SoftwareX 21 (2023) 101282

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

direpack: A Python 3 package for state-of-the-art statistical dimensionality reduction methods

Emmanuel Jordy Menvouta [a,*], Sven Serneels [b,c], Tim Verdonck [c]

[a] Department of Mathematics, KU Leuven, Leuven, Belgium
[b] Gallop Data, Inc., Denver, CO, USA
[c] University of Antwerp, Middelheimlaan 1, 2020 Antwerp, Belgium

ARTICLE INFO

Article history:
Received 9 August 2022
Received in revised form 19 October 2022
Accepted 22 November 2022

Keywords:
Dimensionality reduction
Projection pursuit
Sufficient dimension reduction
Robust statistics
Energy statistics
Statistical learning

ABSTRACT

The **direpack** package establishes a set of modern statistical dimensionality reduction techniques into the Python universe as a single, consistent package. Several of the methods included are only available as open source through **direpack**, whereas the package also offers competitive Python implementations of methods previously only available in other programming languages. In its present version, the package is structured in three subpackages for different approaches to dimensionality reduction: projection pursuit, sufficient dimension reduction and robust M estimators. As a corollary, the package also provides access to regularized regression estimators based on these reduced dimension spaces, as well as a set of classical and robust preprocessing utilities, including very recent developments such as generalized spatial signs. Finally, **direpack** has been written to be consistent with the **scikit-learn** API, such that the estimators can flawlessly be included into (statistical and/or machine) learning pipelines in that framework.

Code metadata

Current code version	1.0.20
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-22-00231
Permanent link to Reproducible Capsule	
Legal Code License	MIT license (MIT)
Code versioning system used	Git
Software code languages, tools, and services used	python
Compilation requirements, operating environments & dependencies	numpy, matplotlib, scipy, sklearn, pandas, statsmodels, dcor, sympy
If available Link to developer documentation/manual	https://direpack.readthedocs.io/en/latest/index.html
Support email for questions	tim.verdonck@uantwerpen.be

1. Introduction

Technological advances have drastically changed the dimension of data sets requiring an increase of the speed with which these data must be analyzed. This large increase in dimension often leads to multicollinearity and overfitting problems and makes visualization and analysis very hard. A popular solution to overcome the challenges above is to compress the data onto a new feature subspace of lower dimensionality, which will also be useful to interpret the data. Therefore, dimensionality reduction is a key building block in machine learning projects and is being applied in diverse fields, such as finance, bioinformatics and chemometrics.

While **scikit-learn** contains some well-established classical statistical dimensionality reduction techniques, e.g. principal component analysis (PCA) or partial least squares (PLS), it does not contain options for some of the more recently developed statistical tools, such as dimensionality reduction techniques based on robust or energy statistics. This is where **direpack** complements **scikit-learn**, by delivering a select, yet extensive, set of state-of-the-art statistical dimensionality reduction and regression techniques consistent with the **scikit-learn** API, meaning that the methods in **direpack** can be included in a **scikit-learn** pipeline. **direpack** can be used in a pre-processing or feature

* Corresponding author.
E-mail addresses: emmanueljordy.menvoutankpwele@kuleuven.be (Emmanuel Jordy Menvouta), tim.verdonck@uantwerpen.be (Tim Verdonck).

https://doi.org/10.1016/j.softx.2022.101282

selection step and the resulting object can be used by a scikit-learn estimator. Furthermore, direpack follows the conventions of the API for **scikit-learn** estimators. Beyond being the only open source implementation for some of these methods, the package unites all methods included into a single API compatible with **scikit-learn** and comprises implementations which can achieve faster computation speed and better accuracy than available implementations in open source versions in other languages, as illustrated in Appendix B.

The dimension reduction methods contained in **direpack** presently resort in three categories. At first, **direpack** contains a subpackage for projection pursuit (PP) dimensionality reduction, **ppdire**, which allows to elegantly switch between different estimators by simply exchanging the projection index. The second subpackage, **sudire**, focuses on sufficient dimension reduction (SDR), which is a dimensionality reduction targeting to find a subspace in the predictor block that is sufficient to explain a predictand. It comprises SDR methods based on energy or ball statistics, which, up to our knowledge, are not available as open source anywhere else. Thirdly, the **sprm** subpackage brings a set of robust M estimators for dimensionality reduction and regression, as well as an efficient implementation of univariate sparse PLS. Moreover, **direpack** contains a set of functions for classical and robust data preprocessing, including recent developments such as generalized spatial signs, as well as ancillary functions to calculate bivariate measures of covariance and association and energy statistics. Finally, **direpack** offers a set of plot functions specific to the methods provided, as well as cross-validation utilities compatible with **scikit-learn**'s hyperparameter tuning.

