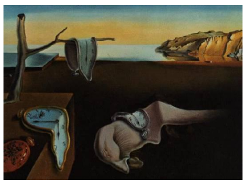

Journal of Statistical Software

October 2021, Volume 99, Issue 12. doi: 10.18637/jss.v099.i12

TRES: An R Package for Tensor Regression and Envelope Algorithms

Jing Zeng
Florida State University

Wenjing Wang
Florida State University

Xin Zhang
Florida State University

Abstract

Recently, there has been a growing interest in tensor data analysis, where tensor regression is the cornerstone of statistical modeling for tensor data. The R package **TRES** provides the standard least squares estimators and the more efficient envelope estimators for the tensor response regression (TRR) and the tensor predictor regression (TPR) models. Envelope methodology provides a relatively new class of dimension reduction techniques that jointly models the regression mean and covariance parameters. Three types of widely applicable envelope estimation algorithms are implemented and applied to both TRR and TPR models.

Keywords: dimension reduction, envelopes, manifold optimization, tensor regression.

1. Introduction

Tensor data, in the form of multidimensional arrays, arise from various applications. Statistical analysis of tensor data has been motivated by many areas of research, including genetics (e.g., Hore, Viñuela, Buil, Knight, McCarthy, Small, and Marchini 2016), signal processing (e.g., Cichocki, Mandic, De Lathauwer, Zhou, Zhao, Caiafa, and Phan 2015), neuroimaging (e.g., Zhou, Li, and Zhu 2013), and relational and networks data analysis (e.g., Hoff 2015). Among these applications, a common statistical analysis task is the study of the relationship between the tensor variable and other (multivariate) variables. This is usually formulated as a tensor regression problem. In particular, this package is developed for two complementary tensor regression models. On one hand, the tensor predictor regression (TPR) model (e.g., Zhou *et al.* 2013; Zhang and Li 2017; Li, Xu, Zhou, and Li 2018b) uses the tensor data along with other additional covariates to predict an outcome of interest. It also includes the scalar-on-image regression (Wang, Zhu, and ADNI 2017) as an important special case. On the other hand, the tensor response regression (TRR) model (Li and Zhang 2017) is aiming to study how the tensor response (e.g., 3D brain images) is affected by a multivariate predictor (e.g., a vector consisting of age, sex, and medical information). However, tensor data often have high dimensionality and complex structures that bring challenges to the estimation procedure. To improve estimation efficiency and to reduce variability in tensor data, Li and Zhang (2017) and Zhang and Li (2017) developed the parsimonious TRR and TPR models by adopting the envelope methodology.

Envelope methodology is a recent research area in multivariate statistical modeling, where the goal is to achieve dimension reduction and efficient parameter estimation simultaneously. The idea is to eliminate variability in the data that is immaterial to the analysis while retaining the material variability by projecting the data onto the latent subspace known as the "envelope". Because of the dimension reduction, more efficient estimators can be derived. The first envelope model was introduced by Cook, Li, and Chiaromonte (2010) for multivariate response reduction in classical multivariate linear regression. The tensor response envelope model (Li and Zhang 2017), which will be discussed in Section 2.3, generalizes the envelope model from multivariate response to tensor response. While Li and Zhang (2017) developed likelihood-based tensor envelope estimators, Zhang and Li (2017) proposed a fast moment-based tensor envelope estimator from the tensor partial least squares algorithm perspective. In this package, we implement both the likelihood-based algorithm and the partial least squares algorithm for model-free and model-based envelope estimation.

The envelope methodology can be viewed as a special type of *sufficient dimension reduction* (SDR) methods (see Li 2018, for background). There have been several packages for sufficient dimension reduction and the envelope models in the literature. The R (R Core Team 2021) package **dr** (Weisberg 2002) implements several non-parametric methods, including SIR (Li 1991), SAVE (Cook and Weisberg 1991), PHD (Li 1992) and IRE (Cook and Ni 2005). The MATLAB (The MathWorks Inc. 2018) package **LDR** (Cook, Forzani, and Tomassi 2011) focuses more on the likelihood-based methods such as LAD (Cook and Forzani 2009). The R package **ldr** (Adragni and Raim 2014) is the counterpart of the MATLAB package **LDR**. Recently, the R package **orthoDr** (Zhu, Zhang, Zhao, Xu, Zhou, and Zhang 2019) implements the semi-parametric SDR method utilizing the techniques developed by Wen and Yin (2013) to efficiently search for solutions in the Stiefel manifold. The MATLAB package **envlp** (Cook, Su, and Yang 2015) implements several envelope estimators under the framework of multivariate linear regression, while its R version is **Renvlp** (Lee and Su 2020). The MATLAB toolbox **TensorReg** (Zhou 2013) implements the tensor predictor regression models with tensor decompositions. Since many of our envelope estimation problems require solving Grassmannian optimizations, we included two of the state-of-the-art manifold optimization algorithms in our package. The first is our implementation of the algorithms in Wen and Yin (2013), which was previously provided in the MATLAB package **OptM** (Wen and Yin 2013). The second one is directly from the R package **ManifoldOptim** (Martin, Raim, Huang, and Adragni 2020). Our package **TRES** (Wang, Zeng, and Zhang 2021) provides options to choose from the various algorithms in Wen and Yin (2013) and Martin *et al.* (2020). Moreover, this package can be easily combined with other general purpose manifold optimization packages, and indeed provides a comprehensive library of tensor regression and envelope estimation problems. The package freely is available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=TRES`.

To the best of our knowledge, there is no existing R package implementing the tensor regression models or tensor envelope models described in this paper. Our R package **TRES** provides both the least squares and the envelope estimation under the framework of tensor regression models. It also implements the general model-free envelope algorithms that are widely applicable. The three types of envelope algorithms covered are: (1) full Grassmannian (FG) algorithm, which solves a likelihood-based general manifold optimization; (2) 1D algorithm (Cook and Zhang 2016) and envelope coordinate descent algorithm (ECD, Cook and Zhang 2018), which are tailored to likelihood-based envelope estimation and solve one-dimensional manifold optimization problems sequentially; (3) moment-based sequential algorithm, which is motivated from the partial least squares algorithm (De Jong 1993). Many envelope models were further proposed in multivariate linear regression (e.g., Su and Cook 2011; Cook, Helland, and Su 2013; Cook and Zhang 2015b). Beyond linear models, envelope methodology can be also applied in generalized linear model (GLM), Cox model (Cook and Zhang 2015a), discriminant analysis (Zhang and Mai 2019) and tensor regression problems (Zhang and Li 2017; Li and Zhang 2017). See Cook (2018) for an overview of envelope methods. The three generic envelope algorithms implemented in this package can be applied to essentially all of these methods by properly incorporating the core estimation functions which will be discussed in Section 4.

The rest of this paper is organized as follows. Section 2 introduces the TRR and the TPR models, and the envelope structured TRR and TPR models. Section 3 demonstrates two main functions for fitting the TRR and TPR models and two dimension selection functions for selecting the envelope dimension. Both simulated and real data examples are included in Section 3. Section 4 presents the envelope algorithms implemented in the **TRES** package and illustrates the core estimation functions via simulation studies. Finally, a brief discussion is given in Section 5.

2. Models

2.1. Notation

We review some matrix, subspace and tensor notations that will be used throughout this article. For more background and a thorough review of tensor related definitions, decompositions and operations, see Kolda and Bader (2009).

- *higher-order tensor*: A multidimensional array $\mathbf{A} \in \mathbb{R}^{r_1 \times \cdots \times r_m}$ is an m-way or mth-order tensor for an integer $m \geq 1$. For example, vectors are first-order tensors, and matrices are second-order tensors. A higher-order tensor is an m-way tensor with $m \geq 3$.

- *vectorization:* The vectorization of mth-order tensor $\mathbf{A}$, denoted by $\mathrm{vec}(\mathbf{A})$, stacks the entries of $\mathbf{A}$ into a column vector. The entry $a_{i_1,\cdots,i_m}$ of $\mathbf{A}$ is mapped to the jth entry of $\mathrm{vec}(\mathbf{A})$, in which $j = 1 + \sum_{k=1}^{m}(i_k - 1)\prod_{k'=1}^{k-1} r_{k'}$.

- *mode-n fiber*: The vector obtained by fixing all indices of a tensor except for the nth one. For a matrix (second-order tensor), the column vector is its mode-1 fiber and the row vector is its mode-2 fiber.

- *mode-n matricization*: The process mapping the tensor $\mathbf{A}$ into a matrix $\mathbf{A}_{(n)} \in \mathbb{R}^{r_n \times \prod_{j \neq n} r_j}$, so that the $(i_1, \cdots, i_m)$ element of $\mathbf{A}$ maps to the (i_n, j) element of the matrix $\mathbf{A}_{(n)}$, where $j = 1 + \sum_{n' \neq n}(i_{n'} - 1)\prod_{n'' < n', n'' \neq n} r_{n''}$.

- *n-mode product*: The n-mode product of a tensor $\mathbf{A}$ and a matrix $\mathbf{G} \in \mathbb{R}^{s \times r_n}$ results in an mth-order tensor denoted as $\mathbf{B} = \mathbf{A} \times_n \mathbf{G} \in \mathbb{R}^{r_1 \times \cdots \times r_{n-1} \times s \times r_{n+1} \times \cdots \times r_m}$, where each element is the product of the mode-n fiber of $\mathbf{A}$ and $\mathbf{G}$. The definition is closely related to the mode-n matricization: $\mathbf{B} = \mathbf{A} \times_n \mathbf{G} \Leftrightarrow \mathbf{B}_{(n)} = \mathbf{G}\mathbf{A}_{(n)}$.

- *n-mode vector product*: The n-mode vector product of a tensor $\mathbf{A}$ and a vector $\mathbf{c} \in \mathbb{R}^{r_n}$ results in an $(m-1)$th-order tensor, denoted as $\mathbf{A} \bar{\times}_n \mathbf{c} \in \mathbb{R}^{r_1 \times \cdots \times r_{n-1} \times r_{n+1} \times \cdots \times r_m}$, where each element is the inner product of each mode-n fiber of $\mathbf{A}$ and vector $\mathbf{c}$.

- *inner product*: The inner product of two tensors $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{r_1 \times \cdots \times r_m}$ is the sum of the products of corresponding entries, denoted as $\langle \mathbf{A}, \mathbf{B} \rangle$.

- *column subspace*: The column subspace of matrix $\boldsymbol{\Gamma} \in \mathbb{R}^{p \times u}$, denoted as $\mathrm{span}(\boldsymbol{\Gamma})$ or $\mathcal{S}_{\boldsymbol{\Gamma}}$, consists of all linear combinations of the column vectors of $\boldsymbol{\Gamma}$.

- *projection*: For the orthogonal basis $(\boldsymbol{\Gamma}, \boldsymbol{\Gamma}_0) \in \mathbb{R}^{p \times p}$, we use $\mathbf{P}_{\boldsymbol{\Gamma}} = \boldsymbol{\Gamma}\boldsymbol{\Gamma}^\top$ to denote the projection onto the subspace $\mathcal{S}_{\boldsymbol{\Gamma}}$ and use $\mathbf{Q}_{\boldsymbol{\Gamma}} = \mathbf{I} - \mathbf{P}_{\boldsymbol{\Gamma}} = \boldsymbol{\Gamma}_0\boldsymbol{\Gamma}_0^\top$ to denote the projection onto the orthogonal complement subspace $\mathcal{S}_{\boldsymbol{\Gamma}_0}$.

- *subspace distance*: Let $\mathcal{S}_{\boldsymbol{\Gamma}_1}$ and $\mathcal{S}_{\boldsymbol{\Gamma}_2}$ be u-dimensional subspaces of $\mathbb{R}^p$, the subspace distance between the two subspaces is defined as $\mathcal{D}(\mathcal{S}_{\boldsymbol{\Gamma}_1}, \mathcal{S}_{\boldsymbol{\Gamma}_2}) = \|\mathbf{P}_{\boldsymbol{\Gamma}_1} - \mathbf{P}_{\boldsymbol{\Gamma}_2}\|_F / \sqrt{2u}$, where $\|\cdot\|_F$ is Frobenius norm.

- *Tucker decomposition*: The Tucker decomposition is used to represent a tensor $\mathbf{A}$ by a small number of components at each mode, which are collected into each factor matrix $\mathbf{C}_k \in \mathbb{R}^{r_k \times p_k}, k = 1, \ldots, m$. And the components in each factor matrix are linked by a dimension reduced core tensor $\mathbf{D} \in \mathbb{R}^{p_1 \times \cdots \times p_m}$ with $p_k \le r_k$. Then, the Tucker decomposition of $\mathbf{A}$ is defined as $\mathbf{A} = \mathbf{D} \times_1 \mathbf{C}_1 \times_2 \cdots \times_m \mathbf{C}_m$ with the notation $[\![\mathbf{D}; \mathbf{C}_1, \ldots, \mathbf{C}_m]\!]$.

