Journal of Statistical Software

April 2023, Volume 106, Issue 9. doi: 10.18637/jss.v106.i09

intRinsic: An R Package for Model-Based Estimation of the Intrinsic Dimension of a Dataset

Francesco Denti
Università Cattolica del Sacro Cuore

Abstract

This article illustrates **intRinsic**, an R package that implements novel state-of-the-art likelihood-based estimators of the intrinsic dimension of a dataset, an essential quantity for most dimensionality reduction techniques. In order to make these novel estimators easily accessible, the package contains a small number of high-level functions that rely on a broader set of efficient, low-level routines. Generally speaking, **intRinsic** encompasses models that fall into two categories: homogeneous and heterogeneous intrinsic dimension estimators. The first category contains the *two nearest neighbors* estimator, a method derived from the distributional properties of the ratios of the distances between each data point and its first two closest neighbors. The functions dedicated to this method carry out inference under both the frequentist and Bayesian frameworks. In the second category, we find the *heterogeneous intrinsic dimension algorithm*, a Bayesian mixture model for which an efficient Gibbs sampler is implemented. After presenting the theoretical background, we demonstrate the performance of the models on simulated datasets. This way, we can facilitate the exposition by immediately assessing the validity of the results. Then, we employ the package to study the intrinsic dimension of the *Alon* dataset, obtained from a famous microarray experiment. Finally, we show how the estimation of homogeneous and heterogeneous intrinsic dimensions allows us to gain valuable insights into the topological structure of a dataset.

Keywords: nearest neighbors, likelihood-based method, heterogeneous intrinsic dimension, Bayesian mixture model, R.

1. Introduction

Statisticians and data scientists are often called to manipulate, analyze, and summarize datasets that present high-dimensional and elaborate dependency structures. In numerous cases, these large datasets contain variables characterized by a considerable amount of redun-

dant information. One can exploit these redundancies to represent a large dataset on a much lower-dimensional scale. This summarization procedure, called *dimensionality reduction*, is a fundamental step in many statistical analyses. For example, dimensionality reduction techniques grant the feasibility of otherwise challenging tasks such as large data manipulation and visualization by reducing computational time and memory requirements.

More formally, dimensionality reduction is possible whenever the data points take place on one or more manifolds characterized by a lower dimension than what has been observed initially. In this context, the word *manifold* is used to indicate a constraint surface embedded in high-dimensional space along which dissimilarities between data points are best represented (Tenenbaum, De Silva, and Langford 2000; Lee, Rodriguez, and Madabhushi 2008). We call the dimension of a latent, potentially nonlinear manifold the *intrinsic dimension* (ID). Several other definitions of ID exist in the literature. For example, we can regard the ID as the minimal number of parameters needed to represent all the information contained in the data without significant information loss (Ansuini, Laio, Macke, and Zoccolan 2019; Rozza, Lombardi, Rosa, Casiraghi, and Campadelli 2011; Bennett 1969).

Intuitively, the ID is an indicator of the complexity of the features of a dataset. It is a necessary piece of information to have before attempting to perform any dimensionality reduction, manifold learning, or visualization tasks. Indeed, most dimensionality reduction methods would be worthless without a reliable estimate of the true ID they need to target: an underestimated ID value can cause needless information loss. At the same time, the reverse can lead to an unnecessary waste of time and computational resources (Hino, Fujiki, Akaho, and Murata 2017). Beyond dimensionality reduction, ID estimation methods have been successfully employed, for instance, in studying physical systems (Mendes-Santos, Turkeshi, Dalmonte, and Rodriguez 2021) and analyzing neural networks (Ansuini *et al.* 2019). For more examples of applications, see also Carter, Raich, and Hero (2010) and the references in the discussion in Bac, Mirkes, Gorban, Tyukin, and Zinovyev (2021).

Over the past few decades, a vast number of methods for ID estimation and dimensionality reduction have been developed. The algorithms can be broadly classified into two main categories: projection and geometric approaches. The former maps the original data to a lower-dimensional space. The projection function can be linear, as in the case of principal component analysis (PCA; Hotelling 1933) or nonlinear, as in the case of locally linear embedding (Roweis and Lawrence 2000), Isomap (Tenenbaum *et al.* 2000), and the tSNE (Van der Maaten and Hinton 2009). For more examples, see Jollife and Cadima (2016) and the references therein. In consequence, there is a plethora of R packages that implement these types of algorithms. To mention some examples, one can use the packages **RDRToolbox** (Bartenhagen 2022), **lle** (Kayo 2006), **Rtsne** (Krijthe 2022), and the classic `princomp()` function from the base package **stats** (R Core Team 2022).

Instead, geometric approaches rely on the topology of a dataset, exploiting the properties of the distances between data points. Within this family, we can find fractal methods (Falconer 2003), graphical methods (Costa and Hero 2004), model-based likelihood approaches (Levina and Bickel 2005), and methods based on nearest neighbors distances (Pettis, Bailey, Jain, and Dubes 1979). Also in this case, numerous packages are available: for example, for fractal methods alone there are **fractaldim** (Sevcikova, Percival, and Gneiting 2021), **nonlinearTseries** (Garcia 2022), and **tseriesChaos** (Di Narzo 2019), among others. For a recent review of the methodologies used for ID estimation we refer to Campadelli, Casiraghi, Ceruti, and Rozza (2015).

Given the abundance of approaches in this area, several R developers have also attempted to provide unifying collections of dimensionality reduction and ID estimation techniques. For example, remarkable ensembles of methodologies are implemented in the packages **ider** (Hino 2017), **dimred** and **coRanking** (Kraemer, Reichstein, and Mahecha 2018), **dyndimred** (Cannoodt and Saelens 2021), **IDmining** (Golay and Kanevski 2017), and **intrinsicDimension** (Johnsson 2019). Among the various options, the package **Rdimtools** (You 2022a) stands out, implementing 150 different algorithms, 17 of which are exclusively dedicated to ID estimation (You 2022b). Finally, it is worth mentioning that there are also Python (Van Rossum *et al.* 2011) packages implementing different methods for ID estimation: two prominent examples are **scikit-learn** (Bac *et al.* 2021) and **DADApy** (Glielmo *et al.* 2022). See Appendix B for more details.

In this paper, we introduce and discuss the R package **intRinsic** (version 1.0.0; Denti and Gilardi 2023). The package is openly available from the Comprehensive R Archive Network (CRAN) at https://CRAN.R-project.org/package=intRinsic, and can be installed by running

```
R> install.packages("intRinsic")
```

Future developments and updates will be uploaded both on CRAN and GitHub at https://github.com/Fradenti/intRinsic.

The package implements the two nearest neighbors (TWO-NN), the generalized ratio ID estimator (GRIDE), and the heterogeneous ID algorithm (HIDALGO) models, three state-of-the-art ID estimators recently introduced in Facco, D'Errico, Rodriguez, and Laio (2017); Denti, Doimo, Laio, and Mira (2022) and Allegra, Facco, Denti, Laio, and Mira (2020), respectively. These methods are likelihood-based estimators that rely on the theoretical properties of the distances among nearest neighbors. The first two models estimate a global, unique ID of a dataset and are implemented under both the frequentist and Bayesian paradigms. Moreover, one can exploit these models to study how the ID depends on the scale of the neighborhood considered for its estimation. On the contrary, HIDALGO is a Bayesian mixture model that allows for the estimation of clusters of points characterized by heterogeneous ID. In this article, we focus our attention on the exposition of TWO-NN and HIDALGO, and we discuss the pros and cons of both models with the aid of simulated data. More details about the additional routines, such as GRIDE, are reported in Appendix A. Finally, in Appendix B, we elaborate more on the strengths and weaknesses of the methods implemented in **intRinsic** in comparison to the other existing packages.

Broadly speaking, the package contains two sets of functions, organized into high-level and low-level routines. The former set contains user-friendly and straightforward R functions. Our goal is to make the package as accessible and intuitive as possible by automating most tasks. The low-level routines are not exported, as they represent the package's core. The most computationally-intensive low-level functions are written in C++, exploiting the interface with R provided by the packages **Rcpp** and **RcppArmadillo** (Eddelbuettel and François 2011; Eddelbuettel and Sanderson 2014). The C++ implementation considerably speeds up time-consuming tasks, like running the Gibbs sampler for the Bayesian mixture model HIDALGO. Moreover, **intRinsic** is well integrated with external R packages. For example, we enriched the package's functionalities defining ad-hoc methods for generic functions like `autoplot()` from the **ggplot2** package (Wickham 2016) to produce the detailed graphical outputs.

The article is structured as follows. Section 2 introduces and describes the theoretical background of the TWO-NN and HIDALGO methods. Section 3 illustrates the basic usage of the implemented routines on simulated data. We show how to obtain, manipulate, and interpret the different outputs. Additionally, we assess the robustness of the methods by monitoring how the results vary when the input parameters change. Section 4 presents an application to a famous real microarray dataset. Finally, Section 5 concludes by discussing future directions and potential extensions to the package.

2. The modeling background

Let $\boldsymbol{X}$ be a dataset with n data points measured over D variables. We denote each observation as $x_i \in \mathbb{R}^D$, with $i = 1, \dots, n$. Despite being observed over a D-dimensional space, we suppose that the points take place on a latent manifold $\mathcal{M}$ with intrinsic dimension $d \leq D$. Generally, we expect that $d \ll D$. Thus, we postulate that a low-dimensional data-generating mechanism can accurately describe the dataset.

Then, consider a single data point x_i. Starting from this point, one can order the remaining $n-1$ observations according to their distance from x_i. This way, we obtain a list of nearest neighbors (NNs) of increasing order. Formally, let $\Delta : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}^+$ be a generic distance function between data points. We denote with $x_i^{(l)}$ the l-th NN of x_i and with $r_{i,l} = \Delta\left(x_i, x_i^{(l)}\right)$ their distance, for $l = 1, \dots, n-1$. Given the sequence of NNs for each data point, we can define the *volume of the hyper-spherical shell enclosed between two successive neighbors of* x_i as

$$\nu_{i,l} = \omega_d \left(r_{i,l}^d - r_{i,l-1}^d\right), \qquad \text{for } l = 1, \dots, n-1, \text{ and } i = 1, \dots, n, \tag{1}$$

where d is the dimensionality of the latent manifold in which the points are embedded (the ID) and ω_d is the volume of the d-dimensional hyper-sphere with unitary radius. For this formula to hold, we need to set $x_{i,0} \equiv x_i$ and $r_{i,0} = 0$. We provide a visual representation of the introduced quantities in Figure 1 for $l = 1, 2$, which depicts the three-dimensional case.

From a modeling perspective, we assume that the dataset $\boldsymbol{X}$ is a realization of a Poisson point process characterized by density function $\rho(x)$. Facco *et al.* (2017) showed that the hyper-spherical shells defined in Equation 1 are the multivariate extension of the well-known *inter-arrival times* (Kingman 1992). Indeed, they proved that under the assumption of homogeneity of the Poisson point process, i.e., $\rho(x) = \rho \ \forall x$, all the $\nu_{i,l}$'s are independently drawn from an Exponential distribution with rate equal to the density ρ: $\nu_{i,l} \sim Exp(\rho)$, for $l = 1, \dots, n-1$, and $i = 1, \dots, n$. This fundamental result motivates the derivation of the estimators we will introduce in the following.

2.1. The TWO-NN estimator

Building on the distribution of the hyper-spherical shells, Facco *et al.* (2017) noticed that if the intensity of the underlying Poisson point process is assumed to be constant on the scale of the second NN, the following distributional result holds:

$$\mu_{i,1,2} = \frac{r_{i,2}}{r_{i,1}} \sim Pareto(1, d), \qquad \mu_{i,1,2} \in [1, +\infty) \qquad i = 1, \dots, n. \tag{2}$$

In other words, if the intensity of the Poisson point process that generates the data can be regarded as locally constant (on the scale of the second NN), the ratio of the first two

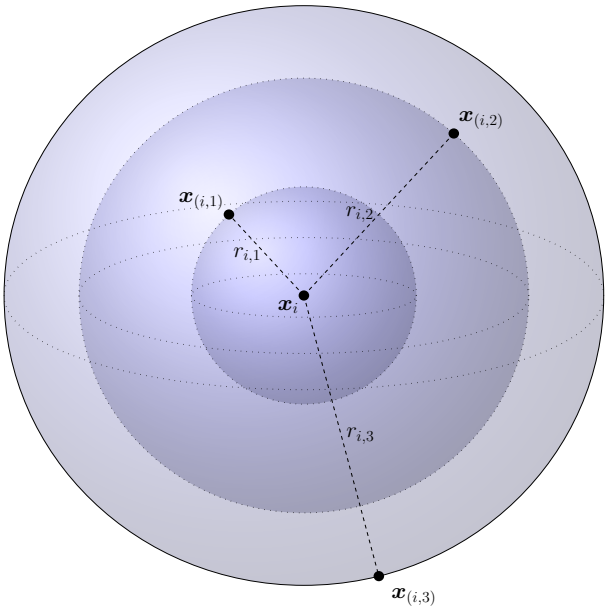

Figure 1: An illustration in three dimensions of the quantities involved in the TWO-NN modeling framework. The central dot represents the i-th data point. The selected observation is connected by dashed lines, representing the distances $r_{i,j}$, $j = 1, 2, 3$, to its first three NNs. The different spherical shells have volumes $v_{i,j}$, $j = 1, 2, 3$.

NN distances from each point is Pareto distributed. Recall that the Pareto random variable is characterized by a shape parameter a, scale parameter b, and density function $f_X(x) = ab^a x^{-a-1}$ defined over $x \in [b, +\infty)$. Remarkably, Equation 2 states that the ratio $r_{i,2}/r_{i,1}$ follows a Pareto distribution with scale $b = 1$ and shape $a = d$, i.e., the shape parameter can be interpreted as the ID of the dataset. Estimating d by exploiting the ratios of distances between each point and its first two NNs is the core of the TWO-NN procedure.

One can also attain more general results by considering ratios of distances with NNs of generic orders. A generalized ratio will be denoted with $\mu_{i,n_1,n_2} = r_{i,n_2}/r_{i,n_1}$ for $i = 1, \ldots, n$. Here, n_1 and n_2 are the NN orders, integer numbers that need to comply with the following constraint: $1 \leq n_1 < n_2 \leq n$. In this paper, we will mainly focus on methods involving the ratio of the first two NN distances. The generalized ratios will be only mentioned when discussing the function `compute_mus()`, for the sake of completeness. Therefore, to simplify the notation, we will write $\mu_i = \mu_{i,1,2}$ and $\boldsymbol{\mu} = (\mu_i)_{i=1}^n$.

Once the vector $\boldsymbol{\mu}$ is computed, we can employ different estimation techniques for the TWO-NN model. All of the following methods can be called via the **intRinsic** function `twonn()`. The reader can find examples of its usage in Section 3.3.

Linear estimator. Facco *et al.* (2017) proposed to estimate the ID via the linearization of the Pareto cumulative distribution function $F(\mu_i) = (1 - \mu_i^{-d})$. The ID estimate is obtained as the least squares solution of the following zero-intercept linear regression problem:

$$-\log(1 - \hat{F}(\mu_{(i)})) = d \log(\mu_{(i)}) + \varepsilon_i, \qquad i = 1, \ldots, n, \tag{3}$$

where the ε_i's are the error terms, $\hat{F}(\cdot)$ denotes the empirical cumulative distribution function

of the sample and the $\mu_{(i)}$'s are the ratios defined in Equation 2 sorted by increasing order. The authors also suggested trimming from $\boldsymbol{\mu}$ a percentage c_{TR} of the most extreme values to obtain a more robust estimation. This choice is justified because the extreme ratios often correspond to observations that do not comply with the local homogeneity assumption.

Maximum likelihood estimator. In a similar spirit, Denti *et al.* (2022) took advantage of the distributional results in Equation 2 to derive a simple maximum likelihood estimator (MLE) and the corresponding confidence interval (CI). Trivially, the (unbiased) MLE for the shape parameter of a Pareto distribution is given by:

$$\hat{d} = \frac{n-1}{\sum_{i=1}^{n} \log(\mu_i)}, \tag{4}$$

while the CI of level $(1-\alpha)$ is defined as

$$CI(d, 1-\alpha) = \left[\frac{\hat{d}}{q^{1-\alpha/2}_{IG_{n,(n-1)}}}, \frac{\hat{d}}{q^{\alpha/2}_{IG_{n,(n-1)}}} \right], \tag{5}$$

where $q^{\alpha/2}_{IG_{a,b}}$ denotes the quantile of order $\alpha/2$ of an Inverse-Gamma distribution of shape a and scale b.

Bayesian estimator. It is also straightforward to derive an estimator according to a Bayesian perspective (Denti *et al.* 2022), obtained with the specification of a prior distribution on the shape parameter d. The most natural prior to choose is a conjugate prior $d \sim Gamma(a, b)$. It is then immediate to derive the posterior distribution for the shape parameter:

$$d|\boldsymbol{\mu} \sim Gamma\left(a+n, b+\sum_{i=1}^{n} \log(\mu_i)\right). \tag{6}$$

With this method, one can quickly recover the principal quantiles of the posterior distribution and obtain point estimates and uncertainty quantification with a credible interval (CRI) of level α.

