Supporting Information for `mvlearnR` for multiview learning

Elise Palzer and Sandra E Safo

June 13, 2023

The package `mvlearnR` and accompanying Shiny App is intended for integrating data from multiple sources (e.g. genomics, proteomics, metabolomics). Most existing software packages for multiview learning are decentralized, making it difficult for users to perform comprehensive integrative analysis. The new package wraps statistical and machine learning methods and graphical tools, providing a convenient and easy data integration workflow. For users with limited programming language, we provide a Shiny Application to facilitate data integration. The methods have potential to offer deeper insights into complex disease mechanisms. Currently, `mvlearnR` can be used for the following:

- Prefiltering of each omics data via differential analysis (DA). We provide both supervised and unsupervised options for DA or for filtering out noise variables prior to performing data integration.
- Integrating data from two sources using a variant of the popular unsupervised method for associating data from two views, i.e. canonical correlation analysis (CCA).
- Predicting a clinical outcome using results from CCA. We provide four outcome data distribution type (i.e. gaussian, binomial, Poisson, and time-to-event data.)
- Supervised integrative analysis (one-step) for jointly associating data from two or more sources and classifying an outcome. This method allows to include covariates.
- Supervised integrative analysis (one-step) for jointly associating structured data from two or more sources and classifying an outcome. This method allows to include covariates.
- Visualizatiing results from the DA or our integrative analysis methods. These plots include: volcano plots, UMAP plots, variable importance plots, discriminant plots, correlation plots, relevance network plots, loadings plots, and within- and between-view biplots. These visualization tools will help unravel complex relationships in the data.
- Demonstrating our integration workflow via already uploaded synthetic and real molecular and clinical data pertaining to COVID-19.

Currently, linear multivariate methods for integrative analysis and biomarker identification are provided in `mvlearnR`. However, we have developed integrative analysis methods for disease subtyping (1) and nonlinear integrative analysis methods for biomarker identification

(2; 3; 4) that will eventually be added to `mvlearnR` and the accompanying web application. Other methods we develop in the future will be added. Thus, we envision `mvlearnR` and our web application to be a one-stop place for comprehensive data integration, for both users of R (or Python) and non-users of these software.

1 The Methods

The methods SELPCCA and SIDA in `mvlearnR` have been described in (5) and (6), respectively. For completeness sake, we describe these methods below. Since these methods are extensions of canonical correlation analysis (CCA)(7) and linear discriminant analysis (LDA) (7), we briefly describe these two methods.

1.1 Linear Discriminant Analysis

Suppose we have one set of data $\mathbf{X}$ with n samples on the rows and p variables on the columns. Assume the n samples are partitioned into K classes, with n_k being the number of samples in class $k, k = 1, \ldots, K$. Assume each sample belongs to one of the K classes and let y_i be the class membership for the ith subject. Let $\mathbf{X}_k = (\mathbf{x}_{1k}, \ldots, \mathbf{x}_{n_k,k})^{\mathrm{T}} \in \Re^{n_k \times p}$, $\mathbf{x}_k \in \Re^p$ be the data matrix for class k. Given these available data, we wish to predict the class membership y_j of a new subject j using their high-dimensional information $\mathbf{z}_j \in \Re^p$. Fishers linear discriminant analysis (LDA) may be used to predict the class membership. For a K class prediction problem, LDA finds $K-1$ low-dimensional vectors (also called discriminant vectors), which are linear combinations of all available variables, such that projected data onto these discriminant vectors have maximal separation between the classes and minimal separation within the classes. The within-class and between-class separation are defined respectively as: $\mathbf{S}_w = \sum_{k=1}^{K} \sum_{i=1}^{n} (\mathbf{x}_{ik} - \hat{\boldsymbol{\mu}}_k)(\mathbf{x}_{ik} - \hat{\boldsymbol{\mu}}_k)^{\mathrm{T}}$; $\quad \mathbf{S}_b = \sum_{k=1}^{K} n_k (\hat{\boldsymbol{\mu}}_k - \hat{\boldsymbol{\mu}})(\hat{\boldsymbol{\mu}}_k - \hat{\boldsymbol{\mu}})^{\mathrm{T}}$. Here, $\hat{\boldsymbol{\mu}}_k = (1/n_k) \sum_{i=1}^{n_k} \mathbf{x}_{ik}$ is the mean vector for class k, $\hat{\boldsymbol{\mu}}$ is the combined class mean vector and is defined as $\hat{\boldsymbol{\mu}} = (1/n) \sum_{k=1}^{K} n_k \hat{\boldsymbol{\mu}}_k$. The solution to the LDA problem is given are the eigenvalue-eigenvector pairs $(\widehat{\lambda}_k, \widehat{\boldsymbol{\beta}}_k)$, $\widehat{\lambda}_1 > \cdots > \widehat{\lambda}_k$ of $\mathbf{S}_w^{-1}\mathbf{S}_b$ for $\mathbf{S}_w \succ 0$. Note that LDA is only applicable to one set of data.