- *tensor normal distribution*: The tensor $\mathbf{X} \in \mathbb{R}^{r_1 \times \cdots \times r_m}$ is from a m-way tensor normal distribution $TN(\mathbf{M}; \boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_m)$ if and only if $\mathrm{vec}(\mathbf{X}) \sim N(\mathrm{vec}(\mathbf{M}), \boldsymbol{\Sigma}_m \otimes \ldots \otimes \boldsymbol{\Sigma}_1)$, where $\otimes$ is the Kronecker product.

- *stacked tensor data*: For n independent and identically distributed (i.i.d.) m-way tensor random variables $\mathbf{X}_i \in \mathbb{R}^{r_1 \times \cdots \times r_m}$, $i = 1, \ldots, n$, we use $\mathbb{X} \in \mathbb{R}^{r_1 \times \cdots \times r_m \times n}$ to represent the $(m+1)$-way tensor dataset constructed by stacking n observations along the $(m+1)$th mode. Similarly, $\mathbb{Y}$ is the stacked tensor dataset constructed from $\mathbf{Y}_i$, $i = 1, \ldots, n$.

2.2. Tensor response regression model

The tensor response regression (TRR) model studies the regression problem with a tensor response and a vector predictor (Li and Zhang 2017). More specifically, for an mth-order tensor response $\mathbf{Y} \in \mathbb{R}^{r_1 \times \cdots \times r_m}$ and a vector of predictors $\mathbf{X} \in \mathbb{R}^p$, the tensor response linear regression model is of the form,

$$\mathbf{Y} = \mathbf{B} \bar{\times}_{(m+1)} \mathbf{X} + \boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim TN(0; \boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_m). \tag{1}$$

The data $(\mathbf{X}_i, \mathbf{Y}_i)_{i=1}^n$ are i.i.d. copies of $(\mathbf{X}, \mathbf{Y})$ in (1). Without loss of generality, we assume that $\mathsf{E}(\mathbf{Y}) = \mathbf{0}$, $\mathsf{E}(\mathbf{X}) = \mathbf{0}$ and that data are centered, i.e., $\sum_{i=1}^n \mathbf{Y}_i = \mathbf{0}$ and $\sum_{i=1}^n \mathbf{X}_i = \mathbf{0}$. The coefficient $\mathbf{B} \in \mathbb{R}^{r_1 \times \cdots \times r_m \times p}$ is an $(m+1)$th-order tensor, which is the target parameter.

The error $\boldsymbol{\varepsilon}$ is an mth-order tensor that is independent of $\mathbf{X}$ and follows the *tensor normal* distribution with mean zero and covariance satisfying $\mathrm{cov}\{\mathrm{vec}(\boldsymbol{\varepsilon})\} = \boldsymbol{\Sigma}_m \otimes \cdots \otimes \boldsymbol{\Sigma}_1$, where $\boldsymbol{\Sigma}_k \in \mathbb{R}^{r_k \times r_k}$ is positive definite, $k = 1, \ldots, m$. This tensor normal assumption can effectively reduce the number of free parameters in the covariance $\boldsymbol{\Sigma}_k$.

The standard ordinary least squares (OLS) estimator of $\mathbf{B}$, denoted as $\widehat{\mathbf{B}}_{\mathrm{OLS}}$, is defined as the minimizer of $\|\mathbf{Y} - \mathbf{B}\bar{\times}_{(m+1)}\mathbf{X}\|_F$. This estimator is the maximum likelihood estimator under the tensor normal assumption on $\boldsymbol{\epsilon}$, and has the following easy-to-compute form,

$$\widehat{\mathbf{B}}_{\mathrm{OLS}} = \mathbb{Y} \times_{(m+1)} (\mathbb{X}\mathbb{X}^\top)^{-1}\mathbb{X}, \tag{2}$$

where $\mathbb{X} \in \mathbb{R}^{p\times n}$ and $\mathbb{Y} \in \mathbb{R}^{r_1\times\cdots\times r_m\times n}$ are the stacked predictor matrix and response tensor. When $p \leq n-1$, the above OLS estimator is always well-defined, regardless of the high dimensionality of tensor response.

2.3. TRR model with envelope structure

The tensor response envelope model (Li and Zhang 2017) generalizes the response envelope model (Cook *et al.* 2010) from multivariate response to tensor response. The key idea is that by utilizing the envelope structure, we decompose the tensor response into two parts, $\mathbf{Y} = \mathcal{P}(\mathbf{Y}) + \mathcal{Q}(\mathbf{Y})$, so that the material part of the tensor response, $\mathcal{P}(\mathbf{Y}) = [\![\mathbf{Y}; \mathbf{P}_{\boldsymbol{\Gamma}_1}, \ldots \mathbf{P}_{\boldsymbol{\Gamma}_m}]\!]$, is affected by the predictor $\mathbf{X}$, but the immaterial part of the response, $\mathcal{Q}(\mathbf{Y}) = [\![\mathbf{Y}; \mathbf{Q}_{\boldsymbol{\Gamma}_1}, \ldots \mathbf{Q}_{\boldsymbol{\Gamma}_m}]\!]$, does not vary by the predictor and is also independent of the material part of response. Then the regression analysis can be efficiently performed by focusing only on the material part.

For the material part $\mathcal{P}(\mathbf{Y})$, each subspace $\mathrm{span}(\boldsymbol{\Gamma}_k), k = 1. \ldots, m$, retains the relevant information in the regression of $\mathbf{Y}$ on $\mathbf{X}$ and is called the envelope at mode k. The tensor envelope model implies the following re-parameterization of the TRR model, where $\boldsymbol{\Gamma}_{0k} \in \mathbb{R}^{r_k\times(r_k-u_k)}$ is the orthogonal completion of the envelope basis $\boldsymbol{\Gamma}_k \in \mathbb{R}^{r_k\times u_k}$, $u_k \leq r_k$,

$$\begin{aligned}\mathbf{B} &= [\![\boldsymbol{\Theta}; \boldsymbol{\Gamma}_1, \ldots, \boldsymbol{\Gamma}_m, \mathbf{I}_p]\!], \quad \text{for some } \boldsymbol{\Theta} \in \mathbb{R}^{u_1\times\cdots\times u_m\times p},\\ \boldsymbol{\Sigma}_k &= \boldsymbol{\Gamma}_k\boldsymbol{\Omega}_k\boldsymbol{\Gamma}_k^\top + \boldsymbol{\Gamma}_{0k}\boldsymbol{\Omega}_{0k}\boldsymbol{\Gamma}_{0k}^\top, \quad \text{for some } \boldsymbol{\Omega}_k, \boldsymbol{\Omega}_{0k},\ k = 1, \ldots, m.\end{aligned}$$

The total number of free parameters in the TRR model is reduced significantly by introducing this tensor envelope structure. If we know the estimated envelope basis $\widehat{\boldsymbol{\Gamma}}_k$, $k = 1, \ldots, m$, then the maximum likelihood estimator for $\mathbf{B}$ has the explicit form

$$\widehat{\mathbf{B}}_{\mathrm{Env}} = [\![\widehat{\mathbf{B}}_{\mathrm{OLS}}; \mathbf{P}_{\widehat{\boldsymbol{\Gamma}}_1}, \cdots, \mathbf{P}_{\widehat{\boldsymbol{\Gamma}}_m}, \mathbf{I}_p]\!]. \tag{3}$$

The main computational cost of envelope models comes from the non-convex optimization for obtaining an estimator of $\boldsymbol{\Gamma}_k$, $k = 1, \ldots, m$. The algorithms estimating the estimated $\boldsymbol{\Gamma}_k$ will be discussed in Section 4.

2.4. Tensor predictor regression model

The tensor predictor regression (TPR) model studies the regression problem with a vector response and a tensor predictor (see Zhang and Li 2017). For a vector response $\mathbf{Y} \in \mathbb{R}^r$, and an mth-order tensor predictor $\mathbf{X} \in \mathbb{R}^{p_1\times\cdots\times p_m}$, the tensor predictor regression model is of the form,

$$\mathbf{Y} = \mathbf{B}_{(m+1)}\mathrm{vec}(\mathbf{X}) + \boldsymbol{\varepsilon} \iff Y_j = \langle \mathbf{B}_j, \mathbf{X}\rangle + \varepsilon_j,\ j = 1, \ldots, r, \tag{4}$$

where the $(m+1)$th-order tensor $\mathbf{B} \in \mathbb{R}^{p_1 \times \cdots \times p_m \times r}$ is obtained by stacking r tensors $\mathbf{B}_j \in \mathbb{R}^{p_1 \times \cdots \times p_m}$ along the $(m+1)$th mode. The jth error term $\varepsilon_j \in \mathbb{R}$ corresponds to the jth response variable Y_j, and is independent of $\mathbf{X}$. The standard OLS estimator involves the inverse of the covariance matrix $\mathbf{\Sigma_X} = \text{cov}\{\text{vec}(\mathbf{X})\}$ with extremely large dimension $\Pi_{k=1}^m p_k \times \Pi_{k=1}^m p_k$. Such matrix inversion is computationally intractable or even not well-defined when the tensor dimension is not small. Therefore, Zhang and Li (2017) proposed a modified OLS estimator under the separable structure of the covariance of $\mathbf{X}$, i.e., $\mathbf{\Sigma_X} = \text{cov}\{\text{vec}(\mathbf{X})\} = \mathbf{\Delta}_m \otimes \cdots \otimes \mathbf{\Delta}_1$, where $\mathbf{\Delta}_k \in \mathbb{R}^{p_k \times p_k}$ is positive definite, $k = 1, \ldots, m$.

Given the i.i.d. copies $(\mathbf{X}_i, \mathbf{Y}_i)_{i=1}^n$ from (4), the modified OLS estimator of $\mathbf{B}$ is obtained as a plug-in estimator as follows,

$$\widehat{\mathbf{B}}_{\text{OLS}} = [\![\widehat{\mathbf{C}}; \widehat{\mathbf{\Delta}}_1^{-1}, \ldots, \widehat{\mathbf{\Delta}}_m^{-1}, \mathbf{I}_r]\!], \tag{5}$$

where $\widehat{\mathbf{C}} = \widehat{\text{cov}}(\mathbf{X}, \mathbf{Y}) \in \mathbb{R}^{p_1 \times \cdots p_m \times r}$ is the sample cross covariance tensor between $\mathbf{X}$ and $\mathbf{Y}$, and $\widehat{\mathbf{\Delta}}_k$ is the sample estimator of $\mathbf{\Delta}_k$. In our implementation, we use the iterative maximum likelihood estimator of $\mathbf{\Delta}_k$ under tensor normality assumption (Manceur and Dutilleul 2013).

2.5. TPR model with envelope structure

The tensor predictor envelope model (Zhang and Li 2017) generalizes the predictor envelope model (Cook *et al.* 2013) from multivariate predictor to tensor predictor. Analogous to the tensor response reduction in TRR model, by using the envelope structure we decompose the tensor predictor into two parts: $\mathbf{X} = \mathcal{P}(\mathbf{X}) + \mathcal{Q}(\mathbf{X})$ where $\mathcal{P}(\mathbf{X}) = [\![\mathbf{X}; \mathbf{P}_{\mathbf{\Psi}_1}, \ldots \mathbf{P}_{\mathbf{\Psi}_m}]\!]$ is the material part of $\mathbf{X}$ and $\mathcal{Q}(\mathbf{X}) = [\![\mathbf{X}; \mathbf{Q}_{\mathbf{\Psi}_1}, \ldots \mathbf{Q}_{\mathbf{\Psi}_m}]\!]$ is the immaterial part of $\mathbf{X}$. And we keep the material part $\mathcal{P}(\mathbf{X})$ while eliminating the immaterial part $\mathcal{Q}(\mathbf{X})$ which is irrelevant to the response and does not intertwine with $\mathcal{Q}(\mathbf{X})$ in regression.