2.2. HIDALGO: The heterogeneous intrinsic dimension algorithm

The TWO-NN model implicitly assumes the existence of a single manifold. However, postulating a global, unique ID value for the entire dataset can often be limiting, especially when the data present complex dependence structures among variables. To extend the previous modeling framework, one can imagine that the data points are divided into clusters, each belonging to a latent manifold with its specific ID. Allegra *et al.* (2020) employed this heterogeneous ID estimation approach in their model: the heterogeneous ID algorithm (HIDALGO). The authors proposed as density function of the generating point process a mixture of K distributions defined on K different latent manifolds, expressed as $\rho(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \rho_k(\boldsymbol{x})$, where $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_K)$ is the vector of mixture weights. This assumption induces a mixture of Pareto distributions as the distribution of the ratios μ_i's:

$$f(\mu_i|\boldsymbol{d}, \boldsymbol{\pi}) = \sum_{k=1}^{K} \pi_k \, d_k \mu_i^{-(d_k+1)}, \qquad i = 1, \ldots, n, \tag{7}$$

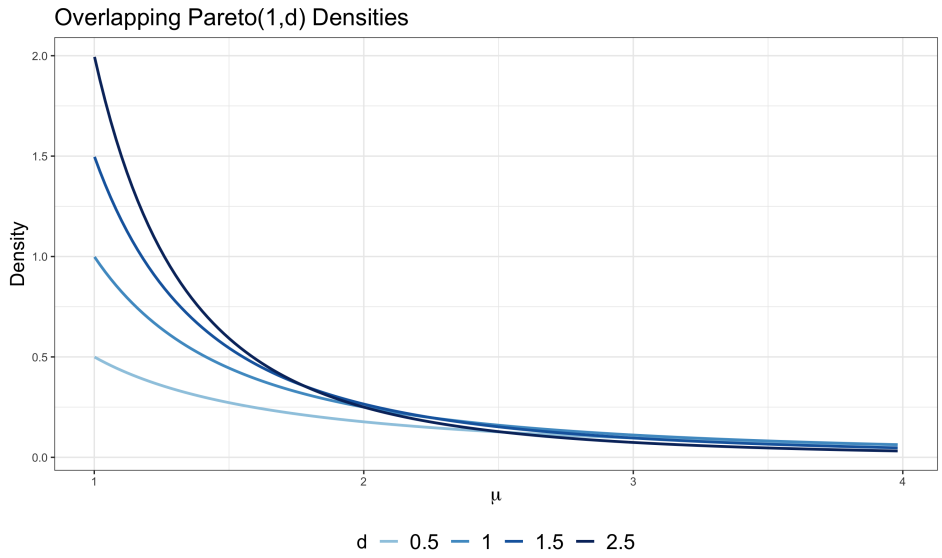

Figure 2: Density functions of Pareto distributions characterized by different shape parameters.

where $\boldsymbol{d} = (d_1, \ldots, d_K)$ is the vector of ID parameters. Allegra *et al.* (2020) adopted a Bayesian perspective, specifying independent Gamma priors for each element of $\boldsymbol{d}$: $d_k \sim Gamma(a_d, b_d)$, and a Dirichlet prior for the mixture weights $\boldsymbol{\pi} \sim Dirichlet(\alpha_1, \ldots, \alpha_K)$. Regarding the latter, we suggest setting $\alpha_1 = \ldots = \alpha_K = \alpha < 0.05$, fitting a sparse mixture model as indicated by Malsiner-Walli, Frühwirth-Schnatter, and Grün (2017). This prior specification encourages the data to populate only the necessary number of mixture components. Thus, we see K as an upper bound on the number of active clusters $K^* \leq K$.

Unfortunately, a model-based clustering approach like the one presented in Equation 7 is ineffective at modeling the distance ratios. The problem lies in the fact that the different Pareto kernels have extremely similar shapes. Therefore, Pareto densities with varying shape parameters can fit the same data points equally well, compromising the clustering allocation and the consequent ID estimation. Even when considering very diverse shape parameters, the right tails of the resulting Pareto distributions overlap to a great extent. This issue is evident in Figure 2, where we depict examples of various $Pareto(1, d)$ densities.

To address this problem, Allegra *et al.* (2020) introduced a local homogeneity assumption, assuming that neighboring points are more likely to be part of the same latent manifold. To incorporate this idea in the model, the authors added an extra penalizing term in the likelihood. We now summarize their approach.

First, they introduced the latent membership labels $\boldsymbol{z} = (z_1, \ldots, z_n)$ to assign each observation to a cluster, where $z_i = k$ means that the i-th observation belongs to the k-th mixture component. Then, they defined the binary adjacency matrix $\mathcal{N}^{(q)}$, whose entries are $\mathcal{N}^{(q)}_{i,j} = 1$ if the point x_j is among the first q NNs of x_i, and 0 otherwise. Finally, they defined N_{z_i} to be the number of observations with label equal to z_i and assumed the following probabilities: $\mathbb{P}\left[\mathcal{N}^{(q)}_{i,j} = 1 | z_i = z_j\right] = \zeta_0$, with $\zeta_0 > 0.5$ and $\mathbb{P}\left[\mathcal{N}^{(q)}_{i,j} = 1 | z_i \neq z_j\right] = \zeta_1$, with $\zeta_1 < 0.5$. These probabilities are employed to define a distribution over the neighboring structure of each data point x_i: $\pi(\mathcal{N}^{(q)}_i | \boldsymbol{z}) = \zeta_0^{n_i^{(in)}(\boldsymbol{z})} \zeta_1^{q - n_i^{(in)}(\boldsymbol{z})} / \mathcal{Z}(\zeta, N_{z_i})$, where $n_i^{(in)}(\boldsymbol{z}) = \sum_{j=1}^{n} \mathcal{N}^{(q)}_{i,j} \mathbb{1}_{\{z_i = z_j\}}$,

$q{-}n_i^{(in)}(\boldsymbol{z}) = \sum_{j=1}^{n} \mathcal{N}_{i,j}^{(q)} \mathbb{1}_{\{z_i \neq z_j\}}$, $\mathcal{Z}(\zeta, N_{z_i})$ is the normalizing constant, and $\mathbb{1}_A$ is the indicator function, equal to 1 when the event A is true, and equal to 0 otherwise. A more technical discussion of this model extension and the validity of the underlying hypotheses can be found in the Supplementary Material of Allegra *et al.* (2020). For simplicity, we assume $\zeta_0 = \zeta$ and $\zeta_1 = 1 - \zeta$. The new likelihood becomes

$$\mathcal{L}\left(\boldsymbol{\mu}, \mathcal{N}^{(q)} | \boldsymbol{d}, \boldsymbol{z}, \zeta\right) = \prod_{i=1}^{n} d_{z_i} \mu_i^{-(d_{z_i}+1)} \times \frac{\zeta^{n_i^{(in)}(\boldsymbol{z})} (1-\zeta)^{q-n_i^{(in)}(\boldsymbol{z})}}{\mathcal{Z}(\zeta, N_{z_i})}, \qquad z_i | \boldsymbol{\pi} \sim Cat_K(\boldsymbol{\pi}), \quad (8)$$

where Cat_K denotes a Categorical distribution over the set $\{1, \ldots, K\}$. A closed-form for the posterior distribution is not available, so we rely on Markov chain Monte Carlo (MCMC) techniques to simulate a posterior sample.

In our package, this Bayesian mixture model is implemented by the function `Hidalgo()`. For the ID parameters, we use conjugate Gamma prior specifications or variations thereof to account for modeling inconsistencies. For example, when the nominal dimension D is low, the unbounded support of a Gamma prior may provide unrealistic results, where the posterior distribution assigns positive density to the interval $(D, +\infty)$. Santos-Fernandez, Denti, Mengersen, and Mira (2022) proposed to employ a more informative prior for $\boldsymbol{d}$:

$$\pi(d_k) = \hat{\rho} \cdot d_k^{a-1} \exp^{-bd_k} \frac{\mathbb{1}_{(0,D)}}{\mathcal{C}_{a,b,D}} + (1 - \hat{\rho}) \cdot \delta_D(d_k) \quad \forall k, \quad (9)$$

where they denoted the normalizing constant of a $Gamma(a, b)$ truncated over $(0, D]$ with $\mathcal{C}_{a,b,D}$. That is, the prior distribution for d_k is a mixture between a truncated Gamma distribution over $(0, D]$ and a point mass located at D. The parameter ρ denotes the mixing proportion. When $\rho = 1$, the distribution in Equation 9 reduces to a simple truncated Gamma. Both approaches are implemented in **intRinsic**, but we recommend using the latter. We report the details of the implemented Gibbs sampler in Appendix C, while in Appendix F we comment more on the concept of *heterogeneous* ID estimation related to global and local ID definitions.

3. Examples using intRinsic

This section illustrates the main routines of the **intRinsic** package. Here, we indicate the number of observations and the observed nominal dimension with `n` and `D`, respectively. Also, $\mathcal{N}_k(m, \Sigma)$ represents a multivariate normal distribution of dimension k, mean m, and covariance matrix Σ. Moreover, let $\mathcal{U}_{(a,b)}^{(k)}$ represent a multivariate Uniform distribution with support (a, b) in k dimensions. Finally, we denote with $\mathbb{I}_k$ an identity matrix of dimension k.

To start, we load our package by running

```
R> library("intRinsic")
```

3.1. Simulated datasets

To illustrate how to apply the different ID estimation techniques available in the package, we will use three simulated datasets: Swissroll, HyperCube, and GaussMix. This way, we can

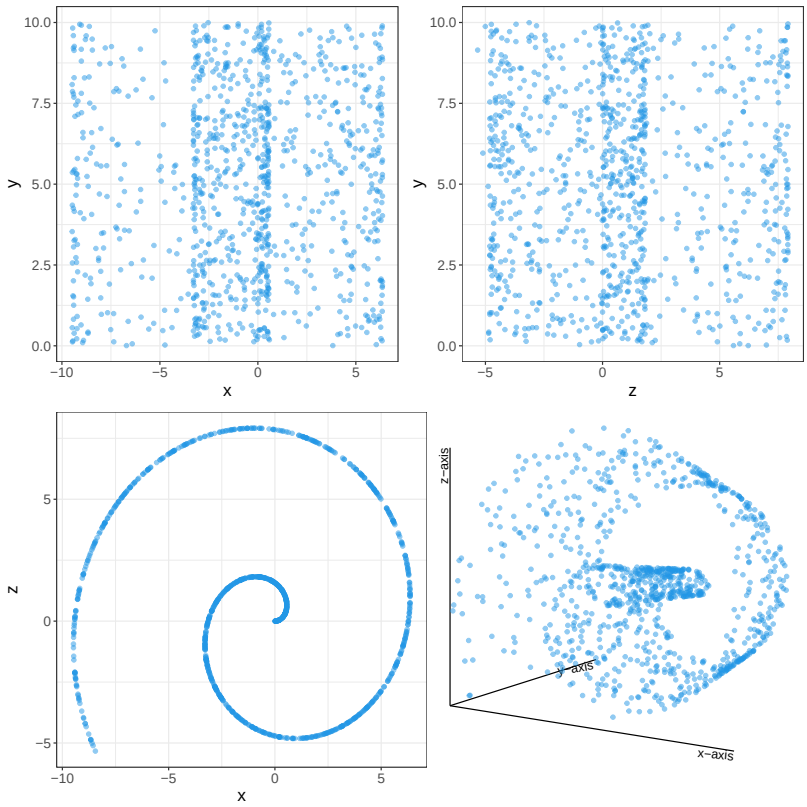

Figure 3: Scatterplots of the three variables in the Swissroll dataset. The dependence among the coordinates x and z is evident.

compare the results of the experiments with the ground truth. One can generate the exact replica of the three simulated datasets used in this paper by running the code reported below. The first dataset, Swissroll, is obtained via the classical Swissroll transformation $\mathcal{S} : \mathbb{R}^2 \to \mathbb{R}^3$ defined as $\mathcal{S}(x, y) = (x\cos(x), y, x\sin(x))$, where each pair of points (x, y) is sampled from two independent Uniform distributions on $(0, 10)$. To simulate such a dataset, we can use the **intRinsic** function `Swissroll()`, specifying the number of observations `n` as the input parameter.

```
R> set.seed(123456)
R> Swissroll <- Swissroll(n = 1000)
```

The second dataset, HyperCube, contains a cloud of 500 points sampled from $\mathcal{U}^{(5)}_{(0,1)}$ embedded in an eight-dimensional space $\mathbb{R}^8$. To fill the gap between the nominal dimension (8) and the ID (5), we add three columns of zeros.

```
R> HyperCube  <- cbind(replicate(5, runif(500)), 0, 0, 0)
```

Lastly, the dataset GaussMix contains 1500 data points generated from three random variables all contained in a five-dimensional Euclidean space. More precisely, we consider a bivariate random variable defined as $X_1 = (X_0, 3X_0)$ where $X_0 \sim \mathcal{N}_1(-5, 1)$, and the three- and five-dimensional variables

$$X_2 \sim \mathcal{N}_3(0, \mathbb{I}_3), \quad X_3 \sim \mathcal{N}_5(5, \mathbb{I}_5).$$

Name	n	D	ID
Swissroll	1000	3	2
HyperCube	500	8	5
GaussMix	1500	5	?

Table 1: Summary of the characteristics of the three simulated datasets.

We embed each of them in a five-dimensional space by adding the corresponding number of columns of zeros, as we did for the HyperCube dataset.

```
R> x0 <- rnorm(500, mean = -5, sd = 1)
R> x1 <- cbind(x0, 3 * x0, 0, 0, 0)
R> x2 <- cbind(replicate(3, rnorm(500)), 0, 0)
R> x3 <- replicate(5, rnorm(500, 5))
R> GaussMix <- rbind(x1, x2, x3)
R> class_GMix <- rep(c("A", "B", "C"), rep(500, 3))
```

Next, we need to establish the true values of the ID for the different datasets. Scatterplots are useful exploratory tools to spot any clear dependence across the columns of a dataset. For example, if we plot all the possible two-dimensional scatterplots from the Swissroll dataset, we obtain Figure 3. The different panels show that two of the three coordinates are free, and we can recover the last coordinate as a function of the others. Therefore, the ID of Swissroll is equal to 2. Moreover, from the description of the data simulation, it is also evident that the ID is equal to 5 for the HyperCube dataset. However, it is not as simple to figure out the value of the true ID for GaussMix, given the heterogeneity of its data-generating mechanism. Table 1 summarizes the sample sizes along with the true nominal dimensions `D` and IDs that characterize the three datasets.

3.2. Ratios of nearest neighbors distances

The ratios of NN distances constitute the core quantities on which the theoretical development presented in Section 2 is based. We can compute the ratios μ_i defined in Equations 2 with the function `compute_mus()`. All in all, the function can also compute the generalized ratios $\boldsymbol{\mu}_{n_1,n_2}$, where `n1 < n2`, as presented in Section 2.1. In fact, the function needs the following arguments:

- `X`: a dataset of dimension `n`×`D` of which we want to compute the distance ratios;
- `dist_mat`: a `n`×`n` symmetric matrix containing the distances between data points which one can pass instead of `X`;
- `n1` and `n2`: the orders of the closest and furthest nearest neighbors to consider, respectively. As default, `n1 = 1` and `n2 = 2`.

The function has two additional arguments, `Nq` and `q`, that we will introduce later in Section 3.4 when we will illustrate the HIDALGO model.

