SEEDCCA: An Integrated R-Package for Canonical Correlation Analysis and Partial Least Squares

by Bo-Young Kim, Yunju Im and Jae Keun Yoo

Abstract Canonical correlation analysis (CCA) has a long history as an explanatory statistical method in high-dimensional data analysis and has been successfully applied in many science fields such as chemomtrics, pattern recognition, genomic sequence analysis and so on. The so-called seedCCA is a newly developed **R** package, and it implements not only the standard and seeded CCA but also partial least squares. The package enables us to fit CCA to large-p and small-n data. The paper provides a complete guide. Also, the seeded CCA application results are compared with the regularized CCA in the existing **R** package. It is believed that the package along with the paper will contribute to high-dimensional data analysis in various science field practitioners and that the statistical methodologies in multivariate analysis become more fruitful.

Introduction

Explanatory studies are important to identify patterns and special structure in data prior to develop a specific model. When a study between two sets of a p-dimensional random variables $\mathbf{X}$ ($\mathbf{X} \in \mathbb{R}^p$) and a r-dimensional random variable $\mathbf{Y}$ ($\mathbf{Y} \in \mathbb{R}^r$) are of primary interest, one of the popular explanatory statistical methods would be canonical correlation analysis (CCA; Hotelling (1936)). The main goal of CCA is the dimension reduction of two sets of variables by measuring an association between the two sets. For this, pairs of linear combinations of variables are constructed by maximizing the Pearson correlation. The CCA has successful application in many science fields such as chemomtrics, pattern recognition, genomic sequence analysis and so on.

In Lee and Yoo (2014) it is shown that the CCA can be used as a dimension reduction tool for high-dimensional data, but also it is connected to least square estimator. Therefore, the CCA is not only explanatory and dimension reduction method but also can be utilized as alternative of least square estimation.

If $\max(p, r)$ is bigger than or equal to the sample size, n, usual CCA application is not plausible due to no incapability of inverting sample covariance matrices. To overcome this, a regularized CCA is developed by Leurgans et al. (1993), whose idea was firstly suggested in Vinod (1976). In practice, the **CCA** package by González et al. (2008) can implement a version of the regularized CCA. To make the sample covariance matrices saying $\hat{\mathbf{\Sigma}}_x$ and $\hat{\mathbf{\Sigma}}_y$, invertible, in González et al. (2008), they are replaced with

$$\hat{\mathbf{\Sigma}}_x^{\lambda_1} = \hat{\mathbf{\Sigma}}_x + \lambda_1 \mathbf{I}_p \text{ and } \hat{\mathbf{\Sigma}}_y^{\lambda_2} = \hat{\mathbf{\Sigma}}_y + \lambda_1 \mathbf{I}_r.$$

The optimal values of λ_1 and λ_2 are chosen by maximizing a cross-validation score throughout the two-dimensional grid search. Although it is discussed that a relatively small grid of reasonable values for λ_1 and λ_2 can lesson intensive computing in González et al. (2008), it is still time-consuming as observed in later sections. Additionally, fast regularized CCA and robust CCA via projection-pursuit are recently developed in Cruz-Cano (2012) and Alfons et al. (2016), respectively.

Another version of CCA to handle $\max(p, r) > n$ is the so-called seeded canonical correlation analysis proposed by Im et al. (2014). Since the seeded CCA does not require any regularization procedure, which is computationally intensive, its implementation to larger data is quite fast. The seeded CCA requires two steps. In the initial step, a set of variables bigger than n is initially reduced based on iterative projections. In next step, the standard CCA is applied to two sets of variables acquired from the initial step to finalize CCA of data. Another advantage is that the procedure of the seeded CCA has a close relation with partial least square, which is one of the popular statistical methods for large p-small n data, so the seed CCA can yield the PLS estimates.

The **seedCCA** package is recently developed mainly to implement the seeded CCA. However, the package can fit a collection of the statistical methodologies, which are standard canonical correlation and partial least squares with uni/multi-dimensional responses including the seeded CCA. The package has been already uploaded to CRAN (https://cran.r-project.org/web/packages/seedCCA/index.html).

The main goal of the paper is to introduce and illustrate the **seedCCA** package. For this, three real data are fitted by the standard CCA, the seeded CCA and partial least square, and two of the three data are available in the package. One of them has been analyzed in González et al. (2008). So, the

implementation results by the seeded and regularized CCA are closely compared.

The organization of the paper is as follows. The collection of three methodologies is discussed in Section 2. The implementation of **seedCCA** is illustrated and compared with **CCA** in Section 3. In Section 4, we summarize the work.

We will use the following notations throughout the rest of the paper. A p-dimensional random variable $\mathbf{X}$ will be denoted as $\mathbf{X} \in \mathbb{R}^p$. So, $\mathbf{X} \in \mathbb{R}^p$ means a random variable, although there is no specific mention. For $\mathbf{X} \in \mathbb{R}^p$ and $\mathbf{Y} \in \mathbb{R}^r$, we define that $\text{cov}(\mathbf{X}) = \mathbf{\Sigma}_x$, $\text{cov}(\mathbf{Y}) = \mathbf{\Sigma}_y$, $\text{cov}(\mathbf{X}, \mathbf{Y}) = \mathbf{\Sigma}_{xy}$ and $\text{cov}(\mathbf{Y}, \mathbf{X}) = \mathbf{\Sigma}_{yx}$. And, it is assumed that $\mathbf{\Sigma}_x$ and $\mathbf{\Sigma}_y$ are positive-definite.

Collection of implemented methodologies in seedCCA

Canonical correlation analysis

Suppose the two sets of variables $\mathbf{X} \in \mathbb{R}^p$ and $\mathbf{Y} \in \mathbb{R}^r$ and consider their linear combinations of $U = \mathbf{a}^T\mathbf{X}$ and $V = \mathbf{b}^T\mathbf{Y}$. Then we have $\text{var}(U) = \mathbf{a}^T\mathbf{\Sigma}_x\mathbf{a}$, $\text{var}(V) = \mathbf{b}^T\mathbf{\Sigma}_y\mathbf{b}$, and $\text{cov}(U, V) = \mathbf{a}^T\mathbf{\Sigma}_{xy}\mathbf{b}$, where $\mathbf{a} \in \mathbb{R}^{p\times 1}$ and $\mathbf{b} \in \mathbb{R}^{r\times 1}$. Then Pearson-correlation between U and V is as follows:

$$\text{cor}(U, V) = \frac{\mathbf{a}^T\mathbf{\Sigma}_{xy}\mathbf{b}}{\sqrt{\mathbf{a}^T\mathbf{\Sigma}_x\mathbf{a}}\sqrt{\mathbf{b}^T\mathbf{\Sigma}_y\mathbf{b}}}. \tag{1}$$

We seek to find $\mathbf{a}$ and $\mathbf{b}$ to maximize $\text{cor}(U, V)$ with satisfying the following criteria.