Each subspace span$(\mathbf{\Psi}_k)$, $k = 1, \ldots, m$, in the material part $\mathcal{P}(\mathbf{X})$ retains the relevant information in the regression of $\mathbf{Y}$ on $\mathbf{X}$ and is called the envelope at mode k. The envelope structure implies the following re-parameterization of the TPR model, where $\mathbf{\Psi}_{0k} \in \mathbb{R}^{p_k \times (p_k - u_k)}$ is the orthogonal completion of the envelope basis $\mathbf{\Psi}_k \in \mathbb{R}^{p_k \times u_k}$, $u_k < p_k$,

$$\begin{aligned}
\mathbf{B} &= [\![\mathbf{\Theta}; \mathbf{\Psi}_1, \ldots, \mathbf{\Psi}_m, \mathbf{I}_r]\!], \quad \text{for some } \mathbf{\Theta} \in \mathbb{R}^{u_1 \times \cdots \times u_m \times r}, \\
\mathbf{\Delta}_k &= \mathbf{\Psi}_k \mathbf{\Phi}_k \mathbf{\Psi}_k^\top + \mathbf{\Psi}_{0k} \mathbf{\Phi}_{0k} \mathbf{\Psi}_{0k}^\top, \quad \text{for some } \mathbf{\Phi}_k, \mathbf{\Phi}_{0k},\ k = 1, \ldots, m.
\end{aligned}$$

If we know the estimated envelope basis $\widehat{\mathbf{\Psi}}_k$, $k = 1, \ldots, m$, then the partial least squares (PLS) estimation of $\mathbf{B}$ is of the form (Zhang and Li 2017),

$$\widehat{\mathbf{B}}_{\text{Env}} = [\![\widehat{\mathbf{B}}_{\text{OLS}}; \mathbf{P}_{\widehat{\mathbf{\Psi}}_1(\widehat{\mathbf{\Delta}}_1)}, \cdots, \mathbf{P}_{\widehat{\mathbf{\Psi}}_m(\widehat{\mathbf{\Delta}}_m)}, \mathbf{I}_p]\!], \tag{6}$$

where $\mathbf{P}_{\widehat{\mathbf{\Psi}}_k(\widehat{\mathbf{\Delta}}_k)}$ is the projection matrix onto subspace span$(\widehat{\mathbf{\Psi}}_k)$ with regard to the $\widehat{\mathbf{\Delta}}_k$-inner product, i.e., $\mathbf{P}_{\widehat{\mathbf{\Psi}}_k(\widehat{\mathbf{\Delta}}_k)} = \widehat{\mathbf{\Psi}}_k(\widehat{\mathbf{\Psi}}_k^\top \widehat{\mathbf{\Delta}}_k \widehat{\mathbf{\Psi}}_k)^{-1} \widehat{\mathbf{\Psi}}^\top \widehat{\mathbf{\Delta}}_k$. To estimate $\mathbf{\Psi}_k$, Zhang and Li (2017) proposed a tensor PLS algorithm, which will be reviewed in Section 4.

3. Functions for fitting the TRR and TPR models

In this section, we illustrate the two main functions, `TRR.fit()` and `TPR.fit()`, along with the two dimension selection functions, `TRRdim()` and `TPRdim()`, in the R package **TRES**. In addition to the standard and modified OLS estimators, envelope estimators based on different estimation algorithms are also implemented in `TRR.fit()` and `TPR.fit()`. Specifically, we included the full Grassmannian (FG), the 1D, the envelope coordinate descend (ECD), and the partial least squares (PLS) algorithms. The details of algorithms will be discussed in Section 4. The results based on FG, 1D and ECD algorithms in the following examples are almost identical for both `TRR.fit()` and `TPR.fit()` functions, because they are all based on the same likelihood-based objective functions. In contrast, the PLS type algorithm is not based on the likelihood and can often lead to very different results. As illustrations, we only include the OLS estimator and the envelope estimators based on 1D and PLS algorithms.

The package **TRES** depends on the R package **ManifoldOptim** (Martin *et al.* 2020), which includes the Grassmannian optimization algorithms for estimating the envelope basis. Another important imported package is **rTensor** (Li, Bien, and Wells 2018a), which is needed for the tensor operation. Some functions and their usage are presented in Table 1. For more details of each function, see the online manual of the package also available at https://CRAN.R-project.org/package=TRES.

We start with installing and loading the package:

```
R> install.packages("TRES")
R> library("TRES")

Loading required package: ManifoldOptim
Loading required package: Rcpp
```

3.1. Example 1: TRR model with envelope structure

In this part, we use the example from Li and Zhang (2017) to demonstrate the main functions under the TRR model (1). The model studies the association between a two-way tensor (matrix) response $\mathbf{Y}_i \in \mathbb{R}^{64\times 64}$ and a binary predictor $X_i \in \{0,1\}$, which follows the TRR model (1): $\mathbf{Y}_i = \mathbf{B}X_i + \boldsymbol{\epsilon}_i,\ i = 1,\ldots,20$. Each observation of response $\mathbf{Y}_i$ is a two-dimensional image, and each binary predictor X_i indicates two different groups by taking value 0 or 1. The regression coefficient $\mathbf{B} \in \mathbb{R}^{64\times 64}$ is a given image with rank 14, representing the mean

Function name	Usage	Section
`TRR.fit()`	The estimation of tensor response regression.	3.1
`TPR.fit()`	The estimation of tensor predictor regression.	3.2
`TRRdim()`	Envelope dimension selection for tensor response regression by 1D-BIC criterion.	3.3
`TPRdim()`	Envelope dimension selection for tensor predictor regression by cross-validation.	3.3

Table 1: Some functions in package **TRES** for the TRR and TPR models.

Component	Details
`x`	A binary vector with dimension 1×20.
`y`	A continuous three-way tensor with dimension $64 \times 64 \times 20$.
`coefficients`	A continuous three-way tensor with dimension $64 \times 64 \times 1$.
`Gamma`	A list of two basis matrices, `Gamma1` and `Gamma2`, with dimension 64×14.

Table 2: The data and parameters in our simulated example `bat`.

difference of the response $\mathbf{Y}$ between two groups. The envelope basis given by $\mathbf{\Gamma}_1, \mathbf{\Gamma}_2 \in \mathbb{R}^{64\times 14}$ is generated from the singular value decomposition of $\mathbf{B}$, namely $\mathbf{B} = \mathbf{\Gamma}_1\mathbf{\Lambda}\mathbf{\Gamma}_2^\top$, where $\mathbf{\Lambda} \in \mathbb{R}^{14\times 14}$ consists of 14 non-zero singular values. The error term $\boldsymbol{\epsilon}_i$ follows a two-way tensor (matrix) normal distribution. To provide users an off-the-shelf dataset, we save the stacked predictor $\mathbb{X} \in \mathbb{R}^{1\times 20}$, stacked response $\mathbb{Y} \in \mathbb{R}^{64\times 64\times 20}$, coefficient $\mathbf{B}$ and the envelope basis $\mathbf{\Gamma}_1, \mathbf{\Gamma}_2$ into the dataset `bat` with names `x`, `y`, `coefficients` and `Gamma`. The dataset can be loaded into the R environment directly using `data("bat", package = "TRES")`. Note that the coefficient $\mathbf{B}$ is converted to a $64 \times 64 \times 1$ tensor in the dataset `bat`. The components of `bat` are summarized in Table 2.

The function `TRR.fit()` in the package **TRES** estimates the coefficient tensor and error covariance matrices and provides other important statistics under the TRR model (1):

```
TRR.fit(x, y, u, method = c("standard", "FG", "1D", "ECD", "PLS"),
  Gamma_init = NULL)
```

The arguments are described as follows:

- `x`: A matrix with each column being a predictor observation or a vector with each entry being a predictor observation.
- `y`: A tensor object representing the response data, which can be constructed by passing an array, a matrix or a vector to the function `rTensor::as.tensor()`. An array or a matrix with appropriate dimensions are also acceptable.
- `u`: A vector representing the envelope dimension at each mode. Note that `u` is not required for `method = "standard"`.
- `method`: A string specifying the implemented method. With `"standard"`, the standard OLS method for the TRR model is used without envelope assumptions. With `"FG"`, `"1D"`, `"ECD"` or `"PLS"`, different envelope estimation algorithms are implemented. By default, the standard OLS is implemented.
- `Gamma_init`: The initial value of envelope basis is only required by the FG algorithm, i.e., `method = "FG"`. By default, the estimated envelope basis from the 1D algorithm is passed as the initial value.

In this example, we assume that the envelope dimension `u` is known, and we will discuss how to choose it using dimension selection function in Section 3.3. We load the dataset `bat` from the **TRES** package and call the function `TRR.fit()` with different methods:

Function name	Usage	Section
`print()`	Simple display (`call` and `coefficients`).	3.1, 3.2
`summary()`	Return an object of class '`summary.Tenv`' containing the call, method, coefficient, sample size, dimensions of data, mean squared error (and p value, standard error for the object returned by the function `TRR.fit()`); the return object has a `print()` method.	3.1
`coef()`	The estimated coefficients.	3.1, 3.2
`fitted()`	The fitted response.	3.1, 3.2
`residuals()`	The residuals.	3.1, 3.2
`predict()`	The predictions for new data.	3.1
`plot()`	The coefficient plot (and p value plot for the object returned by the function `TRR.fit()`).	3.1, 3.2

Table 3: S3 methods for objects of class '`Tenv`'.

```
R> data("bat", package = "TRES")
R> fit_ols1 <- TRR.fit(bat$x, bat$y, method = "standard")
R> fit_1d1 <- TRR.fit(bat$x, bat$y, u = c(14, 14), method = "1D")
R> fit_pls1 <- TRR.fit(bat$x, bat$y, u = c(14, 14), method = "PLS")
```

The output of the function `TRR.fit()` is an object of class '`Tenv`', which contains the original dataset (`x` and `y`), the matched call (`call`), the implemented method (`method`), the estimated coefficient (`coefficients`), a list of estimated envelope basis (`Gamma`), a list of estimated covariance matrices (`Sigma`), the fitted response (`fitted.values`) and the residuals (`residuals`). Since there is no estimated envelope basis for the OLS method, `Gamma` in `fit_ols1` is `NULL`. The details of the S3 methods for the object of class '`Tenv`' are given in Table 3. For demonstration purpose, we display the outputs of the S3 methods `print()`, `coef()`, `fitted()` and `residuals()` using the object `fit_1d1` in the following. Note that the function `print()` is called internally when we type `fit_1d1`, and only the matched call and estimated coefficient are printed out.

```
R> fit_1d1


Call:
TRR.fit(x = bat$x, y = bat$y, u = c(14, 14), method = "1D")

Coefficients:
Numeric Tensor of 3 Modes
Modes:  64 64 1
Data:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-0.00897 -0.00008  0.00000  0.00276  0.00013  0.09285


R> coef(fit_1d1)
```

```
Numeric Tensor of 3 Modes
Modes:  64 64 1
Data:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-0.00897 -0.00008  0.00000  0.00276  0.00013  0.09285

R> fitted(fit_1d1)

Numeric Tensor of 3 Modes
Modes:  64 64 20
Data:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-0.00897  0.00000  0.00000  0.00138  0.00000  0.09285

R> residuals(fit_1d1)

Numeric Tensor of 3 Modes
Modes:  64 64 20
Data:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-0.4202 -0.0083  0.0003  0.0009  0.0095  0.6078
```

The S3 method `summary()` appends the prediction mean squared error, the p value and the standard error of the estimated coefficient to the output of the function `TRR.fit()`.

```
R> summary(fit_1d1)

                Tensor response regression analysis

Call: TRR.fit(x = bat$x, y = bat$y, u = c(14, 14), method = "1D")
Dimensions: x:(1,20), y:(64,64,20)
Method: "1D", sample size: 20, mean squared error: 2.003

Coefficients:
Numeric Tensor of 3 Modes
Modes:  64 64 1
Data:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-0.00897 -0.00008  0.00000  0.00276  0.00013  0.09285

p-value:
Numeric Tensor of 3 Modes
Modes:  64 64 1
Data:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.000   0.916   0.987   0.912   0.998   1.000
```

```
standard error:
Numeric Tensor of 3 Modes
Modes:  64 64 1
Data:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0039  0.0170  0.0261  0.0325  0.0452  0.4975
```

After we fit the model with `TRR.fit()`, we can make the prediction on the new data or the original data with the S3 method `predict()`. For example, we predict on the original data `bat$x` using the fitted model `fit_1d1`:

```
R> predict(fit_1d1, bat$x)
```

```
Numeric Tensor of 3 Modes
Modes:  64 64 20
Data:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-0.00897  0.00000  0.00000  0.00138  0.00000  0.09285
```

Moreover, if the coefficient returned from the function `TRR.fit()` is a matrix or a tensor which can be squeezed into a matrix, we use the S3 method `plot()` to display the coefficient plot and the associated p value plot. We compare the coefficient plots from different methods. And the true coefficient $\mathbf{B}$ is also displayed as a reference.

```
R> plot(fit_ols1)
R> plot(fit_1d1)
R> plot(fit_pls1)
```

In Figure 1, it can be seen that compared to the OLS and PLS estimators, the 1D estimator produces a better recovery of the true coefficient pattern. The PLS estimator identifies almost nothing, because the PLS algorithm usually requires a larger number of components than likelihood-based envelope estimators. Note that with the function `plot()`, the concomitant p value plot is also displayed. To improve the readability, we defer the discussion of p value plots to Section 3.4.