Note that the specification of `dist_mat` overrides the argument passed as `X`. Instead, if the distance matrix `dist_mat` is not specified, `generate_mus()` relies on the function `get.knn()`

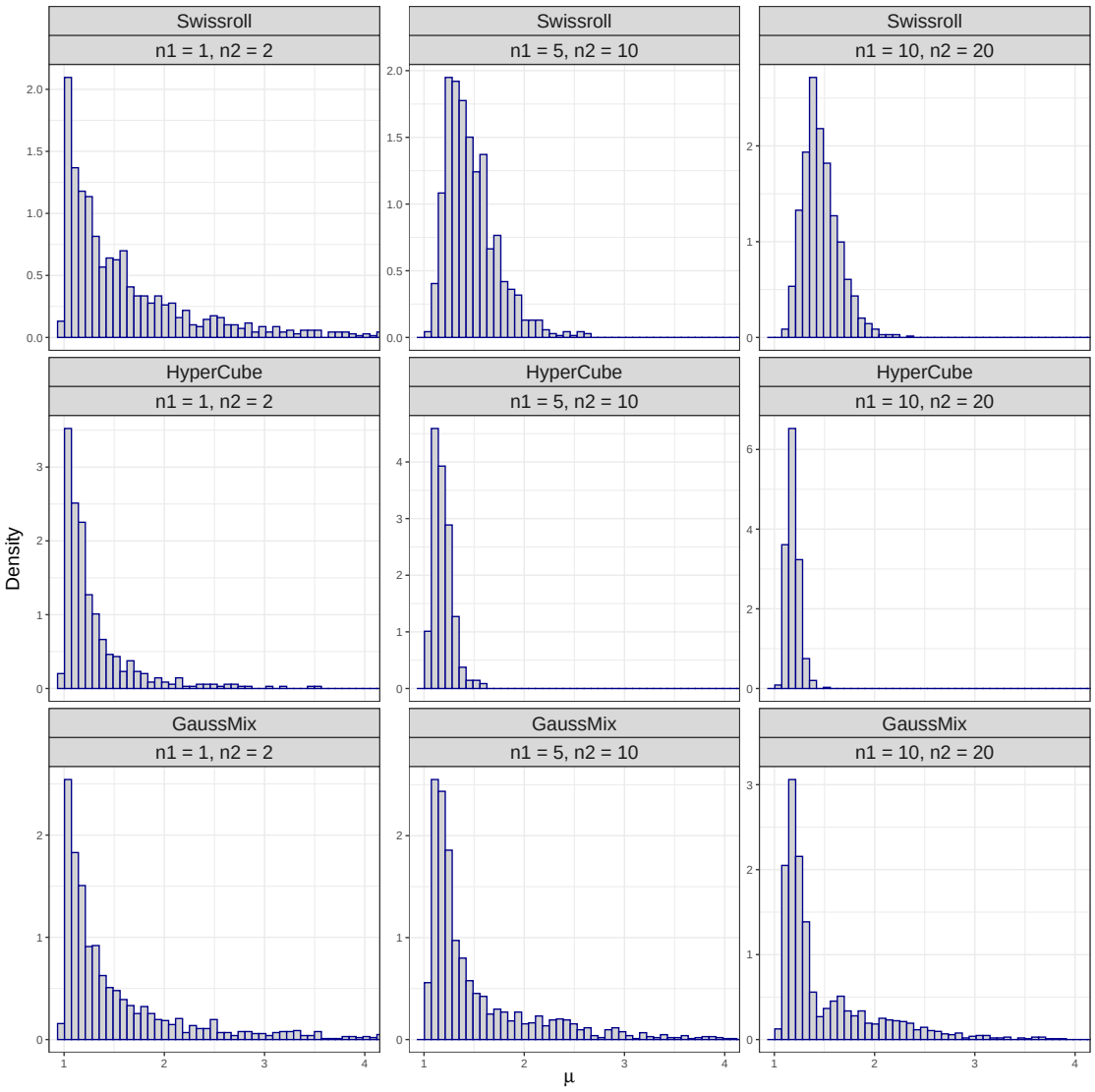

Figure 4: Histograms of the ratios $\boldsymbol{\mu}_{n_1,n_2}$ for the Swissroll, HyperCube, and GaussMix datasets. The shape of the histograms in the first column suggests that a Pareto distribution could be a good fit, according to the TWO-NN model.

from the package **FNN** (Beygelzimer, Kakadet, Langford, Arya, Mount, and Li 2022), which implements fast NN-search algorithms on the original dataset.

The main output of the function is the vector of ratios $\boldsymbol{\mu}_{n_1,n_2}$, an object of class '`mus`' for which appropriate `print()` and `plot()` methods are defined. To use the function, we can easily write

```
R> mus_Swissroll <- compute_mus(X = Swissroll)
R> mus_Hyperube <- compute_mus(X = HyperCube)
R> mus_GaussMix <- compute_mus(X = GaussMix)
```

Calling the function with default arguments produces $\boldsymbol{\mu} = \boldsymbol{\mu}_{1,2}$. To explicitly compute gen-

n1	n2	Minimum	1st quartile	Median	Mean	3rd quartile	Maximum
1	2	1.0002	1.1032	1.3048	18.4239	1.9228	5874.3666
5	10	1.0234	1.1627	1.2858	1.5344	1.7043	6.9533
10	20	1.0472	1.1685	1.2613	1.4987	1.7250	4.0069

Table 2: Summary statistics of the generalized ratios obtained from the GaussMix dataset. Each row corresponds to a different combination of NN orders.

eralized ratios, we need to specify the NN orders `n1` and `n2`. Here, we report two different examples

```
R> mus_Swissroll_1 <- compute_mus(X = Swissroll, n1 = 5, n2 = 10)
R> mus_HyperCube_1 <- compute_mus(X = HyperCube, n1 = 5, n2 = 10)
R> mus_GaussMix_1 <- compute_mus(X = GaussMix, n1 = 5, n2 = 10)
```

and

```
R> mus_Swissroll_2 <- compute_mus(X = Swissroll, n1 = 10, n2 = 20)
R> mus_HyperCube_2 <- compute_mus(X = HyperCube, n1 = 10, n2 = 20)
R> mus_GaussMix_2 <- compute_mus(X = GaussMix, n1 = 10, n2 = 20)
R> mus_GaussMix_2

Ratio statistics mu's:
NN orders: n1 = 10, n2 = 20
Sample size: 1500
Nominal Dimension: 5
```

The histograms of the computed ratios are presented in Figure 4. The panels in each row correspond to different datasets, while the varying NN orders are reported by column. The horizontal axes are truncated over the interval $[0, 4]$ to improve the visualization. The histograms present the right-skewed shape typical of the Pareto distribution. However, some ratios could assume extreme values, especially when low values of NN orders are chosen. To provide an example, in Table 2 we display the summary statistics of the three vectors of ratios computed on GaussMix. The maximum in the first line has a high magnitude, but it significantly reduces when higher NN orders are considered. It is also interesting to observe how the distribution for the ratios of GaussMix when `n1` $= 10$ and `n2` $= 20$ (bottom-right panel) is multimodal, a symptom of the presence of heterogeneous manifolds. We remark again that we will focus on the TWO-NN and HIDALGO models for the rest of the paper. Therefore, we will only use `compute_mus()` in its default specification, simply computing $\boldsymbol{\mu} = (\mu_i)_{i=1}^{n}$. The ratios of NNs of generic order are necessary when using the GRIDE model. See Appendix A and Denti *et al.* (2022) for more details.

Finally, recall that the model is based on the assumption that a Poisson point process is the generating mechanism of the dataset. Ergo, the model cannot handle ties among data points. From a more practical point of view, if $\exists i \neq j$ such that $x_i = x_j$, the computation of μ_i would be unfeasible since $r_{i,1} = 0$. We devised the function `compute_mus()` to automatically detect if duplicates are present in a dataset. In that case, the function removes the duplicates and provides a warning. We showcase this behavior with a simple example:

```
R> Dummy_Data_with_replicates <- rbind(c(1, 2, 3), c(1, 2, 3),
+    c(1, 4, 3), c(1, 4, 3), c(1, 4, 5))
R> mus_shorter <- compute_mus(X = Dummy_Data_with_replicates)

Warning:
  Duplicates are present and will be removed.

  Original sample size: 5. New sample size: 3.
```

The function `compute_mus()` is at the core of many other high-level routines we use to estimate the ID. The following subsection shows how to implement the TWO-NN model to obtain a point estimate of a global, homogeneous ID accompanied by the corresponding CIs or CRIs.

3.3. Estimating a global ID value with TWO-NN

We showcase how to carry out inference on the ID with the TWO-NN model using linear, MLE, and Bayesian estimation methods. The low-level functions that implement these methods are `twonn_mle()`, `twonn_linfit()`, and `twonn_bayes()`, respectively. One can call these low-level functions via the high-level function `twonn()`. Regardless of the preferred estimation method, the `twonn()` function takes the following arguments: the dataset `X` or the distance matrix `dist_mat` (refer to previous input descriptions for more details), along with

- `mus`: the vector of second-to-first NN distance ratios. If this argument is provided, `X` and `dist_mat` will be ignored;
- `method`: a string stating the preferred estimation method. Could be `"mle"` (the default), `"linfit"`, or `"bayes"`;
- `alpha`: the confidence level (for `"mle"` and `"linfit"`) or the posterior probability included in the CRI (`"bayes"`);
- `c_trimmed`: the proportions of most extreme ratios to exclude from the analysis.

The object that the function returns is a list, characterized by a class that varies according to the selected estimation method. Tailored R methods have been devised to extend the generic functions `print()`, `summary()`, `plot()`, and `autoplot()` to interact with these new classes. The first element of the returned list always contains the estimates, while the others provide additional information about the chosen estimation process.

Linear estimator. We apply the linear estimator to the Swissroll dataset. As an example, we fit five linear models by setting `method = "linfit"`, adopting different trimming proportions. The function `summary()` provides an informative recap of the estimation process. We show the results for `lin_2` as an example.

```
R> lin_1 <- twonn(X = Swissroll, method = "linfit", c_trimmed = 0)
R> lin_2 <- twonn(X = Swissroll, method = "linfit", c_trimmed = 0.001)
R> lin_3 <- twonn(X = Swissroll, method = "linfit", c_trimmed = 0.01)
```

Trimming percentage	0%	0.1%	1%	5%	10%
Lower bound	1.9524	1.9867	2.2333	2.5709	2.9542
Estimate	1.9669	1.9997	2.2457	2.5988	2.9974
Upper bound	1.9814	2.0127	2.2581	2.6267	3.0407

Table 3: Point estimates and relative CIs for the ID values retrieved from the Swissroll dataset with the linear estimator. Each column displays the results for a specific trimming level.

```
R> lin_4 <- twonn(X = Swissroll, method = "linfit", c_trimmed = 0.05)
R> lin_5 <- twonn(X = Swissroll, method = "linfit", c_trimmed = 0.1)
R> summary(lin_2)

Model: TWO-NN
Method: Least Squares Estimation
Sample size: 1000, Obs. used: 999. Trimming proportion: 0.1%
ID estimates (confidence level: 0.95)

| Lower Bound| Estimate| Upper Bound|
|-----------:|--------:|-----------:|
|    1.986659| 1.999682|    2.012705|
```

The results of these experiments are collected in Table 3. This first example allows us to comment on the trimming level to choose. Trimming the most extreme observations may be fundamental since outliers may distort the estimate. However, too much trimming would remove important information regarding the tail of the Pareto distribution, which is essential for the correct estimation of the ID. The estimates improve for very low levels of trimming but start to degenerate as more than 5% of observations are removed from the dataset.

We can also visually assess the goodness of fit via a dedicated `autoplot()` function, which displays the data and the estimated regression line. The slope of the regression lines corresponds to the linear fit ID estimates. For example, we obtain the plots in Figure 5 with the following two lines of code:

```
R> autoplot(lin_1, title = "No trimming")
R> autoplot(lin_5, title = "10% trimming")
```

MLE. A second way to obtain an ID estimate, along with its CI, is via MLE. The formulas implemented are presented in Equations 4 and 5. We compute the MLE by calling the low-level `twonn_mle()` function setting `method = "mle"`. In addition to the previous arguments, one can also specify

- `unbiased`: logical, if `TRUE` the point estimate according to the unbiased estimator (where the numerator is $n - 1$, as in Equation 4) is computed.

We compute the ID on HyperCube via MLE using different distance definitions: Euclidean, Manhattan, and Canberra. These distances are calculated with the `dist()` function of the **stats** package.

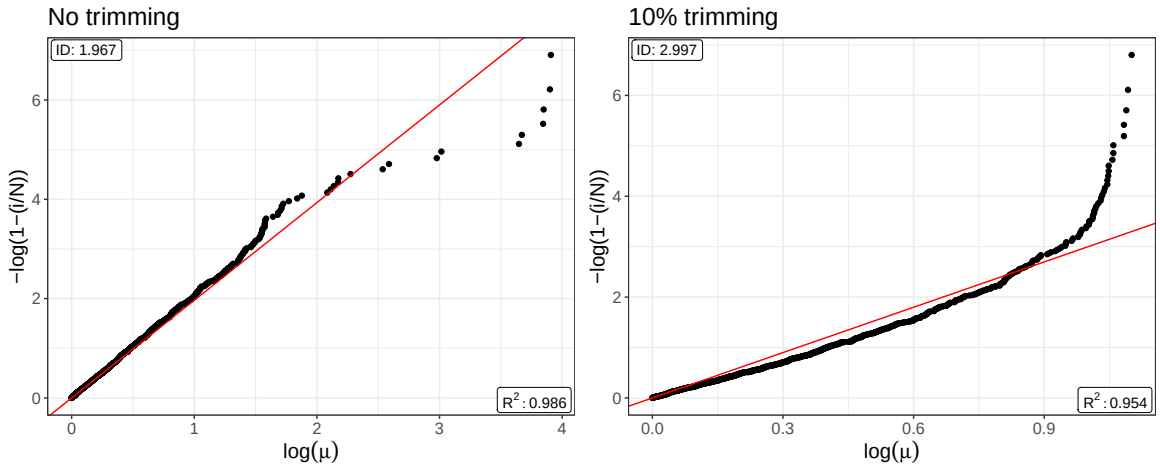

Figure 5: Swissroll dataset. Regression lines estimated from $-\log(1-\hat{F}(\mu_i)) = d\log(\mu_i)$ with no trimming (left panel) and 10% of trimmed observations (right panel).

```
R> dist_Eucl_D2 <- dist(HyperCube)
R> dist_Manh_D2 <- dist(HyperCube, method = "manhattan")
R> dist_Canb_D2 <- dist(HyperCube, method = "canberra")
```

Other distance matrices can be employed as well. In this example, we also show how the widths of the CIs change by varying the confidence levels. We print the results stored in the object `mle_12` as an example. We write:

```
R> mle_11 <- twonn(dist_mat = dist_Eucl_D2)
R> mle_12 <- twonn(dist_mat = dist_Eucl_D2, alpha = 0.99)
R> mle_21 <- twonn(dist_mat = dist_Manh_D2)
R> mle_22 <- twonn(dist_mat = dist_Manh_D2, alpha = 0.99)
R> mle_31 <- twonn(dist_mat = dist_Canb_D2)
R> mle_32 <- twonn(dist_mat = dist_Canb_D2, alpha = 0.99)
R> summary(mle_12)

Model: TWO-NN
Method: MLE
Sample size: 500, Obs. used: 495. Trimming proportion: 1%
ID estimates (confidence level: 0.99)

| Lower Bound| Estimate| Upper Bound|
|-----------:|--------:|-----------:|
|    3.968082| 4.459427|     5.00273|
```

The results are reported in Table 4. The type of distance can lead to differences in the results. Overall, the estimators agree with each other, obtaining values close to the ground truth.

Bayesian estimator. The third option for ID estimation is to adopt a Bayesian perspective and specify a prior distribution for the parameter d. To obtain the Bayesian estimates, we call

`dist()`	`Euclidean`		`Manhattan`		`Canberra`	
α	0.95	0.99	0.95	0.99	0.95	0.99
Lower bound	4.0834	3.9681	4.0736	3.9585	4.6001	4.4701
Estimate	4.4594	4.4594	4.4487	4.4487	5.0237	5.0237
Upper bound	4.8706	5.0027	4.8588	4.9906	5.4868	5.6357

Table 4: MLEs obtained with the TWO-NN model applied to the HyperCube dataset. Different distance functions and confidence level specifications are adopted.

the low-level function `twonn_bayes()` setting `method = "bayes"` in `twonn()`. Along with the arguments mentioned above, we can also specify the following:

- `a_d` and `b_d`: shape and rate parameters of the Gamma prior distribution on d. A vague specification is adopted as default with `a_d = 0.001` and `b_d = 0.001`. This implies $\mathbb{E}(d) = 1$, and $\mathbb{V}ar(d) = 1000$.

Differently from the previous two cases, `alpha` is now assumed to be the probability contained in the CRI computed from the posterior distribution. Along with the CRI, the function outputs the posterior mean, median, and mode. In the following, four examples showcase the usage of this function on the Swissroll dataset with different combinations of `alpha` and Gamma hyperparameters. The results are summarized in Table 5.

```
R> bay_1 <- twonn(X = Swissroll, method = "bayes")
R> bay_2 <- twonn(X = Swissroll, method = "bayes", alpha = 0.99)
R> bay_3 <- twonn(X = Swissroll, method = "bayes", a_d = 1, b_d = 1)
R> bay_4 <- twonn(X = Swissroll, method = "bayes", a_d = 1, b_d = 1,
+    alpha = 0.99)
```

We can plot the posterior density of the parameter d using `autoplot()`, as displayed in Figure 6. When plotting an object of class '`twonn_bayes`', we can also specify the following parameters:

- `plot_low` and `plot_upp`: lower and upper extremes of the support on which the posterior is evaluated;
- `by`: increment of the sequence going from `plot_low` to `plot_upp` that defines the support.

