dimRed and **coRanking**—Unifying Dimensionality Reduction in R

Guido Kraemer Markus Reichstein Miguel D. Mahecha

March 10, 2022

Abstract

This document is based on the manuscript of Kraemer et al. (2018) which was published in the R-Journal and has been modified and extended to fit the format of a package vignette and to match the extended functionality of the **dimRed** package.

"Dimensionality reduction" (DR) is a widely used approach to find low dimensional and interpretable representations of data that are natively embedded in high-dimensional spaces. DR can be realized by a plethora of methods with different properties, objectives, and, hence, (dis)advantages. The resulting low-dimensional data embeddings are often difficult to compare with objective criteria. Here, we introduce the **dimRed** and **coRanking** packages for the R language. These open source software packages enable users to easily access multiple classical and advanced DR methods using a common interface. The packages also provide quality indicators for the embeddings and easy visualization of high dimensional data. The **coRanking** package provides the functionality for assessing DR methods in the co-ranking matrix framework. In tandem, these packages allow for uncovering complex structures high dimensional data. Currently 15 DR methods are available in the package, some of which were not previously available to R users. Here, we outline the **dimRed** and **coRanking** packages and make the implemented methods understandable to the interested reader.

1 Introduction

Dimensionality Reduction (DR) essentially aims to find low dimensional representations of data while preserving their key properties. Many methods exist in literature, optimizing different criteria: maximizing the variance or the statistical independence of the projected data, minimizing the reconstruction error under different constraints, or optimizing for different error metrics, just to name a few. Choosing an inadequate method may imply that much of the underlying structure remains undiscovered. Often the structures of interest in a data set can be well represented by fewer dimensions than exist in the original data. Data compression of this kind has the additional benefit of making the encoded information better

conceivable to our brains for further analysis tasks like classification or regression problems.

For example, the morphology of a plant's leaves, stems, and seeds reflect the environmental conditions the species usually grow in (e.g., plants with large soft leaves will never grow in a desert but might have an advantage in a humid and shadowy environment). Because the morphology of the entire plant depends on the environment, many morphological combinations will never occur in nature and the morphological space of all plant species is tightly constrained. Díaz et al. (2016) found that out of six observed morphological characteristics only two embedding dimensions were enough to represent three quarters of the totally observed variability.

DR is a widely used approach for the detection of structure in multivariate data, and has applications in a variety of fields. In climatology, DR is used to find the modes of some phenomenon, e.g., the first Empirical Orthogonal Function of monthly mean sea surface temperature of a given region over the Pacific is often linked to the El Niño Southern Oscillation or ENSO (e.g., Hsieh, 2004). In ecology the comparison of sites with different species abundances is a classical multivariate problem: each observed species adds an extra dimension, and because species are often bound to certain habitats, there is a lot of redundant information. Using DR is a popular technique to represent the sites in few dimensions, e.g., Aart (1972) matches wolfspider communities to habitat and Morrall (1974) match soil fungi data to soil types. (In ecology the general name for DR is ordination or indirect gradient analysis.) Today, hyperspectral satellite imagery collects so many bands that it is very difficult to analyze and interpret the data directly. Resuming the data into a set of few, yet independent, components is one way to reduce complexity (e.g., see Laparra et al., 2015). DR can also be used to visualize the interiors of deep neural networks (e.g., see Han et al., 2017), where the high dimensionality comes from the large number of weights used in a neural network and convergence can be visualized by means of DR. We could find many more example applications here but this is not the main focus of this publication.

The difficulty in applying DR is that each DR method is designed to maintain certain aspects of the original data and therefore may be appropriate for one task and inappropriate for another. Most methods also have parameters to tune and follow different assumptions. The quality of the outcome may strongly depend on their tuning, which adds additional complexity. DR methods can be modeled after physical models with attracting and repelling forces (Force Directed Methods), projections onto low dimensional planes (PCA, ICA), divergence of statistical distributions (SNE family), or the reconstruction of local spaces or points by their neighbors (LLE).

As an example for how changing internal parameters of a method can have a great impact, the breakthrough for Stochastic Neighborhood Embedding (SNE) methods came when a Student's t-distribution was used instead of a normal distribution to model probabilities in low dimensional space to avoid the "crowding problem", that is, a sphere in high dimensional space has a much larger volume than in low dimensional space and may contain too many points to be represented accurately in few di-

mensions. The t-distribution, allows medium distances to be accurately represented in few dimensions by larger distances due to its heavier tails. The result is called in t-SNE and is especially good at preserving local structures in very few dimensions, this feature made t-SNE useful for a wide array of data visualization tasks and the method became much more popular than standard SNE (around six times more citations of van der Maaten and Hinton (2008) compared to Hinton and Roweis (2003) in Scopus (Elsevier, 2017)).

There are a number of software packages for other languages providing collections of methods: In Python there is scikit-learn (Pedregosa et al., 2011), which contains a module for DR. In Julia we currently find ManifoldLearning.jl for nonlinear and MultivariateStats.jl for linear DR methods. There are several toolboxes for DR implemented in Matlab (Van Der Maaten et al., 2009; Arenas-Garcia et al., 2013). The Shogun toolbox (Sonnenburg et al., 2017) implements a variety of methods for dimensionality reduction in C++ and offers bindings for a many common high level languages (including R, but the installation is anything but simple, as there is no CRAN package). However, there is no comprehensive package for R and none of the former mentioned software packages provides means to consistently compare the quality of different methods for DR.

For many applications it can be difficult to objectively find the right method or parameterization for the DR task. This paper presents the **dimRed** and **coRanking** packages for the popular programming language R. Together, they provide a standardized interface to various dimensionality reduction methods and quality metrics for embeddings. They are implemented using the S4 class system of R, making the packages both easy to use and to extend.

The design goal for these packages is to enable researchers, who may not necessarily be experts in DR, to apply the methods in their own work and to objectively identify the most suitable methods for their data. This paper provides an overview of the methods collected in the packages and contains examples as to how to use the packages.

The notation in this paper will be as follows: $X = [x_i]_{1\leq i\leq n}^T \in \mathbb{R}^{n\times p}$, and the observations $x_i \in \mathbb{R}^p$. These observations may be transformed prior to the dimensionality reduction step (e.g., centering and/or standardization) resulting in $X' = [x_i']_{1\leq i\leq n}^T \in \mathbb{R}^{n\times p}$. A DR method then embeds each vector in X' onto a vector in $Y = [y_i]_{1\leq i\leq n}^T \in \mathbb{R}^{n\times q}$ with $y_i \in \mathbb{R}^q$, ideally with $q \ll p$. Some methods provide an explicit mapping $f(x_i') = y_i$. Some even offer an inverse mapping $f^{-1}(y_i) = \hat{x}_i'$, such that one can reconstruct a (usually approximate) sample from the low-dimensional representation. For some methods, pairwise distances between points are needed, we set $d_{ij} = d(x_i, x_j)$ and $\hat{d}_{ij} = d(y_i, y_j)$, where d is some appropriate distance function.

When referring to `functions` in the **dimRed** package or base R simply the function name is mentioned, functions from other packages are referenced with their namespace, as with `package::function`.