In addition to the visualization of the coefficient, we evaluate the performance of each method by measuring the estimation error of coefficient, $\|\widehat{\mathbf{B}}-\mathbf{B}\|_F$, where $\widehat{\mathbf{B}}$ and $\mathbf{B}$ are the estimated and the true coefficient, respectively. We use the function `fnorm()` called from the package **rTensor** to calculate the Frobenius norm of the tensor object. And for envelope estimation algorithms, we also compare the estimation error of the envelope basis by measuring the total subspace distance defined as $\sum_{j=1}^{2}\mathcal{D}(\mathcal{S}_{\widehat{\mathbf{\Gamma}}_j},\mathcal{S}_{\mathbf{\Gamma}_j}) = \sum_{j=1}^{2}\|\mathbf{P}_{\widehat{\mathbf{\Gamma}}_j}-\mathbf{P}_{\mathbf{\Gamma}_j}\|_F/\sqrt{2u_j}$, where $\widehat{\mathbf{\Gamma}}_j$ and $\mathbf{\Gamma}_j$ are the estimated and the true envelope basis and u_j is the envelope dimension. We use the function `subspace()` in our **TRES** package to compute the subspace distance $\mathcal{D}(\mathcal{S}_{\widehat{\mathbf{\Gamma}}_j},\mathcal{S}_{\mathbf{\Gamma}_j})$ at each mode. The function `subspace()` is used as follows:

```
subspace(A, B)
```

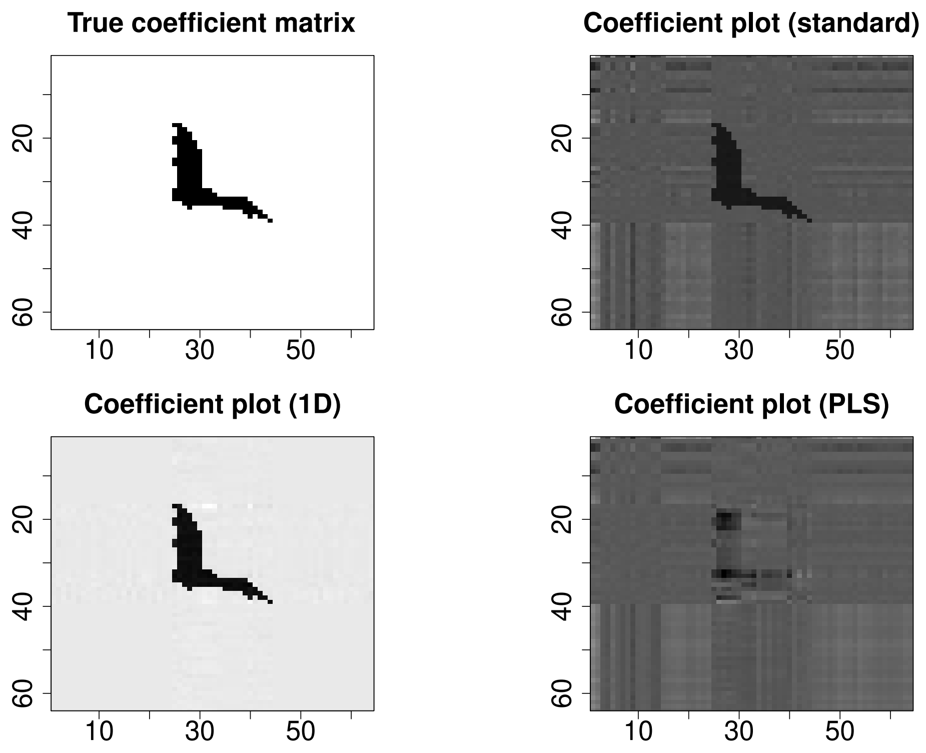

Figure 1: Coefficient plots for the true **B**, OLS estimator, 1D estimator, and PLS estimator on the `bat` dataset.

where the arguments `A` and `B` are two full column rank matrices of the same size. We compare the estimation errors of coefficient among OLS, 1D and PLS methods, and compare the estimation errors of envelope basis only between the envelope methods, 1D and PLS.

```
R> dist_ols1 <- rTensor::fnorm(coef(fit_ols1) - bat$coefficients)
R> dist_1d1 <- rTensor::fnorm(coef(fit_1d1) - bat$coefficient)
R> dist_pls1 <- rTensor::fnorm(coef(fit_pls1) - bat$coefficient)
R> Pdist_1d1 <- rep(NA_real_, 2)
R> Pdist_pls1 <- rep(NA_real_, 2)
R> for (i in 1:2) {
+    Pdist_1d1[i] <- subspace(bat$Gamma[[i]], fit_1d1$Gamma[[i]])
+    Pdist_pls1[i] <- subspace(bat$Gamma[[i]], fit_pls1$Gamma[[i]])
+ }
R> Pdist_1d1 <- sum(Pdist_1d1)
R> Pdist_pls1 <- sum(Pdist_pls1)
R> c(dist_ols1, dist_1d1, dist_pls1)

[1] 0.97204 0.04428 1.19191

R> c(Pdist_1d1, Pdist_pls1)

[1] 0.1143 1.4663
```

Component	Details
`x`	A continuous three-way tensor with dimension $32 \times 32 \times 200$.
`y`	A vector with dimension 1×200.
`coefficients`	A three-way tensor with dimension $32 \times 32 \times 1$.
`Gamma`	A list of two basis matrices, `Gamma1` and `Gamma2`, with dimension 32×2.

Table 4: The details of the components in dataset `square`.

3.2. Example 2: TPR model with envelope structure

In this section, we follow the simulation setting from Zhang and Li (2017) to demonstrate the main functions under the TPR model (4). The model has a scalar response $Y_i \in \mathbb{R}$ and a two-way tensor (matrix) predictor $\mathbf{X}_i \in \mathbb{R}^{32\times32}$, $Y_i = \langle \mathbf{B}, \mathbf{X}_i \rangle + \epsilon_i, i = 1, \ldots, 200$. The coefficient matrix $\mathbf{B} \in \mathbb{R}^{32\times32}$ is a given image with rank 2. All the elements of the coefficient matrix $\mathbf{B}$ are either 0.1 or 1. Similar to Example 1, the two envelope basis $\mathbf{\Gamma}_1, \mathbf{\Gamma}_2 \in \mathbb{R}^{32\times2}$ are generated from the singular value decomposition of $\mathbf{B}$. The error term ϵ_i is generated from a standard normal distribution. We save the stacked predictor $\mathbb{X} \in \mathbb{R}^{32\times32\times200}$, stacked response $\mathbb{Y} \in \mathbb{R}^{1\times200}$, coefficient $\mathbf{B}$ and the envelope basis $\mathbf{\Gamma}_1, \mathbf{\Gamma}_2$ into the dataset `square` with names `x`, `y`, `coefficients` and `Gamma`. And we convert the coefficient to the $32\times32\times1$ tensor in the dataset `square`. The details of the components of `square` are provided in Table 4.

The function `TPR.fit()` estimates the coefficient tensor, error covariance matrices and other important statistics under the TPR model (4) with or without envelope structure assumption:

```
TPR.fit(x, y, u, method = c("standard", "FG", "1D", "ECD", "PLS"),
  Gamma_init = NULL)
```

The meaning of arguments `u` and `Gamma_init` are the same as those of `TRR.fit()` explained in Section 3.1. The rest of the arguments are described as follows:

- `x`: A tensor object representing the predictor data, which can be constructed by passing an array, a matrix or a vector to the function `rTensor::as.tensor()`. An array or a matrix with appropriate dimensions are also acceptable.
- `y`: A matrix with each column being a response observation or a vector with each entry being a response observation.
- `method`: A string specifying the implemented method. With `"standard"`, the modified OLS method for the TPR model is used. With `"FG"`, `"1D"`, `"ECD"` or `"PLS"`, different envelope estimation algorithms are implemented. If not specified, the modified OLS method is implemented by default.

The standard OLS estimator for the TPR model is unavailable due to the ultra-high dimensionality of $\mathbf{X}$. However, to keep the names of the arguments of two main functions `TRR.fit()` and `TPR.fit()` consistent, we still use the name `"standard"` while the modified OLS estimator under the Kronecker separable covariance assumption is returned. Note that the p value and the standard error of estimated coefficient are not provided in the summary output of `TPR.fit()` since they depend on $\widehat{\mathrm{cov}}^{-1}\{\mathrm{vec}(\mathbf{X})\}$ which is unavailable in most cases since the dimension of $\mathrm{vec}(\mathbf{X})$ is ultra-high. Also, the p value plot is not provided for the TPR model.

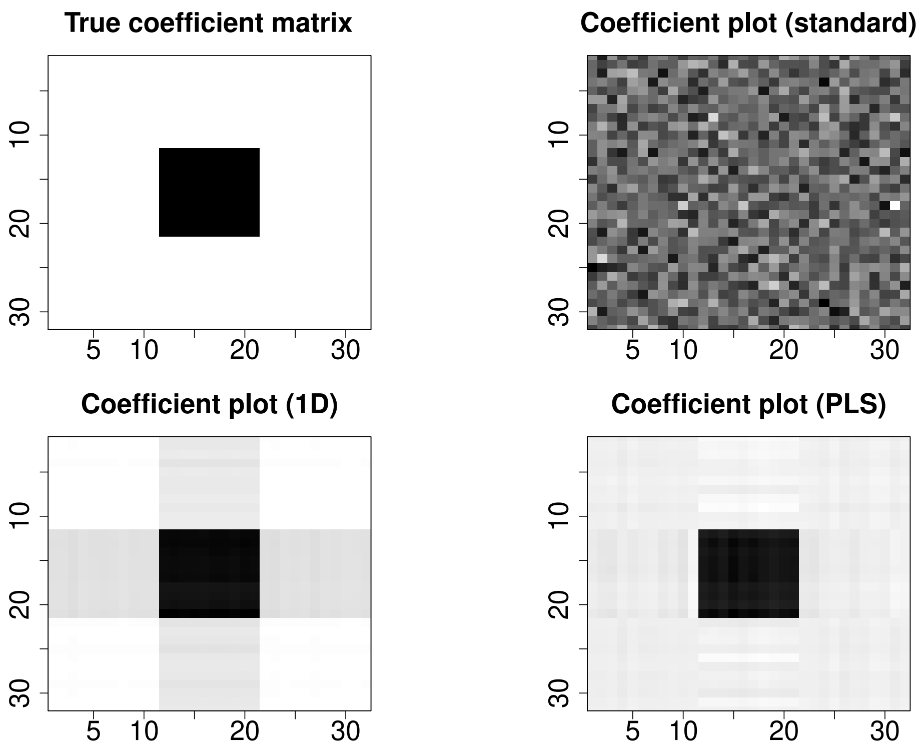

Figure 2: Coefficient plots for the true **B**, OLS estimator, 1D estimator, and PLS estimator on the `square` dataset.

We load the dataset `square` from the **TRES** package and call the function `TPR.fit()` with different methods.

```
R> data("square", package = "TRES")
R> fit_ols2 <- TPR.fit(square$x, square$y, method = "standard")
R> fit_1d2 <- TPR.fit(square$x, square$y, u = c(2, 2), method = "1D")
R> fit_pls2 <- TPR.fit(square$x, square$y, u = c(2, 2), method = "PLS")
```

The output of the function `TPR.fit()` is also an object of class '`Tenv`'. And all the S3 methods in Table 3 are applicable here. We use the '`Tenv`' object `fit_1d2` as an example:

```
R> fit_1d2


Call:
TPR.fit(x = square$x, y = square$y, u = c(2, 2), method = "1D")

Coefficients:
Numeric Tensor of 3 Modes
Modes:  32 32 1
Data:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0052  0.0064  0.0420  0.0724  0.0607  0.4838
```

```
R> coef(fit_1d2)

Numeric Tensor of 3 Modes
Modes:  32 32 1
Data:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0052  0.0064  0.0420  0.0724  0.0607  0.4838

R> dim(fitted(fit_1d2))

[1]   1 200

R> dim(residuals(fit_1d2))

[1]   1 200
```

We also visualize the coefficients from the OLS method and other envelope estimation algorithms in Figure 2, along with the true coefficient **B**.

```
R> plot(fit_ols2)
R> plot(fit_1d2)
R> plot(fit_pls2)
```

In Figure 2, the envelope-based estimators, namely the 1D and PLS estimators, identify the center black square successfully, while the OLS estimator fails to identify any pattern. Moreover, the PLS estimator outperforms the 1D estimator in the sense that the background around the center black square is equally recognized, while the 1D estimator incorrectly highlights four gray squares around the center black square, which results in a different pattern from the true coefficient **B**.