1.2 Canonical Correlation Analysis

Canonical correlation analysis on the other hand is applicable when there are two views. Now, suppose we have two sets of data. $\mathbf{X}^1 = (\mathbf{x}_1^1, \cdots, \mathbf{x}_n^1)^{\mathrm{T}} \in \Re^{n \times p}$ and $\mathbf{X}^2 = (\mathbf{x}_1^2, \cdots, \mathbf{x}_n^2)^{\mathrm{T}} \in \Re^{n \times q}$, $p, q > n$, all measured on the same set of n subjects. The goal of CCA (7) is to find linear combinations of the variables in $\mathbf{X}^1$, say $\mathbf{u} = \mathbf{X}^1\boldsymbol{\alpha}$ and in $\mathbf{X}^2$, say $\mathbf{v} = \mathbf{X}^2\boldsymbol{\beta}$, such that the correlation between these linear combinations is maximized, i.e $\rho = \max cor(\mathbf{u}, \mathbf{v}) = \max cor(\mathbf{X}^1\boldsymbol{\alpha}, \mathbf{X}^2\boldsymbol{\beta})$. But

$$\max_{\boldsymbol{\alpha}.\boldsymbol{\beta}} cor(\mathbf{X}^1\boldsymbol{\alpha}, \mathbf{X}^2\boldsymbol{\beta}) = \frac{\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{\Sigma}_{12}\boldsymbol{\beta}}{\sqrt{\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{\Sigma}_{11}\boldsymbol{\alpha}}\sqrt{\boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{\Sigma}_{22}\boldsymbol{\beta}}},$$

where $\boldsymbol{\Sigma}_{12}$ is the $p \times q$ population covariance between $\mathbf{X}^1$ and $\mathbf{X}^2$, and $\boldsymbol{\Sigma}_{11}, and \boldsymbol{\Sigma}_{22}$ are the population covariances of $\mathbf{X}^1$ and $\mathbf{X}^2$, respectively. The population cross-covariances $\boldsymbol{\Sigma}_{12}$, $\boldsymbol{\Sigma}_{11}$, and $\boldsymbol{\Sigma}_{22}$ are estimated by their sample versions $\mathbf{S}_{12}$, $\mathbf{S}_{11}$, and $\mathbf{S}_{22}$, respectively. For

low-dimensional settings, these sample versions are consistent estimators of the population covariances, for fixed p and q, and large n. However, for high-dimensional settings, these sample versions are regularized. The vectors $\mathbf{u}$ and $\mathbf{v}$ are called the first canonical variates for $\mathbf{X}^1$ and $\mathbf{X}^2$, respectively. We note that the correlation coefficient is invariant to scaling, thus one can choose the denominator be one and then consider the equivalent problem:

$$\max_{\boldsymbol{\alpha},\boldsymbol{\beta}} cor(\mathbf{X}^1\boldsymbol{\alpha}, \mathbf{X}^2\boldsymbol{\beta}) = \boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{\Sigma}_{12}\boldsymbol{\beta} \;\; \text{s.t} \;\; \boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{\Sigma}_{11}\boldsymbol{\alpha} = 1, \quad \boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{\Sigma}_{22}\boldsymbol{\beta} = 1.$$

Using Langragian multipliers, it can be shown that the CCA solution to this optimizaiton problem is given by $\widehat{\boldsymbol{\alpha}} = \mathbf{S}_1^{-1/2}\mathbf{e}_1, \widehat{\boldsymbol{\beta}} = \mathbf{S}_2^{-1/2}\mathbf{f}_1$, where $\mathbf{e}_1$ and $\mathbf{f}_1$ are the first left and right singular vectors of $\mathbf{S}_1^{-1/2}\mathbf{S}_{12}\mathbf{S}_2^{-1/2}$ respectively. We note that maximizing the correlation is equivalent to maximizing the square of the correlation. To obtain subsequent canonical variates, one can impose the following orthogonality constraints in the optimization problem: $\boldsymbol{\alpha}_i^{\mathrm{T}}\boldsymbol{\Sigma}_{11}\boldsymbol{\alpha}_j = \boldsymbol{\beta}_i^{\mathrm{T}}\boldsymbol{\Sigma}_{22}\boldsymbol{\beta}_j = \boldsymbol{\alpha}_i^{\mathrm{T}}\boldsymbol{\Sigma}_{12}\boldsymbol{\beta}_j = 0$, for $i \neq j$, $i, j = \min(p, q, n)$.