2. Overview of the package

2.1. Preprocessing

Preprocessing is the first step in most data science pipelines. A first, well accepted way to pre-process data is to center them and scale them to unit variance on a column wise basis, transforming the $\mathbf{x}$ variable into $\mathbf{z}$-scores:

$$\mathbf{z} = \frac{\mathbf{x} - \hat{\boldsymbol{\mu}}}{\hat{\boldsymbol{\sigma}}}, \tag{1}$$

where $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\sigma}}$ are estimates of location and scale, respectively. For normally distributed data, **scikit-learn**'s `StandardScaler` is suitable and very frequently applied. However, for data deviating from that assumption, **direpack** presents the `VersatileScaler` alternative, which allows to preprocess based on robust location and scale estimators. Location estimators comprised are the column wise median, the spatial median (also called L_1-median) and the k step least trimmed squares (LTS) estimator of location [1]. As scale estimators, the consistency corrected median absolute deviation (MAD) and the τ estimator of scale [2] have been included. Besides standardizing data, it can be beneficial to transform data to spatial signs. In this sense, both conventional [3] and the *generalized* spatial sign transformations [4] are provided, which can be accessed through **direpack**'s `GenSpatialSignPreprocessor`.

2.2. Projection pursuit

Projection pursuit (PP) is a framework to define and construct statistical estimators [5], including dimensionality reduction. Let $\mathbf{X}$ be a sample of n cases of a p dimensional random variable and $\mathbf{y}$ be a sample of a corresponding depending variable, when applicable. The projection pursuit scores $\mathbf{t}_i$ that span the columns of $\mathbf{T}$ are the linear combination of the original data matrix: $\mathbf{T} = \mathbf{XW}$, where the $\mathbf{w}_i$ are given by:

$$\mathbf{w}_i = \underset{\mathbf{a}}{\operatorname{argmax}}\, \mathfrak{P}\left(\mathbb{S}\left(\mathbf{a}^T\mathbf{X}, \mathbf{y}\right)\right), \tag{2a}$$

subject to:

$$\mathbf{w}_i^T \mathbf{X}^T \mathbf{X} \mathbf{w}_j = 0 \text{ and } \|\mathbf{w}_i\|_2 = 1, \tag{2b}$$

where $i, j \in [1, \min(n, p)]$, $j > i$, $\mathfrak{P}$ is called the *projection index* and is a function characterizing the nature of the resulting the projection, e.g. when $\mathfrak{P}$ equals variance, Eq. (2) defines Principal Component Analysis (PCA). The set $\mathbb{S}(\mathbf{X}, \mathbf{y}) = \{\mathbf{X}, \mathbf{y}\}$ if data for a dependent variable Y exist and $\mathbb{S}(\mathbf{X}, \mathbf{y}) = \{\mathbf{X}\}$ otherwise. The properties of the resulting estimator largely derive from the projection index. In **direpack**, projection pursuit can be called through the **ppdire** subpackage, which allows the user to pass any function of appropriate dimensionality as a projection index. A set of popular projection indices deriving from (co-) moments, are provided as well through the **dicomo** subpackage. Plugging in some of these will lead to well-known methods including Principal Component Analysis (PCA), Partial Least Squares (PLS), Independent Component Analysis (ICA), Canonical Correlation Analysis (CCA) and continuum regression [6]. Note that the authors do not recommend to use **direpack** for these techniques instead of the well-optimized open source implementations available. However, **ppdire** allows to access a much broader set of projection indices, such as robust ones or projection indices based on higher order co-moment statistics, such as CAPI [7]. Besides switching the projection index, **ppdire** allows the user to select the numerical optimization techniques used to calculate the result. Presently, either **scipy.optimize**'s sequential least squares quadratic programming optimization (SLSQP) or its native *grid* optimizer can be selected for the numerical optimization of (2). The SLSQP optimizer will be the most computationally efficient option for convex optimization. However, when projection indices are based on ordering or ranking data, such as medians or trimmed (co-)moments, the problem is no longer convex and cannot be solved through SLSQP. For those purposes, the *grid* algorithm is included, which was originally developed in this context to compute Robust Continuum Regression (RCR) [8].