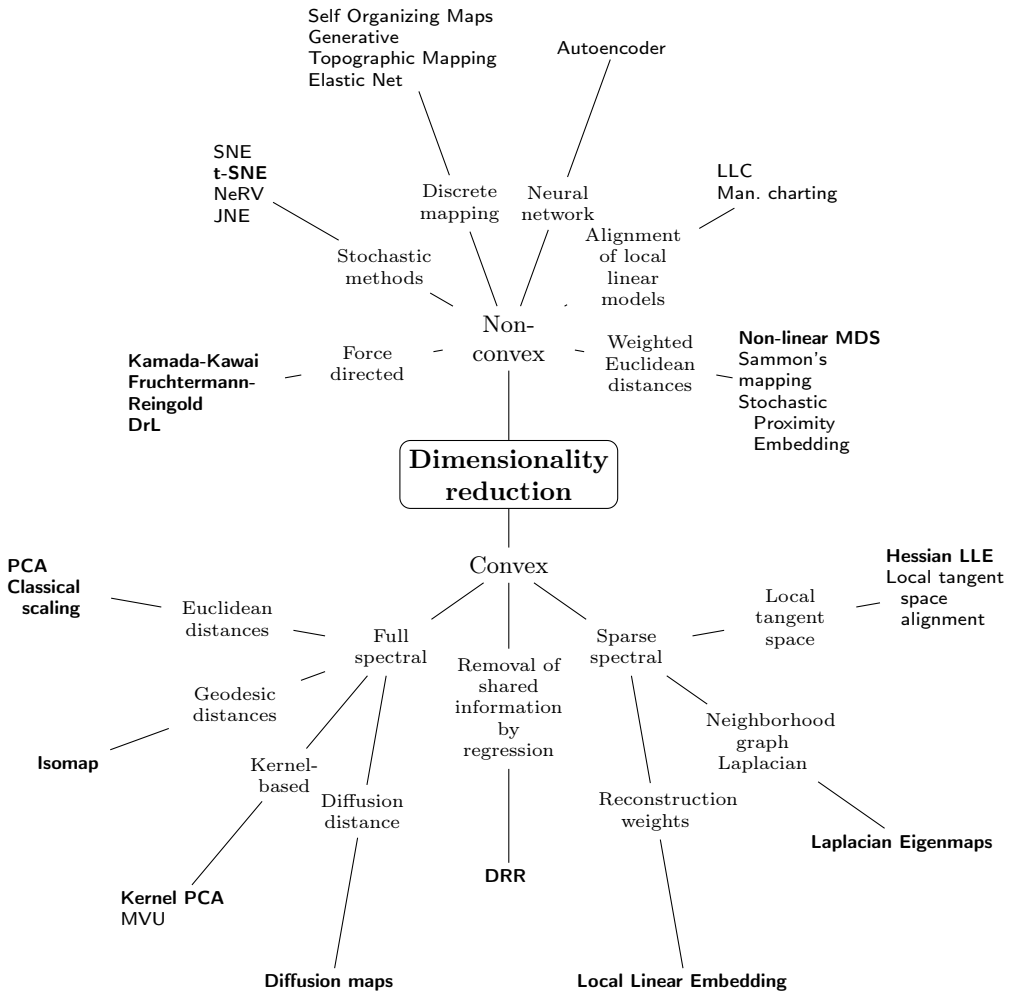

Figure 1: Classification of dimensionality reduction methods. Methods in bold face are implemented in **dimRed**. Modified from Van Der Maaten et al. (2009).

2 Dimensionality Reduction Methods

In the following section we do not aim for an exhaustive explanation to every method in **dimRed** but rather to provide a general idea on how the methods work. An overview and classification of the most commonly used DR methods can be found in Figure 1.

In all methods, parameters have to be optimized or decisions have to be made, even if it is just about the preprocessing steps of data. The **dimRed** package tries to make the optimization process for parameters as easy as possible, but, if possible, the parameter space should be narrowed down using prior knowledge. Often decisions can be made based on theoretical knowledge. For example, sometimes an analysis requires data to be kept in their original scales and sometimes this is exactly what has to be avoided as when comparing different physical units. Sometimes decisions based on the experience of others can be made, e.g., the Gaussian kernel is probably the most universal kernel and therefore should be tested first if there is a choice.

All methods presented here have the embedding dimensionality, q, as a parameter (or `ndim` as a parameter for `embed`). For methods based on eigenvector decomposition, the result generally does not depend on the number of dimensions, i.e., the first dimension will be the same, no matter if we decide to calculate only two dimensions or more. If more dimensions are added, more information is maintained, the first dimension is the most important and higher dimensions are successively less important. This means, that a method based on eigenvalue decomposition only has to be run once if one wishes to compare the embedding in different dimensions. In optimization based methods this is generally not the case, the number of dimensions has to be chosen a priori, an embedding of 2 and 3 dimensions may vary significantly, and there is no ordered importance of dimensions. This means that comparing dimensions of optimization-based methods is computationally much more expensive.

We try to give the computational complexity of the methods. Because of the actual implementation, computation times may differ largely. R is an interpreted language, so all parts of an algorithm that are implemented in R often will tend to be slow compared to methods that call efficient implementations in a compiled language. Methods where most of the computing time is spent for eigenvalue decomposition do have very efficient implementations as R uses optimized linear algebra libraries. Although, eigenvalue decomposition itself does not scale very well in naive implementations ($\mathcal{O}(n^3)$).

2.1 PCA

Principal Component Analysis (PCA) is the most basic technique for reducing dimensions. It dates back to Pearson (1901). PCA finds a linear projection (U) of the high dimensional space into a low dimensional space $Y = XU$, maintaining maximum variance of the data. It is based on solving the following eigenvalue problem:

$$(C_{XX} - \lambda_k I)u_k = 0 \tag{1}$$

where $C_{XX} = \frac{1}{n} X^T X$ is the covariance matrix, λ_k and u_k are the k-th eigenvalue and eigenvector, and I is the identity matrix. The equation has several solutions for different values of λ_k (leaving aside the trivial solution $u_k = 0$). PCA can be efficiently applied to large data sets, because it computationally scales as $\mathcal{O}(np^2 + p^3)$, that is, it scales linearly with the number of samples and R uses specialized linear algebra libraries for such kind of computations.

PCA is a rotation around the origin and there exist a forward and inverse mapping. PCA may suffer from a scale problem, i.e., when one variable dominates the variance simply because it is in a higher scale, to remedy this, the data can be scaled to zero mean and unit variance, depending on the use case, if this is necessary or desired.

Base R implements PCA in the functions `prcomp` and `princomp`; but several other implementations exist i.e., **pcaMethods** from Bioconductor which implements versions of PCA that can deal with missing data. The **dimRed** package wraps `prcomp`.

2.2 kPCA

Kernel Principal Component Analysis (kPCA) extends PCA to deal with nonlinear dependencies among variables. The idea behind kPCA is to map the data into a high dimensional space using a possibly non-linear function ϕ and then to perform a PCA in this high dimensional space. Some mathematical tricks are used for efficient computation.

If the columns of X are centered around 0, then the principal components can also be computed from the inner product matrix $K = X^T X$. Due to this way of calculating a PCA, we do not need to explicitly map all points into the high dimensional space and do the calculations there, it is enough to obtain the inner product matrix or kernel matrix $K \in \mathbb{R}^{n \times n}$ of the mapped points (Schölkopf et al., 1998).

Here is an example calculating the kernel matrix using a Gaussian kernel:

$$K = \phi(x_i)^T \phi(x_j) = \kappa(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right), \tag{2}$$

where σ is a length scale parameter accounting for the width of the kernel. The other trick used is known as the "representers theorem." The interested reader is referred to Schölkopf et al. (2001).

The kPCA method is very flexible and there exist many kernels for special purposes. The most common kernel function is the Gaussian kernel (Equation 2). The flexibility comes at the price that the method has to be finely tuned for the data set because some parameter combinations are simply unsuitable for certain data. The method is not suitable for very large data sets, because memory scales with $\mathcal{O}(n^2)$ and computation time with $\mathcal{O}(n^3)$.

Diffusion Maps, Isomap, Locally Linear Embedding, and some other techniques can be seen as special cases of kPCA. In which case, an out-of-sample extension using the Nyström formula can be applied (Bengio et al., 2004). This can also yield applications for bigger data, where an

embedding is trained with a sub-sample of all data and then the data is embedded using the Nyström formula.

Kernel PCA in R is implemented in the **kernlab** package using the function `kernlab::kpca`, and supports a number of kernels and user defined functions. For details see the help page for `kernlab::kpca`.

