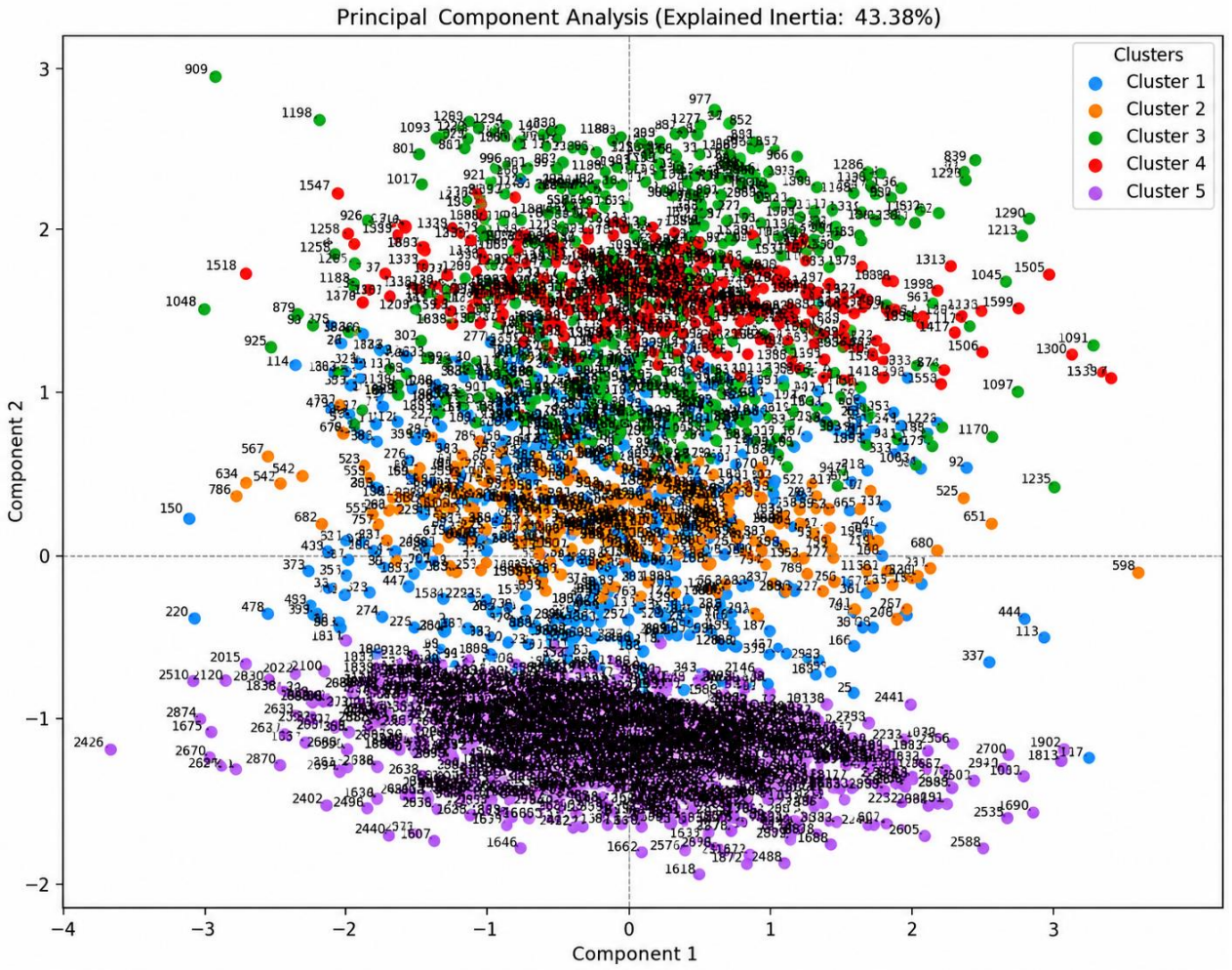
Example 1.
Applications with simulated data:

Classic PCA:



Example 1.
Applications with simulated data:

Riemannian PCA:



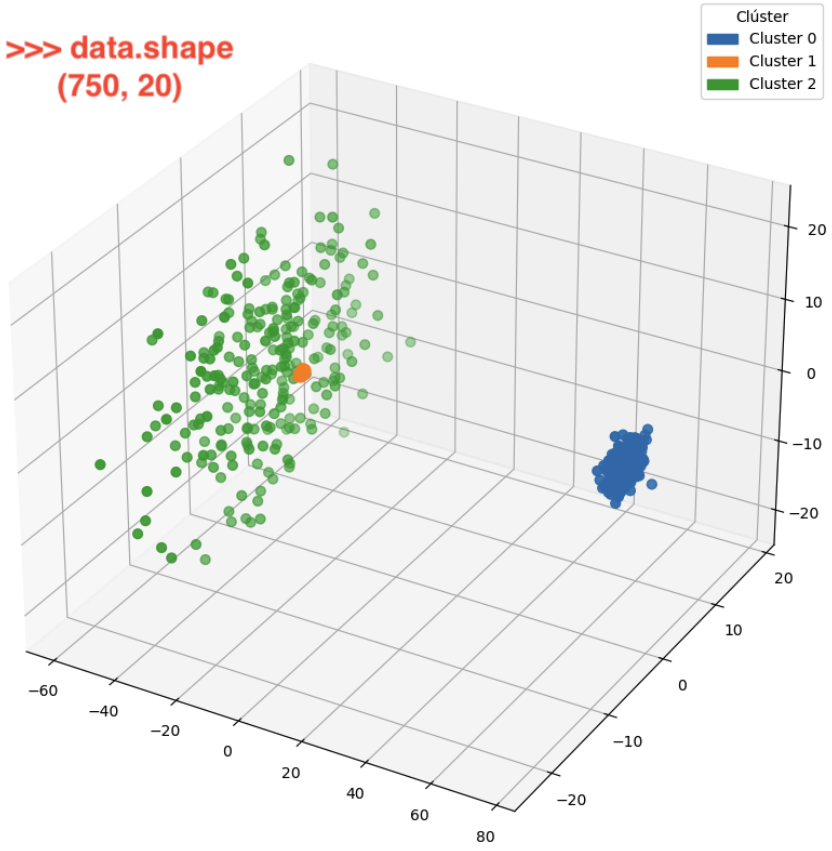
Example 2. Applications with simulated data:

This dataset simulates 3 clusters in 20-dimensional space with different densities:

- Cluster 0: moderately dense.
- Cluster 1: very dense.
- Cluster 2: highly dispersed