2.3. Sufficient dimension reduction

Sufficient dimension reduction (SDR) targets to identify a subspace of the data as a linear combination of the original variables $\mathbf{T} = \mathbf{XW}$, the complement of which is statistically independent of the dependent variable, thereby *sufficiently* explaining the latter. The space of reduced dimension is called the *central subspace* and is identified in such a way that it contains all information relevant to the dependent variable:

$$\mathbf{y} \perp\!\!\!\perp \mathbf{X} \mid \mathbf{T}. \tag{3}$$

The subpackage **sudire** contains implementations of a broad set of approaches to SDR, comprising well-established methods such as Sliced Inverse Regression (SIR), Sliced Average Variance Estimate (SAVE), Principal Hessian Directions (PHD), Iterative Hessian Transformation (IHT) and Directional Regression (DR) (for details, refer to [9]). Beyond these, **sudire** provides implementations to three very recent approaches to SDR, which do not require conditions of linearity or constant covariance, nor do they need distributional assumptions. These methods optimize a criterion based on energy or ball statistics such as distance covariance (`dcov-sdr`, [10]), martingale difference divergence (`mdd-sdr`, [11]) and ball covariance (`bcov-sdr`, [12]). Similarly to the PP framework, SDR algorithms estimate the latent components as:

$$\mathbf{W}_h = \underset{\mathbf{B}}{\operatorname{argmax}}\, \mathfrak{V}^2\left(\mathbf{XB}, \mathbf{y}\right), \tag{4a}$$

subject to:

$$\mathbf{B}^T \mathbf{X}^T \mathbf{X} \mathbf{B} = \mathbf{I}_h, \tag{4b}$$

where $\mathbf{B}$ is an arbitrary $p \times h$ matrix, $h \in [1, \min(n, p)]$ and $\mathfrak{V}$ is the energy or ball statistic. We note that SDR methods typically estimate the latent components in a single step procedure, whereas PP estimates the dimension reduction subspace sequentially. Distribution free SDR methods rely on maximizing a nonlinear objective function, which can be a challenging optimization problem. To overcome this problem, a different SDR method(SIR, SAVE or DR) is used as warm start for the optimization and as a nonlinear optimizer, the **Python** binders for **IPOPT** [13] are used. We note that the formulation in Eq. (4) is similar to that of supervised PCA [14]. However, the aim of SDR is to estimate a basis of the central subspace whereas supervised PCA tries to estimate an orthogonal matrix such that the dependency between the projected data and the outcome is maximized. Hence, SDR produce stronger results, but they do not offer a closed form solution as opposed to supervised PCA.

2.4. Robust M estimators

The third dimensionality reduction subpackage, **sprm**, culminates in sparse and robust dimensionality reduction in the form of sparse partial robust M regression (SPRM). M regression is a generalization of least squares regression which minimizes a more general objective that allows to tune the estimator's efficiency and robustness. In M regression, the vector of regression coefficients is defined as:

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \sum_i \rho\left(\frac{r_i(\boldsymbol{\beta})}{\hat{\sigma}}\right), \tag{5}$$

where r_i are the casewise regression residuals and $\hat{\sigma}$ is a robust scale estimator thereof. The ρ function defines the properties of the estimator. SPRM is a sparse and robust alternative to PLS that can be calculated efficiently [15]. Because SPRM combines the virtues of robust regression with sparse dimensionality reduction, besides the SPRM estimator itself, each of these building blocks are provided themselves as class objects that can be deployed in **sklearn** pipelines: the Sparse Nonlinear Iterative Partial Least Squares (SNIPLS) estimator for univariate sparse Partial Least Squares (PLS) [16], as well as the robust M estimator for multiple linear regression [1]. The package offers options to the resulting robustness through different weighting functions (Fair, Huber or Hampel). Moreover, **direpack** provides a plotting function `sprm_plot`, that conveniently provides parity plots, as well as plots of regression coefficients, scores and caseweights. Where applicable, the plots will distinguish regular cases from those that have been identified as moderate or harsh outliers, as well as between training and test set cases, as is illustrated in Fig. 1.

As the package is tightly integrated with **scikit-learn**, the latter's tools for fine-tuning models, such as model metrics and cross-validation can be used for **direpack** hyperparameter tuning. However, **direpack** also contains cross-validation utilities such as a custom loss function that will weight based on the case weights from the M estimators. Each of the subpackages provide a set of specific utilities as well.