The **dimRed** package wraps `kernlab::kpca` but additionally provides forward and inverse methods (Bakir et al., 2004) which can be used to fit out-of-sample data or to visualize the transformation of the data space.

2.3 Classical Scaling

What today is called Classical Scaling was first introduced by Torgerson (1952). It uses an eigenvalue decomposition of a transformed distance matrix to find an embedding that maintains the distances of the distance matrix. The method works because of the same reason that kPCA works, i.e., classical scaling can be seen as a kPCA with kernel $x^T y$. A matrix of Euclidean distances can be transformed into an inner product matrix by some simple transformations and therefore yields the same result as a PCA. Classical scaling is conceptually more general than PCA in that arbitrary distance matrices can be used, i.e., the method does not even need the original coordinates, just a distance matrix D. Then it tries to find an embedding Y so that $\tilde{d}_{ij}$ is as similar to d_{ij} as possible.

The disadvantage is that it is computationally much more demanding, i.e., an eigenvalue decomposition of an $n \times n$ matrix has to be computed. This step requires $\mathcal{O}(n^2)$ memory and $\mathcal{O}(n^3)$ computation time, while PCA requires only the eigenvalue decomposition of a $d \times d$ matrix and usually $n \gg d$. R implements classical scaling in the `cmdscale` function.

The **dimRed** package wraps `cmdscale` and allows the specification of arbitrary distance functions for calculating the distance matrix. Additionally a forward method is implemented.

2.4 Isomap

As Classical Scaling can deal with arbitrarily defined distances, Tenenbaum et al. (2000) suggested to approximate the structure of the manifold by using geodesic distances. In practice, a graph is created by either keeping only the connections between every point and its k nearest neighbors to produce a k-nearest neighbor graph (k-NNG), or simply by keeping all distances smaller than a value ε producing an ε-neighborhood graph (ε-NNG). Geodesic distances are obtained by recording the distance on the graph and classical scaling is used to find an embedding in fewer dimensions. This leads to an "unfolding" of possibly convoluted structures (see Figure 3).

Isomap's computational cost is dominated by the eigenvalue decomposition and therefore scales with $\mathcal{O}(n^3)$. Other related techniques can use more efficient algorithms because the distance matrix becomes sparse due to a different preprocessing.

In R, Isomap is implemented in the **vegan** package. The `vegan::isomap` calculates an Isomap embedding and `vegan::isomapdist` calculates a

geodesic distance matrix. The **dimRed** package uses its own implementation. This implementation is faster mainly due to using a KD-tree for the nearest neighbor search (from the **RANN** package) and to a faster implementation for the shortest path search in the k-NNG (from the **igraph** package). The implementation in **dimRed** also includes a forward method that can be used to train the embedding on a subset of data points and then use these points to approximate an embedding for the remaining points. This technique is generally referred to as landmark Isomap (De Silva and Tenenbaum, 2004).

2.5 Locally Linear Embedding

Points that lie on a manifold in a high dimensional space can be reconstructed through linear combinations of their neighborhoods if the manifold is well sampled and the neighbohoods lie on a locally linear patch. These reconstruction weights, W, are the same in the high dimensional space as the internal coordinates of the manifold. Locally Linear Embedding (LLE; Roweis and Saul, 2000) is a technique that constructs a weight matrix $W \in \mathbb{R}^{n \times n}$ with elements w_{ij} so that

$$\sum_{i=1}^{n} \left\| x_i - \sum_{j=1}^{n} w_{ij} x_j \right\|^2 \tag{3}$$

is minimized under the constraint that $w_{ij} = 0$ if x_j does not belong to the neighborhood and the constraint that $\sum_{j=1}^{n} w_{ij} = 1$. Finally the embedding is made in such a way that the following cost function is minimized for Y,

$$\sum_{i=1}^{n} \left\| y_i - \sum_{j=1}^{n} w_{ij} y_j \right\|^2. \tag{4}$$

This can be solved using an eigenvalue decomposition.

Conceptually the method is similar to Isomap but it is computationally much nicer because the weight matrix is sparse and there exist efficient solvers. In R, LLE is implemented by the package **lle**, the embedding can be calculated with `lle::lle`. Unfortunately the implementation does not make use of the sparsity of the weight matrix W. The manifold must be well sampled and the neighborhood size must be chosen appropriately for LLE to give good results.

2.6 Laplacian Eigenmaps

Laplacian Eigenmaps were originally developed under the name spectral clustering to separate non-convex clusters. Later it was also used for graph embedding and DR (Belkin and Niyogi, 2003).

A number of variants have been proposed. First, a graph is constructed, usually from a distance matrix, the graph can be made sparse by keeping only the k nearest neighbors, or by specifying an ε neighborhood. Then, a similarity matrix W is calculated by using a Gaussian kernel (see Equation 2), if $c = 2\sigma^2 = \infty$, then all distances are treated equally, the smaller c the more emphasis is given to differences in distance.

The degree of vertex i is $d_i = \sum_{j=1}^{n} w_{ij}$ and the degree matrix, D, is the diagonal matrix with entries d_i. Then we can form the graph Laplacian $L = D - W$ and, then, there are several ways how to proceed, an overview can be found in Luxburg (2007).

The **dimRed** package implements the algorithm from Belkin and Niyogi (2003). Analogously to LLE, Laplacian eigenmaps avoid computational complexity by creating a sparse matrix and not having to estimate the distances between all pairs of points. Then the eigenvectors corresponding to the lowest eigenvalues larger than 0 of either the matrix L or the normalized Laplacian $D^{-1/2}LD^{-1/2}$ are computed and form the embedding.

2.7 Diffusion Maps

Diffusion Maps (Coifman and Lafon, 2006) take a distance matrix as input and calculates the transition probability matrix P of a diffusion process between the points to approximate the manifold. Then the embedding is done by an eigenvalue decompositon of P to calculate the coordinates of the embedding. The algorithm for calculating Diffusion Maps shares some elements with the way Laplacian Eigenmaps are calculated. Both algorithms depart from the same weight matrix, Diffusion Maps calculate the transition probability on the graph after t time steps and do the embedding on this probability matrix.

The idea is to simulate a diffusion process between the nodes of the graph, which is more robust to short-circuiting than the k-NNG from Isomap (see bottom right Figure 3). Diffusion maps in R are accessible via the `diffusionMap::diffuse()` function, which is available in the **diffusionMap** package. Additional points can be approximated into an existing embedding using the Nyström formula (Bengio et al., 2004). The implementation in **dimRed** is based on the `diffusionMap::diffuse` function.

2.8 non-Metric Dimensional Scaling

While Classical Scaling and derived methods (see section Classical Scaling) use eigenvector decomposition to embed the data in such a way that the given distances are maintained, non-Metric Dimensional Scaling (nMDS, Kruskal, 1964a,b) uses optimization methods to reach the same goal. Therefore a stress function,

$$S = \sqrt{\frac{\sum_{i<j} \left(d_{ij} - \hat{d}_{ij}\right)^2}{\sum_{i<j} d_{ij}^2}}, \tag{5}$$

is used, and the algorithm tries to embed y_i in such a way that the order of the d_{ij} is the same as the order of the $\hat{d}_{ij}$ Because optimization methods can fit a wide variety of problems, there are very loose limits set to the form of the error or stress function. For instance Mahecha et al. (2007) found that nMDS using geodesic distances can be almost as powerful as Isomap for embedding biodiversity patterns. Because of the flexibility of

nMDS, there is a whole package in R devoted to Multidimensional Scaling, `smacof` (de Leeuw and Mair, 2009).

Several packages provide implementations for nMDS in R, for example **MASS** and **vegan** with the functions `MASS::isoMDS` and `vegan::monoMDS`. Related methods include Sammons Mapping which con be found as `MASS::sammon`. The **dimRed** package wraps `vegan::monoMDS`.