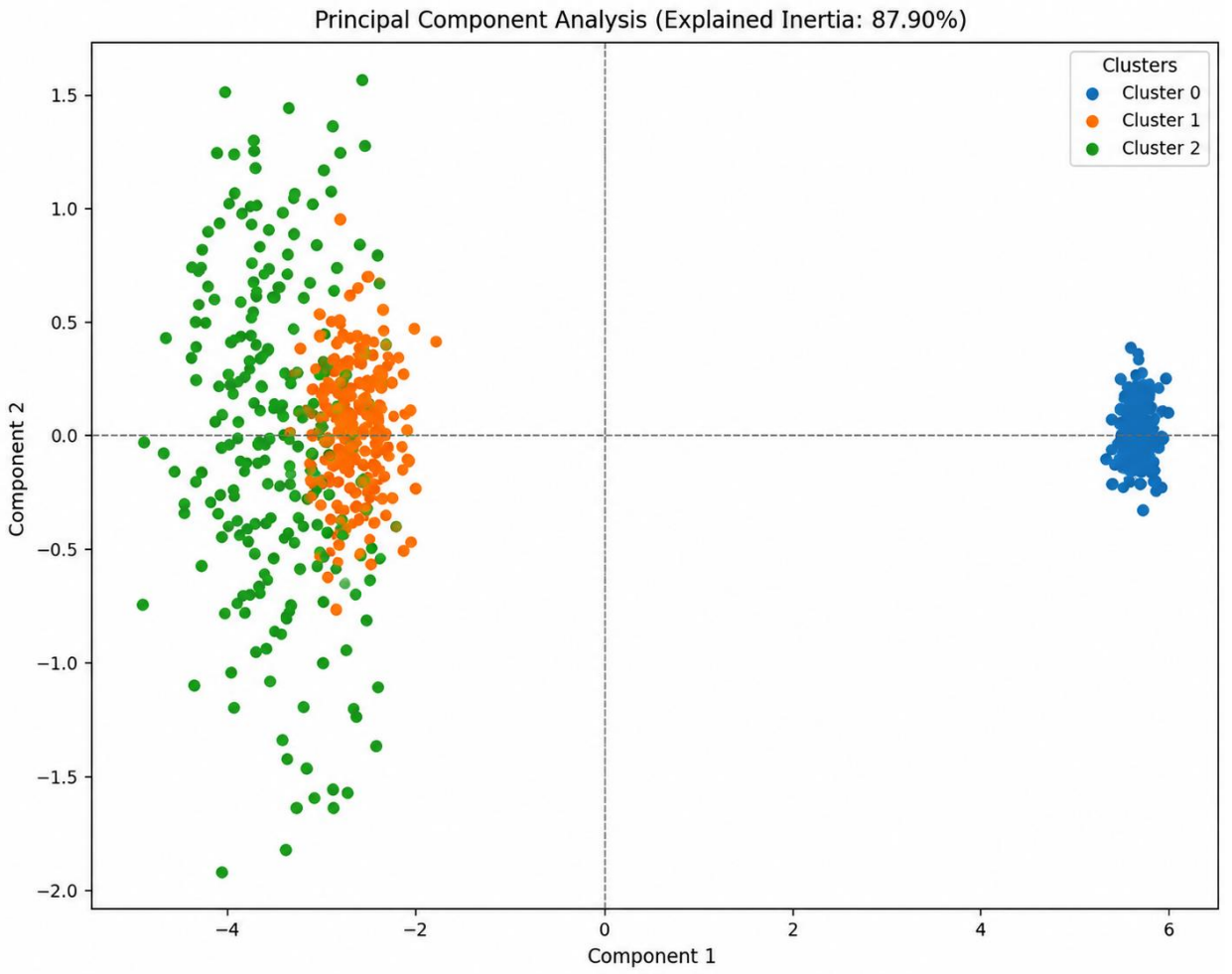
Each cluster has 250 points.

➢ *Cluster 1 is entirely embedded within Cluster 2, creating a nested structure that is difficult to distinguish using linear methods like PCA, which tend to blend them together.*



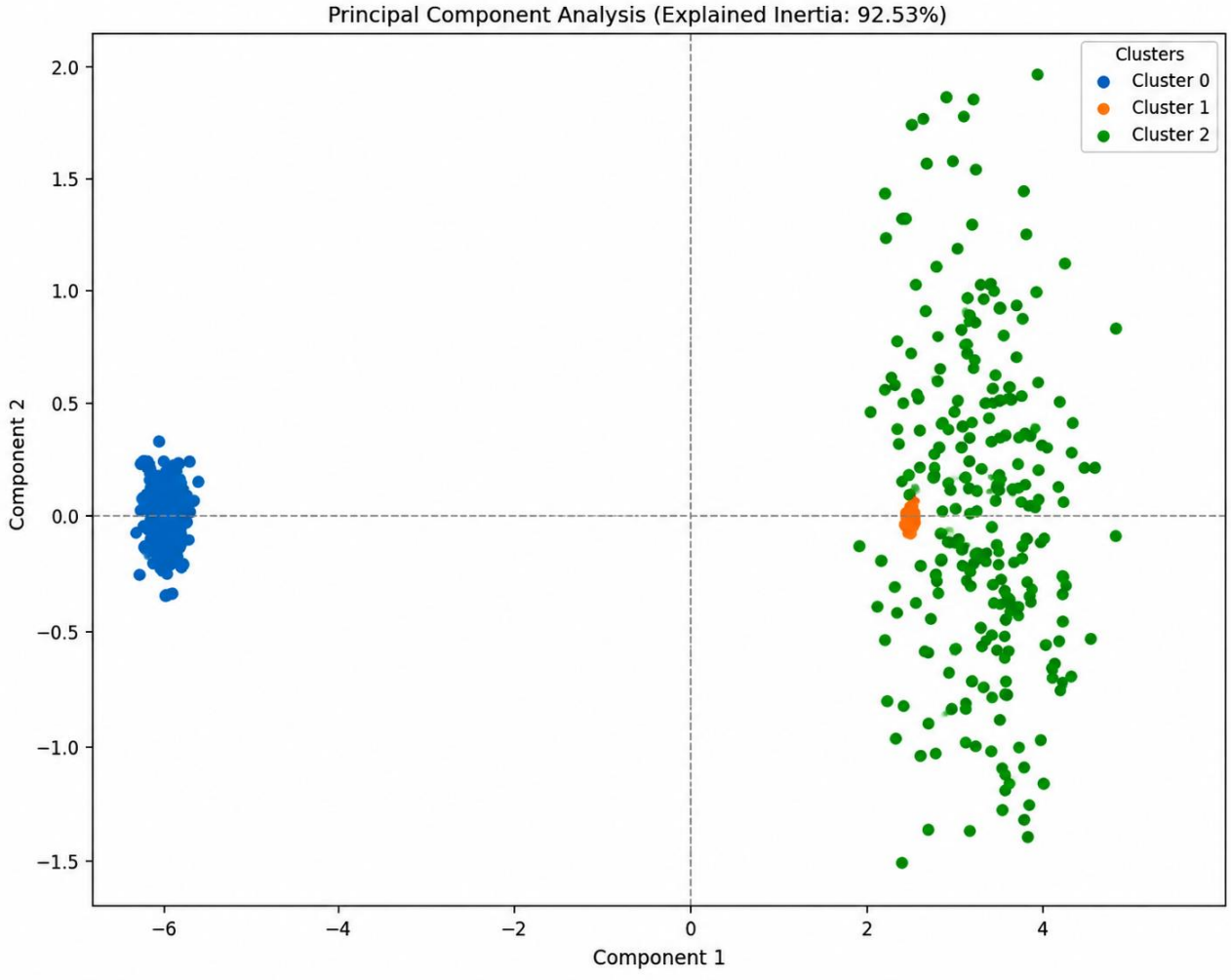
Example 2.
Applications with simulated data:

Classic PCA:



Example 2.
Applications with simulated data:

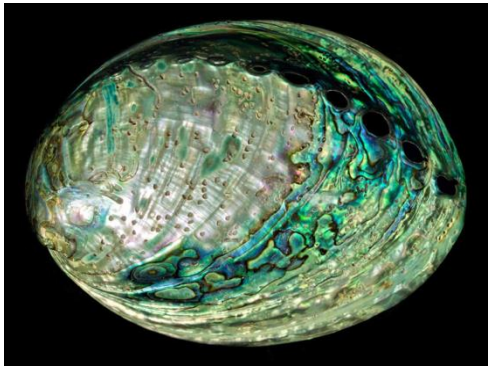
Riemannian PCA:



Example 3 The *Abalone* data table, where abalone is a marine mollusk that lives attached to rocks, contains morphometric information for 4177 abalones, with the goal of studying the relationship between 8 physical characteristics `Length`, `Diameter`, `Height`, `WholeWeight`, `ShuckedWeight`, `VisceraWeight`, `ShellWeight`, `Rings` and their age .

This data set is interesting for regression because the relationship between physical measurements and age is not strictly linear.