As an example, we compare the prior specification used for the object `bay_4` ($d \sim Gamma(1, 1)$) with a more informative one ($d \sim Gamma(10, 10)$) by writing:

```
R> bay_5 <- twonn(X = Swissroll, method = "bayes", a_d = 10, b_d = 10,
+    alpha = 0.99)
R> summary(bay_5)

Model: TWO-NN
Method: Bayesian Estimation
Sample size: 1000, Obs. used: 990. Trimming proportion: 1%
```

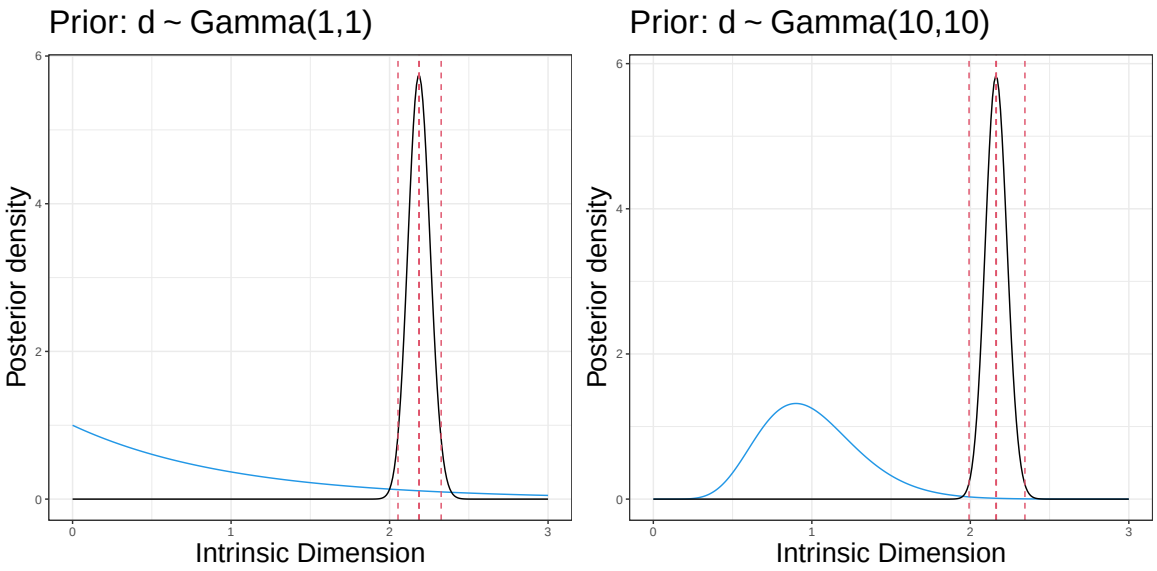

Figure 6: Swissroll dataset. Graphical representation of the posterior distribution (black line), prior distribution (blue line), and main quantiles and average (vertical dotted red lines) under $d \sim Gamma(1, 1)$ (left panel) and $d \sim Gamma(10, 10)$ (right panel) prior specifications.

Prior	Default		$d \sim Gamma(1, 1)$	
α	0.95	0.99	0.95	0.99
Lower bound	2.0556	2.0147	2.0532	2.0124
Mean	2.1899	2.1899	2.1872	2.1872
Median	2.1891	2.1891	2.1865	2.1865
Mode	2.1876	2.1876	2.1850	2.1850
Upper bound	2.3284	2.3733	2.3255	2.3703

Table 5: Swissroll dataset. Posterior estimates under the Bayesian specification according to different prior specifications and CRI levels α.

```
Prior d ~ Gamma(10, 10)
Credible Interval quantiles: 0.5%, 99.5%
Posterior ID estimates:

| Lower Bound|     Mean|   Median|     Mode| Upper Bound|
|-----------:|--------:|--------:|--------:|-----------:|
|    1.991901| 2.164113| 2.163391| 2.161949|    2.344453|
```

The posterior distribution is depicted in black, the prior in blue, and the dashed vertical red lines represent the estimates.

So far, we have discussed methods to determine a global ID estimate accurately and efficiently. Knowing the simulated data-generating processes, we could easily compare the obtained estimates with the ground truth for the Swissroll and HyperCube datasets. However, the same task is not immediate when dealing with GaussMix. For GaussMix, relying only on a global

ID estimate may constitute an oversimplification since the data points are generated from Gaussian distributions defined over different manifolds of heterogeneous dimensions. This scenario is more likely to occur with datasets describing real phenomena, often characterized by complex dependencies, and it will be the focus of the next section.

3.4. Detecting manifolds with heterogeneous ID using HIDALGO

Detecting the presence of multiple manifolds

In contexts where data may exhibit heterogeneous ID, we face two main challenges: (i) detect the actual presence of multiple manifolds in the data, and (ii) accurately estimate their IDs. To tackle these problems, we start by applying the `twonn()` function to GaussMix with `method` equal to `"linfit"` and `"mle"`.

```
R> mus_gm <- compute_mus(GaussMix)
R> summary(twonn(mus = mus_gm, method = "linfit"))

Model: TWO-NN
Method: Least Squares Estimation
Sample size: 1500, Obs. used: 1485. Trimming proportion: 1%
ID estimates (confidence level: 0.95)

| Lower Bound| Estimate| Upper Bound|
|-----------:|--------:|-----------:|
|    1.438657| 1.456325|    1.473992|

R> summary(twonn(mus = mus_gm, method = "mle"))

Model: TWO-NN
Method: MLE
Sample size: 1500, Obs. used: 1485. Trimming proportion: 1%
ID estimates (confidence level: 0.95)

| Lower Bound| Estimate| Upper Bound|
|-----------:|--------:|-----------:|
|    1.739329| 1.830063|    1.925601|
```

The estimates obtained with the different methods do not agree. Figure 7 raises concerns about the appropriateness of a model postulating the existence of a single, global manifold. In the top panel of Figure 7, the data points are colored according to their generating mixture component. The sorted log-ratios present a non-linear pattern, where the slope values vary across the different mixture components `A`, `B`, and `C`.

We can get another empirical assessment by inspecting the evolution of the cumulative average distances between a point and its nearest neighbors. More formally, for each point x_i we consider the evolution of the ergodic mean of the sorted sequence of distances $(r_{i,1}, \ldots, r_{i,n})$, given by $r_i(j) = \sum_{k=1}^{j} r_{i,k}/k$, $\forall i, j$. Figure 8 compares the HyperCube (left panel – exemplifying the homogeneous case) and GaussMix (right panel – representing the heterogeneous

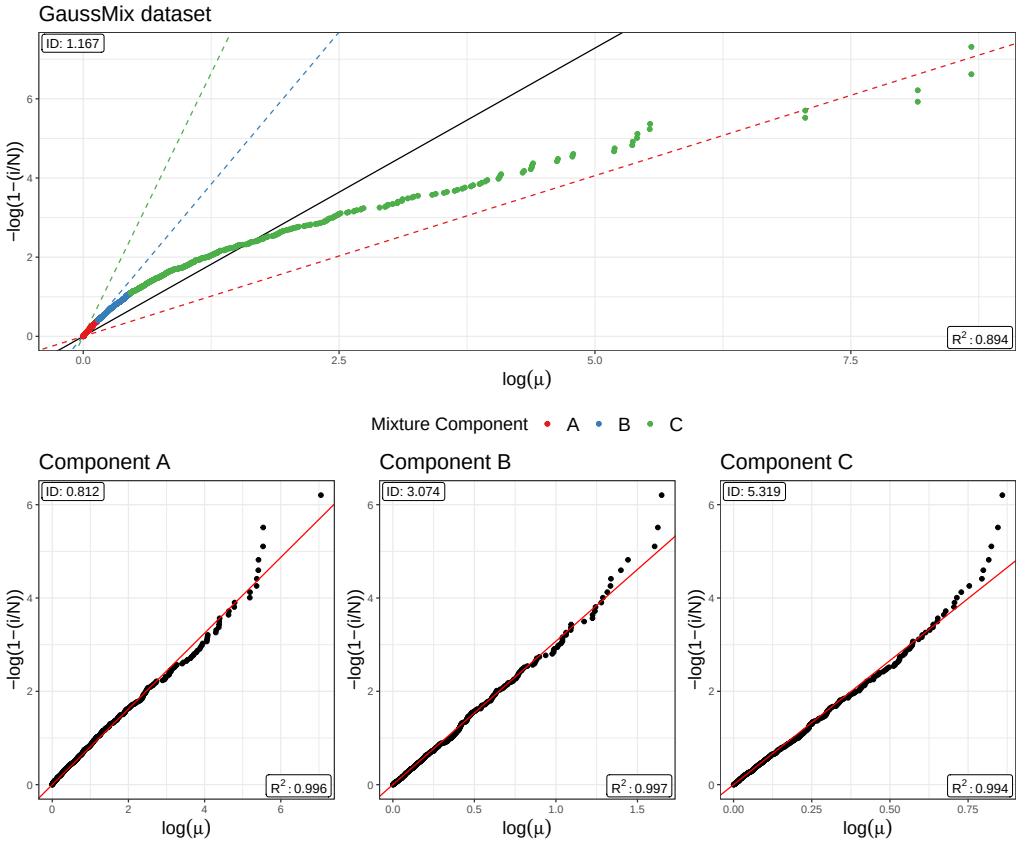

Figure 7: Linear estimators applied to the GaussMix dataset. In the top panel, the estimator is applied to the entire dataset. The points are colored according to the mixture component from which they originate, and the corresponding colored dashed lines result from applying the estimator to each known subset of points. The bottom three panels explicitly report the estimates within each mixture component.

case) datasets. On the one hand, the left plot displays the ideal condition: there are no visible departures from the overall mean distance, which reassures us about the homogeneity assumption we made about the data. But, on the other hand, the right panel tells a different story. We immediately detect the different clusters in the data by focusing on their different starting values. The ergodic means remain approximately constant until the 500th NN, which corresponds to the size of each subgroup. The behavior of the cumulative means abruptly changes after the 500th NN, negating the presence of a unique manifold. This type of plot provides an empirical but valuable overview of the structure between data points, highlighting clusters that may reflect the existence of multiple manifolds.

These visual assessments help detect signs of the inappropriateness of the global ID assumption. The most direct approach to adopt in this case would be to divide the dataset into homogeneous subgroups and apply the TWO-NN estimator within each cluster. Such an approach is highlighted in the bottom panels of Figure 7. However, knowing ex-ante such well-separated groups is not a realistic expectation to have about actual data. Therefore, we will rely on `Hidalgo()`, the Bayesian finite mixture model for heterogeneous ID estimation described in Section 2.2.

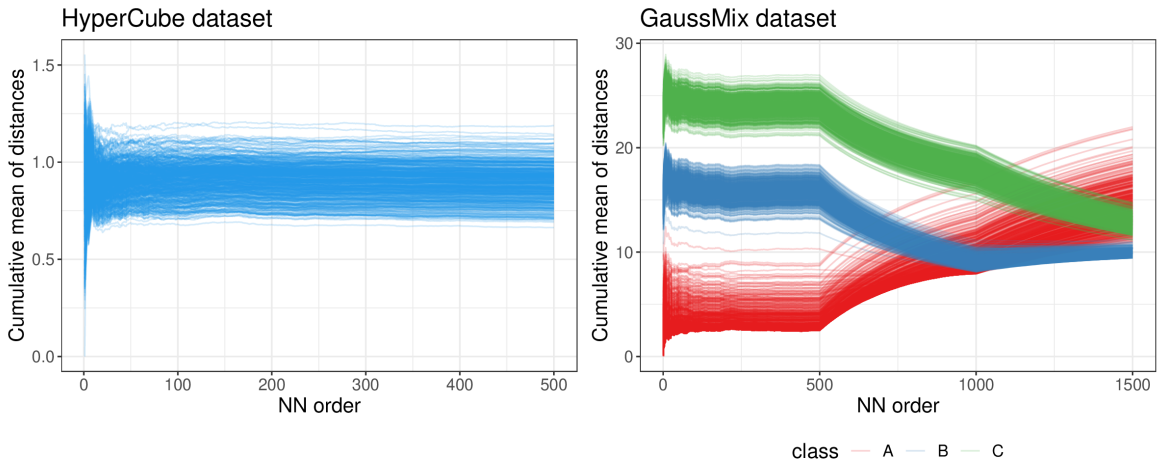

Figure 8: Evolution of the cumulative means of NN distances computed for all the observations in the HyperCube (left panel) and GaussMix (right panel) datasets. In the right panel, the colors highlight the different mixture components.

Fitting the HIDALGO model

HIDALGO allows for the presence of multiple manifolds in the same dataset, yielding a vector of different estimated ID values. As already discussed, estimating a mixture model with Pareto components is challenging because of their extensive overlap. A naive model-based estimation can lead to inaccurate results since there is no clear separation between the kernel densities. The extra term $\prod_{i=1}^{n} \pi(\mathcal{N}_i^{(q)}|\boldsymbol{z})$ added into the likelihood in Equation 8 induces local homogeneity, which helps identify the model parameters.

The adjacency matrix $\mathcal{N}^{(q)}$ can be easily computed by specifying two additional arguments in the function `compute_mus()`:

- `Nq`: logical, if `TRUE`, the function adds the adjacency matrix to the output;
- `q`: integer, the number of NNs to be considered in the construction of the matrix $\mathcal{N}^{(q)}$. The default value is 3.

To provide an idea of the structure of the adjacency matrix $\mathcal{N}^{(q)}$, we report three examples obtained from a random sub-sample of the GaussMix dataset for increasing values of `q`. We display the heatmaps of the resulting matrices in Figure 9.

```
R> set.seed(12345)
R> ind <- sort(sample(1:1500, 100, FALSE))
R> Nq1 <- compute_mus(GaussMix[ind, ], Nq = TRUE, q = 1)$NQ
R> Nq2 <- compute_mus(GaussMix[ind, ], Nq = TRUE, q = 5)$NQ
R> Nq3 <- compute_mus(GaussMix[ind, ], Nq = TRUE, q = 10)$NQ
```

As `q` increases, the binary matrix becomes more populated, uncovering the neighboring structure of the data points. Allegra *et al.* (2020) investigated how the performance of the model changes as `q` varies. They suggest fixing $\texttt{q} = 3$, a value that provides a good trade-off between the flexibility of the mixture allocations and local homogeneity.

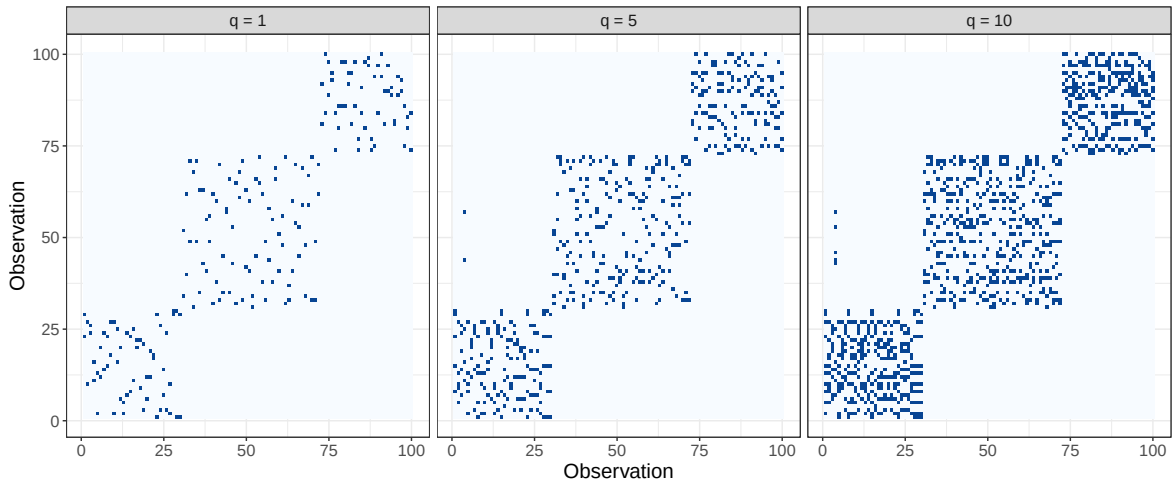

Figure 9: Heatmaps of the adjacency matrices $\mathcal{N}^{(q)}$ computed on a subset of observations of the GaussMix dataset. Different values of q are assumed.

Given this premise, we are now ready to discuss `Hidalgo()`, the high-level function that fits the Bayesian mixture. It implements the Gibbs sampler described in Appendix C, relying on low-level **Rcpp** routines. Also, the function internally calls `compute_mus()` to automatically generate the ratios of distances and the adjacency matrix needed to evaluate the likelihood from the data points.

The function has the following arguments: `X`, `dist_mat`, `q`, `D`, and

- `K`: integer, number of mixture components;
- `nsim`, `burn_in`, and `thinning`: number of MCMC iterations to collect, initial iterations to discard, and thinning interval, respectively;
- `verbose`: logical, if `TRUE`, the progress of the sampler is printed;
- `xi`: real number between 0 and 1, a local homogeneity parameter. The default is 0.75;
- `alpha_Dirichlet`: hyperparameter of the Dirichlet prior on the mixture weights;
- `a0_d` and `b0_d`: shape and rate parameters of the Gamma prior on d. The default is 1 for both values;
- `prior_type`: string, type of Gamma prior on d which can be
  - `"Conjugate"`: a classic Gamma prior is adopted (default);
  - `"Truncated"`: a truncated Gamma prior on the interval $(0, D)$ is used. This specification is advised when dealing with datasets characterized by a small number of columns, to avoid the estimated ID exceeding the nominal dimension `D`;
  - `"Truncated_PointMass"`: same as `"Truncated"`, but a point mass is placed on D. That is, the estimated ID is allowed to be exactly equal to the nominal dimension `D`;

- `pi_mass`: probability placed a priori on D when a `"Truncated_PointMass"` prior specification is chosen.