2.9 Force Directed Methods

The data X can be considered as a graph with weighted edges, where the weights are the distances between points. Force directed algorithms see the edges of the graphs as springs or the result of an electric charge of the nodes that result in an attractive or repulsive force between the nodes, the algorithms then try to minimize the overall energy of the graph.

$$E = \sum_{i<j} k_{ij}(d_{ij} - \hat{d}_{ij})^2, \tag{6}$$

where k_{ij} is the spring constant for the spring connecting points i and j.

Graph embedding algorithms generally suffer from long running times (though compared to other methods presented here they do not scale as badly) and many local optima. This is why a number of methods have been developed that try to deal with some of the shortcomings, for example, the Kamada-Kawai (Kamada and Kawai, 1989), the Fruchtermann-Reingold (Fruchterman and Reingold, 1991), or the DrL (Martin et al., 2007) algorithms.

There are a number of graph embedding algorithms included in the **igraph** package, they can be accessed using the `igraph::layout_with_*` function family. The **dimRed** package only wraps the three algorithms mentioned above; there are many others which are not interesting for dimensionality reduction.

2.10 t-SNE

Stochastic Neighbor Embedding (SNE; Hinton and Roweis, 2003) is a technique that minimizes the Kullback-Leibler divergence of scaled similarities of the points i and j in a high dimensional space, p_{ij}, and a low dimensional space, q_{ij}:

$$KL(P\|Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}. \tag{7}$$

SNE uses a Gaussian kernel (see Equation 2) to compute similarities in a high and a low dimensional space. The t-Distributed Stochastic Neighborhood Embedding (t-SNE; van der Maaten and Hinton, 2008) improves on SNE by using a t-Distribution as a kernel in low dimensional space. Because of the heavy-tailed t-distribution, t-SNE maintains local neighborhoods of the data better and penalizes wrong embeddings of dissimilar points. This property makes it especially suitable to represent clustered data and complex structures in few dimensions.

The t-SNE method has one parameter, perplexity, to tune. This determines the neighborhood size of the kernels used.

The general runtime of t-SNE is $\mathcal{O}(n^2)$, but an efficient implementation using tree search algorithms that scales as $\mathcal{O}(n \log n)$ exists and can be found in the **Rtsne** package in R. The t-SNE implementation in **dimRed** wraps the **Rtsne** package.

There exist a number of derived techniques for dimensionality reduction, e.g., NeRV (Venna et al., 2010) and JNE (Lee et al., 2013), that improve results but for which there do not yet exist packages on CRAN implementing them.

2.11 ICA

Independent Component Analysis (ICA) interprets the data X as a mixture of independent signals, e.g., a number of sound sources recorded by several microphones, and tries to "un-mix" them to find the original signals in the recorded signals. ICA is a linear rotation of the data, just as PCA, but instead of recovering the maximum variance, it recovers statistically independent components. A signal matrix S and a mixing matrix A are estimated so that $X = AS$.

There are a number of algorithms for ICA, the most widely used is fastICA (Hyvarinen, 1999) because it provides a fast and robust way to estimate A and S. FastICA maximizes a measure for non-Gaussianity called negentropy J (Comon, 1994). This is equivalent to minimizing mutual information between the resulting components. Negentropy J is defined as follows:

$$H(u) = -\int f(u) \log f(Y) \, \mathrm{d}u, \tag{8}$$

$$J(u) = H(u_{\text{gauss}}) - H(u), \tag{9}$$

where $u = (u_1, \ldots, u_n)^T$ is a random vector with density $f(\cdot)$ and u_{gauss} is a Gaussian random variable with the same covariance structure as u. FastICA uses a very efficient approximation to calculate negentropy. Because ICA can be translated into a simple linear projection, a forward and an inverse method can be supplied.

There are a number of packages in R that implement algorithms for ICA, the **dimRed** package wraps the `fastICA::fastICA()` function from **fastICA**.

2.12 DRR

Dimensionality Reduction via Regression is a very recent technique extending PCA (Laparra et al., 2015). Starting from a rotated (PCA) solution $X' = XU$, it predicts redundant information from the remaining components using non-linear regression.

$$y_{\cdot i} = x'_{\cdot i} - f_i(x'_{\cdot 1}, x'_{\cdot 2}, \ldots, x'_{\cdot i-1}) \tag{10}$$

with $x_{\cdot i}$ and $y_{\cdot i}$ being the loading of observations on the i-th axis. In theory, any kind of regression can be used. the authors of the original paper choose Kernel Ridge Regression (KRR; Saunders et al., 1998) because it is

a flexible nonlinear regression technique and computational optimizations for a fast calculation exist. DRR has another advantage over other techniques presented here, because it provides an exact forward and inverse function.

The usage of KRR also has the advantage of making the method convex, here we list it under non-convex methods, because other types of regression may make it non-convex.

Mathematically, functions are limited to map one input to a single output point. Therefore, DRR reduces to PCA if manifolds are too complex; but it seems very useful for slightly curved manifolds. The initial rotation is important, because the result strongly depends on the order of dimensions in high dimensional space.

DRR is implemented in the package **DRR**. The package provides forward and inverse functions which can be used to train on a subset.

3 Quality criteria

The advantage of unsupervised learning is that one does not need to specify classes or a target variable for the data under scrutiny. Instead the chosen algorithm arranges the input data. For example, arranged into clusters or into a lower dimensional representation. In contrast to a supervised problem, there is no natural way to directly measure the quality of any output or to compare two methods by an objective measure like for instance modeling efficiency or classification error. The reason is that every method optimizes a different error function, and it would be unfair to compare t-SNE and PCA by means of either recovered variance or KL-Divergence. One fair measure would be the reconstruction error, i.e., reconstructing the original data from a limited number of dimensions, but as discussed above not many methods provide forward and inverse mappings.

However, there are a series of independent estimators on the quality of a low-dimensional embedding. The **dimRed** package provides a number of quality measures which have been proposed in the literature to measure performance of dimensionality reduction techniques.

3.1 Co-ranking matrix based measures

The co-ranking matrix (Lee and Verleysen, 2009) is a way to capture the changes in ordinal distance. As before, let $d_{ij} = d(x_i, x_j)$ be the distances between x_i and x_j, i.e., in high dimensional space and $\hat{d}_{ij} = d(y_i, y_j)$ the distances in low dimensional space, then we can define the rank of y_j with respect to y_i

$$\hat{r}_{ij} = |\{k : \hat{d}_{ik} < \hat{d}_{ij} \text{ or } (\hat{d}_{ik} = \hat{d}_{ij} \text{ and } 1 \leq k < j \leq n)\}|, \tag{11}$$

and, analogously, the rank in high-dimensional space as:

$$r_{ij} = |\{k : d_{ik} < d_{ij} \text{ or } (d_{ik} = d_{ij} \text{ and } 1 \leq k < j \leq n)\}|, \tag{12}$$

where the notation $|A|$ denotes the number of elements in a set A. This means that we simply replace the distances in a distance matrix column

wise by their ranks. Therefore r_{ij} is an integer which indicates that x_i is the r_{ij}-th closest neighbor of x_j in the set X.

The co-ranking matrix Q then has elements

$$q_{kl} = |\{(i,j) : \hat{r}_{ij} = k \text{ and } r_{ij} = l\}|, \tag{13}$$

which is the 2d-histogram of the ranks. That is, q_{ij} is an integer which counts how many points of distance rank j became rank i. In a perfect DR, this matrix will only have non-zero entries in the diagonal; if most of the non-zero entries are in the lower triangle, then the DR collapsed far away points onto each other; if most of the non-zero entries are in the upper triangle, then the DR teared close points apart. For a detailed description of the properties of the co-ranking matrix the reader is referred to Lueks et al. (2011).

The co-ranking matrix can be computed using function `coRanking::coranking()` and can be visualized using `coRanking::imageplot()`. A good embedding should scatter the values around the diagonal of the matrix. If the values are predominantly in the lower triangle, then the embedding collapses the original structure causing far away points to be much closer; if the values are predominantly in the upper triangle the points from the original structure are torn apart. Nevertheless this method requires visual inspection of the matrix. For an automated assessment of quality, a scalar value that assigns a quality to an embedding is needed.