1. The first canonical variate pair ($U_1 = \mathbf{a}_1^T\mathbf{X}, V_1 = \mathbf{b}_1^T\mathbf{Y}$) is obtained from maximizing (1).
2. The second canonical variate pair ($U_2 = \mathbf{a}_2^T\mathbf{X}, V_2 = \mathbf{b}_2^T\mathbf{Y}$) is constructed from the maximization of (1) with restriction that $\text{var}(U_2) = \text{var}(V_2) = 1$ and (U_1, V_1) and (U_2, V_2) are uncorrelated.
3. At the k step, the kth canonical variate pair ($U_k = \mathbf{a}_k^T\mathbf{X}, V_k = \mathbf{b}_k^T\mathbf{Y}$) is obtained from the maximization of (1) with restriction that $\text{var}(U_k) = \text{var}(V_k) = 1$ and (U_k, V_k) are uncorrelated with the previous $(k-1)$ canonical variate pairs.
4. Repeat Steps 1 to 3 until k becomes q $(= \min(p, r))$.
5. Select the first d pairs of (U_k, V_k) to represent the relationship between $\mathbf{X}$ and $\mathbf{Y}$.

Under this criteria, the pairs $(\mathbf{a}_i, \mathbf{b}_i)$ are constructed as follows: $\mathbf{a}_i = \mathbf{\Sigma}_x^{-1/2}\psi_i$ and $\mathbf{b}_i = \mathbf{\Sigma}_y^{-1/2}\phi_i$ for $i = 1, \ldots, q$, where $(\psi_1, ..., \psi_q)$ and $(\phi_1, ..., \phi_q)$ are, respectively, the q eigenvectors of $\mathbf{\Sigma}_x^{-1/2}\mathbf{\Sigma}_{xy}\mathbf{\Sigma}_y^{-1}\mathbf{\Sigma}_{yx}\mathbf{\Sigma}_y^{-1/2}$ and $\mathbf{\Sigma}_y^{-1/2}\mathbf{\Sigma}_{yx}\mathbf{\Sigma}_x^{-1}\mathbf{\Sigma}_{xy}\mathbf{\Sigma}_x^{-1/2}$ with the corresponding common ordered-eigenvalues of $\rho_1^{*2} \geq \cdots \geq \rho_q^{*2} \geq 0$. Then matrices of $\mathbf{M}_x = (\mathbf{a}_1, ..., \mathbf{a}_d)$ and $\mathbf{M}_y = (\mathbf{b}_1, ..., \mathbf{b}_d)$ are called *canonical coefficient matrices* for $d = 1, ..., q$. Also, $\mathbf{M}_x^T\mathbf{X}$ and $\mathbf{M}_y^T\mathbf{Y}$ are called *canonical variates*. In sample, the population quantities are replaced with their usual moment estimators. For more details regarding this standard CCA, readers may refer Johnson and Wichern (2007).

Seeded canonical correlation analysis

Since the standard CCA application requires the inversion of $\hat{\mathbf{\Sigma}}_x$ and $\hat{\mathbf{\Sigma}}_y$ in practice, it is not plausible for high-dimensional data with $\max(p, r) > n$. In Im et al. (2014), a seeded canonical correlation analysis approach is proposed to overcome this deficit. The seeded CCA is a two-step procedure consisting of initialized and finalized steps. In the initialized step, the original two sets of variables are reduced to m-dimensional pairs without loss of information on the CCA application. In the initialized step, it is essential to force $m << n$. In the finalized step, the standard CCA is implemented to the initially-reduced pairs for the repairing and orthonormality. More detailed discussion on the seeded CCA is as follows in next subsections.

Development

Define a notation of $\mathcal{S}(\mathbf{M})$ as the subspace spanned by the columns of $\mathbf{M} \in \mathbb{R}^{p\times r}$. Lee and Yoo (2014) show the following relation:

$$\mathcal{S}(\mathbf{M}_x) = \mathcal{S}(\mathbf{\Sigma}_x^{-1}\mathbf{\Sigma}_{xy}) \text{ and } \mathcal{S}(\mathbf{M}_y) = \mathcal{S}(\mathbf{\Sigma}_y^{-1}\mathbf{\Sigma}_{yx}). \tag{2}$$

The relation in (2) directly indicates that $\mathbf{M}_x$ and $\mathbf{M}_y$ form basis matrices of $\mathcal{S}(\mathbf{\Sigma}_x^{-1}\mathbf{\Sigma}_{xy})$ and $\mathcal{S}(\mathbf{\Sigma}_y^{-1}\mathbf{\Sigma}_{yx})$ and that $\mathbf{M}_x$ and $\mathbf{M}_y$ can be restored from $\mathbf{\Sigma}_x^{-1}\mathbf{\Sigma}_{xy}$ and $\mathbf{\Sigma}_y^{-1}\mathbf{\Sigma}_{yx}$.

Now, we define the following two matrices:

$$\begin{aligned} \mathbf{R}_{x,u_1} \in \mathbb{R}^{p\times ru_1} &= (\mathbf{\Sigma}_{xy}, \mathbf{\Sigma}_x\mathbf{\Sigma}_{xy}, \ldots, \mathbf{\Sigma}_x^{u_1-1}\mathbf{\Sigma}_{xy}) \text{ and} \\ \mathbf{R}_{y,u_2} \in \mathbb{R}^{r\times pu_2} &= (\mathbf{\Sigma}_{yx}, \mathbf{\Sigma}_y\mathbf{\Sigma}_{yx}, \ldots, \mathbf{\Sigma}_y^{u_2-1}\mathbf{\Sigma}_{yx}). \end{aligned} \tag{3}$$

In $\mathbf{R}_{x,u_1}$ and $\mathbf{R}_{y,u_2}$, the numbers of u_1 and u_2 are called termination indexes, and they decide the number of projections of $\mathbf{\Sigma}_{xy}$ and $\mathbf{\Sigma}_{yx}$ onto $\mathbf{\Sigma}_x$ and $\mathbf{\Sigma}_y$, respectively. Also define that

$$\begin{aligned} \mathbf{M}^0_{x,u_1} \in \mathbb{R}^{p\times r} &= \mathbf{R}_{x,u_1}(\mathbf{R}_{x,u_1}^{\mathrm{T}}\mathbf{\Sigma}_x\mathbf{R}_{x,u_1})^{-1}\mathbf{R}_{x,u_1}^{\mathrm{T}}\mathbf{\Sigma}_{xy} \text{ and} \\ \mathbf{M}^0_{y,u_2} \in \mathbb{R}^{r\times p} &= \mathbf{R}_{y,u_2}(\mathbf{R}_{y,u_2}^{\mathrm{T}}\mathbf{\Sigma}_y\mathbf{R}_{y,u_2})^{-1}\mathbf{R}_{y,u_2}^{\mathrm{T}}\mathbf{\Sigma}_{yx}. \end{aligned} \tag{4}$$

In Cook et al. (2007), it is shown that $\mathcal{S}(\mathbf{M}^0_{x,u_1}) = \mathcal{S}(\mathbf{\Sigma}_x^{-1}\mathbf{\Sigma}_{xy})$ and $\mathcal{S}(\mathbf{M}^0_{y,u_2}) = \mathcal{S}(\mathbf{\Sigma}_y^{-1}\mathbf{\Sigma}_{yx})$ in (4), and hence $\mathbf{M}^0_{x,u_1}$ and $\mathbf{M}^0_{y,u_2}$ can be used to infer $\mathbf{M}_x$ and $\mathbf{M}_y$, respectively. One clear advantage to use $\mathbf{M}^0_{x,u_1}$ and $\mathbf{M}^0_{y,u_2}$ is no need of the inversion of $\mathbf{\Sigma}_x$ and $\mathbf{\Sigma}_y$.