"https://archive.ics.uci.edu/ml/machine-learning-databases/abalone/abalone.data"



riemannianStats package in ***R***:





```r
library(riemannianStats)
library(FactoMineR)
```

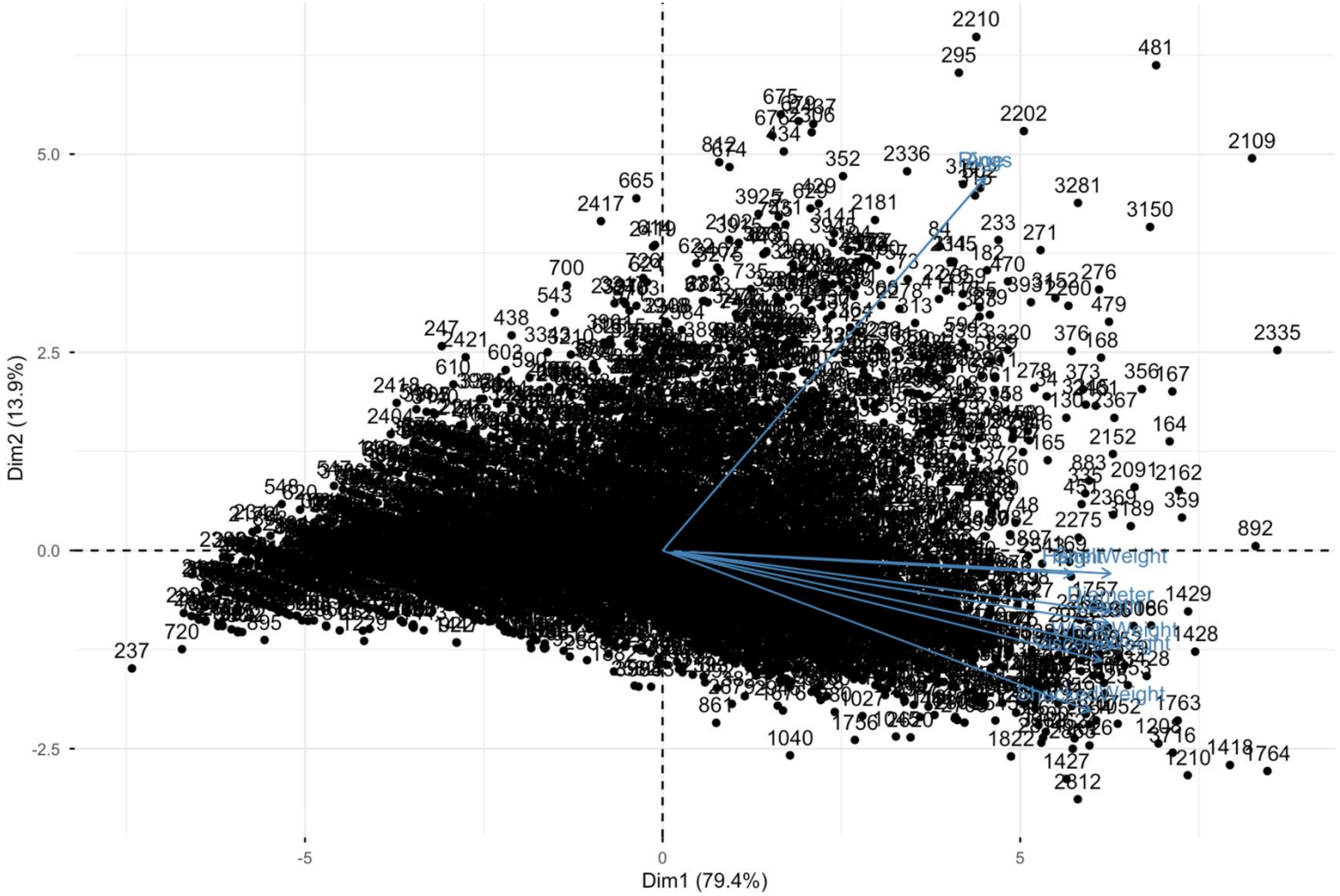
```r
url <- "https://archive.ics.uci.edu/ml/machine-learning-databases/abalone/abalone.data"

abalone <- read.csv(
  url,
  header = FALSE,
  stringsAsFactors = FALSE
)
```

PCA with FactoMineR

```
model<-PCA(abalone, scale.unit=TRUE, ncp=5, graph =
FALSE)
```

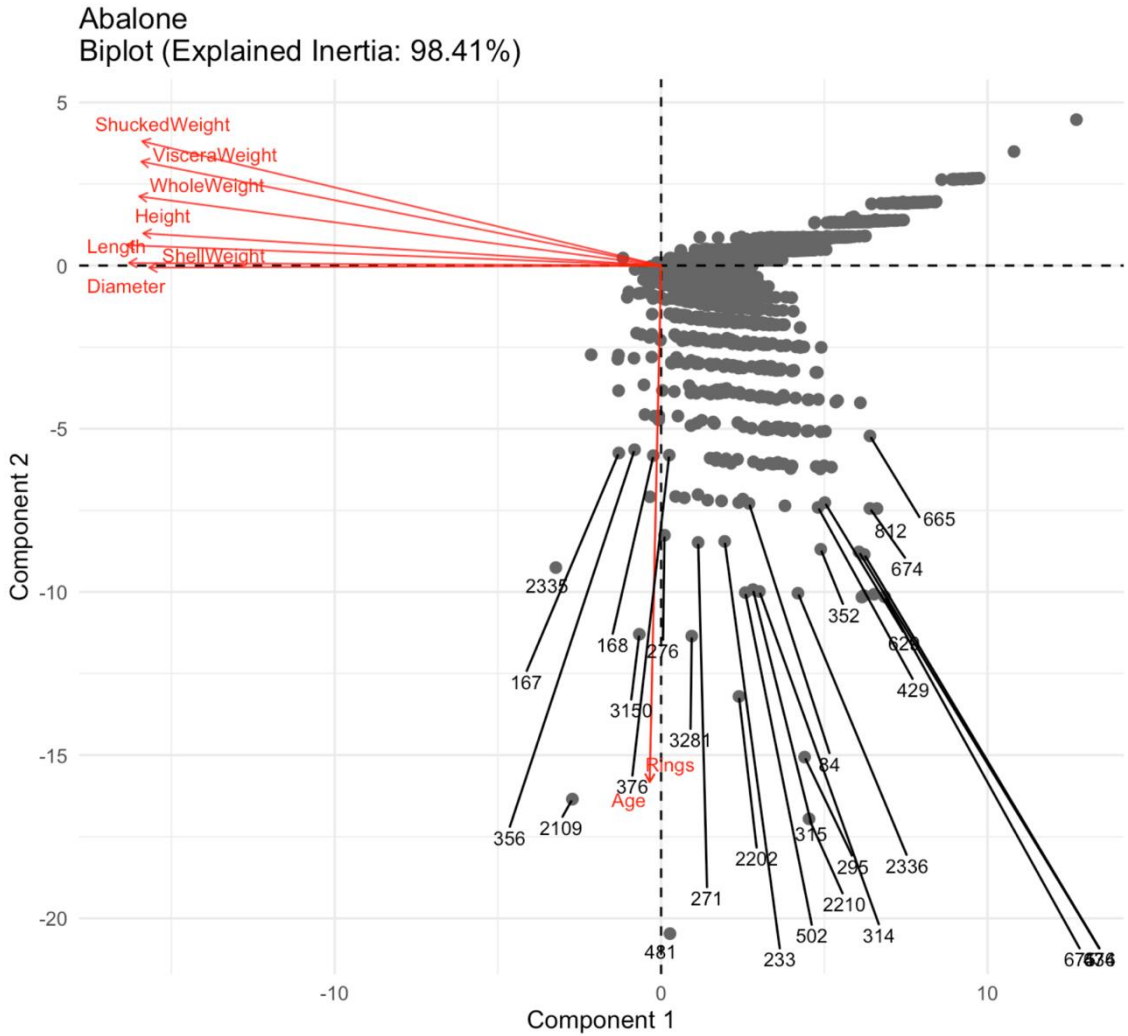
```
library(factoextra)
plot(model, axes=c(1, 2), choix="ind", col.ind="re
d",new.plot=TRUE)
```



Dim1 (79.4%)
Dim2 (13.9%)
5.0
2.5
0.0
-2.5
-5
0
5
Rings
Height
Weight
Diameter
Sex

```r
model <- riem.pca(
  data = abalone,
  scale.unit = TRUE,
  ncp = 5,
  axes = c(1, 2),
  n.neighbors = 50,
  min.dist = 0.1,
  metric = "euclidean",
  method = "dbscan"
)
```

```
riem.biplot(
  model,
  title = "Abalone"
)
```

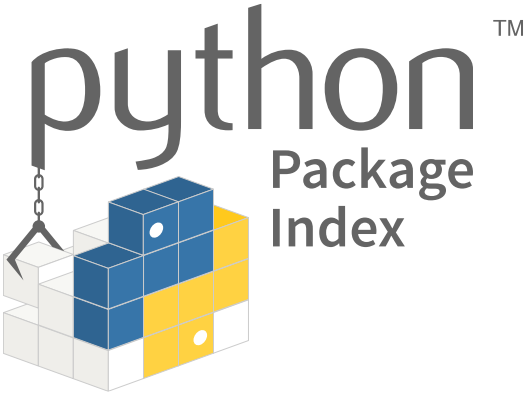


Riemannian STATS package in ***Python***:





Python Riemannian STATS Package:





Install via PyPI

To install the latest version of **Riemannian STATS** directly from PyPI, simply run:

```
pip install riemannian_stats
```

The Riemannian Regression

Definition

Let $X \in M_{n\times p}$ be a predictor data table whose rows are $x_1, \ldots, x_n \in \mathbb{R}^p$, that is, $x_i = (x_{i1} \ \ldots \ x_{ip})$, and let $y = (y_1, \ldots, y_n)^t \in \mathbb{R}^n$ be a response variable. We define the augmented matrix as:

$$Z = \begin{pmatrix} y_1 & x_{11} & \cdots & x_{1p} \\ y_2 & x_{21} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ y_n & x_{n1} & \cdots & x_{np} \end{pmatrix} \in M_{n\times(p+1)}.$$

Then, we apply some algorithm (like UMAP, ISOMAP or DBSCAN) to the rows of Z and obtain the similarity matrix $S(z_i, z_j)$, where $z_i = (y_i, x_{i1}, \ldots, x_{ip}) \in \mathbb{R}^{p+1}$.