A number of metrics can be computed from the co-ranking matrix. For example:

$$Q_{NX}(k) = \frac{1}{kn} \sum_{i=1}^{k} \sum_{j=1}^{k} q_{ij}, \tag{14}$$

which is the number of points that belong to the k-th nearest neighbors in both high- and low-dimensional space, normalized to give a maximum of 1 (Lee and Verleysen, 2009). This quantity can be adjusted for random embeddings, giving the Local Continuity Meta Criterion (Chen and Buja, 2009):

$$\text{LCMC}(k) = Q_{NX}(k) - \frac{k}{n-1} \tag{15}$$

The above measures still depend on k, but LCMC has a well defined maximum at $k_{\max}$. Two measures without parameters are then defined:

$$Q_{\text{local}} = \frac{1}{k_{\max}} \sum_{k=1}^{k_{\max}} Q_{NX}(k) \text{ and} \tag{16}$$

$$Q_{\text{global}} = \frac{1}{n - k_{\max}} \sum_{k=k_{\max}}^{n-1} Q_{NX}(k). \tag{17}$$

These measure the preservation of local and global distances respectively. The original authors advised using Q_{local} over Q_{global}, but this depends on the application.

LCMC(k) can be normalized to a maximum of 1, yielding the following measure for a quality embedding (Lee et al., 2013):

$$R_{NX}(k) = \frac{(n-1)Q_{NX}(k) - k}{n-1-k}, \tag{18}$$

where a value of 0 corresponds to a random embedding and a value of 1 to a perfect embedding into the k-ary neighborhood. To transform $R_{NX}(k)$ into a parameterless measure, the area under the curve can be used:

$$\mathrm{AUC}_{\ln k}\left(R_{NX}(k)\right) = \left(\sum_{k=1}^{n-2} R_{NX}(k)\right) \Bigg/ \left(\sum_{k=1}^{n-2} 1/k\right). \tag{19}$$

This measure is normalized to one and takes k at a log-scale. Therefore it prefers methods that preserve local distances.

In R, the co-ranking matrix can be calculated using the the `coRanking::coranking` function. The **dimRed** package contains the functions `Q_local`, `Q_global`, `Q_NX`, `LCMC`, and `R_NX` to calculate the above quality measures in addition to `AUC_lnK_R_NX`.

Calculating the co-ranking matrix is a relatively expensive operation because it requires sorting every row of the distance matrix twice. It therefore scales with $\mathcal{O}(n^2 \log n)$. There is also a plotting function `plot_R_NX`, which plots the R_{NX} values with log-scaled K and adds the $AUC_{\ln K}$ to the legend (see Figure 2).

There are a number of other measures that can be computed from a co-ranking matrix, e.g., see Lueks et al. (2011); Lee and Verleysen (2009), or Babaee et al. (2013).

3.2 Cophenetic correlation

An old measure originally developed to compare clustering methods in the field of phylogenetics is cophenetic correlation (Sokal and Rohlf, 1962). This method consists simply of the correlation between the upper or lower triangles of the distance matrices (in dendrograms they are called cophenetic matrices, hence the name) in a high and low dimensional space. Additionally the distance measure and correlation method can be varied. In the **dimRed** package this is implemented in the `cophenetic_correlation` function.

Some studies use a measure called "residual variance" (Tenenbaum et al., 2000; Mahecha et al., 2007), which is defined as

$$1 - r^2(D, \hat{D}),$$

where r is the Pearson correlation and D, $\hat{D}$ are the distances matrices consisting of elements d_{ij} and $\hat{d}_{ij}$ respectively.

3.3 Reconstruction error

The fairest and most common way to assess the quality of a dimensionality reduction when the method provides an inverse mapping is the reconstruction error. The **dimRed** package includes a function to calculate the root

mean squared error which is defined as:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} d(x'_i, x_i)^2} \tag{20}$$

with $x'_i = f^{-1}(y_i)$, f^{-1} being the function that maps an embedded value back to feature space.

The **dimRed** package provides the `reconstruction_rmse` and `reconstruction_error` functions.

4 Test data sets

There are a number of test data sets that are often used to showcase a dimensionality reduction technique. Common ones being the 3d S-curve and the Swiss roll, among others. These data sets have in common that they usually have three dimensions, and well defined manifolds. Real world examples usually have more dimensions and often are much noisier, the manifolds may not be well sampled and exhibit holes and large pieces may be missing. Additionally, we cannot be sure if we can observe all the relevant variables.

The **dimRed** package implements a number of test datasets that are being used in literature to benchmark methods with the function `dimRed::loadDataSet()`. For artificial datasets the number of points and the noise level can be adjusted, the function also returns the internal coordinates.

5 The dimRed Package

The **dimRed** package collects DR methods readily implemented in R, implements missing methods and offers means to compare the quality of embeddings. The package is open source and available under the GPL3 license. Released versions of the package are available through CRAN (`https://cran.r-project.org/package=dimRed`) and development versions are hosted on GitHub (`https://github.com/gdkrmr/dimRed`). The **dimRed** package provides a common interface and convenience functions for a variety of different DR methods so that it is made easier to use and compare different methods. An overview of the packages main functions can be found in Table 1.

Internally, the package uses S4 classes but for normal usage the user does not need to have any knowledge on the inner workings of the S4 class system in R (cf. table 2). The package contains simple conversion functions from and to standard R-objects like a data.frame or a matrix. The `"dimRedData"` class provides a container for the data to be processed. The slot `data` contains a matrix with dimensions in columns and observations in rows, the slot `meta` may contain a data frame with additional information, e.g., categories or other information of the data points.

Each embedding method is a class which inherits from `"dimRedMethod"` which means that it contains a function to generate `"dimRedResult"` objects and a list of standard parameters. The class `"dimRedResult"` contains the data in reduced dimensions, the original meta information along

Function	Description
`embed`	Embed data using a DR method.
`quality`	Calculate a quality score from the result of `embed`.
`plot`	Plot a `"dimRedData"` or `"dimRedResult"` object, colors the points automatically, for exploring the data.
`plot_R_NX`	Compares the quality of various embeddings.
`dimRedMethodList`	Returns a character vector that contains all implemented DR methods.
`dimRedQualityList`	Returns a character vector that contains all implemented quality measures.

Table 1: The main interface functions of the **dimRed** package.

Class Name	Function
`"dimRedData"`	Holds the data for a DR. Fed to `embed()`. An `as.dimRedData()` methods exists for `"data.frame"`, `"matrix"`, and `"formula"` exist.
`"dimRedMethod"`	Virtual class, ancestor of all DR methods.
`"dimRedResult"`	The result of `embed()`, the embedded data.

Table 2: The S4 classes used in the **dimRed** package.

with the original data, and, if possible, functions for the forward and inverse mapping.

From a user-perspective the central function of the package is `embed` which is called in the form `embed(data,method,...)`, `data` can take standard R objects such as instances of `"data.frame"`, `"matrix"`, or `"formula"`, as input. The `method` is given as a character vector. All available methods can be listed by calling '`dimRedMethodList()`'. Method-specific parameters can be passed through `...`; when no method-specific parameters are given, defaults are chosen. The `embed` function returns an object of class `"dimRedResult"`.

For comparing different embeddings, **dimRed** contains the function `quality` which relies on the output of `embed` and a method name. This function returns a scalar quality score; a vector that contains the names of all quality functions is returned by calling '`dimRedQualityList()`'.

For easy visual examination, the package contains `plot` methods for `"dimRedData"` and `"dimRedResult"` objects in order to plot high dimensional data using parallel plots and pairwise scatter plots. Automatic coloring of data points is done using the available metadata.

6 Examples

The comparison of different DR methods, choosing the right parameters for a method, and the inspection of the results is simplified by **dimRed**. This section contains a number of examples to highlight the usage of the package.