3. Code structure and usage

The package is structured into a set of subpackages corresponding to the different flavors of dimensionality reduction, as well as ancillary and utility functions. Each of these subpackages have been built following the PEP8 code standards. Unit tests are included and executed by default in the CI/CD workflow, set up as a GitHub Action. Each release is posted to PyPI, whence it can be installed (`pip install direpack`). A thorough documentation is provided on the package's GitHub page,[1] as well as the ReadTheDocs site.[2] More detailed examples are provided in the Jupyter notebooks in the examples folder of the package for each of the core subpackages: **ppdire sudire** and **sprm**, as well as for the **dicomo** utility to calculate (co-)moments through classical, robust or energy statistics. The code snippets in Listings 1, 2 and 3 show how the **ppdire**, **sprm** and **sudire** packages can be used for dimensionality reduction on a data set.

```
import numpy as np
from direpack import dicomo, ppdire
# Generate dummy data set:
X = np.random.rand(1000,5)
y = np.random.rand(1000,1)
# Instantiate a robust projection pursuit
    dimensionality reduction object
lcpca = ppdire(projection_index = dicomo,
    pi_arguments =\      {'mode' : 'var', 'center':
     'median'}, n_components=4,\ optimizer='grid',
    optimizer_options={'ndir':1000,  \            '
    maxiter':10},regopt='robust')
# Perform robust Principal Component Regression

lcpca.fit(X,y=y)
# loadings
lcpca.x_loadings_
# Scores
lcpca.x_scores_
```

Listing 1: Illustration of the use of the ppdire and dicomo subpackages.

```
from direpack import sprm
import numpy as np

# Generate dummy data set:
X = np.random.rand(1000,5)
y = np.random.rand(1000,1)

# Instantiate a Spare Partial Robust M regression
    object
res_sprm = sprm(2,.8,'Hampel',.95,.975,.999,'
    kstepLTS',
    'scaleTau2',True,
    100,.01,'ally','xonly',False,True)
# fit to first half of data
res_sprm.fit(X[:500],y[:500])
# predict on remaining data
predicted = res_sprm.predict(X[500:])
# transform second half of data using the estimated
     weights
transformed = res_sprm.transform(X[500:])
```

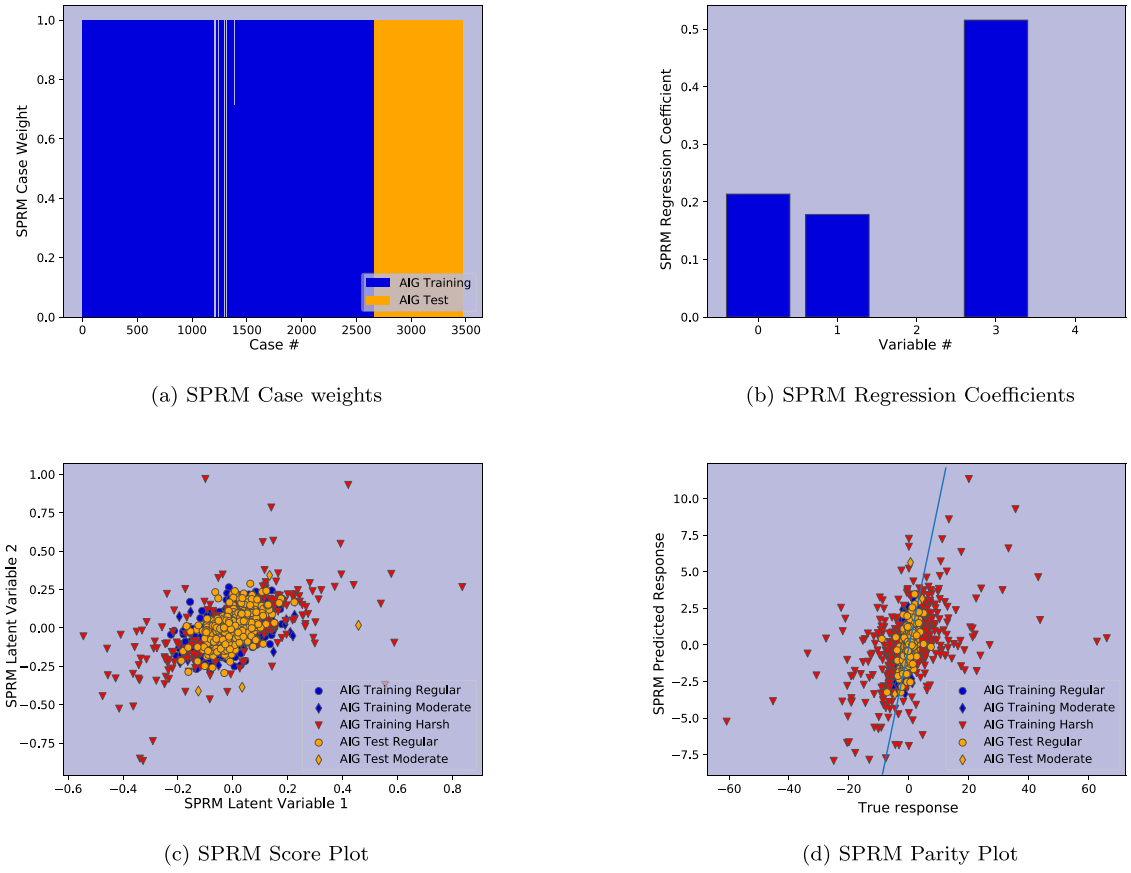
Listing 2: Illustration of the use of the sprm subpackage.

```
from direpack import sudire
import numpy as np
# Generate dummy data set:
X = np.random.rand(1000,5)
y = np.random.rand(1000,1)
# Instantiate a sufficient dimension reduction
    object
dcov_reduce = sudire('dcov-sdr', center_data= True,
     scale_data=True,n_components= 2)
# fit object on data
dcov_reduce.fit(X, y)
# Extract the estimated basis of the central
    subspace
dcov_reduce.x_loadings_
```