1.3 Sparse CCA via SELP (SELPCCA)

The classical CCA method finds linear combinations of all available variables, and since these weights are typically nonzero, it is difficult to interpret the findings. SELPCCA (5) is a variant of CCA that shrinks some of the weights of the low-dimensional representations $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ to zero, thus allowing to identify relevant variables contributing to the overall dependency structure in the data. In particular, the authors in (5) proposed to solve the following optimization problem:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \|\boldsymbol{\alpha}\|_1 \quad s.t \quad \|\mathbf{S}_{12}\tilde{\boldsymbol{\beta}}_1 - \tilde{\rho}_1\tilde{\mathbf{S}}_{11}\boldsymbol{\alpha}\|_\infty \leq \tau_1 \\ \min_{\boldsymbol{\beta}} \|\boldsymbol{\beta}\|_1 \quad s.t \quad \|\mathbf{S}_{21}\tilde{\boldsymbol{\alpha}}_1 - \tilde{\rho}_1\tilde{\mathbf{S}}_{22}\boldsymbol{\beta}\|_\infty \leq \tau_2. \end{aligned} \tag{1}$$

Here, $(\tilde{\boldsymbol{\alpha}}_1, \tilde{\boldsymbol{\beta}}_1)$ are the first nonsparse solution to the CCA optimization problem, and $\tilde{(\rho)}_1$ is the corresponding eigenvalue. Of note, $\tilde{\mathbf{S}}_{11}$ and $\tilde{\mathbf{S}}_{22}$ are regularized versions of $\mathbf{S}_{11}$ and $\tilde{\mathbf{S}}_{22}$, respectively. The authors considered two forms of regularizations: $\tilde{\mathbf{S}}_{11} = \mathbf{I}_p$ and $\tilde{\mathbf{S}}_{22} = \mathbf{I}_q$, for standardized data, and the ridge regularization: $\tilde{\mathbf{S}}_{11} = \mathbf{S}_{11} + \sqrt{\log(p)/n}\mathbf{I}_p$ and $\tilde{\mathbf{S}}_{22} = \mathbf{S}_{22} + \sqrt{\log(q)/n}\mathbf{I}_q$. Also, τ_1 and τ_2 are tuning parameters which are chosen by cross-validation. See (5) for more details. For subsequent canonical directions $\boldsymbol{\alpha}_i$, and $\boldsymbol{\beta}_i$, $i > 1$, the authors solved the above sparse optimization problem on deflated data.

1.4 Prediction with SELPCCA

CCA and SELPCCA are unsupervised multivariate methods, with their main goal of finding canonical vectors that maximize the overall dependency structure between two views. In some biomedical applications, it is usually the case that an outcome variable, say $\mathbf{y}$, is available and a specific interest might be to investigate how these canonical variates are related to the outcome of interest. For this purpose, one can build regression models to associate the outcome with these canonical variates. In our package, we provide `selpPredict()` for this purpose. We provide options gaussian, binomial, poisson, and survival to model continuous, categorical, count, or time-to-event data, respectively. We also provide the function `predict()` to predict out of sample data using the learned low-dimensional representations or canonical variates.