To compare methods of dimensionality reduction, first a test data set is loaded using `loadDataSet`, then the `embed` function is used for DR (`embed` can also handle standard R types like `matrix` and `data.frame`). This makes it very simple to apply different methods of DR to the same data e.g., by defining a character vector of method names and then iterating over these, say with `lapply`. For inspection, **dimRed** provides methods for the `plot` function to visualize the resulting embedding (Figure 2 b and d), internal coordinates of the manifold are represented by color gradients. To visualize how well embeddings represent different neighborhood sizes, the function `plot_R_NX` is used on a list of embedding results (Figure 2 c). The plots in figure 2 are produced by the following code:

```
## define which methods to apply
embed_methods <- c("Isomap", "PCA")
## load test data set
data_set <- loadDataSet("3D S Curve", n = 1000)
## apply dimensionality reduction
data_emb <- lapply(embed_methods, function(x) embed(data_set, x))
names(data_emb) <- embed_methods
## figure \ref{fig:plotexample}a, the data set
plot(data_set, type = "3vars")
## figures \ref{fig:plotexample}b (Isomap) and \ref{fig:plotexample}d (PCA)
lapply(data_emb, plot, type = "2vars")
## figure \ref{fig:plotexample}c, quality analysis
plot_R_NX(data_emb)
```

The function `plot_R_NX` produces a figure that plots the neighborhood size (k at a log-scale) against the quality measure $\mathrm{R}_{NX}(k)$ (see Equation 18). This gives an overview of the general behavior of methods: if R_{NX} is high for low values of K, then local neighborhoods are maintained well; if R_{NX} is high for large values of K, then global gradients are maintained well. It also provides a way to directly compare methods by plotting more than one R_{NX} curve and an overall quality of the embedding by taking the area under the curve as an indicator for the overall quality of the embedding (see fig 19) which is shown as a number in the legend.

Therefore we can see from Figure 2c that t-SNE is very good a maintaining close and medium distances for the given data set, whereas PCA is only better at maintaining the very large distances. The large distances are dominated by the overall bent shape of the S in 3D space, while the close distances are not affected by this bending. This is reflected in the properties recovered by the different methods, the PCA embedding recovers the S-shape, while t-SNE ignores the S-shape and recovers the inner structure of the manifold.

Often the quality of an embedding strongly depends on the choice of parameters, the interface of **dimRed** can be used to facilitate searching

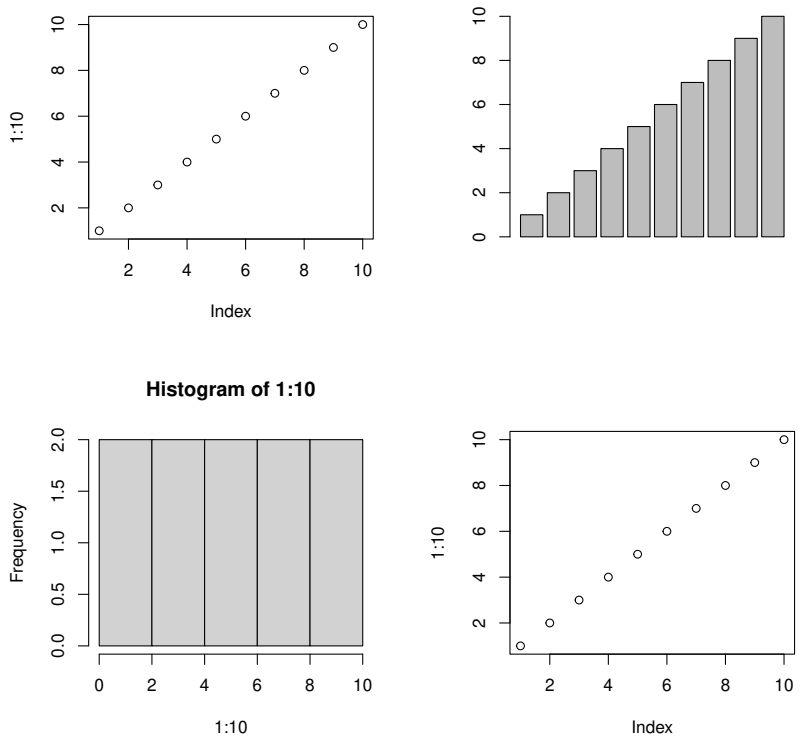

Figure 2: Comparing PCA and Isomap: (a) An S-shaped manifold, colors represent the internal coordinates of the manifold. (b) Isomap embedding, the S-shaped manifold is unfolded. (c) R_{NX} plotted agains neighborhood sizes, Isomap is much better at preserving local distances and PCA is better at preserving global Euclidean distances. The numbers on the legend are the $\mathrm{AUC}_{1/K}$. (d) PCA projection of the data, the directions of maximum variance are preserved.

the parameter space.

Isomap has one parameter k which determines the number of neighbors used to construct the k-NNG. If this number is too large, then Isomap will resemble an MDS (Figure 3 e), if the number is too small, the resulting embedding contains holes (Figure 3 c). The following code finds the optimal value, $k_{\max}$, for k using the Q_{local} criterion, the results are visualized in Figure 3 a:

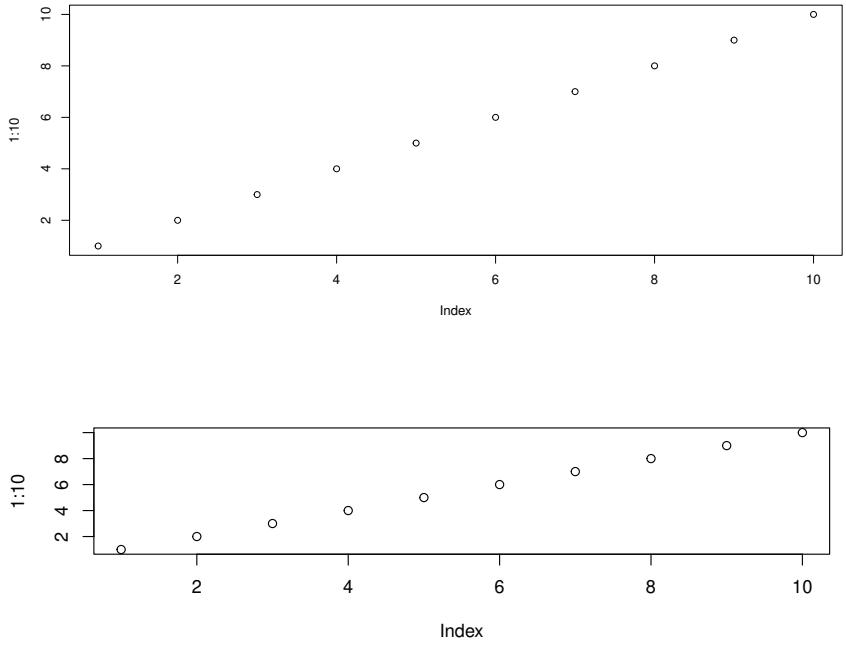

Figure 3: Using **dimRed** and the Q_{local} indicator to estimate a good value for the parameter k in Isomap. (a) Q_{local} for different values of k, the vertical red line indicates the maximum $k_{\max}$. (b) The original data set, a 2 dimensional manifold bent in an S-shape in 3 dimensional space. Bottom row: Embeddings and k-NNG for different values of k. (c) When $k = 5$, the value for k is too small resulting in holes in the embedding, the manifold itself is still unfolded correctly. (d) Choose $k = k_{\max}$, the best representation of the original manifold in two dimensions achievable with Isomap. (e) $k = 100$, too large, the k-NNG does not approximate the manifold any more.

```
## Load data
ss <- loadDataSet("3D S Curve", n = 500)
## Parameter space
kk <- floor(seq(5, 100, length.out = 40))
```

```
## Embedding over parameter space
emb <- lapply(kk, function(x) embed(ss, "Isomap", knn = x))
## Quality over embeddings
qual <- sapply(emb, function(x) quality(x, "Q_local"))
## Find best value for K
ind_max <- which.max(qual)
k_max <- kk[ind_max]
```

Figure 3a shows how the Q_{local} criterion changes when varying the neighborhood size k for Isomap, the gray lines in Figure 3 represent the edges of the k-NN Graph. If the value for k is too low, the inner structure of the manifold will still be recovered, but it will be imperfect (Figure 3c, note that the holes appear in places that are not covered by the edges of the k-NN Graph), therefore the Q_{local} score is lower than optimal. If k is too large, the error of the embedding is much larger due to short circuiting and we observe a very steep drop in the Q_{local} score. The short circuiting can be observed in Figure 3e with the edges that cross the gap between the tips and the center of the S-shape.