Practically, it is important to select proper values for the termination indexes u_1 and u_2. For this define that $\Delta_{x,u_1} = \mathbf{M}^0_{x,u_1+1} - \mathbf{M}^0_{x,u_1}$ and $\Delta_{y,u_2} = \mathbf{M}^0_{y,u_2+1} - \mathbf{M}^0_{y,u_2}$. Finally, the following measure for increment of u_1 and u_2 is defined: $nF_{x,u_1} = n\text{trace}(\mathbf{\Delta}_{x,u_1}^{\mathrm{T}}\mathbf{\Sigma}_x\mathbf{\Delta}_{x,u_1})$ and $nF_{y,u_2} = n\text{trace}(\mathbf{\Delta}_{y,u_2}^{\mathrm{T}}\mathbf{\Sigma}_y\mathbf{\Delta}_{y,u_2})$. Then, a proper value of u is set to have little changes in nF_{x,u_1} and nF_{x,u_1+1} and in nF_{y,u_2} and nF_{y,u_2+1}. It is not necessary that the selected u_1 and u_2 for $\mathbf{M}^0_{x,u_1}$ and $\mathbf{M}^0_{y,u_2}$ are common.

Next, the original two sets of variables of $\mathbf{X}$ and $\mathbf{Y}$ are replaced with $\mathbf{M}^{0\,\mathrm{T}}_{x,u_1}\mathbf{X} \in \mathbb{R}^r$ and $\mathbf{M}^{0\,\mathrm{T}}_{y,u_2}\mathbf{Y} \in \mathbb{R}^p$. This reduction of $\mathbf{X}$ and $\mathbf{Y}$ does not cause any loss of information on CCA in sense that $\mathcal{S}(\mathbf{M}^0_{x,u_1}) = \mathcal{S}(\mathbf{M}_x)$ and $\mathcal{S}(\mathbf{M}^0_{y,u_2}) = \mathcal{S}(\mathbf{M}_y)$, and it is called *initialized CCA*. The initialized CCA has the following two cases.

> **case 1**: Suppose that $\min(p,r) = r << n$. Then, the original $\mathbf{X}$ alone is replaced with $\mathbf{M}^{0\,\mathrm{T}}_{x,u_1}\mathbf{X}$ and the original $\mathbf{Y}$ is kept.
>
> **case 2**: If $\min(p,r) = r$ is not fairly smaller than n, $\mathbf{\Sigma}_{xy}$ and $\mathbf{\Sigma}_{yx}$ are replaced by their m largest eigenvectors in the construction of $\mathbf{R}_{x,u_1}$, $\mathbf{R}_{y,u_2}$, $\mathbf{M}^0_{x,u_1}$ and $\mathbf{M}^0_{y,u_2}$. The following two ways to determine a proper value of m is recommended among many. One is graphical determination by a scree plot for eigenvalues of $\mathbf{\Sigma}_{xy}$. The other is the number of eigenvalues whose sum is to cover 90% or above of the total variations of $\mathbf{\Sigma}_{xy}$.

The primary goal in the initialized step is reduction of $\mathbf{X}$ and $\mathbf{Y}$ less than n without loss of information on CCA. In case 1, $\mathbf{X}$ and $\mathbf{Y}$ are reduced to r-dimensional variates, while they are replaced with the m-dimensional sets of variables in case 2. After the initialized step, r and m are fairly smaller than n.

The next step is to conduct the standard CCA for $\mathbf{M}^{0\,\mathrm{T}}_{x,u_1}\mathbf{X}$ and $\mathbf{M}^{0\,\mathrm{T}}_{y,u_2}\mathbf{Y}$ for the repairing and orthonormality. This CCA application is called *finalized CCA*. Finally, this two-step procedure for CCA is called *seeded CCA*.

Partial least squares

The main goal of the two CCA methods is dimension reduction based on the joint relation of $\mathbf{X}$ and $\mathbf{Y}$ rather than the conditional relation of $\mathbf{Y}|\mathbf{X}$. For simplicity, in this subsection, $\mathbf{Y}$ with $r = 1$ is assumed as a response variable in a regression of $\mathbf{Y}|\mathbf{X}$.

Recall $\mathbf{R}_{x,u_1}$ in (3) and $\mathbf{M}^0_{x,u_1}$ in (4):

$$\mathbf{R}_{x,u_1} = (\mathbf{\Sigma}_{xy}, \mathbf{\Sigma}_x\mathbf{\Sigma}_{xy}, \mathbf{\Sigma}_x^2\mathbf{\Sigma}_{xy}, \ldots, \mathbf{\Sigma}_x^{u_1-1}\mathbf{\Sigma}_{xy}) \text{ and } \mathbf{M}^0_{x,u_1} = \mathbf{R}_{x,u_1}(\mathbf{R}_{x,u_1}^{\mathrm{T}}\mathbf{\Sigma}_x\mathbf{R}_{x,u_1})^{-1}\mathbf{R}_{x,u_1}^{\mathrm{T}}\mathbf{\Sigma}_{xy}.$$

According to Helland (1990), the population partial least square (PLS) with u components on the regression of $\mathbf{Y}|\mathbf{X}$ is as follows:

$$\boldsymbol{\beta}_{u_1,\mathrm{PLS}} = \mathbf{M}^0_{x,u_1}. \tag{5}$$

It is noted that this PLS representation in (5) is equivalent to the canonical matrix for $\mathbf{X}$ via the seeded CCA.

Illustration of seedCCA package

Outline of seedCCA package

The methods discussed in the previous section are implemented through the main function `seedCCA`. The its arguments are as follows.

```
seedCCA(X, Y, type="seed2", ux=NULL, uy=NULL, u=10, eps=0.01, cut=0.9, d=NULL, AS=TRUE,
scale=FALSE)
```

The main function `seedCCA` returns "seedCCA" class and three subclasses depending on the values of `type`. The values of `type` and its resulting subclasses are as follows.

`type="cca"`: standard CCA ($\max(p,r) < n$ and $\min(p,r) > 1$) / "finalCCA" subclass

`type="cca"`: ordinary least squares ($\max(p,r) < n$ and $\min(p,r) = 1$) / "seedols" subclass

`type="seed1"`: seeded CCA with case1 ($\max(p,r) \geq n$) / "finalCCA" subclass

`type="seed2"`: seeded CCA with case2 ($\max(p,r) \geq n$) / "finalCCA" subclass

`type="pls"`: partial least squares ($p \geq n$ and $r < n$) / "seedpls" subclass

The function `seedCCA` prints out estimated canonical coefficient matrices for all subclasses and additionally does canonical correlations for "finalCCA" subclasses, although it produces more outputs. For details, the readers are recommended to run `?seedCCA` after loading **seedCCA** package. It should be noted that the **seedCCA** package must be loaded before using all functions in the package. The **seedCCA** package depends on two packages of **CCA** and **corpcor** (Schafer et al. (2017)).

For illustration purpose, three data sets will be considered. Pulp data is used for the standard CCA, which is available from author's webpage (`http://home.ewha.ac.kr/~yjkstat/pulp.txt`). For the seeded CCA along with the comparison with the regularized CCA and the partial least squares, ~~cooike~~ and `nutrimouse` in **seedCCA** package will be illustrated.