The Riemannian Regression

Definition

We define $\rho_{ij} = 1 - S(z_i, z_j)$. Let $g = z_\lambda$ be the Riemannian mean, that is, the Fréchet mean of the data:

$$g = \arg\min_{z_\lambda \in Z} \sum_{i=1}^{n} d(z_i, z_\lambda)^2.$$

We write $g = (y_\lambda, x_{\lambda_1}, \ldots, x_{\lambda_p})$. For each row z_i, we define the Riemannian centering as $z_i \ominus g = \rho_{i\lambda}(z_i - g)$. In particular, $y_i \ominus y_\lambda = \rho_{i\lambda}(y_i - y_\lambda)$, and $x_{ij} \ominus x_{\lambda j} = \rho_{i\lambda}(x_{ij} - x_{\lambda j})$ for $j = 1, \ldots, p$.

Theorem

Let $g = (x_\lambda, y_\lambda)$ be the Riemannian mean, and consider the centered data matrices $X_c \in \mathbb{R}^{n \times p}$ and $y_c \in \mathbb{R}^n$ defined by:

$$X_c = \begin{pmatrix} x_1 - x_\lambda \\ \vdots \\ x_n - x_\lambda \end{pmatrix}, \qquad y_c = \begin{pmatrix} y_1 - y_\lambda \\ \vdots \\ y_n - y_\lambda \end{pmatrix}.$$

We define the Riemannian weight matrix as $W = \mathrm{diag}(\rho_{1\lambda}, \ldots, \rho_{n\lambda})$. The Riemannian regression estimator β_R and its centered prediction $\widehat{y}_c$ are given by:

1. $\beta_R = (X_c^t W^2 X_c)^{-1} X_c^t W^2 y_c$.
2. $\widehat{y}_c = X_c \beta_R = X_c (X_c^t W^2 X_c)^{-1} X_c^t W^2 y_c$.

Proof.

Riemannian regression seeks to minimize the sum of squares of the differences weighted by the distances to the center $\rho_{i\lambda}$:

$$\min_{\beta} \sum_{i=1}^{n} \left[\rho_{i\lambda}(y_i - y_\lambda) - \rho_{i\lambda} \sum_{j=1}^{p} \beta_j (x_{ij} - x_{\lambda j}) \right]^2 .$$

Equivalently, this objective function can be written as:

$$S(\beta) = \sum_{i=1}^{n} \rho_{i\lambda}^2 \left[(y_i - y_\lambda) - \sum_{j=1}^{p} \beta_j (x_{ij} - x_{\lambda j}) \right]^2 .$$

Thus, we observe that this is a weighted least squares regression with weights $\omega_i = \rho_{i\lambda}^2$.
We introduce the Riemannian-centered predictor matrix X_R and the Riemannian response vector y_R:

$$X_R = W X_c, \qquad y_R = W y_c.$$

Proof.

The objective function, in matrix form, can be written as the squared Euclidean norm:

$$S(\beta) = \|y_R - X_R\beta\|^2 = (y_R - X_R\beta)^t(y_R - X_R\beta).$$

To minimize $S(\beta)$, we differentiate with respect to β and set the derivative equal to zero:

$$\frac{\partial S}{\partial \beta} = -2X_R^t(y_R - X_R\beta) = 0.$$

This leads to the normal equations:

$$X_R^t X_R \beta_R = X_R^t y_R.$$

Substituting $X_R = WX_c$ and $y_R = Wy_c$, and noting that W is a diagonal matrix, hence $W^t = W$, we obtain:

$$(WX_c)^t(WX_c)\beta_R = (WX_c)^t(Wy_c).$$

Proof.

Therefore:

$$(X_c^t W^t W X_c)\beta_R = X_c^t W^t W y_c.$$

Since $W^t W = W^2$, we obtain:

$$(X_c^t W^2 X_c)\beta_R = X_c^t W^2 y_c.$$

Solving for β_R, we get:

$$\beta_R = (X_c^t W^2 X_c)^{-1} X_c^t W^2 y_c.$$

The prediction for the centered data is obtained directly as:

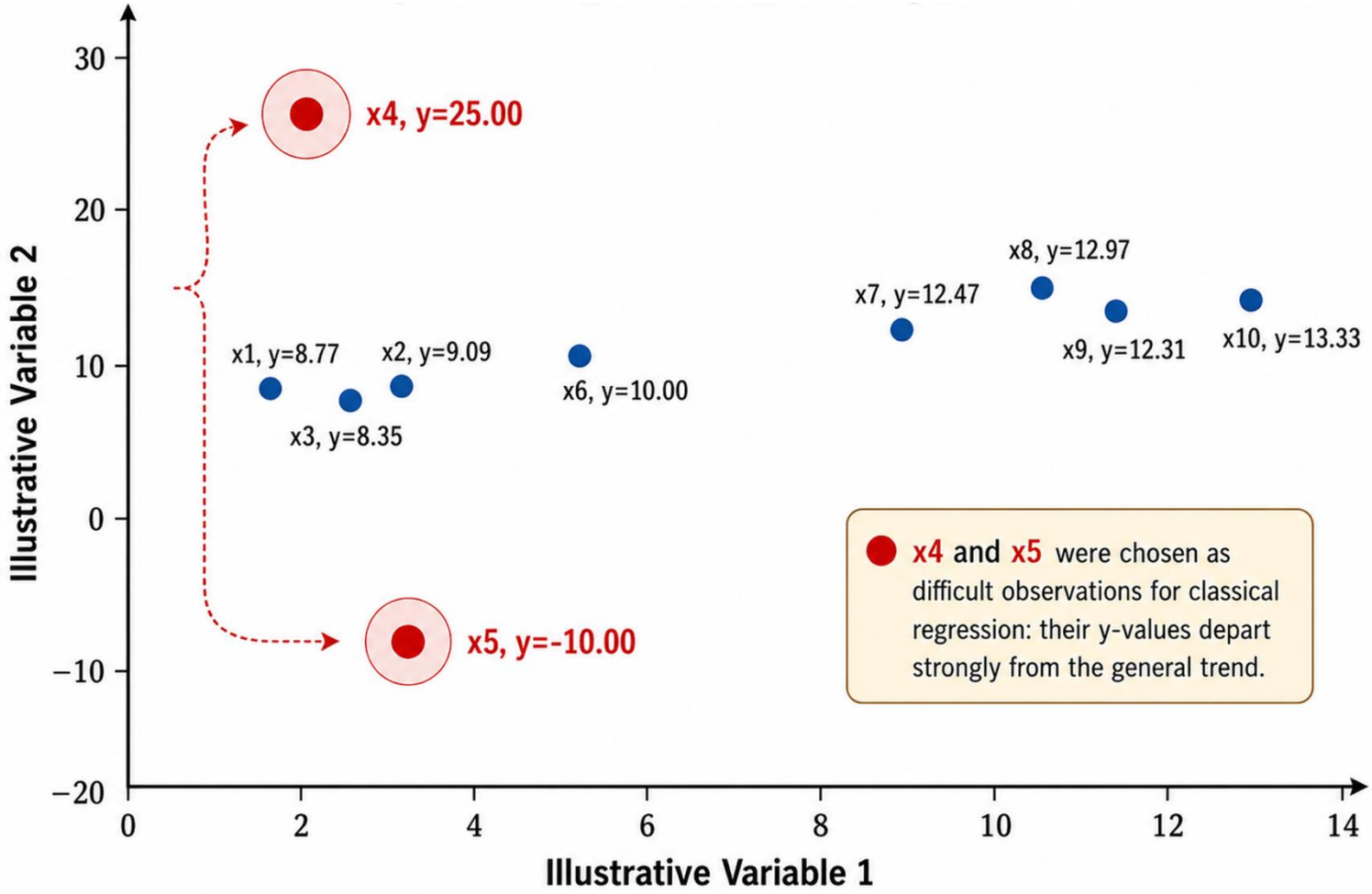
$$\widehat{y}_c = X_c \beta_R = X_c (X_c^t W^2 X_c)^{-1} X_c^t W^2 y_c.$$