1.5 Sparse Integrative Discriminant Analysis (SIDA) for two or more views

Instead of considering the two-step CCA approach proposed above when there is a clinical outcome in addition to the views, the authors in (6) developed the method, SIDA, for jointly maximizing association between two or more views while also maximizing separation between classes in each view. Although CCA is applicable to only two views, SIDA is applicable to two or more views. SIDA combines the advantages of LDA, a supervised learning method for maximizing separation between classes in a view, and CCA, an unsupervised learning method for maximizing correlation between two data types. Suppose there are $d = 1, \ldots, D$ views. Let $\mathbf{X}^d = [\mathbf{X}_1^d, \mathbf{X}_2^d, \ldots, \mathbf{X}_K^d]$, $\mathbf{X}^d \in \Re^{n \times p_d}, \mathbf{X}_k^d \in \Re^{n_k \times p_d}, k = 1, \ldots, K,\ d = 1, 2, \ldots, D$ be a concatenation of the K classes in the d-th view. Let $\mathbf{S}_b^d$ and $\mathbf{S}_w^d$ be the between-class and within-class covariances for the d-th view. Let $\mathbf{S}_{dj}, j < d$ be the cross-covariance between the d-th and j-th views. Define $\mathcal{M}^d = \mathbf{S}_w^{d^{-1/2}} \mathbf{S}_b^d \mathbf{S}_w^{d^{-1/2}}$ and $\mathcal{N}_{dj} = \mathbf{S}_w^{d^{-1/2}} \mathbf{S}_{dj} \mathbf{S}_w^{j^{-1/2}}$. The authors solved the following optimization problem for the basis discriminant vectors $\mathbf{\Gamma}^d$:

$$\max_{\mathbf{\Gamma}^1, \cdots, \mathbf{\Gamma}^D} \rho \sum_{d=1}^{D} \mathrm{tr}(\mathbf{\Gamma}^{d\mathrm{T}} \mathcal{M}^d \mathbf{\Gamma}^d) + \frac{2(1-\rho)}{D(D-1)} \sum_{d=1, d \neq j}^{D} \mathrm{tr}(\mathbf{\Gamma}^{d\mathrm{T}} \mathcal{N}_{dj} \mathbf{\Gamma}^j \mathbf{\Gamma}^{j\mathrm{T}} \mathcal{N}_{jd} \mathbf{\Gamma}^d)$$
$$\text{s.t } \mathrm{tr}(\mathbf{\Gamma}^{d\mathrm{T}} \mathbf{\Gamma}^d) = K - 1.$$

Of note, ρ controls the influence of separation or association in the optimization problem. The second term sums the unique pairwise squared correlations and weights them by $\frac{D(D-1)}{2}$ so that the sum of the squared correlations is one. The authors showed that the nonsparse basis discriminant directions for the d-th view, $\widetilde{\mathbf{\Gamma}}^d$, that maximize both associations and separations are given by the eigenvectors corresponding to the eigenvalues ($\mathbf{\Lambda}^d$) that iteratively solve the following eigensystems:

$$\begin{aligned}
\left(c_1 \mathcal{M}^1 + c_1 \mathcal{M}^{1^{\mathrm{T}}} + c_2 \overline{\mathcal{N}}_{1j} + c_2 \overline{\mathcal{N}}_{1j}^{\mathrm{T}}\right) \mathbf{\Gamma}^1 &= \mathbf{\Lambda}_1 \mathbf{\Gamma}^1, \\
&\vdots \\
\left(c_1 \mathcal{M}^D + c_1 \mathcal{M}^{1^{\mathrm{T}}} + c_2 \overline{\mathcal{N}}_{Dj} + c_2 \overline{\mathcal{N}}_{Dj}^{\mathrm{T}}\right) \mathbf{\Gamma}^D &= \mathbf{\Lambda}_D \mathbf{\Gamma}^D, \qquad (2)
\end{aligned}$$

where $c_1 = \rho$ and $c_2 = \frac{2(1-\rho)}{D(D-1)}$, and $\overline{\mathcal{N}}_{dj} = \sum_{d,j}^{D} \mathcal{N}_{dj} \mathbf{\Gamma}^j \mathbf{\Gamma}^{j^{\mathrm{T}}} \mathcal{N}_{jd},\ d, j = 1, \ldots D, j \neq d$ sums all unique pairwise correlations of the d-th and the j-th views.

For sparsity or smoothness, they considered the following optimization problems:

$$\begin{aligned}
&\min_{\mathbf{\Gamma}^1} \mathcal{P}(\mathbf{\Gamma}^1) \qquad \text{s.t} \qquad \|(c_1 \mathcal{M}^1 + c_1 \mathcal{M}^{1^{\mathrm{T}}} + c_2 \overline{\mathcal{N}}_{1j} + c_2 \overline{\mathcal{N}}_{1j}^{\mathrm{T}}) \widetilde{\mathbf{\Gamma}}^1 - \widetilde{\mathbf{\Lambda}}_1 \mathbf{\Gamma}^1\|_\infty \leq \tau_1 \\
&\qquad\qquad \vdots \\
&\min_{\mathbf{\Gamma}^D} \mathcal{P}(\mathbf{\Gamma}^D) \qquad \text{s.t} \qquad \|(c_1 \mathcal{M}^D + c_1 \mathcal{M}^{D^{\mathrm{T}}} + c_2 \overline{\mathcal{N}}_{Dj} + c_2 \overline{\mathcal{N}}_{Dj}^{\mathrm{T}}) \widetilde{\mathbf{\Gamma}}^D - \widetilde{\mathbf{\Lambda}}_D \mathbf{\Gamma}^D\|_\infty \leq \tau_D. \qquad (3)
\end{aligned}$$