The estimation errors $\|\widehat{\mathbf{B}} - \mathbf{B}\|_F$ and $\sum_{j=1}^{m} \mathcal{D}(\mathcal{S}_{\widehat{\boldsymbol{\Gamma}}_j}, \mathcal{S}_{\boldsymbol{\Gamma}_j})$ are produced in the same way as in Section 3.1 using the functions `fnorm()` from **rTensor** and `subspace()` from our package **TRES**. We thus omit the code and just present the results here.

```
R> c(dist_ols2, dist_1d2, dist_pls2)

[1] 2225.638    5.798    5.591

R> c(Pdist_1d2, Pdist_pls2)

[1] 1.4143 0.1563
```

3.3. Envelope dimension selection

The envelope dimension selection is important in the applications of envelope models since the estimation relies on the knowledge of the envelope dimension. In the TRR model we are more interested in envelope estimation, so an accurate envelope dimension selection method is in demand. Fortunately, Zhang and Mai (2018) proposed a BIC-type criterion called 1D-BIC, which is based on the 1D objective functions, to select the envelope dimension in a model-free context. We use the 1D-BIC criterion to select the dimension under the TRR model. As for the TPR model the prediction is more important, an accurate envelope dimension is not necessary. Therefore, under the TPR model, we use cross-validation to select dimension based on the prediction mean squared error.

The functions `TRRdim()` and `TPRdim()` in the package **TRES** select envelope dimension under the TRR and TPR models respectively:

```
TRRdim(x, y, C = NULL, maxdim = 10, ...)
TPRdim(x, y, maxdim = 10, nfolds = 5)
```

where the arguments are described as follows:

- `x, y`: The predictor and the response.
- `maxdim`: An integer specifying the maximum envelope dimension to be considered. The default value is 10.
- `C`: The constant passed to the internal function `oneD_bic()`. The default value is the dimension of the predictor.
- `...`: Additional user-defined arguments passed to the internal function `oneD_bic()`, which will be discussed in Section 4.4.
- `nfolds`: An integer specifying the number of folds for cross-validation in the function `TPRdim()`. The default value is 5.

The arguments `x` and `y` in `TRRdim()` and `TPRdim()` are of the same data types as those in `TRR.fit()` and `TPR.fit()`. The output of the function `TRRdim()` consists of the minimal 1D-BIC values `bicval` for each mode, the estimated envelope dimensions `u` at which the `bicval` values are obtained, and the prediction mean square error `mse` between the predicted and the original responses. The discussion of the 1D-BIC criterion is deferred to Section 4.4 where the function `oneD_bic()` is introduced. The output of `TPRdim()` is similar. It consists of the minimal cross-validation mean squared error `mincv` and the estimated envelope dimension `u`. The cross-validation mean squared error is the average of mean squared errors between the predicted and the original responses over `nfolds` folds. According to Zhang and Li (2017), the dimensions of envelopes at each mode are assumed to be equal, so the `u` returned by `TPRdim()` is a single value representing the equal dimension.

We illustrate these two functions using the aforementioned examples. For a more exhausted search, we set `maxdim` to the half of the maximum dimension of the tensor, but in practice it should be decided by users' own discretion.

```
R> uhat1 <- TRRdim(bat$x, bat$y, maxdim = 32)
R> uhat1

$bicval
[1] -1262 -1183
```

```
$u
[1] 14 14

$mse
[1] 1.922

R> uhat2 <- TPRdim(square$x, square$y, maxdim = 16)
R> uhat2

$mincv
[1] 1.105

$u
[1] 2
```

We can see that the functions `TRRdim()` and `TPRdim()` correctly identify the envelope dimensions in both examples.

3.4. p values in the TRR model

In some cases, the coefficient $\mathbf{B}$ in the TRR model is sparse, which means that most of the elements are zero. For example, the true coefficient $\mathbf{B}$ in the TRR model in Section 3.1 is sparse since just the elements in the bat shape area are non-zero. The sparsity pattern of coefficient $\mathbf{B}$ can be detected by the corresponding p value tensor at some significance level. With the S3 method `summary()`, we can extract the p value tensor from the '`Tenv`' object returned from the function `TRR.fit()`. In this section, we use the '`Tenv`' objects `fit_ols1`, `fit_1d1` and `fit_pls1` in Section 3.1 as an example.

```
R> summary(fit_ols1)$p_val

Numeric Tensor of 3 Modes
Modes:  64 64 1
Data:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0000  0.0523  0.3791  0.4164  0.7406  0.9997

R> summary(fit_1d1)$p_val

Numeric Tensor of 3 Modes
Modes:  64 64 1
Data:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.000   0.916   0.987   0.912   0.998   1.000

R> summary(fit_pls1)$p_val
```

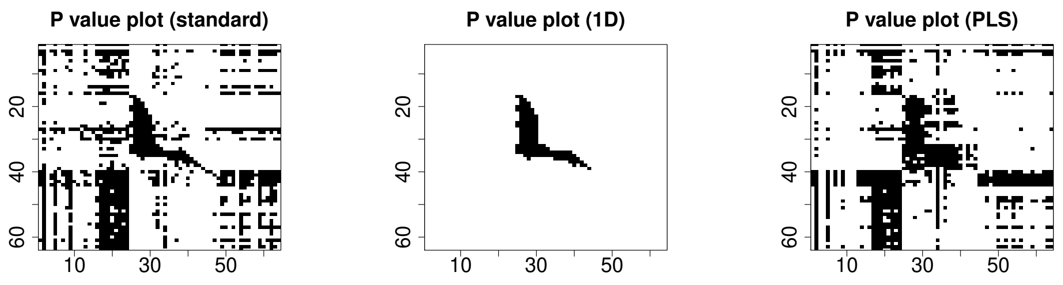

Figure 3: Comparisons of p value plots at significance level 0.05 for OLS, 1D and PLS estimators on the `bat` dataset.

```
Numeric Tensor of 3 Modes
Modes:  64 64 1
Data:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0000  0.0624  0.3538  0.4110  0.7431  1.0000
```

If the estimated coefficient returned from the function `TRR.fit()` is a matrix or a tensor that can be squeezed into a matrix, the p value plot can be generated by the S3 method `plot()`. The significance level is set by specifying the argument `level`. We present the p value plots at significance level 0.05 for each estimator in Figure 3 using the following code:

```
R> plot(fit_ols1, level = 0.05)
R> plot(fit_1d1, level = 0.05)
R> plot(fit_pls1, level = 0.05)
```

The black pixel represents the significant element of coefficient at level 0.05. It can be seen that the pattern detected by the p values approximately complies with the coefficient plots in Figure 1. And the 1D estimator again outperforms the other two estimators.

3.5. EEG data analysis

A variety of problems in neuroimaging can be formulated with the TRR or TPR model with the image tensor being the response or the predictor. In this section, we analyze an electroencephalography (EEG) dataset under the TRR model. The dataset arises from an alcoholism study and can be downloaded from `https://archive.ics.uci.edu/ml/datasets/eeg+database`. It was studied by Li and Zhang (2017). They focus on detecting the difference of EEG images between the two groups, which helps expose the EEG correlates of genetic predisposition to alcoholism. The original dataset contains 122 subjects, each representing a 256×64 EEG image data, which was collected from 64 electrodes placed on the subject's scalp sampled at 256 Hz for 1 second. As suggested by Li and Zhang (2017), we downsize the image to 64×64 by averaging every four consecutive time points. And considering the package size, we randomly select 61 subjects and save them as the dataset `EEG` into the package **TRES**. There are 39 alcoholic individuals and 22 controls in `EEG`, which are coded by 1 and 0 in the predictor vector `x`. The response `y` is a three-way tensor with dimension $64 \times 64 \times 61$, and each matrix obtained by fixing the third mode represents an EEG image for each subject. We

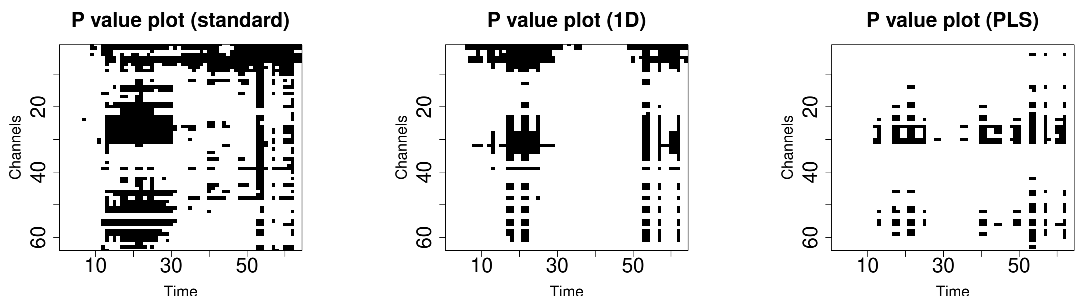

Figure 4: Comparisons of p value plots at significance level 0.05 for OLS, 1D and PLS estimators on the `EEG` dataset.

refer the readers to Li and Zhang (2017) for more details about the dataset and the problem modeling.

We apply the OLS, 1D and PLS methods on this dataset. For envelope-based methods, we first estimate the envelope dimension with the function `TRRdim()`:

```
R> data("EEG", package = "TRES")
R> u_eeg <- TRRdim(EEG$x, EEG$y)
R> u_eeg

$bicval
[1] 1.867 1.608

$u
[1] 1 1

$mse
[1] 38256
```

With the estimated envelope dimension $u_1 = u_2 = 1$, we fit the dataset using the function `TRR.fit()`.

```
R> fit_eeg_ols <- TRR.fit(EEG$x, EEG$y, method = "standard")
R> fit_eeg_1D <- TRR.fit(EEG$x, EEG$y, u_eeg$u, method = "1D")
R> fit_eeg_pls <- TRR.fit(EEG$x, EEG$y, u_eeg$u, method = "PLS")
```

To compare the performances among the three methods in detecting the mean difference of EEG images between the two groups, we present the p value plots with level 0.05 using the following code:

```
R> plot(fit_eeg_ols, xlab = "Time", ylab = "Channels", cex.main = 2,
+    cex.axis = 2, cex.lab = 1.5)
R> plot(fit_eeg_1D, xlab = "Time", ylab = "Channels", cex.main = 2,
+    cex.axis = 2, cex.lab = 1.5)
R> plot(fit_eeg_pls, xlab = "Time", ylab = "Channels", cex.main = 2,
+    cex.axis = 2, cex.lab = 1.5)
```

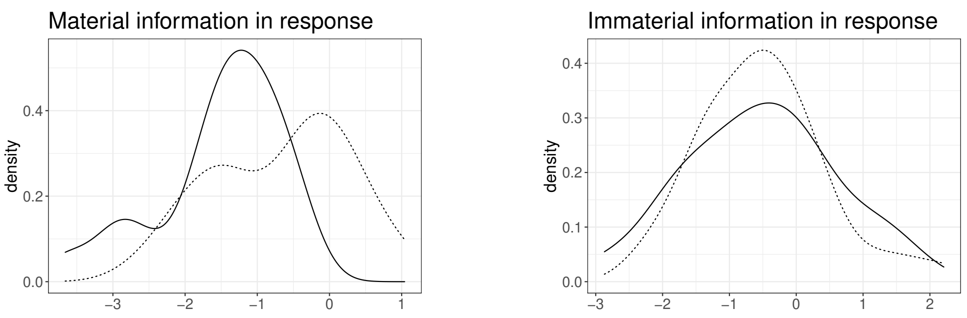

Figure 5: Density plots for the material and immaterial parts of the response. The dotted line represents the alcoholic group ($X = 1$) and the solid line represents the control group ($X = 0$).

From Figure 4, we observe that the 1D estimator identifies several clearly separated significant areas, i.e., the channels between 0 and 15, and between 25 and 40, in the time period from 60 to 120, and from 200 to 240, in the original time scale. In comparison, the plot of the OLS estimator is less clear and more variable. And for the PLS estimator, the significant areas are not well separated, which makes it harder to interpret the plot. In addition, there are fewer significant areas in the plot of the PLS estimator, thus less information about the EEG correlates is contained.

The envelope structure in the TRR model (Section 2.3) identifies the material and immaterial information in the tensor response $\mathbf{Y}$. To illustrate this, we draw density plots of the material part and the immaterial part in the EEG dataset in Figure 5. In these data, the material part of the response is estimated to be one-dimensional: $[\![\mathbf{Y}; \widehat{\mathbf{\Gamma}}_1^\top, \widehat{\mathbf{\Gamma}}_2^\top]\!]$, where $\widehat{\mathbf{\Gamma}}_1, \widehat{\mathbf{\Gamma}}_2 \in \mathbb{R}^{64\times 1}$. By definition the material part of the response $\mathbf{Y}$ changes when we vary the values of the predictor X. This is clearly demonstrated by the two separated density curves on the left panel of Figure 5 corresponding to $X = 0$ or 1. In contrast, the immaterial part of $\mathbf{Y}$ is massive in these data. For visualization purpose, we randomly sampled two immaterial directions $\widetilde{\mathbf{\Gamma}}_{01}, \widetilde{\mathbf{\Gamma}}_{02} \in \mathbb{R}^{64\times 1}$ that are orthogonal to the envelope. From the right panel of Figure 5, the immaterial response has very close conditional distributions given $X = 0$ and $X = 1$. Therefore, the corresponding regression coefficient is zero.