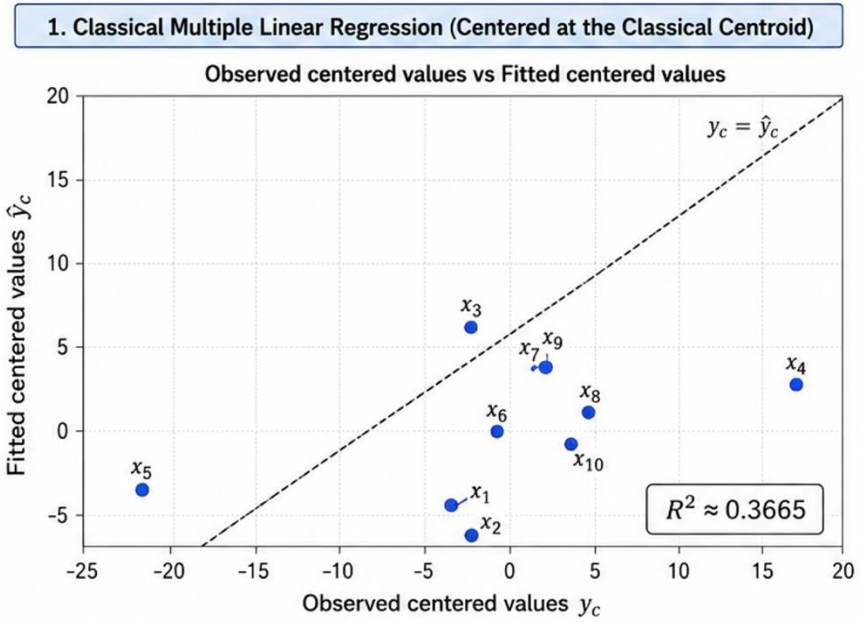
Finally, the prediction in the original scale, $\widehat{y}_R$, for a given center is

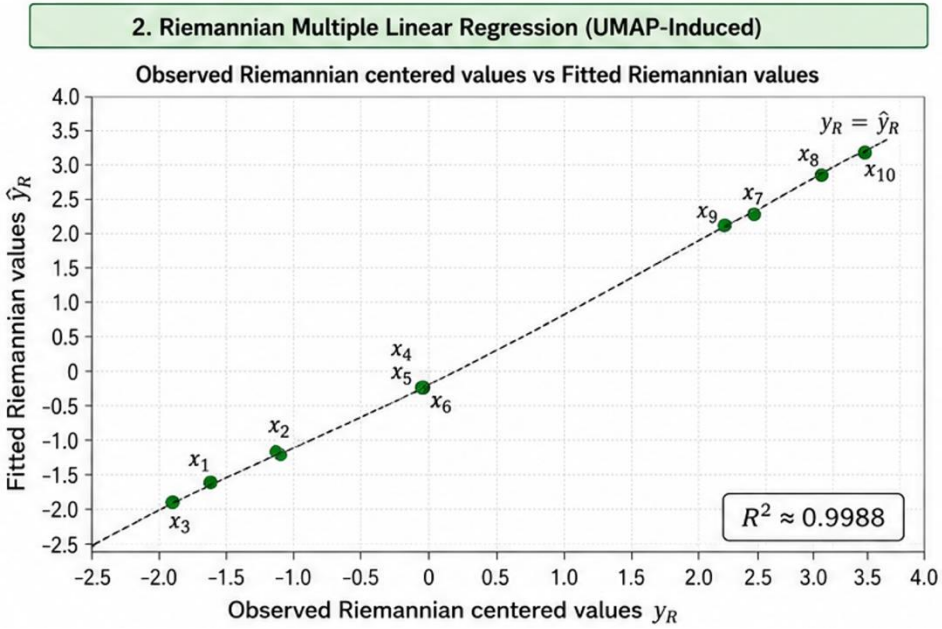
$$\widehat{y}_R = y_\lambda \mathbf{1} + \widehat{y}_c.$$

Example 1 Consider this data table with $n = 10$ individuals and $p = 5$ predictor variables. Observations x_4 and x_5 were deliberately chosen as difficult observations for classical regression, since their y values deviate strongly from the general trend. This will allow us to observe the effect of the local geometry induced by UMAP.

Individual	X_1	X_2	X_3	X_4	X_5	y
x_1	0.0	0.0	0.1	0.2	0.0	8.77
x_2	0.5	0.1	0.2	0.0	0.1	9.09
x_3	0.2	0.8	0.1	0.3	0.0	8.35
x_4	0.9	0.9	0.2	0.2	0.1	25.00
x_5	1.2	0.3	0.3	0.1	0.2	-10.00
x_6	1.5	0.7	0.3	0.3	0.1	10.00
x_7	4.8	5.0	5.0	4.7	4.9	12.47
x_8	5.5	5.1	5.1	4.8	5.0	12.97
x_9	5.2	5.8	4.9	5.3	5.1	12.31
x_{10}	6.0	5.6	5.2	5.4	5.3	13.33


Illustrative Variable 2
Illustrative Variable 1
30
20
10
0
−10
−20
0
2
4
6
8
10
12
14
x4, y=25.00
x5, y=-10.00
x1, y=8.77
x2, y=9.09
x3, y=8.35
x6, y=10.00
x7, y=12.47
x8, y=12.97
x9, y=12.31
x10, y=13.33
x4 and x5 were chosen as difficult observations for classical regression: their y-values depart strongly from the general trend.