Listing 3: Illustration of the use of the sudire subpackage.

[1] https://github.com/SvenSerneels/direpack

[2] https://direpack.readthedocs.io/en/latest/index.html



(a) SPRM Case weights

(b) SPRM Regression Coefficients

(c) SPRM Score Plot

(d) SPRM Parity Plot

Fig. 1. Illustration of the **sprm** subpackage plotting functionality using example data from Yahoo Finance.

4. Impact

The **direpack** package is flexible and can be applied in any domain where the analysis of high dimensional data is necessary. In particular, high dimensional regression or classification problems require feature engineering, and **direpack** helps researchers and practitioners by providing a comprehensive list of functions for dimensionality reduction, while keeping relevant information about the features. Given that **direpack** is compatible with **scikit-learn**, it can be included in a modeling pipeline with ease, which increases its reach and usability. The code is constructed as independent modules that can be modified and adapted to the researcher or practitioners use cases. Each module also contains helper functions that can transform any input tabular data set to a format that is readily usable by the library. **direpack** not only contains statistical methods available in other languages, but also includes functionalities in dimensionality reduction that cannot be found in other software packages e.g. MDD-SDR [11], BCOV-SDR [12], CR [6], CAPI [7], RCR [17]. The implementation of these techniques in **direpack** allow its users access to state-of-the-art methods that have been published in peer reviewed journals, and have not up to now been available in Python.

Some of the methods in **direpack** have been translated from R or MATLAB packages, and hence the relevant code has already been used in industrial domains such as spectrometry [17]. Recently, **direpack** was used in our group to develop a sparse version of sufficient dimension reduction using energy and ball statistics [18]. This software was recently used to calculate generalized betas that also account for higher order moment effects in financial markets [7]. As shown in Appendix B, the implementation of the methods in direpack are competitive in terms of accuracy and computing time with respect to alternatives in other languages. Extensions of the research work implemented in **direpack** are possible, and hence the current and future user community is large. A list of some of the currently available methods in **direpack** is presented in Appendix A.

5. Conclusion and future work

In this article, **direpack** has been presented, a novel package that combines several classes of state-of-the-art dimensionality reduction techniques in one single package and format, consistent with the **scikit-learn** API for statistical and machine learning. Moreover, **direpack** offers some novel preprocessing functions, as well as convenient tools for cross-validation and plotting. The authors hope that thanks to **direpack**'s availability, the methods contained in it will see more widespread adoption in the near future.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data used in the article is available on the package's Github page.

Table A.1
Inexhaustive list of methods accessible through each of the main **direpack** subpackages, along with their most important hyperparameters and references to the literature and open source implementation, as well as the corresponding function to import from **direpack**.

Subpackage	`from direpack import`	Method	Hyperparameters	Ref.	Open Source Ref.
sudire	`sudire`	BCOV−SDR	h	[12]	**None**
	`sudire`	MDD−SDR	h	[11]	**None**
	`sudire`	DCOV−SDR	h	[10]	MATLAB[3]
	`sudire`	IHT	h	[19]	MATLAB[3]
	`sudire`	PHD	h	[20]	R, **dr**
	`sudire`	DR	h, n_{slices}	[21]	R, **dr**
	`sudire`	SAVE	h, n_{slices}	[22]	Python, **sliced** & R, **dr**
	`sudire`	SIR	h, n_{slices}	[23]	Python, **sliced** & R, **dr**
sprm	`sprm`	SPRM	h, η	[15]	R, **sprm**
	`snipls`	SNIPLS	h	[16]	R, **sprm**
	`rm`	RM	h	[1]	R, **lmrob**
ppdire	`ppdire`	LCPCA	h	[24]	R, **pcaPP**
	`ppdire`	CR	h, γ	[6]	**None**
	`ppdire`	RCR	h, γ, α	[25]	**None**
	`capi`	CAPI-GB	h, m, $\mathbf{w}$	[7]	**None**
preprocessing	`VersatileScaler`	k-step LTS	None	[1]	R[3]
	`VersatileScaler`	τ scale	None	[2]	R, **robustbase**
	`GenSpatialSignPreProcessor`	GSS-PP	None	[4]	R[3]
dicomo	`dicomo`	MDD	None	[26]	R, **EDMeasure**
	`dicomo`	MDC	None	[26]	R, **EDMeasure**