4. Envelope algorithms and dimension selection

4.1. Formal definitions

We first review the definitions of reducing subspace and envelope (Cook *et al.* 2010).

Definition 1 (Reducing subspace) *A subspace $\mathcal{R} \subseteq \mathbb{R}^p$ is said to be a reducing subspace of $\mathbf{M} \in \mathbb{R}^{p\times p}$ if $\mathcal{R}$ decomposes $\mathbf{M}$ as $\mathbf{M} = \mathbf{P}_{\mathcal{R}}\mathbf{M}\mathbf{P}_{\mathcal{R}} + \mathbf{Q}_{\mathcal{R}}\mathbf{M}\mathbf{Q}_{\mathcal{R}}$, where $\mathbf{P}_{\mathcal{R}}$ is the projection matrix onto $\mathcal{R}$ and $\mathbf{Q}_{\mathcal{R}} = \mathbf{I}_p - \mathbf{P}_{\mathcal{R}}$ is the projection onto $\mathcal{R}^\perp$.*

Definition 2 (Envelope) *Let $\mathbf{M} \in \mathbb{R}^{p\times p}$ and let $\operatorname{span}(\mathbf{U}) \subseteq \operatorname{span}(\mathbf{M})$. The $\mathbf{M}$-envelope of $\operatorname{span}(\mathbf{U})$, denoted by $\mathcal{E}_{\mathbf{M}}(\mathbf{U})$, is the intersection of all reducing subspaces of $\mathbf{M}$ that contain $\operatorname{span}(\mathbf{U})$.*

Note that $\mathrm{span}(\mathbf{U}) = \mathrm{span}(\mathbf{U}\mathbf{U}^\top)$, we can always denote the envelope by $\mathcal{E}_{\mathbf{M}}(\mathbf{U})$ for some positive semi-definite matrix $\mathbf{U}$. The matrices $\mathbf{U}$ and $\mathbf{M}$ are then parameterized as (Cook *et al.* 2010):

$$\mathbf{U} = \mathbf{\Gamma}\mathbf{\Phi}\mathbf{\Gamma}^\top, \quad \mathbf{M} = \mathbf{\Gamma}\mathbf{\Omega}\mathbf{\Gamma}^\top + \mathbf{\Gamma}_0\mathbf{\Omega}_0\mathbf{\Gamma}_0^\top, \tag{7}$$

where $\mathbf{\Phi} \in \mathbb{R}^{u\times u}, \mathbf{\Omega} \in \mathbb{R}^{u\times u}, \mathbf{\Omega}_0 \in \mathbb{R}^{(p-u)\times(p-u)}$ are positive semi-definite matrices, and $\mathbf{\Gamma} \in \mathbb{R}^{p\times u}$ denotes an orthogonal basis of the envelope $\mathcal{E}_{\mathbf{M}}(\mathbf{U})$. For the TRR model with envelope structure in Section 2.3, the envelope at each mode can be represented using notation $\mathcal{E}_{\mathbf{M}}(\mathbf{U})$ where $\mathbf{M} = \mathbf{\Sigma}_k$ and $\mathbf{U} = \mathbf{B}_{(k)}$, i.e., $\mathrm{span}(\mathbf{\Gamma}_k) = \mathcal{E}_{\mathbf{\Sigma}_k}(\mathbf{B}_{(k)})$, $k = 1,\ldots,m$. And for the TPR model with envelope structure in Section 2.5, the envelope at each mode can be represented as $\mathrm{span}(\mathbf{\Psi}_k) = \mathcal{E}_{\mathbf{\Delta}_k}(\mathbf{B}_{(k)})$, $k = 1,\ldots,m$, where $\mathbf{M} = \mathbf{\Delta}_k$ and $\mathbf{U} = \mathbf{B}_{(k)}$.

The definitions above do not rely on the regression model and can be applied in any model-free context. In the next section, we will show how to obtain a $\sqrt{n}$-consistent estimator for envelope $\mathcal{E}_{\mathbf{M}}(\mathbf{U})$ in the model-free context via different algorithms, while the matrices $\mathbf{M}$ and $\mathbf{U}$ are substituted by the $\sqrt{n}$-consistent estimators $\widehat{\mathbf{M}}$ and $\widehat{\mathbf{U}}$. The corresponding algorithms are incorporated into the package **TRES**. For more details about the estimators $\widehat{\mathbf{M}}$ and $\widehat{\mathbf{U}}$ in TRR and TPR models with envelope structure, see Li and Zhang (2017) and Zhang and Li (2017).

4.2. Three types of algorithms

Full Grassmannian (FG) optimization

To elaborate the full Grassmannian (FG) algorithm, we introduce an equivalent definition of $\mathcal{E}_{\mathbf{M}}(\mathbf{U})$ by optimizing the generic population objective function $F(\mathbf{\Gamma})$ as follows,

$$\widetilde{\mathbf{\Gamma}} = \arg\min_{\mathbf{\Gamma}\in\mathcal{G}_{u,p}} F(\mathbf{\Gamma}), \tag{8}$$

$$\text{where } F(\mathbf{\Gamma}) = \log|\mathbf{\Gamma}^\top\mathbf{M}\mathbf{\Gamma}| + \log|\mathbf{\Gamma}^\top(\mathbf{M}+\mathbf{U})^{-1}\mathbf{\Gamma}|. \tag{9}$$

In (8), $\mathcal{G}_{u,p}$ denotes a Grassmannian consisting of all u-dimensional subspaces of $\mathbb{R}^p$. It follows that $\mathrm{span}(\widetilde{\mathbf{\Gamma}}) = \mathcal{E}_{\mathbf{M}}(\mathbf{U})$ in population. The gradient-based full Grassmannian optimization relies on the following closed-form expression of the gradient,

$$\frac{\partial}{\partial\mathbf{\Gamma}}F(\mathbf{\Gamma}) = 2\,\mathbf{M}\mathbf{\Gamma}(\mathbf{\Gamma}^\top\mathbf{M}\mathbf{\Gamma})^{-1} + 2(\mathbf{M}+\mathbf{U})^{-1}\,\mathbf{\Gamma}\left(\mathbf{\Gamma}^\top(\mathbf{M}+\mathbf{U})^{-1}\mathbf{\Gamma}\right)^{-1}. \tag{10}$$

The 1D algorithm and the ECD algorithm

Instead of optimizing $F(\mathbf{\Gamma})$ over the u-dimensional manifold, which is usually intractable, Cook and Zhang (2016) extracted one dimension in the envelope at a time and solved the problem sequentially by breaking down the u-dimensional manifold optimization problem into a series of one-dimensional manifold optimization problems.

The 1D algorithm starts with initializing $\mathbf{g}_0 = \mathbf{G}_0 = \mathbf{0}$. Let $\mathbf{G}_k = (\mathbf{g}_1,\ldots,\mathbf{g}_k)$ and $(\mathbf{G}_k, \mathbf{G}_{0k})$ be an orthogonal basis matrix for $\mathbb{R}^p$. Then given the envelope dimension $\dim(\mathcal{E}_{\mathbf{M}}(\mathbf{U})) = u$, repeat the following steps for $k = 0,\ldots,u-1$:

(i) Calculate $\mathbf{U}_k = \mathbf{G}_{0k}^\top \mathbf{U} \mathbf{G}_{0k}$ and $\mathbf{M}_k = \mathbf{G}_{0k}^\top \mathbf{M} \mathbf{G}_{0k}$, and construct the following stepwise objective function for $\mathbf{w} \in \mathbb{R}^{p-k}$,

$$f_k(\mathbf{w}) = \log(\mathbf{w}^\top \mathbf{M}_k \mathbf{w}) + \log(\mathbf{w}^\top (\mathbf{M}_k + \mathbf{U}_k)^{-1} \mathbf{w}). \tag{11}$$

(ii) Solve $\mathbf{w}_{k+1} = \arg\min f_k(\mathbf{w})$, subject to $\mathbf{w}^\top \mathbf{w} = 1$.

(iii) Set $\mathbf{g}_{k+1} = \mathbf{G}_{0k} \mathbf{w}_{k+1}$ as the $(k+1)$th direction of the envelope.

At termination, $\mathbf{G}_u = (\mathbf{g}_1, \ldots, \mathbf{g}_u)$ satisfies $\mathrm{span}(\mathbf{G}_u) = \mathcal{E}_{\mathbf{M}}(\mathbf{U})$ in population.

The gradient of $f_k(\mathbf{w})$ can also be explicitly expressed as

$$\frac{\partial}{\partial \mathbf{w}} f_k(\mathbf{w}) = \frac{2\mathbf{M}_k \mathbf{w}}{\mathbf{w}^\top \mathbf{M}_k \mathbf{w}} + \frac{2(\mathbf{M}_k + \mathbf{U}_k)^{-1} \mathbf{w}}{\mathbf{w}^\top (\mathbf{M}_k + \mathbf{U}_k)^{-1} \mathbf{w}}, \tag{12}$$

so the gradient-based algorithm can be adaptively applied in minimizing $f_k(\mathbf{w})$. Compared to the FG algorithm, the 1D algorithm optimizes $f_k(\mathbf{w})$ over an one-dimensional manifold at a time, which takes fewer parameters into consideration.

More recently, a faster algorithm called the envelope coordinate descent (ECD, Cook and Zhang 2018) was proposed to minimize the objective function $f_k(\mathbf{w})$. The key idea of the ECD algorithm is to separate the coordinates of $\mathbf{w}$ by transformation and iteratively updating each coordinate of the transformed $\mathbf{w}$ by coordinate descent, which in fact speeds up the algorithm and makes the optimization more accurate.

It is worth noting that the FG algorithm, the 1D algorithm and the ECD algorithm are all proposed to minimize or approximately minimize the objective function $F(\boldsymbol{\Gamma})$. But we prefer the 1D algorithm since it is faster, more stable than the FG algorithm, and does not require a carefully chosen initial value. Next, we introduce the PLS-type algorithm that is not based on solving the objective function.

The PLS-type algorithm

The PLS-type algorithm is a sequential algorithm for envelope estimation motivated by the SIMPLS (statistically inspired modification of the partial least squares; De Jong 1993) algorithm in the partial least squares (PLS) literature, and can efficiently estimate the envelope under the TPR model. To estimate the envelope $\mathcal{E}_{\mathbf{M}}(\mathbf{U})$, given $\dim(\mathcal{E}_{\mathbf{M}}(\mathbf{U})) = u$, the algorithm starts with setting $\mathbf{w}_0 = \mathbf{0}$ and $\boldsymbol{\Gamma}_0 = \mathbf{w}_0$. Then for $k = 0, \ldots, u-1$,

$$\begin{aligned} \mathcal{E}_k &= \mathrm{span}(\mathbf{M}\boldsymbol{\Gamma}_k), \\ \mathbf{w}_{k+1} &= \mathbf{v}_{\max}(\mathbf{Q}_{\mathcal{E}_k} \mathbf{U}\mathbf{U}^\top \mathbf{Q}_{\mathcal{E}_k}), \\ \boldsymbol{\Gamma}_{k+1} &= (\mathbf{w}_0, \ldots, \mathbf{w}_k, \mathbf{w}_{k+1}), \end{aligned}$$

where $\mathbf{v}_{\max}(\mathbf{A})$ denotes the eigenvector that is associated with the largest eigenvalue of the matrix $\mathbf{A}$, and $\mathbf{Q}_{\mathcal{E}_k}$ is the projection matrix onto the orthogonal complement of the subspace $\mathcal{E}_k$. At termination, we have $\mathcal{E}_{\mathbf{M}}(\mathbf{U}) = \mathrm{span}(\boldsymbol{\Gamma}_u)$.