$$R^2_{\text{classical}} \approx 0.3665.$$

$$R^2_{\text{Riem}} \approx 0.9988.$$

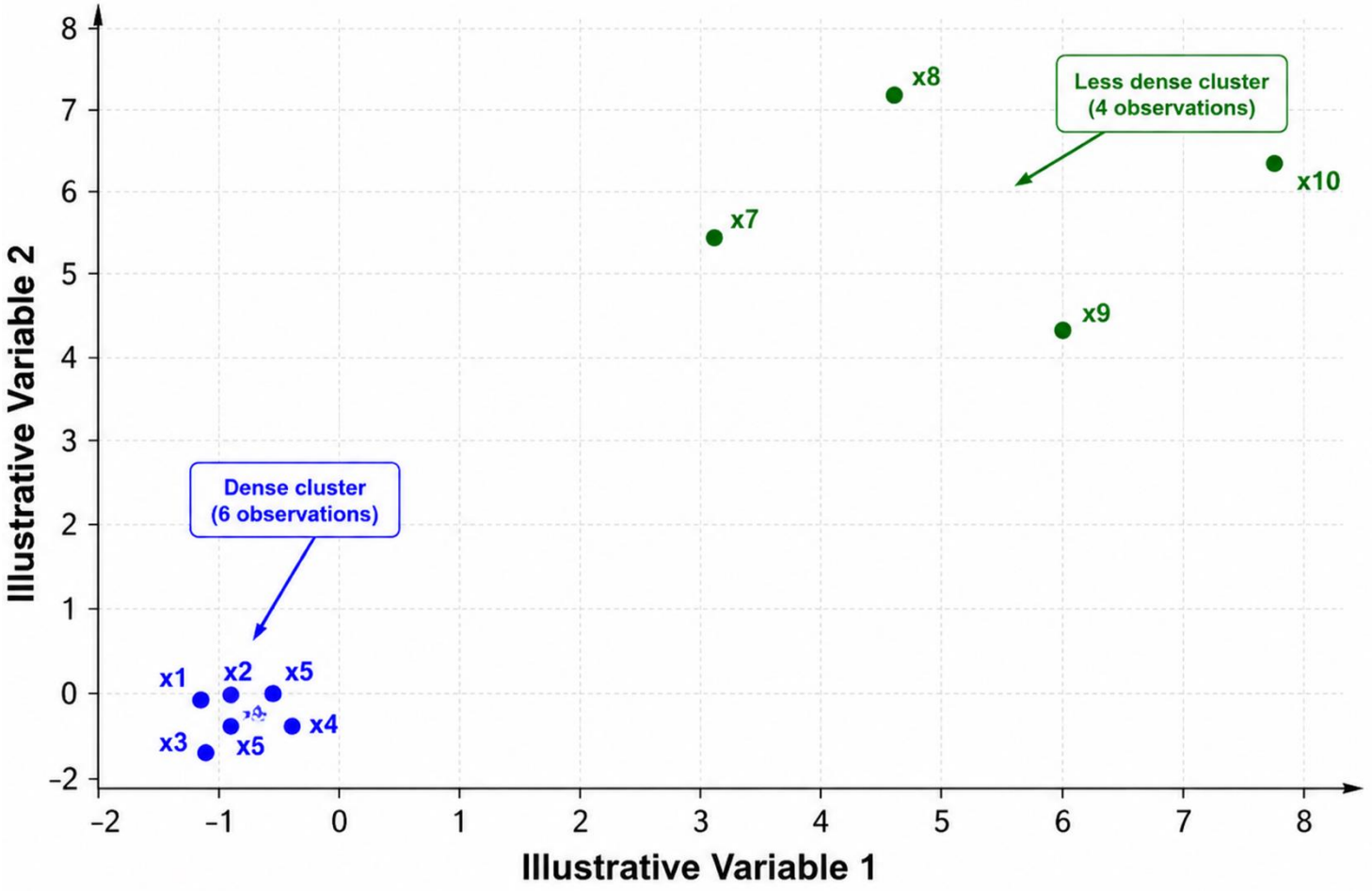
Example 2 Consider a data table with $n = 10$ individuals and $p = 5$ predictor variables. The data are formed by two clusters:

- A dense cluster with 6 observations.
- A less dense cluster with 4 observations.

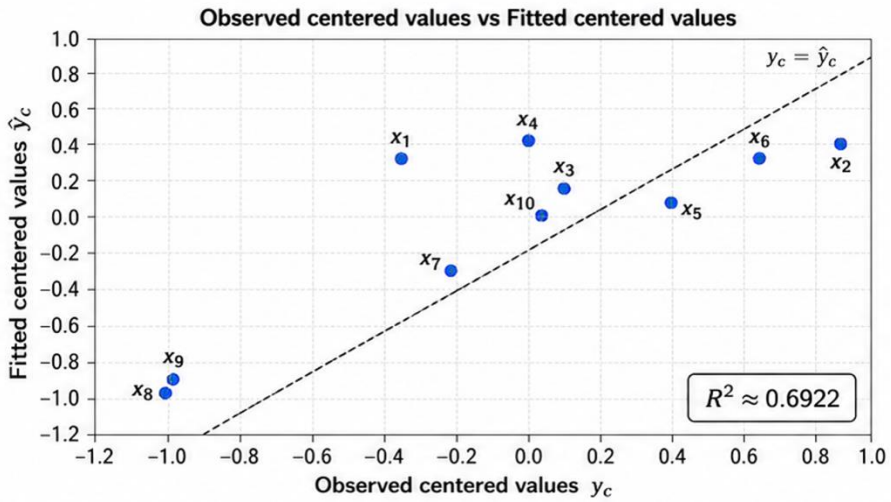
The data table is:

Individual	Cluster	X_1	X_2	X_3	X_4	X_5	y
x_1	A	0.00	0.02	−0.01	0.03	0.00	4.286
x_2	A	0.04	−0.01	0.02	0.01	−0.02	5.486
x_3	A	−0.03	0.01	0.00	−0.02	0.02	4.776
x_4	A	0.02	0.04	−0.02	0.00	0.01	4.762
x_5	A	−0.01	−0.03	0.01	0.02	−0.01	5.015
x_6	A	0.03	0.00	0.03	−0.01	0.00	5.289
x_7	B	3.00	3.50	2.80	3.20	3.10	4.500
x_8	B	4.00	2.90	3.40	3.80	2.70	3.698
x_9	B	2.70	4.10	3.60	2.90	3.90	3.718
x_{10}	B	4.50	4.20	2.90	4.30	3.50	4.655

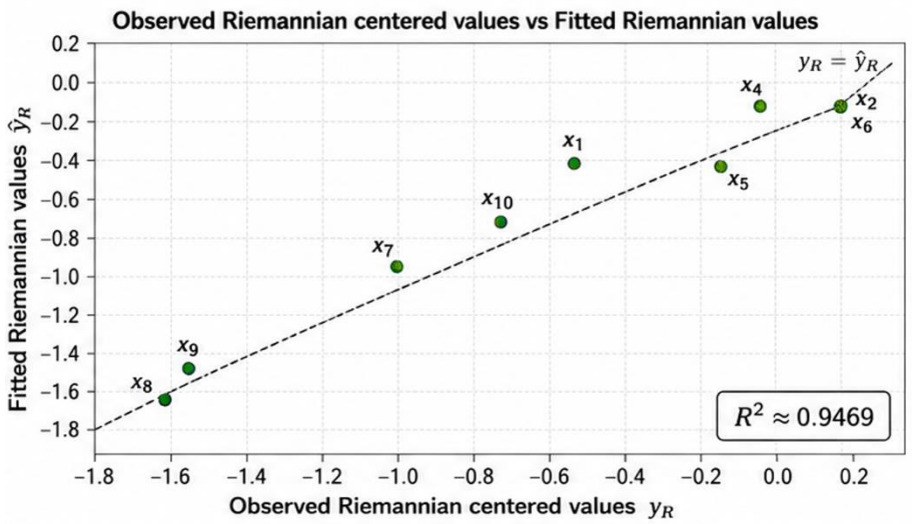
The data are organized into two clusters: a dense cluster with 6 observations and a less dense cluster with 4 observations.



1. Classical Multiple Linear Regression (Centered at the Classical Centroid)



2. Riemannian Multiple Linear Regression (UMAP-Induced)



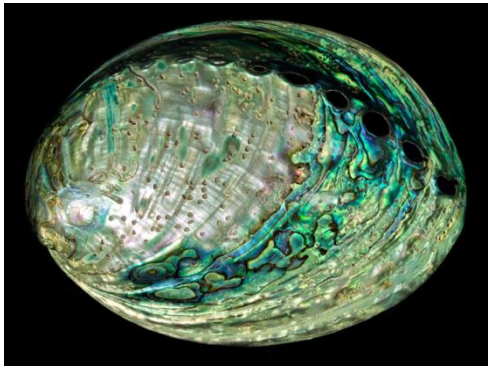
$$R^2_{\text{classical}} \approx 0.6922.$$

$$R^2_{\text{Riem}} \approx 0.9469.$$

Example 3 The *Abalone* data table, where abalone is a marine mollusk that lives attached to rocks, contains morphometric information for 4177 abalones, with the goal of studying the relationship between 8 physical characteristics `Length`, `Diameter`, `Height`, `WholeWeight`, `ShuckedWeight`, `VisceraWeight`, `ShellWeight`, `Rings` and their age .

This data set is interesting for regression because the relationship between physical measurements and age is not strictly linear.