The authors considered two forms of penalty term $\mathcal{P}(\mathbf{\Gamma}^d)$, depending on whether sparsity only (data-driven) or sparsity and smoothness (knowledge-driven) is desired: $\mathcal{P}(\mathbf{\Gamma}^d) = \sum_{i=1}^{p_d} \|\boldsymbol{\gamma}_i^d\|_2$, or $\mathcal{P}(\mathbf{\Gamma}^d) = \eta \sum_{i=1}^{p_d} \|\boldsymbol{\gamma}_i^{\mathcal{L}_n}\|_2 + (1-\eta) \sum_{i=1}^{p_d} \|\boldsymbol{\gamma}_i\|_2$. In the latter, $\boldsymbol{\gamma}_i^{\mathcal{L}_n}$ is the i-th row of the matrix product $\mathcal{L}_n \mathbf{\Gamma}^d$, and $\mathcal{L}_n$, is the normalized Laplacian of a graph, which is

view-dependent. Essentially, the first term in this penalty acts as a smoothing operator for the weight matrices $\mathbf{\Gamma}^d$ so that variables that are connected within the d-th view are encouraged to be selected or neglected together. This was termed SIDANet (SIDA for structured or network data). SIDANet is applicable when there exists prior biological information in the form of variable-variable connections, which guides the detection of connected variables that maximize both separation and association.

2 Visualizations of discriminant scores and canonical correlation variates

Once the discriminant vectors or canonical correlation vectors have been estimated using SIDA/SIDANet or SELPCCA respectively, the scores for each view, representing projections of each view onto these vectors can be estimated. We provide `DiscriminantPlots()` and `CorrelationPlots()` to visualize these scores. The `DiscriminantPlots()` function can be used to visualize the first (assuming only two classes) or the first and second discriminant scores (assuming $>$ two classes). Each point on this plot represent a score for an individual. Discriminant plots can showcase how well the discriminant vectors separate the classes. Correlation plots on the other hand can be used to visualize the strength of association between pairs of views as well as the separation of the classes. Figure 1 demonstrates sample figures that can be generated by our `DiscriminantPlots()` and `CorrelationPlots()` in `mvlearnR`. We provide different color options for the graphs.

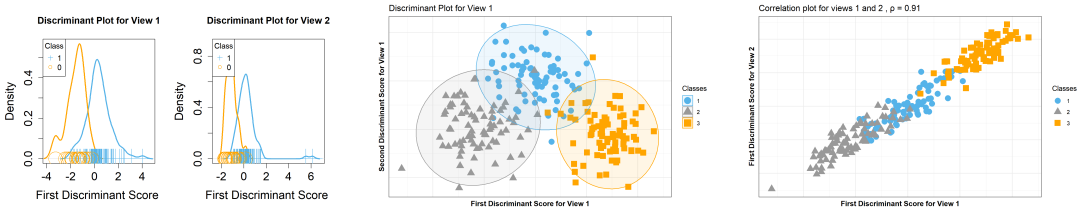


Figure 1: Sample discriminant plots for two classes (left panel), for three classes (middle panel) and correlation plot (right panel). Discriminant plots can demonstrate whether the discriminant vectors separate the classes. Correlation plots demonstrate the strength of association between pairs of views. It also shows how well the classes are separated.