We apply the HIDALGO model on the GaussMix dataset with two different prior configurations: conjugate and truncated with point mass at `D` $= 5$. The code we used to run the models is:

```
R> set.seed(1234)
R> hid_fit <- Hidalgo(X = GaussMix, K = 10, alpha_Dirichlet = 0.05,
+    nsim = 2000, burn_in = 2000, thinning = 5, verbose = FALSE)
R> set.seed(12345)
R> hid_fit_TR <- Hidalgo(X = GaussMix, K = 10, alpha_Dirichlet = 0.05,
+    prior_type = "Truncated_PointMass", D = 5, nsim = 2000,
+    burn_in = 2000, thinning = 5, verbose = FALSE)
```

We can print one of the returned objects to visualize a short summary of the run:

```
R> hid_fit_TR

Model: Hidalgo
Method: Bayesian Estimation
Prior d ~ Gamma(1, 1), type = Truncated_PointMass
Prior on mixture weights: Dirichlet(0.05) with 10 mixture components
MCMC details:
Total iterations: 4000, Burn in: 2000, Thinning: 5
Used iterations: 2000
Elapsed time: 2.2978 mins
```

By using `alpha_Dirichlet = 0.05`, we have adopted a sparse mixture modeling approach in the spirit of Malsiner-Walli, Frühwirth-Schnatter, and Grün (2016). The sparse mixture approach would automatically let the data estimate the number of mixture components required. As a consequence, the argument `K` should be interpreted as an upper bound on the number of active clusters. Nonetheless, we stress that estimating the number of well-separated clusters with Pareto kernels is challenging. Hence, we will discuss how to analyze the output to perform proper inference. The output object `hid_fit` is a list of class '`Hidalgo`', containing six elements:

- `cluster_prob`: matrix of dimension `nsim`×`K`. Each column contains the MCMC sample of a mixing weight for every mixture component;
- `membership_labels`: matrix of dimension `nsim`×`n`. Each column contains the MCMC sample of a membership label for every observation;
- `id_raw`: matrix of dimension `nsim`×`K`. Each column contains the MCMC sample for the ID estimated in every cluster;
- `id_postpr`: matrix of dimension `nsim`×`n`. It contains a chain for each observation, corrected for label-switching;

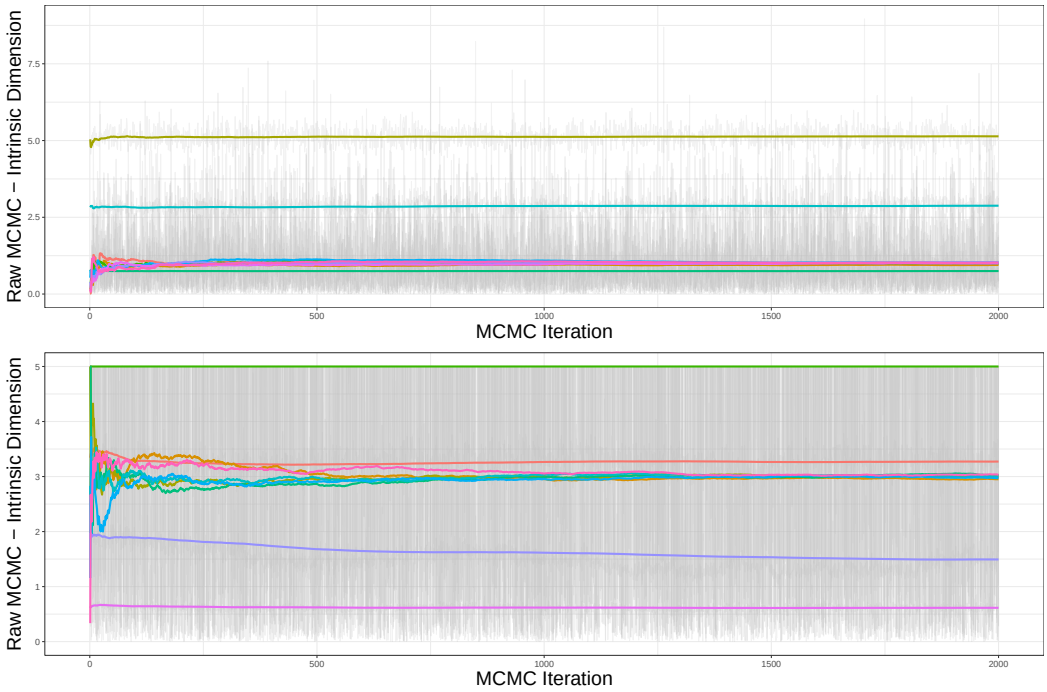

Figure 10: MCMC traceplots and superimposed ergodic means of the components of the ID vector. Top panel: conjugate prior specification. Bottom panel: truncated with point mass prior specification.

- `id_summary`: a matrix containing the posterior mean and the 5%, 25%, 50%, 75%, 95% quantiles for each observation;
- `recap`: a list with the specifications passed to the function as inputs.

To inspect the output, we can employ the dedicated `autoplot()` function devised for objects of class '`Hidalgo`'. There are several arguments that can be specified, producing different graphs. The most important is

- `type`: string that indicates the type of plot that is requested. It can be:
  - `"raw_chains"`: plot the MCMC and the ergodic means **not** corrected for label-switching (default);
  - `"point_estimates"`: plot the posterior mean and median ID for each observation, along with their CRIs;
  - `"class_plot"`: plot the estimated ID distributions stratified by the groups specified in an additional `class` vector;
  - `"clustering"`: plot the posterior co-clustering matrix. Rows and columns can be stratified by an exogenous `class` and/or a clustering structure.

For example, we can plot the raw chains of the two models with the aid of the **patchwork** package (Pedersen 2022), producing Figure 10, via:

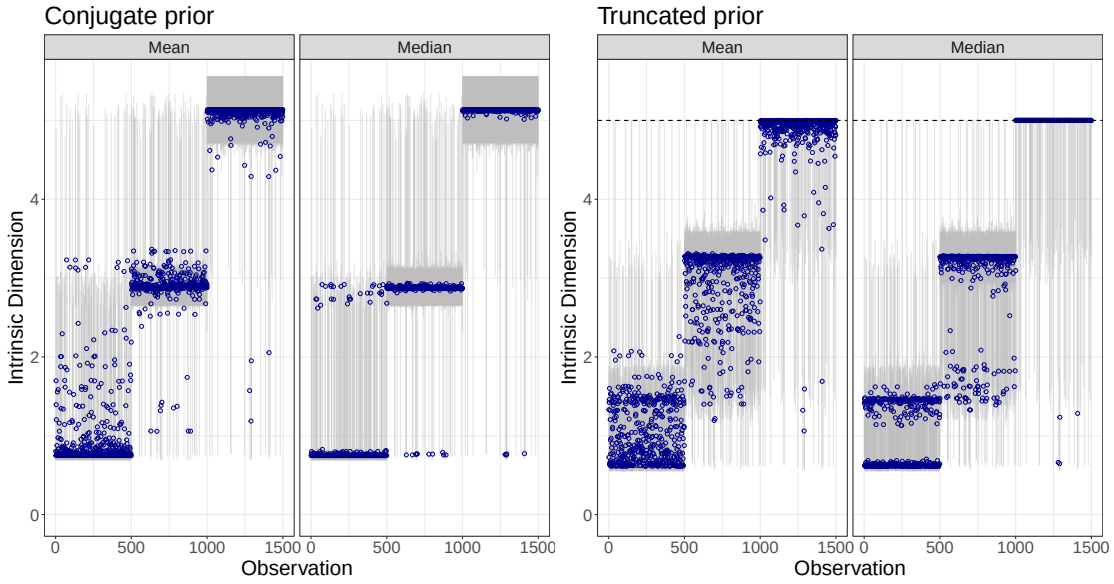

Figure 11: Observation-specific posterior means (left panels) and medians (right panels) ID represented with blue dots. The gray bars represent the 90% CRIs. The two plots correspond to two different prior specifications.

```
R> autoplot(hid_fit) / autoplot(hid_fit_TR)
```

Plotting the traceplots of the elements in $\boldsymbol{d}$ allows us to assess the convergence of the algorithm. First, however, we need to be aware that these chains may suffer from label-switching issues, preventing us from directly drawing inference from the MCMC output. Due to label-switching, mixture components can be discarded, emptied, or repopulated across iterations. This behavior is observed in Figure 10, which shows the MCMC traceplots of the two models, with the ergodic means for each mixture component superimposed. In this type of plot, we can often notice that various chains overlap around the prior mean of d_k. These chains represent the parameters of the empty clusters, which are sampled from the prior. For example, in the top panel of Figure 10 (`"Conjugate"` prior), $\mathbb{E}[d_k] = a_d/b_d = 1$. Recall that the presence of empty clusters is favored by the sparse mixture setting.

Additionally, we can see that if no constraint is imposed on the support of the prior distribution for $\boldsymbol{d}$ (top panel), the posterior estimates can exceed the nominal dimension `D = 5` of the GaussMix dataset. However, this problem disappears when imposing a truncation on the prior support (bottom panel).

To address the label-switching issue and perform meaningful inference, the raw MCMC sample needs to be postprocessed. In Appendix D, we discuss the algorithm used to map the K chains to n observation-specific chains that can be employed for inference. The algorithm is already implemented in `Hidalgo()`, and produces the elements `id_postpr` and `id_summary` in the returned list. We can obtain a visual summary of the postprocessed estimates via

```
R> autoplot(hid_fit, type = "point_estimates") +
+    autoplot(hid_fit_TR, type = "point_estimates")
```

The resulting plots are shown in Figure 11. The panels display the mean and median ID estimates for each data point. Here, the separation of the data into different generating manifolds is evident. Also, we notice that some of the estimates in the conjugate case are incorrectly above the nominal value `D = 5`, once again justifying the need for a truncated prior. The default plots were modified with `coord_cartesian(ylim = c(0, 5.5))` to highlight the effect of the truncation.

Estimated clustering solutions

It is natural to seek model-based clustering solutions when dealing with a mixture model. To this extent, the key source of information is the posterior similarity – or co-clustering – matrix (PSM). The entries $\{s_{i,j}\}_{i,j=1}^{n}$ of this matrix are computed as the proportion of times in which two observations have been assigned to the same mixture component across the MCMC iterations. Thus, the PSM describes the underlying clustering structure of the data detected by HIDALGO. Given the PSM, one can evaluate various loss functions on the space of the partitions. By minimizing the loss functions, we can retrieve the optimal partition of the dataset into clusters. To obtain such an estimate, we rely on the function `salso()` from the R package **salso** (Dahl, Johnson, and Müller 2022a). Otherwise, a faster alternative method proceeds by building a dendrogram from the implied posterior dissimilarity matrix (PDM), whose entries are given by $\{d_{i,j}\}_{i,j=1}^{n}$ where $d_{i,j} = 1 - s_{i,j}$, $\forall i, j$. Once the dendrogram is built, we can threshold it to segment the data into a pre-specified number of clusters `K`.

These approaches are implemented in the dedicated function `clustering()` which takes as arguments, along with the `object` output from the `Hidalgo()` function,

- `clustering_method`: string indicating the method to use to perform clustering. It can be `"dendrogram"` or `"salso"`. The former method thresholds the dendrogram constructed from the PDM to retrieve exactly `K` clusters. The latter method estimates the optimal clustering solution by minimizing a loss function on the space of the partitions. The default loss function is the variation of information (VI, Wade and Ghahramani 2018). For additional details about the VI loss function, see Appendix E;
- `K`: integer, used when `"dendrogram"` is chosen. It corresponds to the number of clusters to recover when thresholding the dendrogram obtained from the PDM;
- `nCores`: integer, argument used in the functions called from package **salso**. It represents the number of cores used to compute the PSM and the optimal clustering solution.

Additional arguments can be passed to modify the partition estimation via `salso()`. Given the large sample size of the GaussMix dataset, we opt for the dendrogram approach, truncating the dendrogram at `K = 3` groups. We highlight that relying on the minimization of a loss function is a more principled approach. However, the method can be misled by the strongly overlapping clusters estimated across the MCMC iterations, providing overly conservative solutions.

```
R> psm_cl <- clustering(object = hid_fit_TR,
+    clustering_method = "dendrogram", K = 3, nCores = 5)
R> psm_cl
```

```
Estimated clustering solution summary:

Method: dendrogram
Retrieved clusters: 3
Clustering frequencies:

| Cluster 1| Cluster 2| Cluster 3|
|---------:|---------:|---------:|
|       554|       450|       496|
```

To visualize the results, we can also plot the PSM by passing an object of class 'Hidalgo' to `autoplot()` with `type = "clustering"`. The `autoplot()` method internally calls the function `clustering()` to compute the PSM. One can also specify an additional argument `class` to stratify the observations according to exogenous factors. Finally, we remark that the user can also rely on partitioning around medoids on the dissimilarity matrix PDM, a robust clustering procedure implemented in the R package **PReMiuM** (Liverani, Hastie, Azizi, Papathomas, and Richardson 2015).

The presence of patterns in the data uncovered by the ID

Once the observation-specific estimates are computed, we can investigate the presence of potential patterns between the recovered IDs and given exogenous variables. To explore these possible relations, we can use the function `id_by_class()`. Along with an object of class 'Hidalgo', we need to specify:

- `class`: factor, a variable used to stratify the ID posterior estimates.

For the GaussMix dataset, the exogenous information is contained in the `class_GMix` vector, which we pass as `class`.

```
R> id_by_class(object = hid_fit_TR, class = class_GMix)

Posterior ID by class:

|class |     mean|    median|        sd|
|:-----|--------:|---------:|---------:|
|A     | 1.031105| 0.9019793| 0.3781802|
|B     | 2.961009| 3.2068593| 0.4683262|
|C     | 4.864018| 4.9685900| 0.3774186|
```

The estimates in the three classes are very close to the ground truth. The same argument, `class`, can be passed to the `autoplot()` function, in combination with

- `class_plot_type`: string, if `type = "class_plot"`, one can visualize the stratified ID estimates with a `"density"` plot or a `"histogram"`, or using `"boxplots"` or `"violin"` plots;
- `class`: a vector containing class information used to stratify the observations;

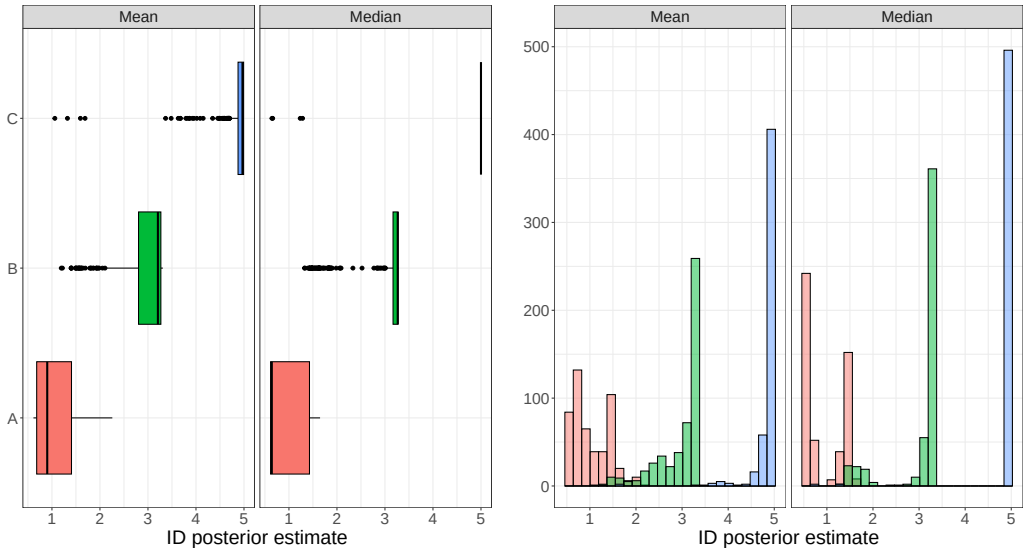

Figure 12: Two different types of graphs where the estimated IDs are stratified by a given exogenous variable.

to visualize ID estimates of the GaussMix dataset stratified by the generating manifold of the observations. As an example of possible graphs, Figure 12 shows the stratified boxplots (left panel) and histograms (right panel).

```
R> autoplot(hid_fit_TR, type = "class", class = class_GMix,
+       class_plot_type = "boxplot") +
+    autoplot(hid_fit_TR, type = "class", class = class_GMix,
+       class_plot_type = "histogram")
```

We have introduced and discussed the principal functions of the **intRinsic** package concerning the TWO-NN and HIDALGO models. Then, by employing simulated data with known IDs, we have suggested a pipeline to guide our study. In the next section, we present a real data analysis, highlighting how the ID estimation can be used to effectively reduce the size of a dataset while capturing and preserving important features.