Appendix A. List of methods contained in direpack per subpackage

An overview of the methods that stand out the most provided in each of the **direpack** subpackages, as well as the corresponding objects to import from **direpack** and sets of most important hyperparameters, is presented in Table A.1. The Table also lists the references to the original publications for these methods and the name of previously developed open-source implementations, if such a package exists. As to the hyperparameters, h, γ and α denote the number of dimension of the reduced space, the continuum parameter and the percentage of trimming, respectively. For CAPI, $\mathbf{w}$ denotes the vector of relative weights for the moments and m denotes the maximal order of the moment to consider.

Table A.1 lists the methods that most represent what makes **direpack** unique, but neither the list of methods therein, nor the corresponding list of hyperparameters, are exhaustive. Particularly **ppdire** is a framework too general to cite all methods accessible through it and so is its ancillary package **dicomo** that provides a convenient wrapper to access classical, robust and energy estimators for moment and comoment statistics. As to the hyperparameters, the ones listed in Table A.1 are the hyperparameters that would most commonly need to be tuned, such as the number of components. However, many methods possess additional parameters, such as the maximal number of iterations, tolerance for convergence or the cutoff values in the Hampel function, all of which are settings that the user would most commonly not want to tune, but may occasionally want to access. The latter have not been mentioned in Table A.1. Finally, for most of the methods listed in Table A.1, open source implementations in other languages are distributed through the corresponding package repositories, but not for all.[3]

Appendix B. Scaling comparisons

This Section presents an inexhaustive overview of how **direpack** compares to existing packages in other languages in terms of scaling and accuracy. It is noted again that (i) several methods are not available as open source outside **direpack** and (ii) while some other well-known methods, such as PCA, can be calculated through **ppdire**, **ppdire**'s attractiveness does not derive from this option but rather from the flexibility to calculate a larger number of methods under the same API. Based on this observation, a set of five methods has been selected for this comparative study, one from **ppdire**, three from **sudire** and **sprm** subpackages. These methods can be considered as having the right tradeoff between an objective function that cannot be calculated through analytical derivation, yet also are available in a benchmark implementation in another language. These are:

- Projection Pursuit Robust PCA based on the Median Absolute Deviation as a projection index. This method was first published by [24] and an implementation, based on the grid algorithm as well, is available in the R package **pcaPP**.
- Sufficient dimension reduction based on distance covariance (dcov-SDR). The method was first introduced by [10] and MATLAB code has been made available by the authors, using MATLAB's Sequential Quadratic Programming (SQP) optimizer.
- Sufficient dimension reduction via Sliced Inverse Regression (SIR) compared to the implementation in the python package **sliced**. Five slices are used for both the **direpack** and **sliced** implementations.
- Sufficient dimension reduction via Sliced Average Variance Estimate (SAVE), compared to the implementation in the python package **sliced**. Five slices are used for both the **direpack** and **sliced** implementations.
- Sparse partial robust M-regression [15], code for which has also been published as the R package **sprm**. Care has been taken to set parameters and starting values such that the results R and Python become equivalent.

For the scaling comparison of the 3 methods, the number of variables p is fixed at 20 and the number of cases is varied as $n \in \{200, 400, 600, \ldots, 1600\}$.

For robust PCA, data are simulated as in [27]. Set The number of components to $k = 2$, then simulate $\mathbf{X} \sim N(\mathbf{0}, \text{diag}(\lambda_1, \ldots, \lambda_p))$, with $\lambda_j = 1/j^2$. The error of estimation is measured as $1/I_k$, with $I_k = \sum_{j=1}^{k} \text{MAD}(a_j^t \mathbf{x}_1, \ldots, a_j^t \mathbf{x}_n)$, a_j the j^{th} principal component, and $\mathbf{x}_1, \ldots, \mathbf{x}_n$ denoting the n observations.

For SDR, the setup from [10] is adopted, with $h = 2$, $\beta_1 = (1, 0, 0, \ldots, 0)^T$, $\beta_2 = (0, 1, 0, \ldots, 0)^T$, $\epsilon \sim N(0, 1)$, $\mathbf{X} \sim N(\mathbf{0}, \mathbb{I}_p)$, with $\mathbb{I}_p$ the identity matrix, $\mathbf{Y} = (\beta_1^T \mathbf{X})^2 + (\beta_2^T \mathbf{X}) + 0.1\epsilon$ for dcov-SDR and SAVE, and $\mathbf{Y} = (\beta_1^T \mathbf{X}) + 0.1\epsilon$ for SIR.. The error

[3] For these methods, open source implementations can only be retrieved as supplementary material to the corresponding publications.