Standard CCA: pulp data

Pulp data is ~~for~~ measurements of properties of pulp fibers and the paper made from them. It contains two sets of variables with 62 sample sizes. The first set $\mathbf{Y}$ is for the pulp fiber characteristics, which are arithmetic fiber length, long fiber fraction, fine fiber fraction and zero span tensile. The second set $\mathbf{X}$ is regarding the paper properties such as breaking length, elastic modulus, stress at failure and burst strength. To implement the standard CCA application, the function `seedCCA` with `type="cca"` should be used. In this case, `seeCCA` results in "finalCCA" subclass. The function requires two matrix-type arguments, and it returns the following five components of `cor`, `xcoef`, `ycoef`, `Xscores` and `Yscores`. The first component is `cor` is the sample canonical correlations. The next two ones, `xcoef` and `ycoef`, are the estimated canonical matrices for $\mathbf{X}$ and $\mathbf{Y}$. The last two components, which are `Xscores` and `Yscores`, are the estimated canonical variates for $\mathbf{X}$ and $\mathbf{Y}$. A command `plot(object)` constructs a plot of the cumulative correlations against the number of canonical pairs. The `plot(object)` will provide a 90% reference line as default, and users can change the reference line with `plot(object,ref=percent)`.

```
## loading pulp data
> pulp <- read.table("http://home.ewha.ac.kr/~yjkstat/pulp.txt", header=TRUE)
> Y <- as.matrix(pulp[,1:4])
> X <- as.matrix(pulp[,5:8])

## standard CCA for X and Y
> fit.cca <- seedCCA(X, Y, type="cca")
NOTE:  The standard CCA is fitted for the two sets.

> names(fit.cca)
[1] "cor"     "xcoef"   "ycoef"   "Xscores" "Yscores"

## plotting cumulative canonical correlation
> par(mfrow=c(1, 2))
> plot(fit.cca, ref=80)
> plot(fit.cca)
```

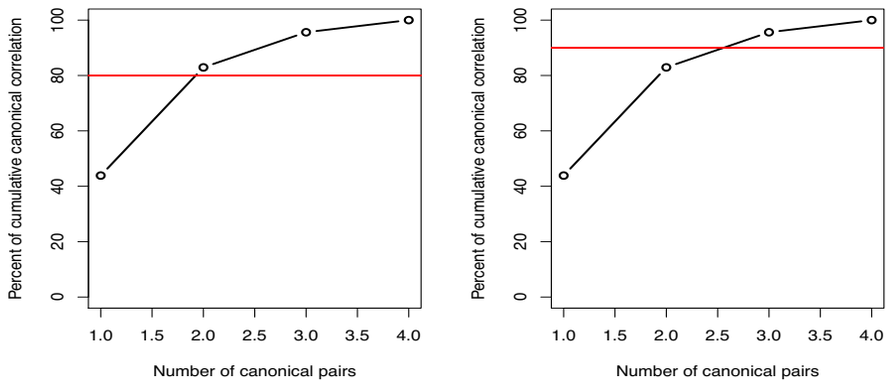

Figure 1: Cumulative canonical correlation plot in pulp data in Section 3.2

```
## first two canonical pairs
> X.cc <- fit.cca$Xscores[,1:2]
> Y.cc <- fit.cca$Yscores[,1:2]
```

According to Figure 1(a) and (b), with 80% cumulative canonical correlations, two canonical pairs are enough, while three canonical pairs should be good with the default 90%.

Ordinary least squares: pulp data

If the dimension of either **X** or **Y** is equal to one, the estimated canonical coefficient matrix from the standard CCA is equivalent to that from ordinary least squares. In such case, the command `seedCCA(X,Y[,1],type="cca")` results in the ordinary least squares estimate, which is "seedols" subclass. The output of "seedols" has three components, which are the estimated coefficients, and the two sets of variables. For example, assume that a regression study of arithmetic fiber length, which is the first column of **Y**, given **X** is of specific interest. It should be noted that the order of `seedCCA(X,Y[,1],type="cca")` and `seedCCA(Y[,1],X,type="cca")` does not matter, and any of them yields the same results. Also, the commands of `coef(object)` and `fitted(object)` return the estimated coefficients and fitted values from the ordinary least squares, respectively.

```
## extracting arithmatic fiber from Y
> fit.ols <- seedCCA(X, Y[, 1], type="cca")
NOTE:  One of the two sets are 1-dimensional, so a linear regression via ordinary least
square is fitted.

> names(fit.ols)
[1] "coef"  "X"  "Y"

> coef(fit.ols)
> fitted(fit.ols)
```

Seeded CCA (case 1): cookie data

The biscuit dough data set called `cookie` in **seedCCA** comes from the experiment of analyzing the composition of biscuit by NIR spectroscopy. Two sets of variables are obtained from 72 biscuit samples. The first set of variables is wavelengths measured by spectroscopy. In the original data set, wavelengths at 700 different points from 1100 to 2798 nanometers (NM) at the steps of 2nm were measured. However, since some of figures seemed to contain little information, wavelengths from 1380nm to 2400 at an interval of 4nm were analyzed. The second set of variables is the percentages of four ingredients of biscuits, fat, sucrose, dry flour and water. Since the 23th and 61st samples in the data set were believed to be outliers, they were deleted from the data set. The standard CCA is not applicable because of $p = 256 > n = 72$, and case 1 of the seeded CCA should be fitted, considering that $n = 72 >> r = 4$.

The basic command for this is `seedCCA(X,Y,type="seed1")`, which results in "finalCCA" subclass. Regardless of the order of `X` and `Y`, the lower dimensional set alone is reduced in the initial step.

Therefore, `seedCCA(X,Y,type="seed1")` and `seedCCA(Y,X,type="seed1")` basically produce the same seeded CCA results. For `type="seed1"`, the values of the options of `ux`, `uy`, `u`, `eps` and `AS` affect the implementation, whose defaults are `NULL`, `NULL`, 10, 0.01 and `TRUE`, respectively.

The option `u` controls the maximum number of projections unless both `ux` and `uy` are specified. The option `ux=k` works only when the dimension of the first set `X` is bigger than that of the second set `Y`. Then, the maximum number of projections becomes the value given in `ux=k`. The option `uy` works in the opposite way to `ux`. The options of `AS=TRUE` and `eps` control automatic termination of the projections before reaching the maximum given in `u`, `ux` or `uy`. The projection is terminated if the increment gets less than the value given in `eps`. Then, the first candidate value, which satisfies the stopping criteria, is suggested as a proper value of projections. If any of `ux`, `uy` and `u` is specified not enough to guarantee the automatic stopping, a notice is provided to increase it.

After running `seedCCA(X,Y,type="seed1")`, a plot for the proper selection of `u` is automatically constructed, and a blue vertical bar in the plot is the suggested value of `u`.

```
## loading cookie data
> data(cookie)
> myseq<-seq(141, 651, by=2)
> A <- as.matrix(cookie[-c(23, 61), myseq])
> B <- as.matrix(cookie[-c(23, 61), 701:704])

## seedec CCA with case 1
> fit.seed1.ab <- seedCCA(A, B, type="seed1") ## the first set A has been initial-CCAed.
NOTE:  Seeded CCA with case 1 is fitted. The set with larger dimension is initially reduced.
The first and second sets are denoted as X and Y, respectively.

> fit.seed1.ba <- seedCCA(B, A, type="seed1") ##  the second set A has been initial-CCAed.
NOTE:  Seeded CCA with case 1 is fitted. The set with larger dimension is initially reduced.
The first and second sets are denoted as X and Y, respectively.

> names(fit.seed1.ab)
[1] "cor"  "xcoef"  "ycoef"  "proper.u"  "initialMX0"  "newX" "Y"  "Xscores"  "Yscores"

> names(fit.seed1.ba)
[1] "cor"  "xcoef"  "ycoef"  "proper.u"  "X"  "initialMY0"  "newY"  "Xscores"  "Yscores"

> fit.seed1.ab$xcoef[, 3] <- -fit.seed1.ab$xcoef[, 3] ## changing the sign
> fit.seed1.ab$xcoef[, 4] <- -fit.seed1.ab$xcoef[, 4] ## changing the sign

> all(round(fit.seed1.ab$cor, 5)== round(fit.seed1.ba$cor, 5))
[1] TRUE

> fit.seed1.ab$proper.u
[1] 3

> fit.seed1.ba$proper.u
[1] 3

> all(round(fit.seed1.ab$xcoef, 5) == round(fit.seed1.ba$ycoef, 5))
[1] TRUE

> fit.seed1.ab.ux <- seedCCA(A, B, type="seed1", ux=2)
The maximum number of iterations is reached. So, users must choose u bigger than 2.

> fit.seed1.ab.ux$proper.u
[1] 2
```