4. The ID of gene microarray measurements

In this section, we present a real data example investigating the ID of the Alon dataset. The dataset, first presented in Alon *et al.* (1999), contains microarray measurements for 2000 genes measured on 62 patients. Among the patients, 40 were diagnosed with colon cancer, and 22 were healthy subjects. A factor variable named `status` describes the patient health condition (coded as `"Cancer"` vs. `"Healthy"`). A copy of this famous dataset can be found in the R package **HiDimDA** (Duarte Silva 2015). We store the gene measurements in the object `Xalon`, a matrix of nominal dimension `D = 2000`, with `n = 62` observations. To load and prepare the data, we write:

```
R> data("AlonDS", package = "HiDimDA")
```

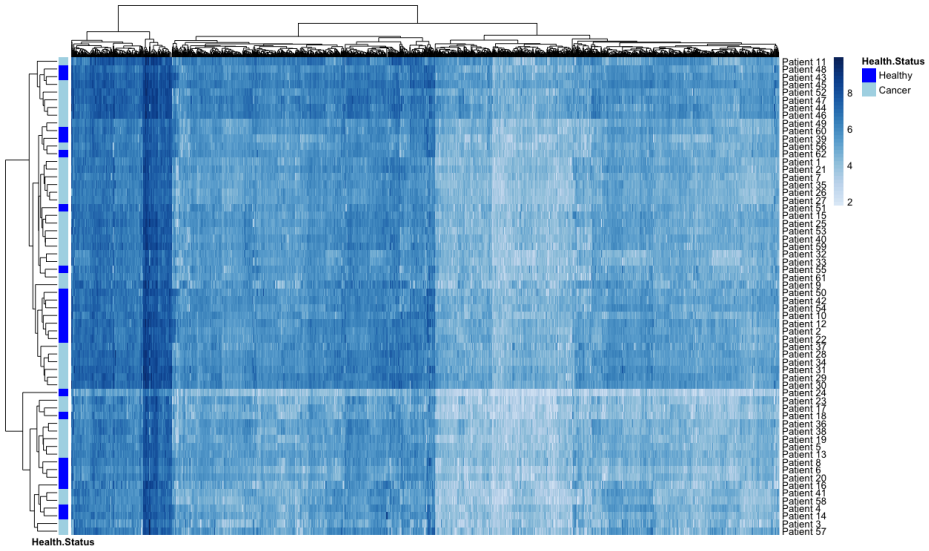

Figure 13: Heatmap of the log-values of the Alon microarray dataset. The patients on the rows are labeled according to their health status.

```
R> status <- factor(AlonDS$grouping, labels = c("Cancer", "Healthy"))
R> Xalon <- as.matrix(AlonDS[, -1])
```

To obtain a visual summary of the dataset, we plot the heatmap of the log-data values annotated by `status`. The result is shown in Figure 13. No clear structure is immediately visible.

We ultimately seek to uncover hidden patterns in this dataset. The task is challenging, especially given the small number of available observations. As a first step, we investigate how well a unique, global ID estimate can represent the data.

4.1. Homogeneous ID estimation

Let us start by describing the overall complexity of the dataset by estimating a homogeneous ID value. Using the TWO-NN model, we can compute:

```
R> Alon_twonn_1 <- twonn(Xalon, method = "linfit")
R> summary(Alon_twonn_1)

Model: TWO-NN
Method: Least Squares Estimation
Sample size: 62, Obs. used: 61. Trimming proportion: 1%
ID estimates (confidence level: 0.95)

| Lower Bound| Estimate| Upper Bound|
|-----------:|--------:|-----------:|
|    10.00382| 10.34944|    10.69506|

R> Alon_twonn_2 <- twonn(Xalon, method = "bayes")
R> summary(Alon_twonn_2)
```

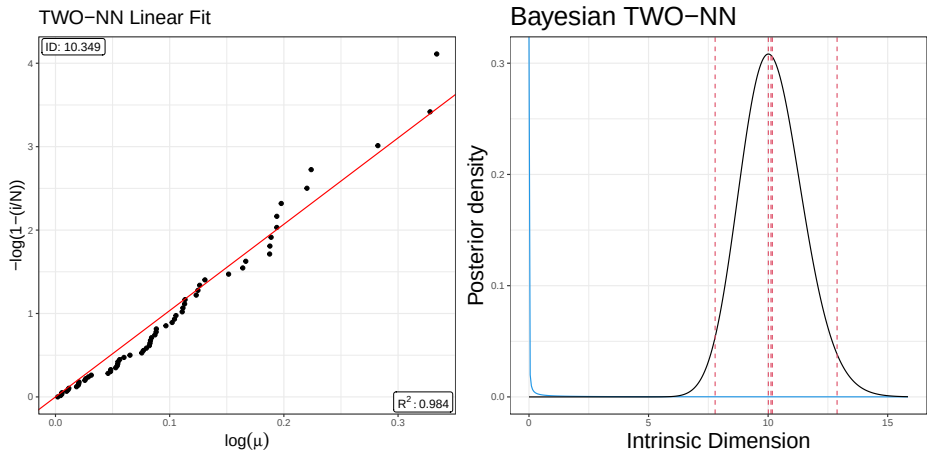

Figure 14: Alon dataset. The left panel shows the result of the linear estimator, while the right panel depicts the posterior distribution obtained via the Bayesian approach.

```
Model: TWO-NN
Method: Bayesian Estimation
Sample size: 62, Obs. used: 61. Trimming proportion: 1%
Prior d ~ Gamma(0.001, 0.001)
Credible Interval quantiles: 2.5%, 97.5%
Posterior ID estimates:

| Lower Bound|     Mean|   Median|     Mode| Upper Bound|
|-----------:|--------:|--------:|--------:|-----------:|
|    7.784152| 10.17639| 10.12084| 10.00957|    12.88427|
```

```
R> Alon_twonn_3 <- twonn(Xalon, method = "mle")
R> summary(Alon_twonn_3)

Model: TWO-NN
Method: MLE
Sample size: 62, Obs. used: 61. Trimming proportion: 1%
ID estimates (confidence level: 0.95)

| Lower Bound| Estimate| Upper Bound|
|-----------:|--------:|-----------:|
|    7.785304| 10.01107|    12.88623|
```

The estimates based on the TWO-NN model obtained with different methods are very similar. The results are also illustrated in Figure 14, which shows the linear fit (left panel) and posterior distribution (right panel) for the TWO-NN model. According to these results, we conclude that the information contained in the `D = 2000` genes can be summarized with approximately ten variables. For example, the first ten eigenvalues computed from the spectral decomposition of the matrix $\Lambda = X_{Alon}^{\top} X_{Alon}$ contribute to explaining 95.4% of the total variance.

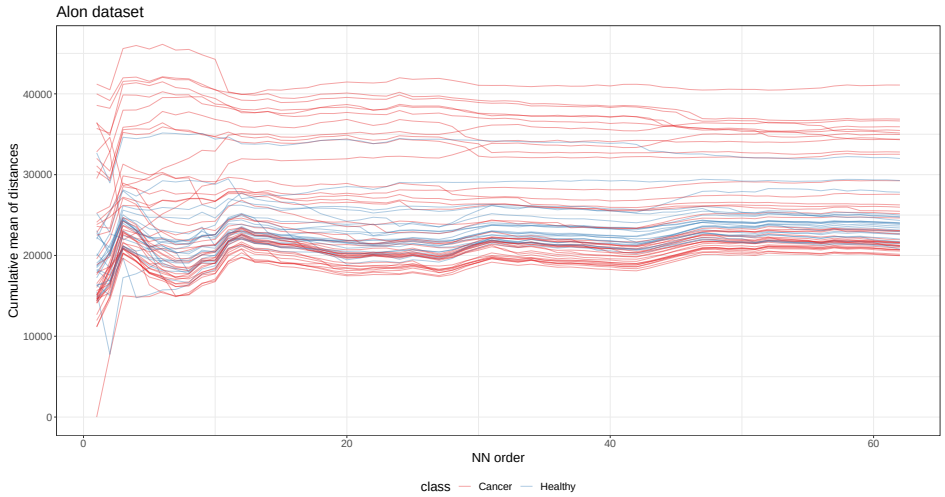

Figure 15: Evolution of the cumulative means of NN distances computed for all the observations in the Alon dataset.

Although the linear fit plot and the TWO-NN estimates do not raise any evident sign of concern, as a final check, we explore the evolution of the average distances between NN, reported in Figure 15. As expected, the plot does not highlight any abrupt change in the evolution of the ergodic means. However, it suggests that investigating the presence of multiple manifolds could be interesting. In fact, despite the evolution of most of the ergodic means being stationary, their heterogeneous levels highlight some potential data inhomogeneities that should be deepened.

4.2. Heterogeneous ID estimation

To investigate the presence of heterogeneous latent manifolds in the Alon dataset, we employ `Hidalgo()`. Since the nominal dimension `D` is large, we do not need to truncate the prior on d. Moreover, given the small number of data points, we opt for an informative and regularizing prior $Gamma(1, 1)$ (the default) instead of a vague specification. Also, we set a conservative upper bound for the mixing components `K = 15` and choose again $\alpha = 0.05$ to fit a sparse mixture. We run:

```
R> set.seed(1234)
R> Alon_hid <- Hidalgo(X = Xalon, K = 15, a0_d = 1, b0_d = 1,
+    alpha_Dirichlet = 0.05, nsim = 10000, burn_in = 100000, thin = 5)
R> Alon_hid

Model: Hidalgo
Method: Bayesian Estimation
Prior d ~ Gamma(1, 1), type = Conjugate
Prior on mixture weights: Dirichlet(0.05) with 15 mixture components
MCMC details:
Total iterations: 110000, Burn in: 1e+05, Thinning: 5
Used iterations: 10000
Elapsed time: 38.3067 secs
```

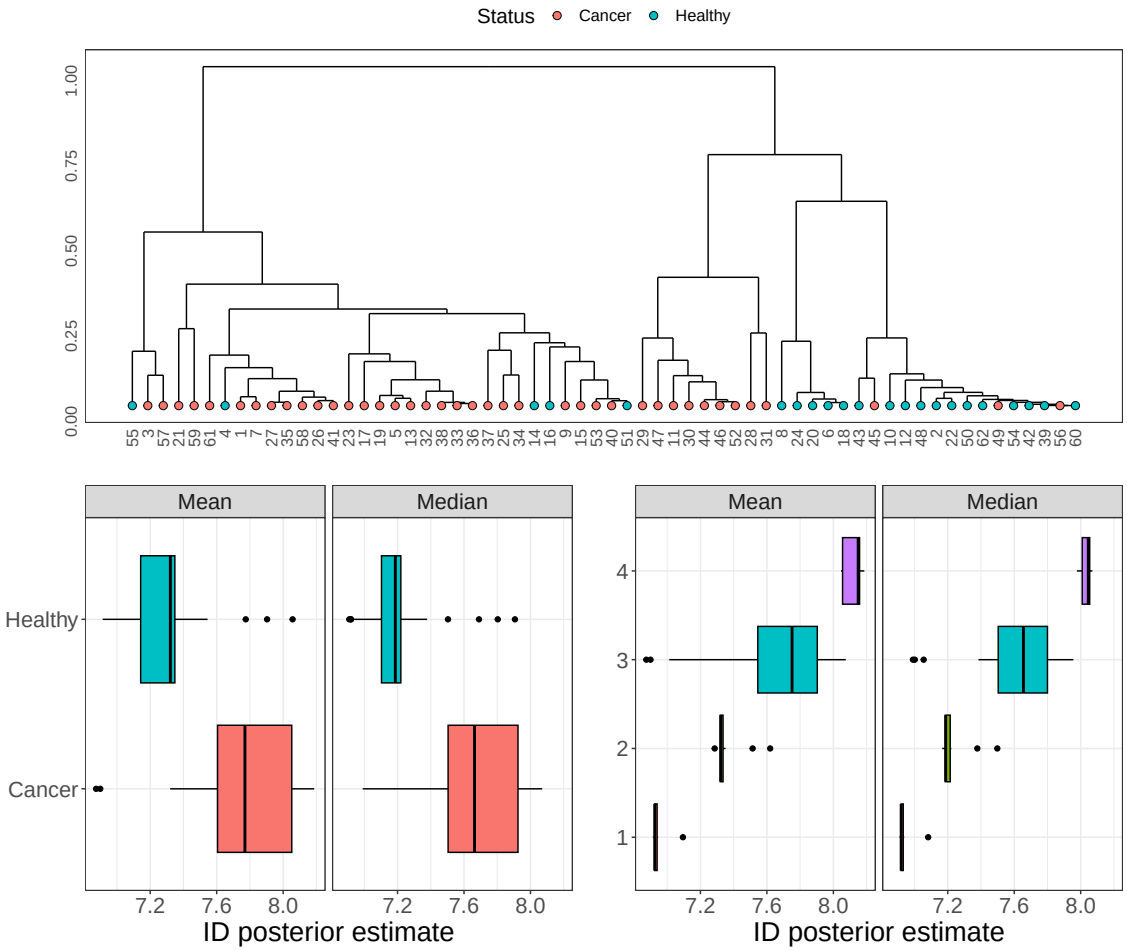

Figure 16: Alon dataset. Top panel: dendrogram obtained from the PDM. Bottom panel: boxplots of the ID estimates stratified by health status (left) and estimated partition (right).

Once the model is fitted, we first explore the estimated clustering structure. Here, instead of directly plotting the heatmap of the PSM, we build the dendrogram from the PDM, and we report it in the top panel of Figure 16. We construct such a plot with the help of the package **ggdendro** (De Vries and Ripley 2022). We can detect four clusters, and therefore we decide to set `K = 4` when running

```
R> Alon_psm <- clustering(Alon_hid, K = 4)
```

As illustrated in the previous section, all these plots can be obtained by calling `autoplot()` with the proper argument specifications. The next natural step is to investigate how strongly the estimated partition and, in general, the estimated IDs are associated with the health `status`. The bottom two panels of Figure 16 display the boxplots of the values (means and medians) of the postprocessed, observation-specific IDs stratified by `status` (left) and estimated cluster (right). We can also run

```
R> id_by_class(Alon_hid,class = status)
```

Cluster	# Cancer	# Healthy	% Healthy	Mean ID	Median ID	Std. Dev. ID
1	0	5	1.0000	6.9595	6.9252	0.0761
2	3	12	0.8000	7.3564	7.3222	0.0889
3	28	5	0.1515	7.6883	7.7513	0.3083
4	9	0	0.0000	8.1198	8.1493	0.0546

Table 6: Stratification by cluster of the health status (frequencies and proportions of healthy patients) and ID estimates (mean, median, and standard deviation).

rfm1	Cancer	Healthy	class.err	rfm2	Cancer	Healthy	class.err
Cancer	36	4	0.100	Cancer	35	5	0.125
Healthy	9	13	0.409	Healthy	6	16	0.273
Dataset:	X_OR	OOB err:	20.97%	Dataset:	X_ID	OOB err:	17.74%

Table 7: Confusion matrices summarizing the classification performance of the two random forest models, trained with the original gene expressions (`rfm1`, left) and with a summary of the ID estimates (`rfm2`, right).

```
R> id_by_class(Alon_hid,class = Alon_psm$clust)
```

to obtain summary results linking the variations in the ID with health status and cluster. We report the output in Table 6. As the estimated ID increases, the proportion of healthy subjects in each cluster decreases. This result suggests that the microarray profiles of people diagnosed with cancer are slightly more complex than healthy patients' ones.

The analyses conducted so far helped us uncover interesting descriptive characteristics of the IDs in the dataset. Nevertheless, these results can also be effectively used as a representative data summary. Here, we show how the estimated individual IDs are valid to potentially classify the health `status` of new patients according to their genomic profiles. As a simple example, we perform a classification analysis using two random forest models, predicting the target variable `Y = status`. To train the models, we use two different sets of covariates: `X_OR`, the original dataset composed of 2000 genes,

```
R> X_OR <- data.frame(Y = status, X = Xalon)
R> set.seed(1231)
R> rfm1 <- randomForest::randomForest(Y ~ ., data = X_OR,
+    type = "classification", ntree = 100)
```

and `X_ID`, the observation-specific ID summary returned by `Hidalgo()`, along with our estimated partition.

```
R> X_ID <- data.frame(Y = status, X = summary(Alon_hid),
+    clust = factor(Alon_psm$clust))
R> set.seed(1231)
R> rfm2 <- randomForest::randomForest(Y ~ ., data = X_ID,
+    type = "classification", ntree = 100)
```

The classification results are reported in Table 7.

Remarkably, a simple dataset with seven variables summarizing the main distributional traits of the observation-specific posterior IDs obtains good performance in predicting health status, similar to the original dataset. More precisely, the random forest on the original dataset got an out-of-bag estimated error rate of 20.97%, while the error is reduced to 17.74% when using our ID-based covariates. We can conclude that, in this case, the topological properties of the dataset are associated with the outcome of interest and convey important information.

We showed how the estimation of heterogeneous ID provides a reliable complexity index for elaborate data structures and helps unveil relationships among data points hidden at the topological level. The application to the Alon dataset showcases how reliable ID estimates give additional fundamental perspectives that help us discover non-trivial data patterns. Furthermore, one can exploit the extracted information in many downstream investigations, such as patient segmentation or predictive analyses.

5. Summary and discussion

In this paper, we illustrated **intRinsic**, an R package that implements novel routines for the ID estimation according to the models recently developed in Facco *et al.* (2017); Allegra *et al.* (2020); Denti *et al.* (2022), and Santos-Fernandez *et al.* (2022). **intRinsic** consists of a collection of high-level, user-friendly functions that, in turn, rely on efficient, low-level routines implemented in R and C++. We also remark that **intRinsic** integrates functionalities from external packages. For example, all the graphical outputs returned by the functions are built using the well-known package **ggplot2**. Therefore, they are easily customizable using the grammar of graphics (Wilkinson 2005).