It is also very easy to compare across methods and quality scores. The following code produces a matrix of quality scores and methods, where `dimRedMethodList` returns a character vector with all methods. A visualization of the matrix can be found in Figure 4.

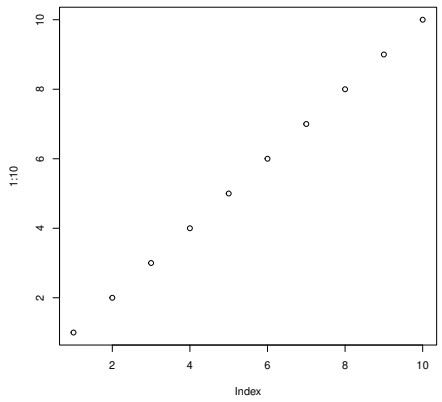

Figure 4: A visualization of the `quality_results` matrix. The methods are ordered by mean quality score. The reconstruction error was omitted, because a higher value means a worse embedding, while in the present methods a higher score means a better embedding. Parameters were not tuned for the example, therefore it should not be seen as a general quality assessment of the methods.

```
embed_methods <- dimRedMethodList()
quality_methods <- c("Q_local", "Q_global", "AUC_lnK_R_NX",
                     "cophenetic_correlation")
```

```
scurve <- loadDataSet("3D S Curve", n = 2000)
quality_results <- matrix(
  NA, length(embed_methods), length(quality_methods),
  dimnames = list(embed_methods, quality_methods)
)

embedded_data <- list()
for (e in embed_methods) {
embedded_data[[e]] <- embed(scurve, e)
for (q in quality_methods)
  try(quality_results[e, q] <- quality(embedded_data[[e]], q))
}
```

This example showcases the simplicity with which different methods and quality criteria can be combined. Because of the strong dependencies on parameters it is not advised to apply this kind of analysis without tuning the parameters for each method separately. There is no automatized way to tune parameters in **dimRed**.

7 Conclusion

This paper presents the **dimRed** and **coRanking** packages and it provides a brief overview of the methods implemented therein. The **dimRed** package is written in the R language, one of the most popular languages for data analysis. The package is freely available from CRAN. The package is object oriented and completely open source and therefore easily available and extensible. Although most of the DR methods already had implementations in R, **dimRed** adds some new methods for dimensionality reduction, and **coRanking** adds methods for an independent quality control of DR methods to the R ecosystem. DR is a widely used technique. However, due to the lack of easily usable tools, choosing the right method for DR is complex and depends upon a variety of factors. The **dimRed** package aims to facilitate experimentation with different techniques, parameters, and quality measures so that choosing the right method becomes easier. The **dimRed** package wants to enable the user to objectively compare methods that rely on very different algorithmic approaches. It makes the life of the programmer easier, because all methods are aggregated in one place and there is a single interface and standardized classes to access the functionality.

8 Acknowledgments

We thank Dr. G. Camps-Valls and an anonymous reviewer for many useful comments. This study was supported by the European Space Agency (ESA) via the Earth System Data Lab project (`http://earthsystemdatacube.org`) and the EU via the H2020 project BACI, grant agreement No 640176.

References

P. J. M. V. D. Aart. Distribution Analysis of Wolfspiders (Araneae, Lycosidae) in a Dune Area By Means of Principal Component Analysis. *Netherlands Journal of Zoology*, 23 (3):266–329, Jan. 1972. ISSN 1568-542X. doi: 10.1163/002829673X00076. URL `http://booksandjournals.brillonline.com/content/journals/10.1163/002829673x00076`.

J. Arenas-Garcia, K. B. Petersen, G. Camps-Valls, and L. K. Hansen. Kernel Multivariate Analysis Framework for Supervised Subspace Learning: A Tutorial on Linear and Kernel Multivariate Methods. *IEEE Signal Processing Magazine*, 30(4):16–29, July 2013. ISSN 1053-5888. doi: 10.1109/MSP.2013.2250591.

M. Babaee, M. Datcu, and G. Rigoll. Assessment of dimensionality reduction based on communication channel model; application to immersive information visualization. In *Big Data 2013*, pages 1–6. IEEE Xplore, 2013. doi: 10.1109/BigData.2013.6691726. URL `http://elib.dlr.de/88828/`.

G. H. Bakir, J. Weston, and P. B. Schölkopf. Learning to Find Pre-Images. In S. Thrun, L. K. Saul, and P. B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 449–456. MIT Press, 2004. doi: 10.1007/978-3-540-28649-3_31. URL `http://papers.nips.cc/paper/2417-learning-to-find-pre-images.pdf`.

M. Belkin and P. Niyogi. Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. *Neural Computation*, 15(6):1373, June 2003. ISSN 08997667. doi: 10.1162/089976603321780317.

Y. Bengio, O. Delalleau, N. L. Roux, J.-F. Paiement, P. Vincent, and M. Ouimet. Learning Eigenfunctions Links Spectral Embedding and Kernel PCA. *Neural Computation*, 16(10):2197–2219, Oct. 2004. ISSN 0899-7667. doi: 10.1162/0899766041732396. URL `http://dx.doi.org/10.1162/0899766041732396`.

L. Chen and A. Buja. Local multidimensional scaling for nonlinear dimension reduction, graph drawing, and proximity analysis. *Journal of the American Statistical Association*, 104(485):209–219, 2009. doi: 10.1198/jasa.2009.0111. URL `https://doi.org/10.1198/jasa.2009.0111`.

R. R. Coifman and S. Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21(1):5–30, July 2006. ISSN 10635203. doi: 10.1016/j.acha.2006.04.006. URL `http://linkinghub.elsevier.com/retrieve/pii/S1063520306000546`.

P. Comon. Independent component analysis, A new concept? *Signal Processing*, 36(3):287–314, Apr. 1994. ISSN 01651684. doi: 10.1016/0165-1684(94)90029-9. URL `http://linkinghub.elsevier.com/retrieve/pii/0165168494900299`.

J. de Leeuw and P. Mair. Multidimensional Scaling Using Majorization: SMACOF in R. *JOURNAL OF STATISTICAL SOFTWARE*, 31(3): 1–30, AUG 2009. ISSN 1548-7660.

V. De Silva and J. B. Tenenbaum. Sparse multidimensional scaling using landmark points. Technical report, 2004.

S. Díaz, J. Kattge, J. H. C. Cornelissen, I. J. Wright, S. Lavorel, S. Dray, B. Reu, M. Kleyer, C. Wirth, I. Colin Prentice, E. Garnier, G. Bönisch, M. Westoby, H. Poorter, P. B. Reich, A. T. Moles, J. Dickie, A. N. Gillison, A. E. Zanne, J. Chave, S. Joseph Wright, S. N. Sheremet'ev, H. Jactel, C. Baraloto, B. Cerabolini, S. Pierce, B. Shipley, D. Kirkup, F. Casanoves, J. S. Joswig, A. Günther, V. Falczuk, N. Rüger, M. D. Mahecha, and L. D. Gorné. The global spectrum of plant form and function. *Nature*, 529(7585):167–171, Jan. 2016. ISSN 0028-0836. doi: 10.1038/nature16489. URL `http://www.nature.com/nature/journal/v529/n7585/full/nature16489.html`.