For `fit.seed1.ab`, the first set `A` is reduced in the initial step. The output component `initialMX0` is the estimate of $\mathbf{M}^0_{x,u_1}$ and `newX` is $\hat{\mathbf{M}}^{0\,\mathrm{T}}_{x,u_1}\mathbf{X}$. On the contrary, in case of `fit.seed1.ba`, the second set `A` is initially reduced, so `initialMY0` and `newY` are produced. So, it is observed that the canonical correlations and suggested values of `u` from `fit.seed1.ab` and `fit.seed1.ba` are equal, not to mentioning that `fit.seed1.ab$xcoef` and `fit.seed1.ba$ycoef` are the same. The selection plot for `u` is reported in Figure 2, and three projections are suggested. Since `ux` is not given big enough in `seedCCA(A,B,type="seed1",ux=2)`, the following warning is given:

```
The maximum number of iterations is reached. So, users must choose u bigger than 2.
```

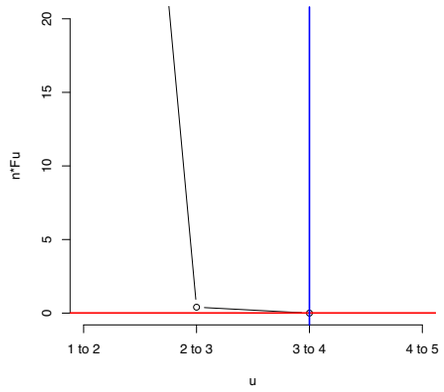

Figure 2: Selection plot of u generated from `seedCCA(A, B, type="seed1")` in Section 3.4

Next, we change the values of `ux`, `uy`, `AS` and `eps`. Since the usage of these options for `type="seed1"` are the same as that for `type="seed2"` and `type="pls"`. To measure the computing time, the **tictoc** package (Izrailev (2014)) is used with Intel(R) Core(TM)i7 2.9GHz and 12GB Ram computer.

```
> seedCCA(A, B, type="seed1", ux=5)$proper.u
[1] 3

> seedCCA(B, A, type="seed1", eps=0.000001)$proper.u
[1] 4

> library(tictoc)
> tic()
> seedCCA(B, A, type="seed1", u=30)$proper.u
> toc()
0.03 sec elapsed

> tic()
> seedCCA(B, A, type="seed1", u=30, AS=FALSE)$proper.u
> toc()
0.29 sec elapsed
```

Usage of `AS` should be noted. With bigger choices of u and `AS=FALSE`, the running time of the function will be longer.

Seeded CCA (case 2) versus Regularized CCA: nutrimouse data

The nutrimouse data was collected from a nutrition study in 40 mice ($n = 40$). One of two sets of variables was expressions of 120 genes measured in liver cells by microarray technology. The other set of variables was concentrations of 21 hepatic fatty acids(FA) measured through gas chromatography. In addition, the forty mice are cross-classified based on two factors, genotype and diet. There are two genotypes, wild-type (WT) and PPARα deficient (PPARα) mice and five diets: corn and colza oils (50/50 REF), hydrogenated coconut oil for a saturated FA diet (COC), sunflower oil for ω6 FA-rich diet (SUN), linseed oil for ω3-rich diet (LIN) and corn/colza/enriched fish oils (42.5/42.5/15, FISH). The nutrimouse data is contained in **seedCCA** package.

In this data, case 2 of the seeded CCA should be used because $\min(120, 21)$ is relatively big compared to $n = 40$. Then, case 2 of the seeded CCA requires to choose how many eigenvectors of $\hat{\mathbf{\Sigma}}_{xy}$ should be enough to replace it. This is another tuning parameter for case 2 of the seeded CCA along with the number of projections. The option `cut` in `seedCCA` controls automatic selection of the number of eigenvectors of $\hat{\mathbf{\Sigma}}_{xy}$. The option `cut=`$\alpha$ determines a set of the eigenvectors whose cumulative proportions of their corresponding eigenvalues is bigger than equal to α. For the set of eigenvectors to be chosen conservatively, we set the default of `cut` at 0.9. Also, users can directly give the number of eigenvectors using `d`. Unless `d` is `NULL`, the option `cut` is discarded. This means that `cut` works only when `d=NULL`. If users want to use `d`, then a function `covplot` should be run first. The function `covplot` has the option `mind`, which set the number of the eigenvalues to show their cumulative percentages. Its default is `NULL`, and then it becomes $\min(p, r)$. The function returns the

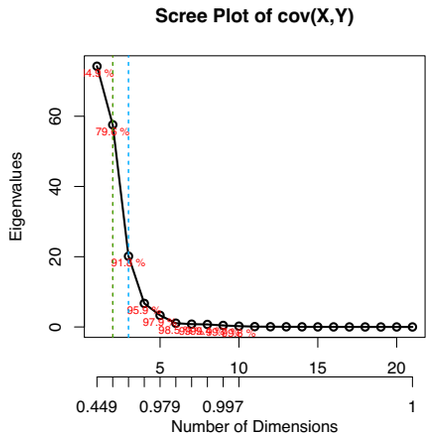

Figure 3: Scree plots for the selection of sets of eigenvectors to replace $\mathrm{cov}(\mathbf{X}, \mathbf{Y})$ generated from `covplot(X, Y, mind=10)` in Section 3.5

eigenvalues, the cumulative percentages and the number of the eigenvectors to account 60%, 70%, 80% and 90% of the total variation along with the scree plot of the eigenvalues.

The results by the seeded and regularized CCAs are compared. Since the regularized CCA is necessary to choose proper values of the two parameters, we compare running times for the automatic searches for the regularized and seeded CCAs via **tictoc** package. For **seedCCA**, we use the default value of `cut`.

```
> library(CCA)
> library(tictoc)

## loading nutrimouse data
> data(nutrimouse)
> X <- scale(as.matrix(nutrimouse$gene))
> Y <- scale(as.matrix(nutrimouse$lipid))

## determining the number of the eigenvectors of cov(X,Y) with cut=0.9
> tic("SdCCA")
> fit.seed2 <- seedCCA(X, Y)
> toc()
SdCCA: 0.13 sec elapsed

## finding the optimal values of lambda1 and lambda2 for RCCA
> tic("Regularized CCA")
> res.regul <- estim.regul(X, Y, plt=TRUE, grid1=seq(0.0001, 0.2, l=51), grid2=seq(0, 0.2, l=51))
> toc()
Regularized CCA 819.58 sec elapsed

## scree plot of cov(X, Y)
> names(covplot(X, Y, mind=10))
[1] "eigenvalue"  "cum.percent" "num.evecs"

> names(fit.seed2)
[1] "cor"  "xcoef"  "ycoef"  "proper.ux"  "proper.uy"  "d"  "initialMX0"  "initialMY0"
[9] "newX"  "newY"  "Xscores"    "Yscores"

> fit.seed2$d
[1] 3
```

Since `type="seed2"` reduces the dimensions of `X` and `Y` at the initialized CCA step, the output components of `initialMX0`, `initialMY0`, `newX` and `newY` and `d` are reported.