4.3. Illustrations of algorithms

In Section 3, we showed that by passing different options to the argument `method`, the functions `TRR.fit()` and `TPR.fit()` implement different algorithms in tensor regression. In fact,

Function name	Usage	Section
`simplsMU()`	The SIMPLS-type moment-based algorithm.	4.3
`ECD()`	The envelope coordinate descent (ECD) algorithm.	4.3
`manifold1D()`	The 1D algorithm based on the R package **ManifoldOptim**.	4.3
`OptM1D()`	The 1D algorithm based on the MATLAB package **OptM**.	4.3
`manifoldFG()`	The FG algorithm based on the R package **ManifoldOptim**.	4.3
`OptMFG()`	The FG algorithm based on the MATLAB package **OptM**.	4.3
`oneD_bic()`	Selects the envelope dimension via 1D-BIC.	4.4
`MenvU_sim()`	Generates matrices $\mathbf{M}$ and $\mathbf{U}$ for the generic envelope estimation problem.	4.3

Table 5: Core and auxiliary functions in package **TRES**.

for envelope estimation algorithms, the functions `TRR.fit()` and `TPR.fit()` first construct the matrices $\widehat{\mathbf{M}}$ and $\widehat{\mathbf{U}}$ from the data `x` and `y`, then invoke different core functions in the package **TRES** to implement the corresponding algorithms. We also demonstrate with examples another auxiliary function `MenvU_sim()` which generates matrices $\mathbf{M}$ and $\mathbf{U}$ for the generic envelope estimation problem. All the core and auxiliary functions are listed in Table 5.

In this section, we compare the performances of different core functions implementing the FG algorithm, 1D algorithm, ECD algorithm and PLS algorithm with the simulated data. To be more specific, we compare the core function of the ECD algorithm, `ECD()`, the core function of the PLS algorithm, `simplsMU()`, the core functions of the FG algorithm, `manifoldFG()` and `OptMFG()`, and the core functions of the 1D algorithm, `manifold1D()` and `OptM1D()`. Note that the functions `manifoldFG()` and `manifold1D()` are based on the R package **ManifoldOptim**, while the functions `OptMFG()` and `OptM1D()` are based on the MATLAB package **OptM**. The six core functions are used in a similar way as follows:

```
simplsMU(M, U, u)
ECD(M, U, u, ...)
manifold1D(M, U, u, ...)
OptM1D(M, U, u, ...)
manifoldFG(M, U, u, Gamma_init = NULL, ...)
OptMFG(M, U, u,  Gamma_init = NULL, ...)
```

where the arguments are described below:

- `M`, `U`: The input matrices $\mathbf{M}$ and $\mathbf{U}$ for which the envelope $\mathcal{E}_{\mathbf{M}}(\mathbf{U})$ is estimated.
- `u`: An integer between 0 and p, representing the dimension of the envelope.
- `Gamma_init`: The initial envelope subspace basis for FG algorithms `manifoldFG()` and `OptMFG()`.
- `...`: Additional user-specified arguments. The acceptable arguments are different for different functions, while the common ones include the maximal number of iterations `maxiter` and the tolerance of convergence `tol`. The default values of `maxiter` and `tol` are 500 and 10^{-8}.

The FG algorithm functions `manifoldFG()` and `OptMFG()` require the initial envelope basis `Gamma_init` to start the iteration, as we already mentioned in Section 3.1. By default, the estimated envelope basis from 1D algorithms, `manifold1D()` and `OptM1D()`, are passed to `manifoldFG()` and `OptMFG()` as initial values. For the functions `manifold1D()` and `manifoldFG()`, the type of optimization method supported by package **ManifoldOptim** can also be passed as an additional argument. And for the functions `OptM1D()` and `OptMFG()`, the tolerances of different convergence criteria are acceptable as additional arguments. The outputs of the six core functions above are the estimated envelope basis matrices.

The simulation settings are adopted from Cook and Zhang (2018). Under the envelope parameterization (7), the matrices $\mathbf{M}, \mathbf{U} \in \mathbb{R}^{p\times p}$ in three models are given as

$$\mathbf{U} = \mathbf{\Gamma}\mathbf{\Phi}\mathbf{\Gamma}^\top,\ \mathbf{M} = \begin{cases} \mathbf{\Gamma}\mathbf{\Omega}\mathbf{\Gamma}^\top + \mathbf{\Gamma}_0\mathbf{\Omega}_0\mathbf{\Gamma}_0^\top, & \text{(M1)} \\ \mathbf{\Gamma}\mathbf{\Gamma}^\top + 0.01\mathbf{\Gamma}_0\mathbf{\Gamma}_0^\top, & \text{(M2)} \\ 0.01\mathbf{\Gamma}\mathbf{\Gamma}^\top + \mathbf{\Gamma}_0\mathbf{\Gamma}_0^\top. & \text{(M3)} \end{cases}$$

Note that Model (M2) and Model (M3) are special cases of Model (M1) by specifying $\mathbf{\Omega} = \mathbf{I}_u, \mathbf{\Omega}_0 = 0.01\mathbf{I}_{p-u}$ and $\mathbf{\Omega} = 0.01\mathbf{I}_u, \mathbf{\Omega}_0 = \mathbf{I}_{p-u}$, respectively. After the matrix $\mathbf{M}$ in Model (M1) is generated, the matrix $0.00001\mathbf{I}_p$ is added to $\mathbf{M}$ to guarantee its positive-definiteness. We use the function `MenvU_sim()` to generate the matrices $\mathbf{M}$, $\mathbf{U}$ and the envelope basis $\mathbf{\Gamma}$:

```
MenvU_sim(p, u, Omega = NULL, Omega0 = NULL, Phi = NULL, jitter = FALSE,
  wishart = FALSE, n = NULL)
```

The arguments are described as follows:

- `p`: The dimension of the p-by-p matrix $\mathbf{M}$.
- `u`: The envelope dimension.
- `Omega, Omega0, Phi`: The matrices $\mathbf{\Omega}$, $\mathbf{\Omega}_0$, and $\mathbf{\Phi}$ in parameterization (7). The default for `Omega` is $\mathbf{\Omega} = \mathbf{A}\mathbf{A}^\top$ where $\mathbf{A} \in \mathbb{R}^{u\times u}$ is a randomly generated matrix, elementwise from the Uniform(0, 1) distribution. The matrices $\mathbf{\Omega}_0$ and $\mathbf{\Phi}$ are generated in the same way as $\mathbf{\Omega}$ with $\mathbf{A}$ matching the dimensions of $\mathbf{\Omega}_0$ and $\mathbf{\Phi}$.
- `jitter`: A logical or numeric value. If it is numeric, a scalar matrix `jitter * diag(1, p, p)` is added to matrix $\mathbf{M}$ to ensure its positive definiteness. If it is `TRUE`, `jitter` is set as `1e-5`. If it is `FALSE`, no jitter is added.
- `wishart`: A logical value. If it is `TRUE`, after $\mathbf{M}$ and $\mathbf{U}$ are constructed, two samples from the Wishart distribution $W_p(\mathbf{M}/n, n)$ and $W_p(\mathbf{U}/n, n)$ are output as `M` and `U`.
- `n`: An integer value denoting the sample size, which is required if `wishart = TRUE`.

In all three models, $\mathbf{\Gamma} \in \mathbb{R}^{p\times u}$ is randomly generated elementwise from the Uniform$(0, 1)$ distribution and then transformed to a semi-orthogonal matrix. The matrix $\mathbf{\Gamma}_0 \in \mathbb{R}^{p\times(p-u)}$ is the orthogonal completion of $\mathbf{\Gamma}$ such that the basis $(\mathbf{\Gamma}, \mathbf{\Gamma}_0)$ is orthogonal. The outputs of the function `MenvU_dim` consist of the matrices `M`, `U` and the true envelope basis `Gamma`.

Models	PLS	ECD	1D (**Mani.**)	1D (**OptM**)	FG (**Mani.**)	FG (**OptM**)
M1	0.0033	0.0279	0.0705	0.1376	0.0762	0.1415
M2	0.0034	0.0214	0.0478	0.0372	0.0529	0.0393
M3	0.0033	0.0205	0.0463	0.0362	0.0592	0.0390

Table 6: Median of the execution time (in seconds) for each function on the simulated dataset. The maximum standard errors of the execution time for the three models are 0.01, 0.001, 0.001. The median of the estimation error is less than `1e-7` for all methods at each of these settings and is thus not reported in the table.

Then under each model, we fix $p = 20, u = 5$ and generate 50 pairs of population matrices $\mathbf{M}$ and $\mathbf{U}$ using `MenvU_sim()` with the argument `wishart` set as `FALSE`. We record the median execution time (in seconds) and the median estimation error $\bar{\mathcal{D}}(\mathcal{S}_{\widehat{\boldsymbol{\Gamma}}}, \mathcal{S}_{\boldsymbol{\Gamma}})$ over 50 replicates. The median of the execution time is reported in Table 6. The median of the estimation error is less than `1e-7` for all methods at each of these settings and is thus not reported in the table. The six core functions for estimating the envelope $\mathcal{E}_{\mathbf{M}}(\mathbf{U})$ are used as follows:

```
R> p <- 20
R> u <- 5
R> data <- MenvU_sim(p, u, jitter = 1e-5)
R> Gamma <- data$Gamma
R> M <- data$M
R> U <- data$U
R> G <- vector("list", 8)
R> G[[1]] <- simplsMU(M, U, u)
R> G[[2]] <- ECD(M, U, u)
R> G[[3]] <- manifold1D(M, U, u)
R> G[[4]] <- OptM1D(M, U, u)
R> G[[5]] <- manifoldFG(M, U, u)
R> G[[6]] <- OptMFG(M, U, u)
```

We compare the estimation error of the envelope basis using the function `subspace()`:

```
R> d <- rep(NA_real_, 8)
R> for (i in 1:6) {
+    d[i] <- subspace(G[[i]], Gamma)
+  }
R> d[1:6]

[1] 1.300e-08 5.650e-13 1.369e-07 2.574e-08 1.315e-07 2.434e-08
```

The results show that all functions estimate the envelope basis accurately and there is almost no difference among the estimation errors.

We are also interested in how the initial value impacts the performance of the FG algorithm. By passing a randomly generated $p \times u$ matrix `A` as initial value, we obtain another two results from `manifoldFG()` and `OptMFG()`. We compare the estimation errors as follows.

```
R> A <- matrix(runif(p * u), p, u)
R> G[[7]] <- manifoldFG(M, U, u, Gamma_init = A)
R> G[[8]] <- OptMFG(M, U, Gamma_init = A)
R> for (i in 7:8) {
+   d[i] <- subspace(G[[i]], Gamma)
+ }
R> d[5:8]

[1] 1.315e-07 2.434e-08 6.325e-01 7.216e-01
```

Obviously, with the random initial value, the FG algorithm produces a poor estimation of ethe nvelope that is far from the true envelope. But with a good initial value, e.g., the 1D estimator, the FG algorithm provides accurate estimation.

4.4. Dimension selection

We mentioned the 1D-BIC criterion for selecting envelope dimension under the TRR model in Section 3.3. The function `oneD_bic()` in the **TRES** package applies the 1D-BIC criterion and selects the envelope dimension. Specifically, the 1D-BIC criterion selects the envelope dimension via minimizing the following function,

$$\mathcal{I}_n(m) = \sum_{k=1}^{m} f_{k,n}(\widehat{\mathbf{w}}_k) + \frac{C \cdot m \cdot \log(n)}{n}, \quad m = 1, \dots, p, \tag{13}$$

where $C > 0$ is a constant, n is the sample size, and the function $f_{k,n}(\mathbf{w})$ is the sample version of the objective function $f_k(\mathbf{w})$ in (11) by replacing $\mathbf{M}, \mathbf{U}$ with $\sqrt{n}$-consistent estimators $\widehat{\mathbf{M}}, \widehat{\mathbf{U}}$. As suggested by Zhang and Mai (2018), the number C should be set as 1 when there is no additional model assumption or prior information. However, if additional model assumption or prior information are known, C should be set such that Cm best matches the degree-of-freedom or total number of free parameters of the model or estimation procedure. For example, in the TRR model, C should be set as the dimension of the predictor. We refer readers to Zhang and Mai (2018) for the detailed discussion on the role of the constant C. Then we select the envelope dimension as $\widehat{u} = \arg\min_{0 \le m \le p} \mathcal{I}_n(m)$. Note that the 1D-BIC criterion does not rely on the regression model and can be applied in any model-free context. The function `oneD_bic()` is used as:

```
oneD_bic(M, U, n, C = 1, maxdim = 10, ...)
```

where the arguments are:

- `M, U, ...`: The same as the ones in function `OptM1D` introduced in Section 4.3.
- `n`: The sample size.
- `C`: The constant used in the definition of 1D-BIC criterion (13).
- `maxdim`: The maximum dimension to be considered. The default value is 10.

The output of `oneD_bic()` is a list consisting of a vector `bicval` which contains the BIC values for different envelope dimensions, the selected dimension `u` which corresponds to the smallest BIC, and the estimated envelope basis `Gamma`.