"https://archive.ics.uci.edu/ml/machine-learning-databases/abalone/abalone.data"



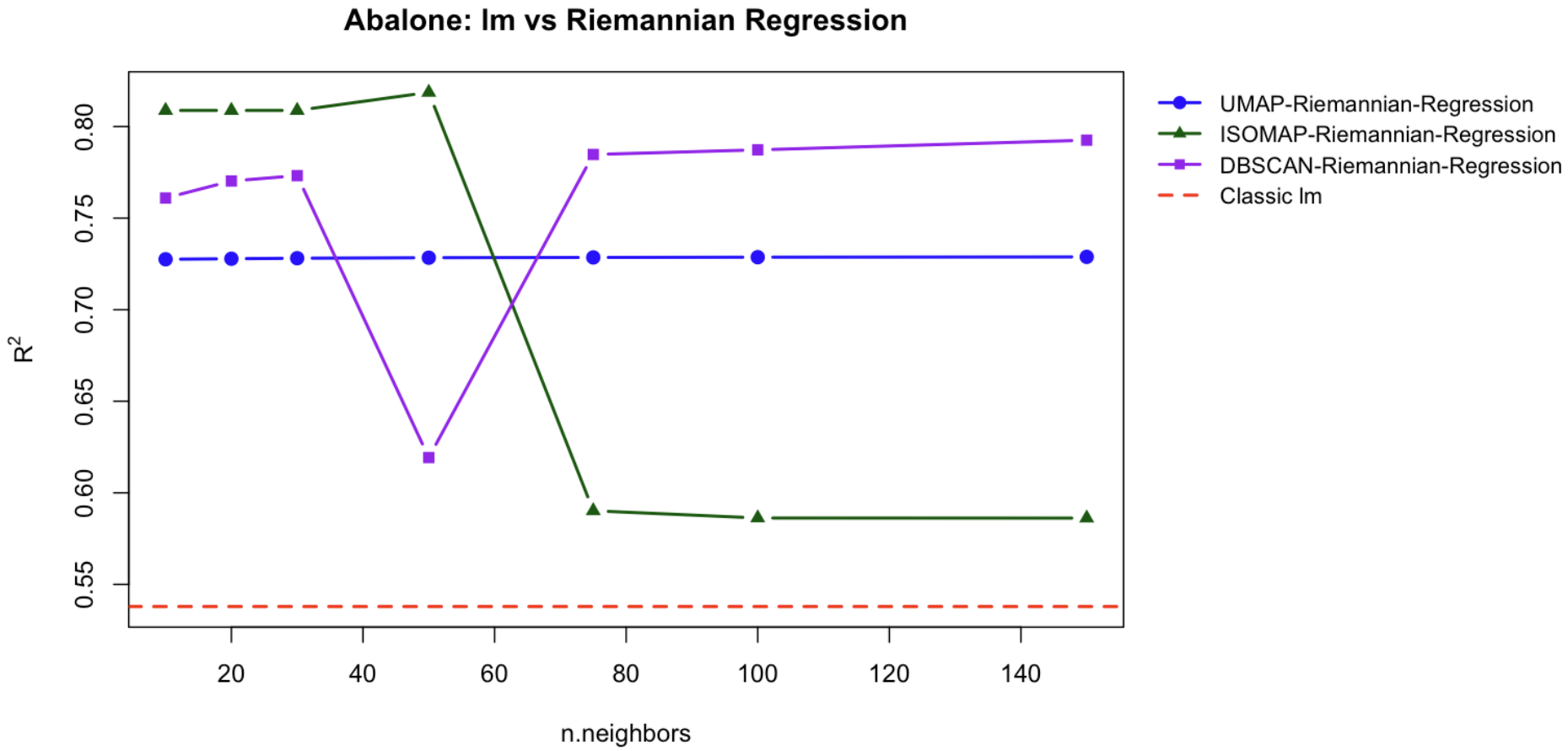
```r
library(riemannianStats)
```

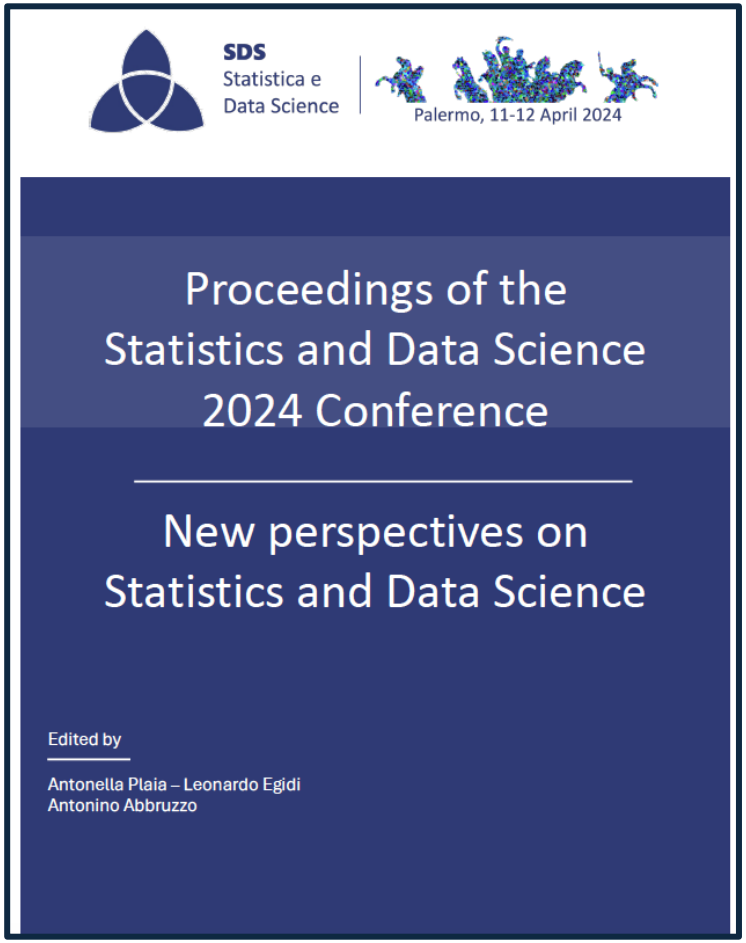
```r
modelo.lm <- lm(
  Age ~ Length + Diameter + Height +
    WholeWeight + ShuckedWeight + VisceraWeight + ShellWeight,
  data = abalone
)

# R^2
R2.lm <- summary(modelo.lm)$r.squared
R2.lm
#> [1] 0.5276299
```

```r
# Abalone with ISOMAP
# ============================================================
# Riemannian regression with ISOMAP
# ============================================================
n.neighbors <- 25
fit.R <- riem.lm(
  data = data.analysis.scaled,
  y = y,
  n.neighbors = n.neighbors,
  similarities = "isomap",
  min.dist = 0.1,
  metric = "euclidean"
)

R2.R <- fit.R$R2.R
R2.R
#> [1] 0.7569302
```


Abalone: lm vs Riemannian Regression
UMAP-Riemannian-Regression
ISOMAP-Riemannian-Regression
DBSCAN-Riemannian-Regression
Classic lm
R2
0.55
0.60
0.65
0.70
0.75
0.80
20
40
60
80
100
120
140
n.neighbors

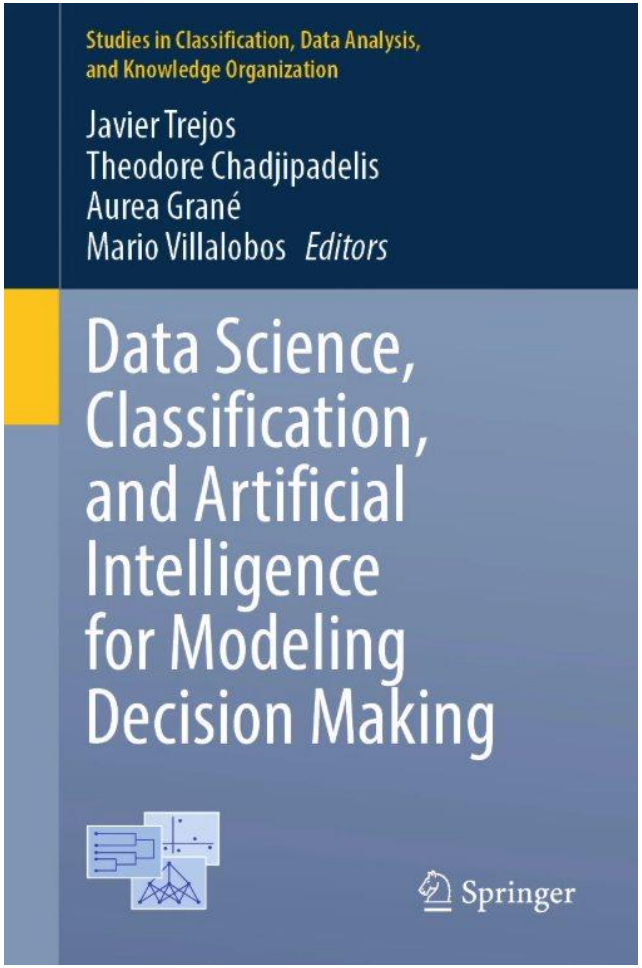


Riemannian Statistics for Any Type of Data

Oldemar Rodríguez Rojas

Abstract This paper introduces a novel approach to statistics and data analysis, departing from the conventional assumption of data residing in Euclidean space to consider a Riemannian Manifold. The challenge lies in the absence of vector space operations on such manifolds. Pennec X. et al. in their book Riemannian Geometric Statistics in Medical Image Analysis proposed analyzing data on Riemannian manifolds through geometry, this approach is effective with structured data like medical images, where the intrinsic manifold structure is apparent. Yet, its applicability to general data lacking implicit local distance notions is limited. We propose a solution to generalize Riemannian statistics for any type of data.