The plot generated from `covplot(X,Y,mind=10)` is given in Figure 3. According to Figure 3, the first two, three and four eigenvalues account for 79.6%, 91.8% and 95.9% of the total variation of $\hat{\mathbf{\Sigma}}_{xy}$, respectively. Using 90% conservative guideline, it is determined that the first three largest eigenvectors replace $\hat{\mathbf{\Sigma}}_{xy}$ well enough.

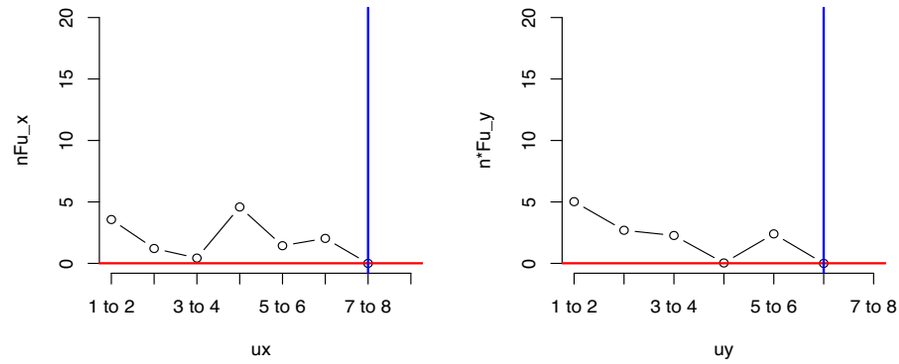

Figure 4: Selection plot of ux and uy generated from `seedCCA(X, Y)` in Section 3.5

The selection plot of ux and uy is given in Figure 4. The figure suggests that u_x and u_y are equal to 7 and 6, respectively.

Now we compare the parameter selection time. For the regularized CCA, it can be done with `estim.regul`, and users must provide small enough range for them to reduce the computing time. The resulted optimal λ_1 and λ_2 are 0.168016 and 0.004, respectively. With Intel(R) Core(TM)i7 2.9GHz and 12GB Ram, the seeded CCA took 0.32 second, while 819.58 seconds, around 13.5 minutes, lapsed for the regularized CCA. This difference is really huge, so time consumed in the selection of ux, uy and d is trivially small compared to the regularized CCA. This is a clear desirable aspect and advantage of the seeded CCA over the regularized one.

Next, we compare the first two pairs of estimated canonical variates. The results shown in Figures 5–6 are equivalent to the analysis discussed in González et al. (2008).

```
## Extracting the first two pairs of canonical variates
> sx1 <- fit.seed2$Xscores[, 1]
> sx2 <- fit.seed2$Xscores[, 2]
> sy1 <- fit.seed2$Yscores[, 1]
> sy2 <- fit.seed2$Yscores[, 2]

## fitting the regularized CCA
> res.rcc <- rcc(X, Y, 0.168016, 0.004)
> RCCA.X <- X%*%res.rcc$xcoef
> RCCA.Y <- Y%*%res.rcc$ycoef
> rx1 <- RCCA.X[,1]
> rx2 <- RCCA.X[,2]
> ry1 <- RCCA.Y[,1]
> ry2 <- RCCA.Y[,2]

par(mfrow=c(1,2))
> with(plot(rx1, ry1, col=c(2,4)[genotype], pch=c(1,2)[genotype],
+ main="1st pair from RCCA", xlab="rx1", ylab="ry1"), data=nutrimouse)
> with(legend(-1.4, 1.4, legend=levels(genotype), col=c(2,4), pch=c(1,2), cex=1.5),
+ data=nutrimouse)
> with(plot(-sx1, -sy1, col=c(2,4)[genotype], pch=c(1,2)[genotype],
+ main="1st pair from seedCCA", xlab="sx1", ylab="sy1"), data=nutrimouse)
> with(legend(-1.5, 1.6, legend=levels(genotype), col=c(2,4), pch=c(1,2), cex=1.5),
+ data=nutrimouse)

> par(mfrow=c(1,2))
> with(plot(rx2, ry2, col=c(1:4,6)[diet], pch=c(15,16,17,18,20)[diet], cex=1.5,
+ main="2nd pair from RCCA", xlab="rx2", ylab="ry2"), data=nutrimouse)
> with(legend(-2.3, 1.9, legend=levels(diet), col=c(1:4,6), pch=c(15:18,20)),
+ data=nutrimouse)
> with(plot(sx2, sy2, col=c(1:4,6)[diet], pch=c(15,16,17,18,20)[diet], cex=1.5,
+ main="2nd pair from seedCCA", xlab="sx2", ylab="sy2"), data=nutrimouse)
with(legend(-2.5, 1.9, legend=levels(diet), col=c(1:4,6), pch=c(15:18,20)),
```

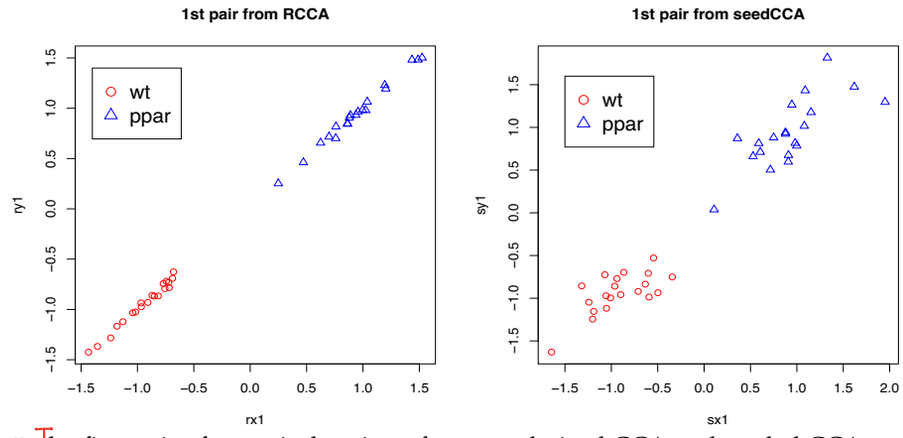

Figure 5: The first pair of canonical variates from regularized CCA and seeded CCA marked with genotype in Section 3.5

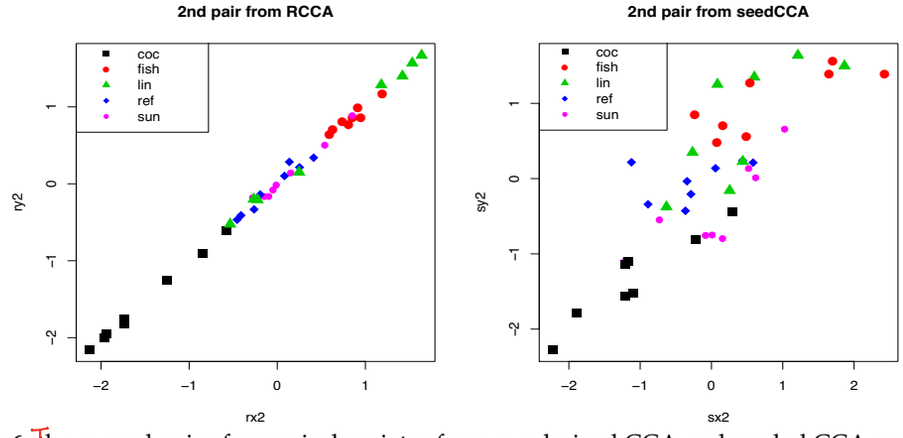

Figure 6: The second pair of canonical variates from regularized CCA and seeded CCA marked with diet in Section 3.5

```
+ data=nutrimouse)
```