Table B.2
Scaling comparisons when varying the number of observations.

Method	200		400		600		800	
	Error	Time (s)	Error	Time (s)	Error	Time (s)	Error	Time (s)
R-**pcaPP** grid robust PCA	0.607	0.07	0.617	0.130	0.644	0.200	0.644	0.270
direpack grid robust PCA	0.375	1.273	0.405	1.141	0.422	1.258	0.424	1.484
MATLAB dcov-SDR	0.277	21.3	0.193	105.9	0.144	287.2	0.122	614.7
direpack dcov-SDR	0.273	2.186	0.186	27.445	0.145	37.195	0.128	82.242
sliced SIR	0.976	0.001	0.972	0.001	0.964	0.001	0.968	0.001
direpack SIR	0.976	0.001	0.972	0.125	0.965	0.125	0.968	0.125
sliced SAVE	0.423	0.001	0.196	0.001	0.161	0.001	0.133	0.001
direpack SAVE	0.482	0.001	0.198	0.125	0.162	0.125	0.133	0.125
R SPRM	0.102	0.050	0.061	0.080	0.052	0.080	0.042	0.090
direpack SPRM	0.082	0.055	0.049	0.078	0.035	0.078	0.034	0.109
Method	1000		1200		1400		1600	
	Error	Time (s)	Error	Time (s)	Error	Time (s)	Error	Time (s)
R-**pcaPP** grid robust PCA	0.651	0.340	0.645	0.410	0.658	0.520	0.653	0.565
direpack grid robust PCA	0.431	1.305	0.447	1.414	0.445	1.711	0.448	2.656
MATLAB dcov-SDR	0.104	1780.5	0.099	3179.5	0.087	5378.9	0.081	7867.7
direpack dcov-SDR	0.104	181.523	0.102	253.695	0.096	464.61	0.088	827.17
sliced SIR	0.973	0.001	0.968	0.001	0.970	0.001	0.956	0.001
direpack SIR	0.973	0.25	0.968	0.25	0.970	0.25	0.956	0.375
sliced SAVE	0.112	0.001	0.109	0.001	0.097	0.001	0.092	0.001
direpack SAVE	0.112	0.25	0.109	0.25	0.097	0.25	0.092	0.375
R SPRM	0.041	0.135	0.035	0.180	0.035	0.230	0.029	0.240
direpack SPRM	0.030	0.156	0.027	0.180	0.026	0.203	0.022	0.234

of estimation is measured as $\Delta_m(S_1, S_2) = \left\| \mathbb{P}_{(\beta_1,\beta_2)} - \mathbb{P}_{(\hat{\beta}_1,\hat{\beta}_2)} \right\|$, with $\mathbb{P}_{(\beta_1,\beta_2)}$ the orthogonal projection onto the space spanned by (β_1, β_2), $\|\cdot\|$ the maximum singular value of a matrix and $(\hat{\beta}_1, \hat{\beta}_2)$ an estimated basis of the central subspace by SDR.

For the SPRM comparison, h is set to 6, and the predictand is generated as $\mathbf{Y} = \mathbf{T}\boldsymbol{\gamma} + \epsilon$, where $\mathbf{T} = \mathbf{XA}$, with $\mathbf{X}$ a $n \times p$ matrix simulated from a multivariate standard normal distribution, and the columns $\mathbf{a}_l$ $(l = 1, \ldots, h)$ of $\mathbf{A} \in \mathbb{R}^{p\times h}$ are simulated such that only the first $q \leq p$ elements of a_h correspond to informative variables. To this end, the nonzero part of $\mathbf{A}$ is given by the eigenvectors of $\mathbf{X}_q^T\mathbf{X}_q$, where $\mathbf{X}_q$ denotes the first q columns of $\mathbf{X}$. The components of $\boldsymbol{\gamma}$ are drawn from a uniform distribution on [0.5, 1.5]. The error term, ϵ is generated from an independent standard normal distribution. The regression coefficient vector can be expressed as $\beta = \mathbf{A}\boldsymbol{\gamma}$ and the error of estimation is computed as $\sum_{i=1}^{p}(\beta_i - \hat{\beta}_i)^2$, where $\hat{\beta}$ is the regression coefficient vector estimated by SPRM.

Table B.2 shows the median time (in seconds) and median error for the different methods, with 50 repetitions each, while varying the number of observations. The conclusion from this comparative study is that the implementations provided in **direpack** are competitive.