Key words: Riemannian manifold; simplicial complexes; UMAP; homeomorphism.



Riemannian Statistics for Any Type of Data

Oldemar Rodríguez Rojas

Abstract This paper introduces a novel approach to statistics and data analysis, departing from the conventional assumption of data residing in Euclidean space to consider a Riemannian Manifold. The challenge lies in the absence of vector space operations on such manifolds. Pennec X. et al. in their book Riemannian Geometric Statistics in Medical Image Analysis proposed analyzing data on Riemannian manifolds through geometry, this approach is effective with structured data like medical images, where the intrinsic manifold structure is apparent. Yet, its applicability to general data lacking implicit local distance notions is limited. We propose a solution to generalize Riemannian statistics for any type of data.

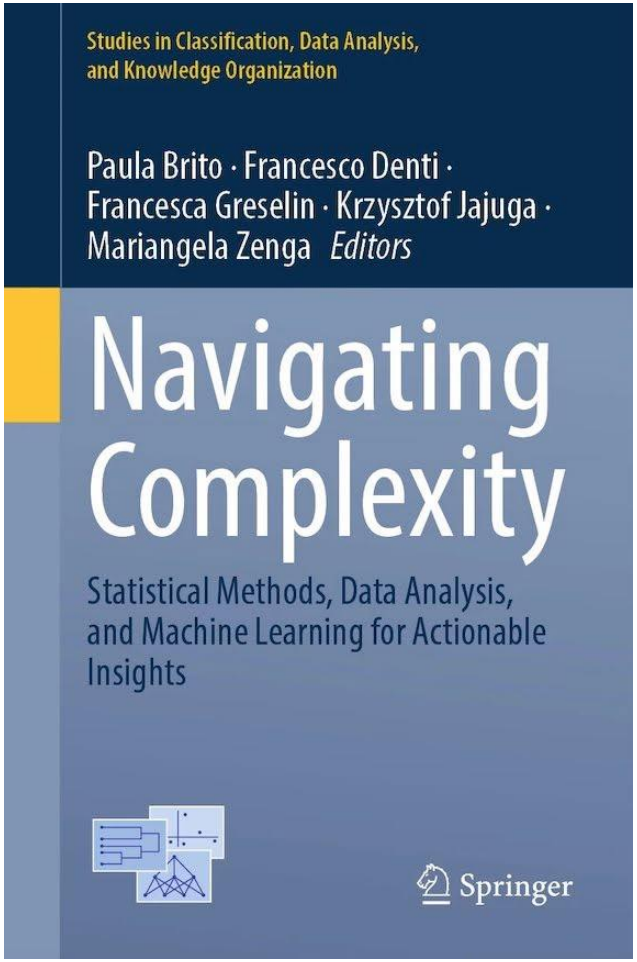
Keywords: Statistics; data analysis; Riemannian manifold; simplicial complexes; UMAP; homeomorphism; topology

1 Introduction

Each point in a data table can be imagined as a star or planet in the universe, especially when dealing with big data issues. In the universe, due to the infinitely different sizes of constellations, there are vastly different perceptions of distances between celestial bodies. For example, two constellations or galaxies that appear to be the same size from a distance (from Earth, for example) could be infinitely different, and one could even fit inside the other in a very small portion or empty space within it. For this reason, especially in problems involving Big Data, *thinking that the data is in Euclidean space is just as wrong as thinking that the earth is flat.*

Similarly, in data, there are local notions of distance corresponding to different regions of the data, and this should be considered when calculating indices or

Oldemar Rodríguez Rojas
School of Mathematics, CIMPA, University of Costa Rica, San José, Costa Rica, e-mail: oldemar.rodriguez@ucr.ac.cr



© The Author(s), under exclusive license to Springer Nature Switzerland AG 2026
P. Brito et al. (Eds.): IFCS 2026, STUDIES CLASS, pp. 243–251, 2026.
https://doi.org/10.1007/978-3-032-32009-4_30

Riemannian Principal Component Analysis

Oldemar Rodríguez(✉)

School of Mathematics and Research Center for Pure and Applied Mathematics (CIMPA), University of Costa Rica, San José, Costa Rica
oldemar.rodriguez@ucr.ac.cr
https://oldemarrodriguez.com/

Abstract. This paper proposes an extension of Principal Component Analysis (PCA) beyond the assumption of Euclidean data, enabling its application to Riemannian manifolds. A central challenge is the absence of vector space operations on such manifolds. Fletcher et al., in [1] and [4], introduced Principal Geodesic Analysis (PGA) as a geometric method for analyzing manifold-valued data, particularly effective for structured datasets such as medical images, where intrinsic geometry is explicit.



arXiv:2609.40242v1 [math.ST] 30 Sep 2026

Riemannian Regression

Oldemar Rodríguez[1]

School of Mathematics and Research Center for Pure and Applied Mathematics (CIMPA), University of Costa Rica, San José, Costa Rica,
oldemar.rodriguez@ucr.ac.cr,
WWW home page: https://oldemarrodriguez.com/

Abstract. Classical linear regression assumes that the relevant geometry of the predictor space is Euclidean and that all centered observations contribute to the least-squares fit in the same geometric scale. This paper proposes *Riemannian Regression*, a regression framework in which the usual vector differences are replaced by locally weighted differences induced by a data-dependent similarity structure. We introduce a generalized framework, termed *Riemannian Regression*, extending classic regression to any data endowed with a local distance structure. By equipping data tables with local metrics, we adapt regression model to incorporate manifold geometry. Given a similarity matrix $S = (S_{ij})$, obtained from UMAP, ISOMAP, or DBSCAN [3–5], we define the dissimilarity coefficient $\rho_{ij} = 1 - S_{ij}$ and the induced subtraction $x_i \ominus x_j = \rho_{ij}(x_i - x_j)$. A Riemannian center $g = x_\lambda$ is selected as a discrete Fréchet mean, and regression is performed on the Riemannian-centered variables $X_R = WX_{c,\lambda}$ and $y_R = Wy_{c,\lambda}$, where $W = \mathrm{diag}(\rho_{1\lambda}, \ldots, \rho_{n\lambda})$. The resulting estimator has the weighted least-squares form $\widehat{\beta}_R = (X_{c,\lambda}^t W^2 X_{c,\lambda})^{-1} X_{c,\lambda}^t W^2 y_{c,\lambda}$. The proposed approach preserves the linear form of the regression model while changing the geometry of the fit. The paper develops three ways to construct the local metric: UMAP-based fuzzy similarities, ISOMAP-based normalized geodesic distances, and DBSCAN-based density similarities. Simulated examples and the Abalone data set illustrate how Riemannian Regression can reduce the influence of locally anomalous observations and adapt to regions with different local densities.

Keywords: Riemannian regression, UMAP, ISOMAP, DBSCAN, local metrics, weighted least squares, Riemannian statistics

1 Introduction

Classical multiple linear regression models a response variable $y \in \mathbb{R}^n$ from a predictor matrix $X \in \mathbb{R}^{n\times p}$ by assuming that the relevant differences between observations are Euclidean. If $\bar{x}$ and $\bar{y}$ denote the usual arithmetic means, the centered least-squares estimator is $\widehat{\beta}_c = (X_c^t X_c)^{-1} X_c^t y_c$, where X_c has rows $x_i - \bar{x}$ and y_c has entries $y_i - \bar{y}$.

This approach is effective when the global Euclidean geometry adequately represents the structure of the data. However, many data sets contain nonlinear

THANK YOU….