To mimic the finite sample behavior and investigate how the sample size influences the behavior of the 1D-BIC criterion, we sample $\widehat{\mathbf{M}}$ and $\widehat{\mathbf{U}}$ from the Wishart distribution $W_p(\mathbf{M}/n, n)$ and $W_p(\mathbf{U}/n, n)$ separately, which can be done by the function `MenvU_sim()` with the argument `wishart` set as `TRUE`. Here the matrices $\mathbf{M} \in \mathbb{R}^{p\times p}$ and $\mathbf{U} \in \mathbb{R}^{p\times p}$ are randomly generated from model (M1) and n is the sample size. We fix $p = 50$, the envelope dimension $u = 5$, and vary the sample size $n = 50, 70, 100, 200, 400, 800$. We omit `bicval` and `Gamma` from the output of the function `oneD_bic` since we are only interested in the selected dimension.

```
R> p <- 50
R> u <- 5
R> n0 <- c(50, 70, 100, 200, 400, 800)
R> uhat3 <- rep(NA_integer_, length(n0))
R> for (i in seq_along(n0)) {
+    n <- n0[i]
+    data <- MenvU_sim(p, u, jitter = 1e-5, wishart = TRUE, n = n)
+    M <- data$M
+    U <- data$U
+    output <- oneD_bic(M, U, n, maxdim = p/2)
+    uhat3[i] <- output$u
+ }
R> uhat3

11  9  5  5  5  5
```

The results indicate that as n increases, the function `oneD_bic()` consistently selects the correct dimension.

5. Discussion

The R package **TRES** implements several estimation procedures for the tensor regression models. It also provides a general framework for both the model-based and model-free envelope estimation. Inferential tools such as the tensor regression coefficient p-value map and envelope dimension selection are also provided. In this paper, we demonstrate the main functions of the package for fitting two tensor regression models and various envelope estimation algorithms.

Computational details

Computations were done on a macOS High Sierra 10.13.6 with Intel Core i5 CPU@1.6GHz processor, 8GB memory (RAM). The results in this paper were obtained using R 3.6.3 with Fortran library **LAPACK** 3.9.0 (Anderson *et al.* 1999) and R packages **TRES** 1.1.4, **ManifoldOptim** 1.0.0 (Martin *et al.* 2020), **MASS** 7.3.52 (Venables and Ripley 2002), **methods** 3.6.3,

pracma 2.2.9 (Borchers 2019), **Rcpp** 1.0.5 (Eddelbuettel and François 2011), **rTensor** 1.4.1 (Li *et al.* 2018a), **remotes** 2.2.0 (Hester, Csárdi, Wickham, Chang, Morgan, and Tenenbaum 2020), **stats** 3.6.3. Slightly different results might be obtained, in particular also depending on the linear algebra library used. The R packages are all available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/`. The package **TRES** will be updated on CRAN and the first author's Github repository. One way to download from Github is:

```
R> library("remotes")
R> install_github("leozeng15/TRES")
```

Acknowledgments

The authors are grateful to the Journal of Statistical Software editorial team and two anonymous reviewers for very helpful comments that led to significant improvements in this article and the **TRES** package. The authors also thank Dr. Daniel Eck from University of Illinois Urbana-Champaign and Dr. Dootika Vats from Indian Institute of Technology Kanpur, for helpful discussion. Research for this paper was supported in part by grants CCF-1908969, CCF-1617691 and DMS-1613154 from the U.S. National Science Foundation.

References

Adragni KP, Raim A (2014). "**ldr**: An R Software Package for Likelihood-Based Sufficient Dimension Reduction." *Journal of Statistical Software*, **61**(3), 1–21. `doi:10.18637/jss.v061.i03`.

Anderson E, Bai Z, Bischof C, Blackford S, Demmel J, Dongarra J, Du Croz J, Greenbaum A, Hammarling S, McKenney A, Sorensen D (1999). ***LAPACK** Users' Guide*. 3rd edition. Society for Industrial and Applied Mathematics, Philadelphia.

Borchers HW (2019). ***pracma**: Practical Numerical Math Functions*. R package version 2.2.5, URL `https://CRAN.R-project.org/package=pracma`.

Cichocki A, Mandic D, De Lathauwer L, Zhou G, Zhao Q, Caiafa C, Phan HA (2015). "Tensor Decompositions for Signal Processing Applications: From Two-Way to Multiway Component Analysis." *IEEE Signal Processing Magazine*, **32**(2), 145–163. `doi:10.1109/msp.2013.2297439`.

Cook RD (2018). *An Introduction to Envelopes: Dimension Reduction for Efficient Estimation in Multivariate Statistics*. John Wiley & Sons.

Cook RD, Forzani L (2009). "Likelihood-Based Sufficient Dimension Reduction." *Journal of the American Statistical Association*, **104**(485), 197–208. `doi:10.1198/jasa.2009.0106`.

Cook RD, Forzani LM, Tomassi DR (2011). "**LDR**: A Package for Likelihood-Based Sufficient Dimension Reduction." *Journal of Statistical Software*, **39**(3), 1–20. `doi:10.18637/jss.v039.i03`.

Cook RD, Helland IS, Su Z (2013). "Envelopes and Partial Least Squares Regression." *Journal of the Royal Statistical Society B*, **75**(5), 851–877. doi:10.1111/rssb.12018.

Cook RD, Li B, Chiaromonte F (2010). "Envelope Models for Parsimonious and Efficient Multivariate Linear Regression." *Statistica Sinica*, **20**(3), 927–960.

Cook RD, Ni L (2005). "Sufficient Dimension Reduction via Inverse Regression: A Minimum Discrepancy Approach." *Journal of the American Statistical Association*, **100**(470), 410–428. doi:10.1198/016214504000001501.

Cook RD, Su Z, Yang Y (2015). "**envlp**: A MATLAB Toolbox for Computing Envelope Estimators in Multivariate Analysis." *Journal of Statistical Software*, **62**(8), 1–20. doi:10.18637/jss.v062.i08.

Cook RD, Weisberg S (1991). "Sliced Inverse Regression for Dimension Reduction: Comment." *Journal of the American Statistical Association*, **86**(414), 328–332.

Cook RD, Zhang X (2015a). "Foundations for Envelope Models and Methods." *Journal of the American Statistical Association*, **110**(510), 599–611. doi:10.1080/01621459.2014.983235.

Cook RD, Zhang X (2015b). "Simultaneous Envelopes for Multivariate Linear Regression." *Technometrics*, **57**(1), 11–25. doi:10.1080/00401706.2013.872700.

Cook RD, Zhang X (2016). "Algorithms for Envelope Estimation." *Journal of Computational and Graphical Statistics*, **25**(1), 284–300. doi:10.1080/10618600.2015.1029577.

Cook RD, Zhang X (2018). "Fast Envelope Algorithms." *Statistica Sinica*, **28**(3), 1179–1197. doi:10.5705/ss.202016.0037.

De Jong S (1993). "SIMPLS: An Alternative Approach to Partial Least Squares Regression." *Chemometrics and Intelligent Laboratory Systems*, **18**(3), 251–263. doi:10.1016/0169-7439(93)85002-x.

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

Hester J, Csárdi G, Wickham H, Chang W, Morgan M, Tenenbaum D (2020). ***remotes**: R Package Installation from Remote Repositories, Including GitHub*. R package version 2.2.0, URL https://CRAN.R-project.org/package=remotes.

Hoff PD (2015). "Multilinear Tensor Regression for Longitudinal Relational Data." *The Annals of Applied Statistics*, **9**(3), 1169–1193. doi:10.1214/15-aoas839.

Hore V, Viñuela A, Buil A, Knight J, McCarthy MI, Small K, Marchini J (2016). "Tensor Decomposition for Multiple-Tissue Gene Expression Experiments." *Nature Genetics*, **48**(9), 1094. doi:10.1038/ng.3624.

Kolda TG, Bader BW (2009). "Tensor Decompositions and Applications." *SIAM Review*, **51**(3), 455–500. doi:10.1137/07070111x.

Lee M, Su Z (2020). ***Renvlp**: Computing Envelope Estimators.* R package version 2.8, URL https://CRAN.R-project.org/package=Renvlp.

Li B (2018). *Sufficient Dimension Reduction: Methods and Applications with R*. Chapman & Hall/CRC. doi:10.1201/9781315119427.

Li J, Bien J, Wells MT (2018a). "**rTensor**: An R Package for Multidimensional Array (Tensor) Unfolding, Multiplication, and Decomposition." *Journal of Statistical Software*, **87**(10), 1–31. doi:10.18637/jss.v087.i10.

Li KC (1991). "Sliced Inverse Regression for Dimension Reduction." *Journal of the American Statistical Association*, **86**(414), 316–327. doi:10.1080/01621459.1991.10475035.

Li KC (1992). "On Principal Hessian Directions for Data Visualization and Dimension Reduction: Another Application of Stein's Lemma." *Journal of the American Statistical Association*, **87**(420), 1025–1039. doi:10.1080/01621459.1992.10476258.

Li L, Zhang X (2017). "Parsimonious Tensor Response Regression." *Journal of the American Statistical Association*, **112**(519), 1131–1146. doi:10.1080/01621459.2016.1193022.

Li X, Xu D, Zhou H, Li L (2018b). "Tucker Tensor Regression and Neuroimaging Analysis." *Statistics in Biosciences*, **10**(3), 520–545. doi:10.1007/s12561-018-9215-6.

Manceur AM, Dutilleul P (2013). "Maximum Likelihood Estimation for the Tensor Normal Distribution: Algorithm, Minimum Sample Size, and Empirical Bias and Dispersion." *Journal of Computational and Applied Mathematics*, **239**, 37–49. doi:10.1016/j.cam.2012.09.017.

Martin S, Raim AM, Huang W, Adragni KP (2020). "**ManifoldOptim**: An R Interface to the **ROPTLIB** Library for Riemannian Manifold Optimization." *Journal of Statistical Software*, **93**(1), 1–32. doi:10.18637/jss.v093.i01.

R Core Team (2021). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Su Z, Cook RD (2011). "Partial Envelopes for Efficient Estimation in Multivariate Linear Regression." *Biometrika*, **98**(1), 133–146. doi:10.1093/biomet/asq063.

The MathWorks Inc (2018). *MATLAB – The Language of Technical Computing, Version 8 (R2018b)*. Natick. URL http://www.mathworks.com/products/matlab/.

Venables WN, Ripley BD (2002). *Modern Applied Statistics with S*. 4th edition. Springer-Verlag, New York.

Wang W, Zeng J, Zhang X (2021). ***TRES**: Tensor Regression with Envelope Structure*. R package version 1.1.5, URL https://CRAN.R-project.org/package=TRES.

Wang X, Zhu H, ADNI (2017). "Generalized Scalar-On-Image Regression Models via Total Variation." *Journal of the American Statistical Association*, **112**(519), 1156–1168. doi:10.1080/01621459.2016.1194846.

Weisberg S (2002). "Dimension Reduction Regression in R." *Journal of Statistical Software*, **7**(1), 1–22. doi:10.18637/jss.v007.i01.

Wen Z, Yin W (2013). "A Feasible Method for Optimization with Orthogonality Constraints." *Mathematical Programming*, **142**(1–2), 397–434. doi:10.1007/s10107-012-0584-1.

Zhang X, Li L (2017). "Tensor Envelope Partial Least Squares Regression." *Technometrics*, **59**(4), 426–436. doi:10.1080/00401706.2016.1272495.

Zhang X, Mai Q (2018). "Model-Free Envelope Dimension Selection." *Electronic Journal of Statistics*, **12**(2), 2193–2216. doi:10.1214/18-ejs1449.

Zhang X, Mai Q (2019). "Efficient Integration of Sufficient Dimension Reduction and Prediction in Discriminant Analysis." *Technometrics*, **61**(2), 259–272. doi:10.1080/00401706.2018.1512901.

Zhou H (2013). "MATLAB **TensorReg** Toolbox Version 1.0." URL https://hua-zhou.github.io/TensorReg/.

Zhou H, Li L, Zhu H (2013). "Tensor Regression with Applications in Neuroimaging Data Analysis." *Journal of the American Statistical Association*, **108**(502), 540–552. doi:10.1080/01621459.2013.776499.

Zhu R, Zhang J, Zhao R, Xu P, Zhou W, Zhang X (2019). "**orthoDr**: Semiparametric Dimension Reduction via Orthogonality Constrained Optimization." *The R Journal*, **11**(2), 24–37. doi:10.32614/rj-2019-006.

Affiliation:

Jing Zeng, Wenjing Wang, Xin Zhang
Department of Statistics
Florida State University
Tallahassee, FL 32306, United States of America
E-mail: jing.zeng@stat.fsu.edu, wenjing.wang@stat.fsu.edu, henry@stat.fsu.edu
URL: https://ani.stat.fsu.edu/~henry/

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/

October 2021, Volume 99, Issue 12 *Submitted:* 2019-11-21
doi:10.18637/jss.v099.i12 *Accepted:* 2021-02-08