References

[1] Rousseeuw PJ, Leroy AM. Robust regression and outlier detection. New York: Wiley and Sons; 1987.

[2] Maronna RA, Zamar RH. Robust estimates of location and dispersion for high-dimensional datasets. Technometrics 2002;44(4):307–17.

[3] Serneels S, De Nolf E, Van Espen PJ. Spatial sign preprocessing: A simple way to impart moderate robustness to multivariate estimators. J Chem Inform Model 2006;46:1402–9.

[4] Raymaekers J, Rousseeuw PJ. A generalized spatial sign covariance matrix. J Multivariate Anal 2019;171:94–111.

[5] Kruskal JB. Toward a practical method which helps uncover the structure of a set of multivariate observations by finding the linear transformation which optimizes a new 'index of condensation'. In: Milton RC, Nelder JA, editors. Statistical computation. New York, NY: Academic Press; 1969.

[6] Stone M, Brooks RJ. Continuum regression: Cross-validated sequentially constructed prediction embracing ordinary least squares, partial least squares and principal components regression. J R Stat Soc Ser B Stat Methodol 1990;52:237–69.

[7] Serneels S. Projection pursuit based generalized betas accounting for higher order co-moment effects in financial market analysis. In: JSM proceedings,business and economic statistics section. Alexandria, VA, USA: American Statistical Association; 2019, p. 3009–35.

[8] Filzmoser P, Serneels S, Croux C, Van Espen PJ. Robust multivariate methods: The projection pursuit approach. In: Spiliopoulou M, Kruse R, Borgelt C, Nürnberger A, Gaul W, editors. From data and information analysis to knowledge engineering. Berlin, Germany: Springer Verlag; 2006, p. 270–7.

[9] Li B. Sufficient dimension reduction: methods and applications with R. New York: Chapman & Hall /CRC, Monographs on Statistics and Applied Probability; 2018.

[10] Sheng W, Yin X. Sufficient dimension reduction via distance covariance. J Comput Graph Statist 2016;25:91–104.

[11] Zhang Y, Liu J, Wu Y, Fang X. A martingale-difference-divergence-based estimation of central mean subspace. Stat Interface 2019;12:489–500.

[12] Zhang J, Chen X. Robust sufficient dimension reduction via ball covariance. Comput Statist Data Anal 2019;140:144–54.

[13] Wächter A, Biegler LT. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Math Program 2006;106:25–57.

[14] Barshan E, Ghodsi A, Azimifar Z, Zolghadri Jahromi M. Supervised principal component analysis: Visualization, classification and regression on subspaces and submanifolds. Pattern Recognit 2011;44(7):1357–71.

[15] Hoffmann I, Serneels S, Filzmoser P, Croux C. Sparse partial robust m regression. Chemometr Intell Lab Syst 2015;149:50–9.

[16] Hoffmann I, Filzmoser P, Serneels S, Varmuza K. Sparse and robust PLS for binary classification. J Chemometr 2016;30(4):153–62.

[17] Serneels S, Van Espen PJ. Bootstrap confidence intervals for trilinear partial least squares regression. Anal Chim Acta 2005;544:153–8.

[18] Menvouta EJ, Serneels S, Verdonck T. Sparse dimension reduction based on energy and ball statistics. Adv Data Anal Classif 2022;16:951–75.

[19] Cook RD, Li B. Dimension reduction for the conditional mean in regression. Ann Statist 2002;30:455–74.

[20] Li K-C. On principal Hessian directions for data visualization and dimension reduction: Another application of Stein's lemma. J Amer Statist Assoc 1992;87:1025–39.

[21] Li B. On directional regression for dimension reduction. J Amer Statist Assoc 2007;102:997–1008.

[22] Cook RD. SAVE: A method for dimension reduction and graphics in regression. Commun Stat Theory Methods 2000;29:2109–21.
[23] Li K-C. Sliced inverse regression for dimension reduction. J Amer Statist Assoc 1991;86:316–27.
[24] Chen Z, Li G. Robust principal components and dispersion matrices via projection pursuit. Research report, Department of Statistics, Harvard University; 1981.
[25] Serneels S, Filzmoser P, Croux C, Van Espen PJ. Robust continuum regression. Chemometr Intell Lab Syst 2005;76:197–204.
[26] Shao X, Zhang J. Martingale difference correlation and its use in high-dimensional variable screening. J Amer Statist Assoc 2014;109:1302–18.
[27] Croux C, Filzmoser P, Oliveira MR. Algorithms for projection-pursuit robust principal component analysis. Chemometr Intell Lab Syst 2008;87:218–25.