The package includes frequentist and Bayesian model specifications for the TWO-NN global ID estimator. Moreover, it implements the Gibbs sampler for posterior simulation of the HIDALGO model, which can capture the presence of heterogeneous ID within a single dataset. We showed how discovering multiple latent manifolds could help unveil the topological traits of a dataset, primarily when additional exogenous variables are used to stratify the ID estimates.

As a general analysis pipeline for practitioners, we suggested starting with the efficient TWO-NN functions to understand how appropriate the hypothesis of homogeneity is for the data at hand. If departures from the assumptions are visible from nonuniform estimates obtained with different estimation methods and from visual assessment of the evolution of the average NN distances, one should rely on HIDALGO.

The most promising future research directions stem from HIDALGO. First, we plan to develop more reliable methods to obtain an optimal partition of the data based on the ID estimates since the one proposed heavily relies on a mixture model of overlapping distributions. Moreover, another research avenue worth exploring is a version of HIDALGO with likelihood distributions based on generalized NN ratios, exploiting the information coming from varying neighborhood sizes.

We also know that the mixture model fitting may become computationally expensive if the analyzed datasets are large. Therefore, faster solutions, such as the Variational Bayes approach, will be explored. Also, we highlight that HIDALGO, a mixture model within a Bayesian framework, lacks a frequentist-based estimation counterpart, such as an expectation maximization algorithm. Its derivation is not immediate since the neighboring structure introduced via the $\mathcal{N}^{(q)}$ matrix makes the problem non-trivial. We plan to keep working on this

package and continuously update it in the long run as contributions to this line of research become available. The novel ID estimators we discussed have started a lively research branch, and we intend to include all the future advancements in **intRinsic**.

Acknowledgments

The author thanks the editorial team and the two anonymous reviewers for their constructive comments. Moreover, the author is extremely grateful to Andrea Gilardi for his valuable guidance. Finally, the author also thanks Michelle N. Ngo, Derenik Haghverdian, Wendy N. Rummerfield, Andrea Cappozzo, and Riccardo Corradin for their comments on earlier versions of this manuscript.

References

Allegra M, Facco E, Denti F, Laio A, Mira A (2020). "Data Segmentation Based on the Local Intrinsic Dimension." *Scientific Reports*, **10**(1), 16449. doi:10.1038/s41598-020-72222-0.

Alon U, Barkai N, Notterman DA, Gish K, Ybarra S, Mack D, Levine AJ (1999). "Broad Patterns of Gene Expression Revealed by Clustering Analysis of Tumor and Normal Colon Tissues Probed by Oligonucleotide Arrays." *Proceedings of the National Academy of Sciences of the United States of America*, **96**(12), 6745–6750. doi:10.1073/pnas.96.12.6745.

Amsaleg L, Chelly O, Houle ME, Kawarabayashi K, Radovanović M, Treeratanajaru W (2019). "Intrinsic Dimensionality Estimation within Tight Localities." In *Proceedings of the 2019 SIAM International Conference on Data Mining (SDM)*, pp. 181–189. doi:10.1137/1.9781611975673.21.

Ansuini A, Laio A, Macke JH, Zoccolan D (2019). "Intrinsic Dimension of Data Representations in Deep Neural Networks." In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pp. 6111–6122. Article ID 549.

Bac J, Mirkes EM, Gorban AN, Tyukin I, Zinovyev A (2021). "**scikit-dimension**: A Python Package for Intrinsic Dimension Estimation." *Entropy*, **23**(10), 1–12. doi:10.3390/e23101368.

Bartenhagen C (2022). ***RDRToolbox**: A Package for Nonlinear Dimension Reduction with Isomap and LLE*. doi:10.18129/B9.bioc.RDRToolbox. R package version 1.48.0.

Bennett RS (1969). "The Intrinsic Dimensionality of Signal Collections." *IEEE Transactions on Information Theory*, **15**(5), 517–525. doi:10.1109/tit.1969.1054365.

Beygelzimer A, Kakadet S, Langford J, Arya S, Mount D, Li S (2022). ***FNN**: Fast Nearest Neighbor Search Algorithms and Applications*. R package version 1.1.3.1, URL https://CRAN.R-project.org/package=FNN.

Campadelli P, Casiraghi E, Ceruti C, Rozza A (2015). "Intrinsic Dimension Estimation: Relevant Techniques and a Benchmark Framework." *Mathematical Problems in Engineering*, **2015**, 759567. doi:10.1155/2015/759567.

Cannoodt R, Saelens W (2021). ***dyndimred**: Dimensionality Reduction Methods in a Common Format*. R package version 1.0.4, URL https://CRAN.R-project.org/package=dyndimred.

Carter KM, Raich R, Hero AO (2010). "On Local Intrinsic Dimension Estimation and Its Applications." *IEEE Transactions on Signal Processing*, **58**(2), 650–663. doi:10.1109/tsp.2009.2031722.

Costa JA, Hero AO (2004). "Geodesic Entropic Graphs for Dimension and Entropy Estimation in Manifold Learning." *IEEE Transactions on Signal Processing*, **52**(8), 2210–2221. doi:10.1109/tsp.2004.831130.

Dahl DB, Johnson DJ, Müller P (2022a). ***salso**: Search Algorithms and Loss Functions for Bayesian Clustering*. R package version 0.3.29, URL https://CRAN.R-project.org/package=salso.

Dahl DB, Johnson DJ, Müller P (2022b). "Search Algorithms and Loss Functions for Bayesian Clustering." *Journal of Computational and Graphical Statistics*, **31**(4), 1189–1201. doi:10.1080/10618600.2022.2069779.

De Vries A, Ripley BD (2022). ***ggdendro**: Create Dendrograms and Tree Diagrams Using **ggplot2***. R package version 0.1.23, URL https://CRAN.R-project.org/package=ggdendro.

Denti F, Doimo D, Laio A, Mira A (2022). "The Generalized Ratios Intrinsic Dimension Estimator." *Scientific Reports*, **12**(1), 20005. doi:10.1038/s41598-022-20991-1.

Denti F, Gilardi A (2023). ***intRinsic**: Likelihood-Based Intrinsic Dimension Estimators*. R package version 1.0.0, URL https://CRAN.R-project.org/package=intRinsic.

Di Narzo AF (2019). ***tseriesChaos**: Analysis of Nonlinear Time Series*. R package version 0.1-13.1, URL https://CRAN.R-project.org/package=tseriesChaos.

Duarte Silva AP (2015). ***HiDimDA**: High Dimensional Discriminant Analysis*. R package version 0.2-4, URL https://CRAN.R-project.org/package=HiDimDA.

Eddelbuettel D, François R (2011). "**Rcpp**: Seamless R and C++ Integration." *Journal of Statistical Software*, **40**(8), 1–18. doi:10.18637/jss.v040.i08.

Eddelbuettel D, Sanderson C (2014). "**RcppArmadillo**: Accelerating R with High-Performance C++ Linear Algebra." *Computational Statistics & Data Analysis*, **71**, 1054–1063. doi:10.1016/j.csda.2013.02.005.

Facco E (2017). *The Intrinsic Dimension of Biological Data Landscapes*. Ph.D. thesis, SISSA, Trieste. URL https://core.ac.uk/download/pdf/144263715.pdf.

Facco E, D'Errico M, Rodriguez A, Laio A (2017). "Estimating the Intrinsic Dimension of Datasets by a Minimal Neighborhood Information." *Scientific Reports*, **7**(1), 12140. doi:10.1038/s41598-017-11873-y.

Falconer K (2003). *Fractal Geometry: Mathematical Foundations and Applications*. 2nd edition. John Wiley & Sons. doi:10.1002/0470013850.

Garcia CA (2022). ***nonlinearTseries**: Nonlinear Time Series Analysis*. R package version 0.2.12, URL https://CRAN.R-project.org/package=nonlinearTseries.

Glielmo A, Macocco I, Doimo D, Carli M, Zeni C, Wild R, D'Errico M, Rodriguez A, Laio A (2022). "**DADApy**: Distance-Based Analysis of Data-Manifolds in Python." *Patterns*, **3**(10), 100589. doi:10.1016/j.patter.2022.100589.

Golay J, Kanevski M (2017). "Unsupervised Feature Selection Based on the Morisita Estimator of Intrinsic Dimension." *Knowledge-Based Systems*, **135**, 125–134. doi:10.1016/j.knosys.2017.08.009.

Hino H (2017). "**ider**: Intrinsic Dimension Estimation with R." *The R Journal*, **9**(2), 329–341. doi:10.32614/rj-2017-054.

Hino H, Fujiki J, Akaho S, Murata N (2017). "Local Intrinsic Dimension Estimation by Generalized Linear Modeling." *Neural Computation*, **29**(7), 1838–1878. doi:10.1162/neco_a_00969.

Hotelling H (1933). "Analysis of a Complex of Statistical Variables into Principal Components." *Journal of Educational Psychology*, **24**(7), 498–520. doi:10.1037/h0070888.

Johnsson K (2019). ***intrinsicDimension**: Intrinsic Dimension Estimation*. R package version 1.2.0, URL https://CRAN.R-project.org/package=intrinsicDimension.

Johnsson K, Soneson C, Fontes M (2015). "Low Bias Local Intrinsic Dimension Estimation from Expected Simplex Skewness." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **37**(1), 196–202. doi:10.1109/tpami.2014.2343220.

Jollife IT, Cadima J (2016). "Principal Component Analysis: A Review and Recent Developments." *Philosophical Transactions of the Royal Society A*, **374**(2065). doi:10.1098/rsta.2015.0202.

Kayo O (2006). *Locally Linear Embedding Algorithm: Extensions and Applications*. Ph.D. thesis, University of Oulu, Faculty of Technology, Department of Electrical and Information Engineering. URL http://jultika.oulu.fi/files/isbn9514280415.pdf.

Kingman JFC (1992). *Poisson Processes*, volume 3. Oxford University Press.

Kraemer G, Reichstein M, Mahecha MD (2018). "**dimRed** and **coRanking** – Unifying Dimensionality Reduction in R." *The R Journal*, **10**(1), 342–358. doi:10.32614/rj-2018-039.

Krijthe JH (2022). ***Rtsne**: T-Distributed Stochastic Neighbor Embedding Using Barnes-Hut Implementation*. R package version 0.16, URL https://github.com/jkrijthe/Rtsne.

Lee G, Rodriguez C, Madabhushi A (2008). "Investigating the Efficacy of Nonlinear Dimensionality Reduction Schemes in Classifying Gene and Protein Expression Studies." *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, **5**(3), 368–384. doi:10.1109/tcbb.2008.36.

Levina E, Bickel PJ (2005). "Maximum Likelihood Estimation of Intrinsic Dimension." In LK Saul, Y Weiss, L Bottou (eds.), *Advances in Neural Information Processing Systems 17*, pp. 777–784. MIT Press. URL https://papers.nips.cc/paper/2577-maximum-likelihood-estimation-of-intrinsic-dimension.pdf.

Liverani S, Hastie DI, Azizi L, Papathomas M, Richardson S (2015). "PReMiuM: An R Package for Profile Regression Mixture Models Using Dirichlet Processes." *Journal of Statistical Software*, **64**(7), 1–30. doi:10.18637/jss.v064.i07.

Malsiner-Walli G, Frühwirth-Schnatter S, Grün B (2016). "Model-Based Clustering Based on Sparse Finite Gaussian Mixtures." *Statistics and Computing*, **26**(1–2), 303–324. doi:10.1007/s11222-014-9500-2.

Malsiner-Walli G, Frühwirth-Schnatter S, Grün B (2017). "Identifying Mixtures of Mixtures Using Bayesian Estimation." *Journal of Computational and Graphical Statistics*, **26**(2), 285–295. doi:10.1080/10618600.2016.1200472.

Meilă M (2007). "Comparing Clusterings – An Information Based Distance." *Journal of Multivariate Analysis*, **98**(5), 873–895. doi:10.1016/j.jmva.2006.11.013.

Mendes-Santos T, Turkeshi X, Dalmonte M, Rodriguez A (2021). "Unsupervised Learning Universal Critical Behavior via the Intrinsic Dimension." *Physical Review X*, **11**(1). doi:10.1103/physrevx.11.011040. Article number: 011040.

Pedersen TL (2022). ***patchwork**: The Composer of Plots*. R package version 1.1.2, URL https://CRAN.R-project.org/package=patchwork.

Pettis KW, Bailey TA, Jain AK, Dubes RC (1979). "An Intrinsic Dimensionality Estimator from Near-Neighbor Information." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI-1**(1), 25–37. doi:10.1109/tpami.1979.4766873.

Rastelli R, Friel N (2018). "Optimal Bayesian Estimators for Latent Variable Cluster Models." *Statistics and Computing*, **28**(6), 1169–1186. doi:10.1007/s11222-017-9786-y.

R Core Team (2022). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Roweis TS, Lawrence KS (2000). "Nonlinear Dimensionality Reduction by Locally Linear Embedding." *Science*, **290**(5500), 2323–2326. doi:10.1126/science.290.5500.2323.

Rozza A, Lombardi G, Rosa M, Casiraghi E, Campadelli P (2011). "**IDEA**: Intrinsic Dimension Estimation Algorithm." In G Maino, GL Foresti (eds.), *Image Analysis and Processing – ICIAP 2011*, volume 6978 of *Lecture Notes in Computer Science*, pp. 433–442. Springer-Verlag. doi:10.1007/978-3-642-24085-0_45.

Santos-Fernandez E, Denti F, Mengersen K, Mira A (2022). "The Role of Intrinsic Dimension in High-Resolution Player Tracking Data – Insights in Basketball." *The Annals of Applied Statistics*, **16**(1), 326–348. doi:10.1214/21-aoas1506.

Sevcikova H, Percival D, Gneiting T (2021). ***fractaldim**: Estimation of Fractal Dimensions*. R package version 0.8-5, URL https://CRAN.R-project.org/package=fractaldim.

Tenenbaum JB, De Silva V, Langford JC (2000). "A Global Geometric Framework for Nonlinear Dimensionality Reduction." *Science*, **290**(5500), 2319–2323. doi:10.1126/science.290.5500.2319.

Van der Maaten L, Hinton G (2009). "Visualizing Data Using t-SNE." *Journal of Machine Learning Research*, **9**(86), 2579–2605. URL https://www.jmlr.org/papers/v9/vandermaaten08a.html.

Van Rossum G, *et al.* (2011). *Python Programming Language*. URL https://www.python.org/.

Vinh NX, Epps J, Bailey J (2010). "Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance." *Journal of Machine Learning Research*, **11**(95), 2837–2854. URL https://jmlr.org/papers/v11/vinh10a.html.

Wade S, Ghahramani Z (2018). "Bayesian Cluster Analysis: Point Estimation and Credible Balls (with Discussion)." *Bayesian Analysis*, **13**(2), 559–626. doi:10.1214/17-ba1073.

Wickham H (2016). ***ggplot2**: Elegant Graphics for Data Analysis*. Springer-Verlag. doi:10.1007/978-0-387-98141-3.

Wilkinson L (2005). *The Grammar of Graphics*. Springer-Verlag.

You K (2022a). ***Rdimtools**: Dimension Reduction and Estimation Methods*. R package version 1.1.2, URL https://CRAN.R-project.org/package=Rdimtools.

You K (2022b). "**Rdimtools**: An R Package for Dimension Reduction and Intrinsic Dimension Estimation." *Software Impacts*, **14**. doi:10.1016/j.simpa.2022.100414.

A. Additional methods implemented in the package

In this paper, we have focused our attention on the TWO-NN and the HIDALGO models. In Section 2, we explained that both methods are based on the distributional properties of the ratios of distances between a point and its first two NNs. However, this modeling framework has been extended by Denti *et al.* (2022), where the authors developed a novel ID estimator called GRIDE. This new estimator is based upon the ratios of distances between a point and two of its NNs of generic order, namely n_1 and n_2. Extending the neighborhood size leads to two major implications: more stringent local homogeneity assumptions and the possibility of computing ID estimates as a function of the chosen NN orders. Monitoring the ID evolution as the order of the furthest NN n_2 increases allows the extraction of meaningful information regarding the link between the ID and the scale of the considered neighborhood. In doing so, GRIDE produces estimates that are more robust to noise present in the data, which is not directly addressed by the model formulation.

The GRIDE model is implemented in **intRinsic**, and the estimation can be carried out under both the frequentist and Bayesian frameworks via the function `gride()`, which is very similar to `twonn()` in its usage. Additionally, one can use the functions `twonn_decimation()` and `gride_evolution()` to study the ID dynamics. More details about these functions are available in the package documentation.

The map in Figure 17 provides a visual summary of the most important functions contained in the package. The main topics are reported in the diamonds, while the high-level, exported functions are displayed in the blue rectangles. These routines are linked to the (principal) low-level function via dotted lines. Finally, the light-blue area highlights the functions discussed in this paper.