According to Figures 5–6, the first pair of canonical variates from both CCAs distinguish genotype very well, while However, their second pairs marked with diet are quite complex. To have more insight into the results for the second pair on diet, multivariate analysis of variance is fitted. Further, pairwise comparison is done via **lsmeans** (Lenth (2016)) with level 5% and p-values adjusted by false discovery rate Benjamini and Hochberg (1995).

```
> library(lsmeans)
> fit2r <- manova(cbind(rx2, ry2)~diet, data=nutrimouse)
> fit3sd <- manova(cbind(sx2, sy2)~diet, data=nutrimouse)
> test( contrast( lsmeans(fit2r, "diet"), "pairwise"), side = "=",  adjust = "fdr")
 contrast     estimate        SE df t.ratio p.value
 coc - fish -2.3842686 0.2684019 35  -8.883  <.0001
 coc - lin  -2.1749708 0.2684019 35  -8.103  <.0001
 coc - ref  -1.4881111 0.2684019 35  -5.544  <.0001
 coc - sun  -1.6582635 0.2684019 35  -6.178  <.0001
 fish - lin  0.2092978 0.2684019 35   0.780  0.4897
 fish - ref  0.8961575 0.2684019 35   3.339  0.0040
 fish - sun  0.7260051 0.2684019 35   2.705  0.0175
 lin - ref   0.6868597 0.2684019 35   2.559  0.0214
 lin - sun   0.5167073 0.2684019 35   1.925  0.0780
 ref - sun  -0.1701524 0.2684019 35  -0.634  0.5302

Results are averaged over the levels of: rep.meas
P value adjustment: fdr method for 10 tests
```

```
> test( contrast (lsmeans(fit3sd, "diet"), "pairwise"), side = "=",  adjust = "fdr")
 contrast      estimate        SE df t.ratio p.value
 contrast      estimate        SE df t.ratio p.value
 coc - fish -2.14838660 0.3163067 35  -6.792  <.0001
 coc - lin  -1.79396325 0.3163067 35  -5.672  <.0001
 coc - ref  -1.07697196 0.3163067 35  -3.405  0.0035
 coc - sun  -1.03303440 0.3163067 35  -3.266  0.0041
 fish - lin  0.35442334 0.3163067 35   1.121  0.3001
 fish - ref  1.07141463 0.3163067 35   3.387  0.0035
 fish - sun  1.11535219 0.3163067 35   3.526  0.0035
 lin - ref   0.71699129 0.3163067 35   2.267  0.0371
 lin - sun   0.76092885 0.3163067 35   2.406  0.0308
 ref - sun   0.04393756 0.3163067 35   0.139  0.8903

Results are averaged over the levels of: rep.meas
P value adjustment: fdr method for 10 tests
```

For the regularized CCA, the "coc" diet is different from the others ~~and~~ Moreover "fish" differs from "sun". However, the other pairwise comparisons are quite mixed. It is determined that there are no significant differences between "fish-lin", "lin-sun" and "ref-sun". On the contrary, reasonable pairwise comparison results come from the seeded CCA. Like the others, the "coc" diet is different from the others. ~~And~~ Furthermore, "fish-lin" is not significantly different and "ref-sun" is concluded to be similar. Fish oil is known to contain $\omega 3$ and linseed oil is designed for it. Therefore, this conclusion would be reasonable. Also, The reference oil diet consists of corn and colza oil, which is known to contain $\omega 6$. Since sun flower oil is, indeed, for $\omega 6$-rich diet, this result is also reasonable. In this regard, the seeded CCA results would be ~~more~~ preferable ~~than~~ to the regularized CCA.

Partial least square application with nutrimouse data

With the nutrimouse data, consider a regression of the first one " C14.0" in concentrations of 21 hepatic fatty acids given expressions of 120 genes measured in liver cells. In ~~the~~ this case, partial least squares is a front-runner choice. Then, to obtain the partial least square estimator in **seedCCA**, one needs to implement ~~to~~ `seedCCA(X,Y,type="pls")`. This results in "seedpls" subclass. An important matter in partial least squares is that the first set of variable must be predictors. Response variable can be either univariate or multivariate. ~~The~~ the option `u` is recommended to set reasonably small, because the estimated coefficients are reported upto the value given in `u`. If `scale=TRUE`, the predictors are standardized to have zero sample means and sample correlation matrix.

The estimated coefficients and fitted values by partial least square can be obtained via `coef(object,u=NULL)` and `fitted(object,u=NULL)`. The default of `u` in both `coef` and `fitt` is `NULL`. In both functions, usage of `u` is equivalent. If `u=k` is specified, only the estimated coefficients and fitted values computed from `k` projections are reported. All of the coefficient estimates and fitted values are reported upto `u`, if `u=NULL`.

For `type="pls"`, the automatic procedure to suggest a proper value of projections is not conducted. For "seedpls" subclass, `plot(object)` suggests a proper value of projections along with other output components. If the terminating condition is not satisfied before reaching the value of `u`, then `plot(object)` provides a caution to increase the value of `u`.

```
> data(nutrimouse)
> Y <- as.matrix(nutrimouse$lipid)
> X <- as.matrix(nutrimouse$gene)
> Y1 <- as.matrix(Y[, 1])  ## univariate response
> Y12 <- as.matrix(Y[, 1:2])  ## multivariate response

## fitting partial least square and obtaining the estimated coefficient vector
> fit.pls1.10 <- seedCCA(X, Y1, u=10, type="pls")
> fit.pls1.3 <- seedCCA(X, Y1, u=3, type="pls", scale=TRUE)

> names(fit.pls1.10)
[1] "coef"  "u"     "X"     "Y"     "scale"

> names(fit.pls1.10$coef)
[1] u=1"  "u=2"  "u=3"  "u=4"  "u=5"  "u=6"  "u=7"  "u=8"  "u=9"  "u=10"
```

```
> names(fit.pls1.3$coef)
[1] "u=1" "u=2" "u=3"

> fit.pls1.3$scale
[1] TRUE

> par(mfrow=c(1,2))
> plot(fit.pls1.10)
$proper.u
[1] 6
$nFu
[1] 6.344725e+00 2.383108e+00 1.681329e+00 2.669394e+00 1.853061e+00 3.217472e-04 5.296046e-05
[8] 6.017641e-06 4.895905e-07 3.117371e-08
$u
[1] 10
$eps
[1] 0.01
> title("fit.pls1.10")

> plot(fit.pls1.3)
Caution: The terminating condition is NOT satisfied. The number of projections should be bigger than  3.
$proper.u
[1] 3
$nFu
[1] 6.344725 2.383108 1.681329
$u
[1] 3
$eps
[1] 0.01
> title("fit.pls1.3")

> names(fitted(fit.pls1.10))
[1] "u=1"  "u=2"  "u=3"  "u=4"  "u=5"  "u=6"  "u=7"  "u=8"  "u=9"  "u=10"

> fitted(fit.pls1.10, u=6)
     1      2     3     4     5     6     7     8     9    10    11    12    13    14
 0.137  0.368 0.317 0.346 0.492 1.620 0.722 0.003 0.065 1.212 0.458 0.640 0.272 0.397
    15     16    17    18    19    20    21    22    23    24    25    26    27    28
-0.103  0.426 1.448 0.287 1.264 0.517 2.803 0.914 0.043 0.028 0.234 0.598 0.875 0.434
    29     30    31    32    33    34    35    36    37    38    39    40
 0.694  0.666 2.958 2.350 0.620 0.958 0.495 2.790 0.701 0.168 0.767 0.535

> fit.pls.m <- seedCCA(X, Y12, u=5, type="pls")
> dim(fit.pls.m$coef$'u=1')
[1] 120   2
```