Elsevier. Scopus - Advanced search, 2017. URL `https://www.scopus.com/`.

T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Softw: Pract. Exper.*, 21(11):1129–1164, Nov. 1991. ISSN 1097-024X. doi: 10.1002/spe.4380211102.

Y. Han, J. Kim, and K. Lee. Deep Convolutional Neural Networks for Predominant Instrument Recognition in Polyphonic Music. *IEEE-ACM TRANSACTIONS ON AUDIO SPEECH AND LANGUAGE PROCESSING*, 25(1):208–221, JAN 2017. ISSN 2329-9290. doi: {10.1109/TASLP.2016.2632307}.

G. E. Hinton and S. T. Roweis. Stochastic Neighbor Embedding. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 857–864. MIT Press, 2003. URL `http://papers.nips.cc/paper/2276-stochastic-neighbor-embedding.pdf`.

W. W. Hsieh. Nonlinear multivariate and time series analysis by neural network methods. *Rev. Geophys.*, 42(1):RG1003, Mar. 2004. ISSN 1944-9208. doi: 10.1029/2002RG000112.

A. Hyvarinen. Fast and robust fixed-point algorithms for independent component analysis. *IEEE Transactions on Neural Networks*, 10(3): 626–634, May 1999. ISSN 1045-9227. doi: 10.1109/72.761722.

T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31(1):7–15, Apr. 1989. ISSN 0020-0190. doi: 10.1016/0020-0190(89)90102-6. URL `http://www.sciencedirect.com/science/article/pii/0020019089901026`.

G. Kraemer, M. Reichstein, and M. D. Mahecha. dimRed and coRanking - Unifying Dimensionality Reduction in R. *The R Journal*, 2018. URL `https://journal.r-project.org/archive/2018/RJ-2018-039/index.html`.

J. B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, Mar. 1964a. ISSN 0033-3123, 1860-0980. doi: 10.1007/BF02289565. URL http://link.springer.com/article/10.1007/BF02289565.

J. B. Kruskal. Nonmetric multidimensional scaling: A numerical method. *Psychometrika*, 29(2):115–129, June 1964b. ISSN 0033-3123, 1860-0980. doi: 10.1007/BF02289694. URL http://link.springer.com/article/10.1007/BF02289694.

V. Laparra, J. Malo, and G. Camps-Valls. Dimensionality Reduction via Regression in Hyperspectral Imagery. *IEEE Journal of Selected Topics in Signal Processing*, 9(6):1026–1036, Sept. 2015. ISSN 1932-4553. doi: 10.1109/JSTSP.2015.2417833.

J. A. Lee and M. Verleysen. Quality assessment of dimensionality reduction: Rank-based criteria. *Neurocomputing*, 72(7–9):1431–1443, Mar. 2009. ISSN 0925-2312. doi: 10.1016/j.neucom.2008.12.017. URL http://www.sciencedirect.com/science/article/pii/S0925231209000101.

J. A. Lee, E. Renard, G. Bernard, P. Dupont, and M. Verleysen. Type 1 and 2 mixtures of Kullback–Leibler divergences as cost functions in dimensionality reduction based on similarity preservation. *Neurocomputing*, 112:92–108, July 2013. ISSN 0925-2312. doi: 10.1016/j.neucom.2012.12.036. URL http://www.sciencedirect.com/science/article/pii/S0925231213001471.

W. Lueks, B. Mokbel, M. Biehl, and B. Hammer. How to Evaluate Dimensionality Reduction? - Improving the Co-ranking Matrix. *arXiv:1110.3917 [cs]*, Oct. 2011. URL http://arxiv.org/abs/1110.3917. arXiv: 1110.3917.

U. V. Luxburg. *A Tutorial on Spectral Clustering*. 2007.

M. D. Mahecha, A. Martínez, G. Lischeid, and E. Beck. Nonlinear dimensionality reduction: Alternative ordination approaches for extracting and visualizing biodiversity patterns in tropical montane forest vegetation data. *Ecological Informatics*, 2(2):138–149, June 2007. ISSN 1574-9541. doi: 10.1016/j.ecoinf.2007.05.002. URL http://www.sciencedirect.com/science/article/pii/S1574954107000325.

S. Martin, W. M. Brown, and B. N. Wylie. Dr.l: Distributed Recursive (graph) Layout. Technical Report dRl; 002182MLTPL00, Sandia National Laboratories, Nov. 2007. URL http://www.osti.gov/scitech/biblio/1231060-dr-distributed-recursive-graph-layout.

R. A. A. Morrall. Soil microfungi associated with aspen in Saskatchewan: synecology and quantitative analysis. *Can. J. Bot.*, 52(8):1803–1817, Aug. 1974. ISSN 0008-4026. doi: 10.1139/b74-233. URL http://www.nrcresearchpress.com/doi/abs/10.1139/b74-233.

K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(6):559–572, 1901. doi: 10.1080/14786440109462720.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

S. T. Roweis and L. K. Saul. Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, 290(5500):2323–2326, Dec. 2000. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.290.5500.2323. URL http://science.sciencemag.org/content/290/5500/2323.

C. Saunders, A. Gammerman, and V. Vovk. Ridge regression learning algorithm in dual variables. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, pages 515–521, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc. ISBN 1-55860-556-8. URL http://dl.acm.org/citation.cfm?id=645527.657464.

B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear Component Analysis as a Kernel Eigenvalue Problem. *Neural Computation*, 10(5):1299–1319, 1998. ISSN 08997667. doi: 10.1162/089976698300017467.

B. Schölkopf, R. Herbrich, and A. J. Smola. A Generalized Representer Theorem. In *Computational Learning Theory*, pages 416–426. Springer, Berlin, Heidelberg, July 2001. doi: 10.1007/3-540-44581-1_27. URL https://link.springer.com/chapter/10.1007/3-540-44581-1_27.

R. R. Sokal and F. J. Rohlf. The Comparison of Dendrograms by Objective Methods. *Taxon*, 11(2):33–40, 1962. ISSN 0040-0262. doi: 10.2307/1217208. URL http://www.jstor.org/stable/1217208.

S. Sonnenburg, H. Strathmann, S. Lisitsyn, V. Gal, F. J. I. García, W. Lin, S. De, C. Zhang, frx, tklein23, E. Andreev, JonasBehr, sploving, P. Mazumdar, C. Widmer, P. D. . Zora, G. D. Toni, S. Mahindre, A. Kislay, K. Hughes, R. Votyakov, khalednasr, S. Sharma, A. Novik, A. Panda, E. Anagnostopoulos, L. Pang, A. Binder, serialhex, and B. Esser. shogun-toolbox/shogun: Shogun 6.1.0, Nov. 2017. URL https://doi.org/10.5281/zenodo.1067840.

J. B. Tenenbaum, V. d. Silva, and J. C. Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290(5500):2319–2323, Dec. 2000. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.290.5500.2319. URL http://science.sciencemag.org/content/290/5500/2319.

W. S. Torgerson. Multidimensional scaling: I. Theory and method. *Psychometrika*, 17(4):401–419, 1952. ISSN 0033-3123, 1860-0980. doi: 10.1007/BF02288916. URL http://link.springer.com/article/10.1007/BF02288916.

L. van der Maaten and G. Hinton. Visualizing Data using t-SNE. *J. Mach. Learn. Res.*, 9:2579–2605, Nov. 2008. ISSN 1532-4435. WOS:000262637600007.

L. Van Der Maaten, E. Postma, and J. Van den Herik. Dimensionality reduction: a comparative review. *J Mach Learn Res*, 10:66–71, 2009.

J. Venna, J. Peltonen, K. Nybo, H. Aidos, and S. Kaski. Information Retrieval Perspective to Nonlinear Dimensionality Reduction for Data Visualization. *J. Mach. Learn. Res.*, 11:451–490, Feb. 2010. ISSN 1532-4435. WOS:000277186500001.