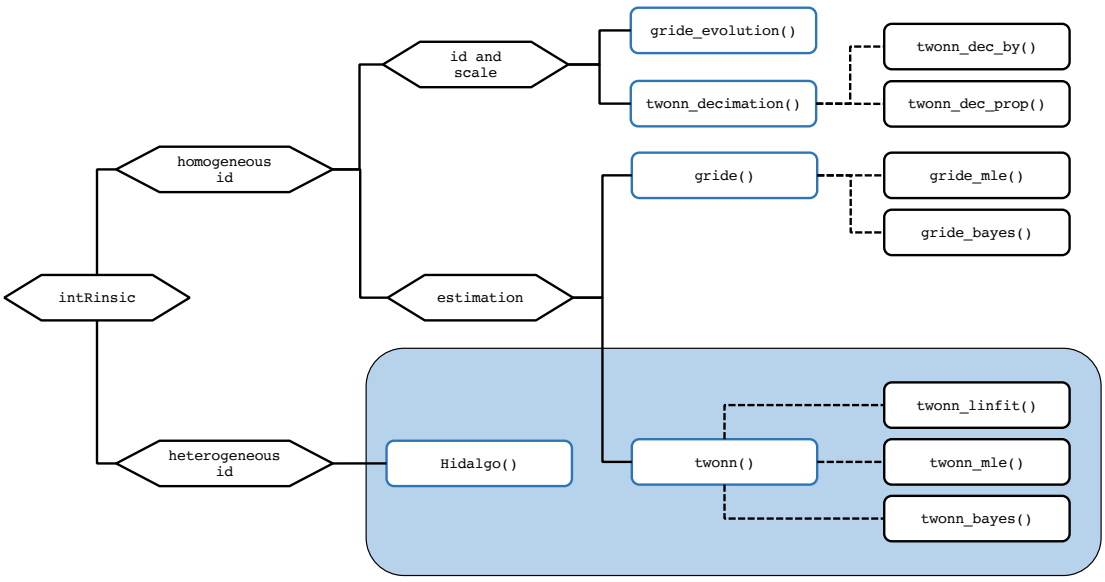

Figure 17: A conceptual map summarizing the most important functions contained in **intRinsic**. The blue squares contain the names of the principal high-level functions. Dotted lines connect these functions with the most important low-level functions (not exported). The light-blue area represents the topics that have been discussed in this paper.

B. intRinsic and other packages

As mentioned in Section 1, there is a large number of ID estimators available in the literature, and many of them have been implemented in R. A valuable survey of the availability of dimensionality reduction methods and ID estimators has been recently reported in You (2022b). From there, we see that two packages are the most important when it comes to ID estimation: **Rdimtools** and **intrinsicDimension**. **Rdimtools**, in particular, is comprised of an unprecedented collection of methods – including also the least-squares TWO-NN.

At the moment of writing, two main traits of **intRinsic** are unique to this package. First, our package is devoted to the recently proposed likelihood-based estimation methods introduced by the seminal work of Facco *et al.* (2017) and the literature that followed. Therefore, as of today, many of the R implementations presented here are exclusively contained in this package. This is true, for example, for the MLE and Bayesian versions of the TWO-NN model, the HIDALGO model, and all the routines linked to GRIDE. To the best of our knowledge, the function `Hidalgo()` is available outside this package. However, one can only find it on GitHub repositories, coded in Python and C++. Note that Python versions of the TWO-NN estimator have also been implemented in the recent **scikit-dimension** and **DADApy** packages (Bac *et al.* 2021; Glielmo *et al.* 2022). Moreover, **DADApy** contains routines dedicated to GRIDE. Table 8 presents a summary of recent software packages containing ensembles of ID estimation methods. Second, all the functions in our package allow – and emphasize – the uncertainty quantification around the ID estimates, which is a crucial component granted by our model-based approach. This feature is often overlooked in other implementations.

Overall, the wide variety of methods and ongoing research in this area indicate that there is no globally optimal estimator to employ regardless of the application. Thus, a practitioner should be aware of the strengths and limitations of every method.

One limitation of the likelihood-based models offered in this package, shared with many other ID estimators in general, is the underestimation of the ID when the true latent manifold's dimension is large. As an empirical rule, for cases where the estimated ID is large (e.g., $d > 20$), the retrieved value should be cautiously regarded as a lower bound for the actual ID (Ansuini *et al.* 2019). An alternative method we found particularly robust to this issue is the expected simplex skewness (ESS) algorithm proposed by Johnsson, Soneson, and Fontes (2015). For example, consider 5000 observations sampled from a $D = d = 50$ dimensional Gaussian distribution. With the following code, we can see how `twonn()` underestimates the true ID, which the ESS instead recovers well.

Package	Language	ID estimation methods	Overlap with **intrRinsic**
intrinsicDimension	R	5	–
Rdimtools	R	17	TWO-NN§
ider	R	7	–
DADApy	Python	2	TWO-NN†,‡,§, GRIDE†
scikit-dimension	Python	19	TWO-NN§

Table 8: A non-exhaustive list of the most recent software packages for ID estimation in R and Python. The superscripts indicate the implemented estimation procedures. In detail: †MLE; ‡Bayesian estimation; §linear fit.

```
R> set.seed(12211221)
R> X_highdim <- replicate(50, rnorm(5000))
R> intrinsicDimension::essLocalDimEst(X_highdim)

Dimension estimate: 49.05083
Additional data: ess

R> summary(twonn(X_highdim))

Model: TWO-NN
Method: MLE
Sample size: 5000, Obs. used: 4950. Trimming proportion: 1%
ID estimates (confidence level: 0.95)

| Lower Bound| Estimate| Upper Bound|
|-----------:|--------:|-----------:|
|    34.92154| 35.90791|    36.92254|
```

However, the ESS algorithm is not uniformly optimal. For example, for the Swissroll data, `twonn()` performs better:

```
R> intrinsicDimension::essLocalDimEst(Swissroll)

Dimension estimate: 2.898866
Additional data: ess

R> summary(twonn(Swissroll, c_trimmed = 0.001))

Model: TWO-NN
Method: MLE
Sample size: 1000, Obs. used: 999. Trimming proportion: 0.1%
ID estimates (confidence level: 0.95)

| Lower Bound| Estimate| Upper Bound|
|-----------:|--------:|-----------:|
|    1.945607|  2.07005|    2.202571|
```

When dealing with a dataset characterized by many columns, we suggest checking the discrepancy between our methods and different competitors. A marked difference in the results should flag the likelihood-based findings as less reliable. At this point, a legitimate doubt that may arise regards the validity of the findings we obtained studying the Alon dataset. Because of its high number of columns (`D = 2000`), the ID we recovered may have been strongly underestimated. To validate our results, we run the ESS estimator on the Alon dataset, obtaining

```
R> intrinsicDimension::essLocalDimEst(data = Xalon)
```

```
Dimension estimate: 7.752803
Additional data: ess
```

which is very close to the estimates obtained with our methods, reassuring us about our conclusions.

C. Gibbs sampler for `Hidalgo()`

The steps of the Gibbs sampler are the following:

1. Sample the mixture weights according to

$$\boldsymbol{\pi}|\cdots \sim \textit{Dirichlet}\left(\alpha_1 + \sum_{i=1}^{n} \mathbb{1}_{\{z_i=1\}}, \ldots, \alpha_K + \sum_{i=1}^{n} \mathbb{1}_{\{z_i=K\}}\right).$$

2. Let $\boldsymbol{z}_{-i}$ denote the vector $\boldsymbol{z}$ without its i-th element. Sample the cluster indicators z_i according to:

$$\mathbb{P}\left(z_i = k|\boldsymbol{z}_{-i}, \cdots\right) \propto \pi_{z_i} f\left(\mu_i, \mathcal{N}_i^{(q)}|z_1, \ldots, z_{i-1}, k, z_{i+1}, \ldots, z_n, \boldsymbol{d}\right).$$

   We emphasize that, given the new likelihood we are considering, the cluster labels are no longer independent given all the other parameters. Let us define

$$\boldsymbol{z}_i^k = (z_1, \ldots, z_{i-1}, k, z_{i-1}, \ldots, z_n).$$

   Then, let $N_{z_i}(\boldsymbol{z}_{-i})$ be the number of elements in the $(n-1)$-dimensional vector $\boldsymbol{z}_{-i}$ that are assigned to the manifold (mixture component) indicated by the label z_i. Moreover, let $m_i^{in} = \sum_l \mathcal{N}_{l,i}^{(q)} \mathbb{1}_{\{z_l=z_i\}}$ be the number of points sampled from the same manifold of the i-th observation that have x_i as neighbor, and let $n_i^{in}(\boldsymbol{z}) = \sum_l \mathcal{N}_{i,l}^{(q)} \mathbb{1}_{\{z_l=z_i\}} \leq q$ be the number of neighbors of x_i sampled from the same manifold. Then, we can simplify the previous formula, obtaining the following full conditional:

$$\mathbb{P}\left(z_i = k|\boldsymbol{z}_{-i}, \cdots\right) \propto \frac{\pi_k d_k \mu_i^{-(d_k+1)}}{\mathcal{Z}\left(\zeta, N_k(\boldsymbol{z}_{-i})+1\right)} \times \left(\frac{\zeta}{1-\zeta}\right)^{n_i^{in}(\boldsymbol{z}_i^k)+m_i^{in}(\boldsymbol{z}_i^k)} \times \left(\frac{\mathcal{Z}\left(\zeta, N_k(\boldsymbol{z}_{-i})\right)}{\mathcal{Z}\left(\zeta, N_k(\boldsymbol{z}_{-i})+1\right)}\right)^{N_k(\boldsymbol{z}_{-i})}. \quad (10)$$

   See Facco (2017) for a detailed derivation of this result.

3. The posterior distribution for $\boldsymbol{d}$ depends on the prior specification we adopt:

   (a) If we assume a conjugate Gamma prior, we obtain

$$d_k|\cdots \sim \textit{Gamma}\left(a_0 + n_k, b_0 + \sum_{i:z_i=k} \log \mu_i\right),$$

   where $n_k = \sum_{i=1}^{n} \mathbb{1}_{\{z_i=k\}}$ is the number of observations assigned to the k-th group;

(b) If G_0 is assumed to be a truncated Gamma distribution on $(0, D)$, then

$$d_k|\cdots \sim Gamma\left(a_0 + n_k, b_0 + \sum_{i:z_i=k} \log \mu_i\right) \mathbb{1}(\cdot)_{(0,D)};$$

(c) Finally, let us define $a^* = a_0 + n_k$ and $b^* = b_0 + \sum_{i:z_i=k} \log \mu_i$. If G_0 is assumed to be a truncated Gamma with point mass at D, we obtain

$$d_k|\cdots \sim \frac{\hat{\rho}_1^*}{\hat{\rho}_1^* + \hat{\rho}_0^*} Gamma\,(a^*, b^*)\, \mathbb{1}(\cdot)_{(0,D)} + \frac{\hat{\rho}_0^*}{\hat{\rho}_1^* + \hat{\rho}_0^*}\, \delta_D(\cdot),$$

where $\hat{\rho}_1^* = \hat{\rho} \cdot (\mathcal{C}_{a^*,b^*,D}/\mathcal{C}_{a,b,D})$ and $\hat{\rho}_0^* = (1 - \hat{\rho}) \cdot D^{n_k} \cdot \exp\{-D \sum_{i:z_i=k} \log \mu_i\}$.

D. Postprocessing to address label-switching

The postprocessing procedure adopted for the raw chains fitted by `Hidalgo()` works as follows.

Recall that we are working with n observations and K mixture components. Let us consider an MCMC sample of length T, and denote a particular MCMC iteration with t, $t = 1, \ldots, T$. Let $z_i(t)$ indicate the cluster membership of observation i at the t-th iteration, with $i = 1, \ldots, n$. Similarly, $d_k(t)$ represents the value of the estimated ID in the k-th mixture component at the t-th iteration, where $k = 1, \ldots, K$.

We map the K chains of the parameters in $\boldsymbol{d}$ to each data point via the values of $\boldsymbol{z}$. That is, we construct n chains, one for each observation. At the t-th iteration, we will have $\{d_{z_i(t)}(t)\}_{i=1}^n$. In so doing, we obtain a collection of n chains that link every observation to its ID estimate. When the chains have been postprocessed, the local observation-specific ID can be estimated by the ergodic mean or median.

E. The variation of information metric

This section provides additional details regarding the variation of information (VI) distance between partitions, a quantity often employed to estimate optimal posterior clustering configurations.

First, let $|A|$ denote the cardinality of a generic set A. Then, consider two different partitions of n elements defined as $\rho_1 = \{S_1^1 \ldots, S_{p_1}^1\}$ and $\rho_2 = \{S_1^2 \ldots, S_{p_2}^2\}$. By definition, for $q = 1, 2$, we have that $S_i^q \cap S_j^q = \varnothing$ when $i \neq j$, that $|\rho_1| = p_1$, $|\rho_2| = p_2$, and that $\sum_{l=1}^{p_q} |S_l^q| = n$. Following the notation in Dahl, Johnson, and Müller (2022b), we define the individual entropy function as $H(\rho_q) = -\sum_{S \in \rho_q} |S|/n \log_2 (|S|/n)$ for $q = 1, 2$. Moreover, the joint entropy $H(\rho_1, \rho_2)$ and mutual information $I(\rho_1, \rho_2)$ are defined as

$$H(\rho_1, \rho_2) = -\sum_{S^1 \in \rho_1} \sum_{S^2 \in \rho_2} \frac{|S^1 \cap S^2|}{n} \log_2 \left(\frac{|S^1 \cap S^2|}{n}\right),$$
$$I(\rho_1, \rho_2) = H(\rho_1) + H(\rho_2) - H(\rho_1, \rho_2).$$

Given these quantities, Meilă (2007) and Vinh, Epps, and Bailey (2010) considered the variation of information as a distance between partitions:

$$\mathcal{L}_{VI}(\rho_1, \rho_2) = H(\rho_1) + H(\rho_2) - 2I(\rho_1, \rho_2) = -H(\rho_1) - H(\rho_2) + 2H(\rho_1, \rho_2). \tag{11}$$

Wade and Ghahramani (2018) used Equation 11 as a loss function to measure the discrepancy between the targeted posterior partition $\rho_1 = \rho$ and an estimated one $\rho_2 = \hat{\rho}$. By minimizing the posterior expectation of $\mathcal{L}_{VI}(\rho, \hat{\rho})$ w.r.t. $\hat{\rho}$, one obtains the optimal clustering solution:

$$\hat{\rho}^* = \arg\min_{\hat{\rho}} \mathbb{E}\left[\mathcal{L}_{VI}(\rho, \hat{\rho}) \mid \mathcal{D}\right],$$

where $\mathcal{D}$ denotes the data. Unfortunately, minimizing the previous quantity (or one of its variations) over the partition space is extremely challenging. Therefore, numerous authors focused on developing efficient algorithms for this task; see, for example, Wade and Ghahramani (2018); Rastelli and Friel (2018) and the review in Dahl *et al.* (2022b).

F. Global and local intrinsic dimensions

In Section 2.2, we introduced HIDALGO as a heterogeneous ID estimator. The Bayesian mixture segments the data into groups, each belonging to a specific manifold with a specific ID. However, to be more precise, we provide additional comments on the meaning of the word *heterogeneous*. In the ID literature, there is a clear distinction between *global* and *local* ID estimators.

On the one hand, methods in the former group estimate a single ID value for the whole dataset. A single measure for the complexity of the data is extremely useful, for example, as a starting point for dimensionality reduction techniques.

On the other hand, the latter group contains methods that attribute a specific ID to each data point. Such output is instrumental in monitoring the behavior of the ID across the entire dataset to detect significant differences in its topology. Moreover, local ID estimation has an impact when employed for subspace outlier detection, subspace clustering, or, more in general, other applications in which the ID can vary from location to location (Amsaleg, Chelly, Houle, Kawarabayashi, Radovanović, and Treeratanajaru 2019). Finally, we can recover a global ID value by aggregating local ID estimates.

Broadly speaking, HIDALGO can be seen as both, a global and local estimator. Its mixture formulation allows the segmentation of the observations in homogeneous and spatially related model-based clusters. Hence, we can see HIDALGO as an estimator for multiple global IDs (one for each manifold but not for each point). Nonetheless, we can take advantage of the Bayesian framework and the postprocessing procedure we propose to deal with the label-switching issue (see Appendix D for more details). Indeed, while solving for label-switching, our procedure delivers a valuable byproduct. By mapping the K mixture parameters $\{d_k\}_{k=1}^K$ into n different observation-specific chains, we can effortlessly obtain an ID value for each data point. For example, see Figure 11, where observation-specific posterior mean and median ID estimates are displayed.

G. System configuration

We obtained the results in this vignette by running our R code on a MacBook Pro with a 2.6 GHz 6-Core Intel Core i7 processor.

Affiliation:

Francesco Denti
Università Cattolica del Sacro Cuore
Largo Gemelli 1
Milan 20123, Italy
E-mail: francesco.denti@unicatt.it
URL: https://fradenti.github.io/

Journal of Statistical Software https://www.jstatsoft.org/
published by the Foundation for Open Access Statistics https://www.foastat.org/
April 2023, Volume 106, Issue 9 *Submitted:* 2022-01-16
doi:10.18637/jss.v106.i09 *Accepted:* 2023-01-22