The selection of projections for two partial least squares by `seedCCA(X,Y1,u=10,type="pls")` and `seedCCA(X,Y1,u=3,type="pls",scale=TRUE)` is given in Figure 7. According to Figure 7, the proper value of projection is suggested at 6 for `fit.pls1.10` object, while the termination condition is not satisfied for `fit.pls1.3` object, so a caution statement is given.

Discussion

When a study between two sets of variables, saying ($\mathbf{X} \in \mathbb{R}^p, \mathbf{Y} \in \mathbb{R}^r$), ~~are~~ is of primary interest, canonical correlation analysis (CCA; Hotelling (1936)) is still popularly used in explanatory studies. The CCA has successful application in many science fields such as ~~chemomtrics~~ chemometrics, pattern recognition, genomic sequence analysis and so on.

The recently developed **seedCCA** package implements a collection of CCA methodologies including the standard CCA application, seeded CCA and partial least squares. The package enables us to fit CCA to large-p and small-n data. The paper provides a complete guide for the package to implement all the methods along with three real data examples. Also, the seeded CCA application results are compared with the regularized CCA in the existing **CCA** package.

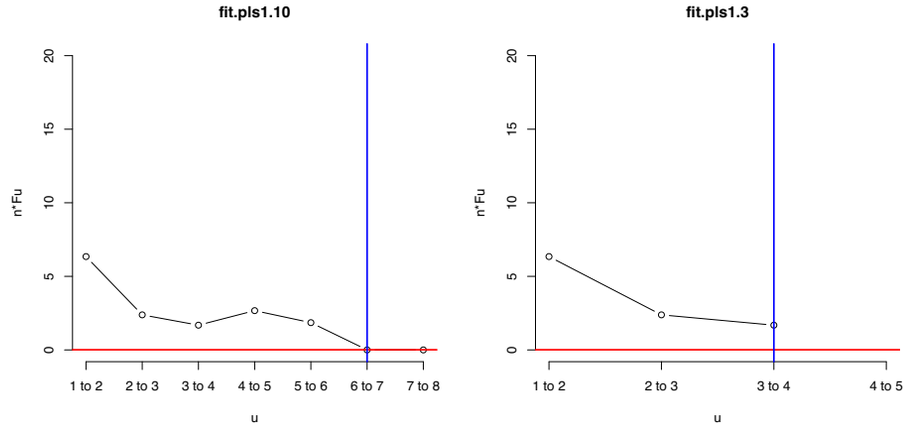

Figure 7: Selection plot of u generated from `seedCCA(X, Y1, u=10, type="pls")`(left) and `seedCCA(X, Y1, u=3, type="pls", scale=TRUE)`(right) in Section 3.6

It is believed that the package along with the paper will contribute to high-dimensional data analysis in various ~~science~~ scientific field practitioners and that the statistical methodologies in multivariate analysis become more fruitful.

Acknowledgments

For the corresponding author Jae Keun Yoo, this work was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Korean Ministry of Education (NRF-2019R1F1A1050715). For Bo-Young Kim, this work was supported by the BK21 Plus Project through the National Research Foundation of Korea (NRF) funded by the Korean Ministry of Education (22A20130011003).

Bibliography

A. Alfons, C. Croux, and P. Filzmoser. Robust maximum association between data sets: The R package ccaPP. *Austrian Journal of Statistics*, 45(1):71–79, 2016. [p]

Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society, Series B*, 57(1):289–300, 1995. URL https://doi.org/10.1111/j.2517-6161.1995.tb02031.x. [p]

R. D. Cook, B. Li, and F. Chiaromonte. Dimension reduction in regression without matrix inversion. *Biometrika*, 94(3):569–584, 2007. URL https://doi.org/10.1093/biomet/asm038. [p]

R. Cruz-Cano. *FRCC: Fast Regularized Canonical Correlation Analysis*, 2012. URL https://CRAN.R-project.org/package=FRCC. R package version 1.0. [p]

I. González, S. Déjean, P. G. P. Martin, and A. Baccini. Cca: an r package to extend canonical correlation analysis. *Journal of Statistical Software*, 23(12):1–13, 2008. URL https://doi.org/10.18637/jss.v023.i12. [p]

I. S. Helland. Partial least squares regression and statistical models. *Scandinavian Journal of Statistics*, 17(2):97–114, 1990. URL https://www.jstor.org/stable/4616159. [p]

H. Hotelling. Relations between two sets of variates. *Biometrika*, 28(3):321–377, 1936. URL https://www.jstor.org/stable/2333955. [p]

Y. Im, H. Gang, and J. Yoo. High-throughput data dimension reduction via seeded canonical correlation analysis. *Journal of Chemometrics*, 29(3):193–199, 2014. URL http://dx.doi.org/10.1002/cem.2691. [p]

S. Izrailev. *tictoc: Functions for timing R scripts, as well as implementations of Stack and List structures.*, 2014. URL https://CRAN.R-project.org/package=tictoc. R package version 1.0. [p]

R. Johnson and D. W. Wichern. *Applied Multivariate Statistical Analysis*. Pearson Prentice Hall, 2007. [p]

K. Lee and J. Yoo. Canonical correlation analysis through linear modeling. *Australian and New Zealand Journal of Statistics*, 56(1):59–72, 2014. URL http://dx.doi.org/10.1111/anzs.12057. [p]

R. V. Lenth. Least-squares means: The R package lsmeans. *Journal of Statistical Software*, 69(1):1–33, 2016. doi: 10.18637/jss.v069.i01. [p]

S. E. Leurgans, R. A. Moyeed, and B. W. Silverman. Canonical correlation analysis when the data are curves. *Journal of the Royal Statistical Society, Series B*, 55(3):725–740, 1993. URL https://www.jstor.org/stable/2345883. [p]

J. Schafer, R. Opgen-Rhein, M. A. V. Zuber, A. P. D. Silva, and K. Strimmer. *corpcor: Efficient Estimation of Covariance and (Partial) Correlation*, 2017. URL https://CRAN.R-project.org/package=corpcor. R package version 1.6.9. [p]

H. D. Vinod. Canonical ridge and econometrics of joint production. *Journal of Econometrics*, 4(2): 147–166, 1976. URL https://doi.org/10.1016/0304-4076(76)90010-5. [p]

Bo-Young Kim, Researcher
Celltrion
Incheon, 22014
Republic of Korea

Yunju Im, Postdoctoral Associate
Department of Biostatistics, Yale University
New Haven, CT 06520
United States of America

Jae Keun Yoo, Professor
Department of Statistics, Ewha Womans University
Seoul, 03760
Republic of Korea
peter.yoo@ewha.ac.kr