3 Visualizations of variables selected

3.1 Relevance Network

Relevance Network (RN) (8; 9) have been proposed to visualize pairwise relationships of variables in integrative analysis. Here, the nodes represent variables, and edges represent variable associations. To construct RN, the correlation matrix of pairwise associations is obtained from the data. Then, for a specific cutoff (say 0.7), if the estimated correlation coefficients is greater (in absolute value) than this cutoff, an edge is drawn between these two variables; otherwise, no edge is drawn and these two variables are deemed not associated for this threshold. Further, variables/nodes with no link are not represented in the RN. The

cutoff is used to make the graph less dense, especially when the number of variables is high. RN have potential to shed lignt into the complex associations between different types of data. We provide the function `networkPlot()` to visualize the pairwise relationships of variables selected by SELPCCA, SIDA, and SIDANet. We follow ideas (9) to construct the RN for SELPCCA, SIDA, and SIDANet. In particular, instead of computing the Pearson correlation coefficients between each pair of variables in each view separately, we construct bipartite graph (or bigraph) where variables/nodes from view $\mathbf{X}^d$ are connected to variables/nodes from view $\mathbf{X}^j$, $d \neq j, d, j = 1, \ldots, D$. Similar to (9), we construct the bipartite graphs using a pairwise similarity matrix directly obtained from the outputs of our integrative analysis methods.

Suppose we have applied SELPCCA to two views and we have obtained the canonical loadings $\hat{\boldsymbol{\alpha}}_l$ and $\hat{\boldsymbol{\beta}}_l$, $l = 1, \ldots, L$, for views 1 and 2, respectively. For $l > 2$, it is likely that within a view, different variables will have nonzero coefficients for each canonical loading. Therefore, we deem a variable to be selected if it has a nonzero coefficient in at least one canonical loading. Let $\widehat{\mathbf{A}} = (\hat{\boldsymbol{\alpha}}_1^{\mathrm{T}}, \ldots, \hat{\boldsymbol{\alpha}}_L^{\mathrm{T}})^{\mathrm{T}} \in \Re^{p \times L}$ be a concatenation of all L canonical loadings for view 1. Let $\widehat{\mathbf{B}} = (\hat{\boldsymbol{\beta}}_1^{\mathrm{T}}, \ldots, \hat{\boldsymbol{\beta}}_L^{\mathrm{T}})^{\mathrm{T}} \in \Re^{q \times L}$ be defined similarly for view 2. Given data with these selected variables, we construct the canonical variates: $\mathbf{U} = \mathbf{X}^1_{selected}\widehat{\mathbf{A}}$ and $\mathbf{V} = \mathbf{X}^2_{selected}\widehat{\mathbf{B}}$. Let p' and q' be the number of selected variables in views 1 and 2, respectively. To construct the $p' \times q'$ similarity matrix for views 1 and 2, we first project data for the selected variables to $\mathbf{U}$, for view 1, $\mathbf{V}$ for view 2, or to a combination of $\mathbf{U}$ and $\mathbf{V}$. Since $\mathbf{X}^1$ and $\mathbf{X}^2$ are analyzed simultaneously in CCA, the projection of the data onto a combination of $\mathbf{U}$ and $\mathbf{V}$ seems natural. Thus, we define $\mathbf{Z} = \mathbf{U} + \mathbf{V}$. As noted in (9), the $\mathbf{Z}$ variables are closest to $\mathbf{X}^d$ and $\mathbf{X}^j$. Then, for the ith variable $X_i^1 \in \mathbf{X}^1_{selected}$ (similarly $X_i^2 \in \mathbf{X}^2_{selected}$) and the lth component $\mathbf{z}_l \in \mathbf{Z}$, we compute the scalar inner product $x_{il}^d = \langle X_i^d, \mathbf{z}_l \rangle, d = 1, 2; j = 1 \ldots p' (\text{or } q'),\ l = 1, \ldots, L$. Assuming that the variables X_i^d, and $\mathbf{z}_l$ are standardized to have variance 1, $x_{il}^d = \langle X_i^d, \mathbf{z}_l \rangle = \mathrm{cor}(X_i^d, \mathbf{z}_l)$. We compute the similarity matrix $\mathbf{M}$ between views 1 and 2 as $\mathbf{M} = \mathbf{x}^1 \mathbf{x}^{2'_{\mathrm{T}}} \in \Re^{p' \times q'}$, where $\mathbf{x}^1$ is a $p' \times L$ matrix with the ilth entry x_{il}^1 and $\mathbf{x}^2$ is a $q' \times L$ matrix with the ilth entry x_{il}^2. Each entry in the similarity matrix is between -1 and 1.

We construct the similarity matrix for SIDA and SIDANet in a similar fashion. Of note, in SIDA and SIDANet, $l = K - 1$. Although, the $l_{2,1}$ norm encourages the same variables to be selected across all $K - 1$ components, empirically, this is not usually the case. Therefore, we obtain overall variable selection as described above. We obtain the discriminant vectors for the dth and jth views as $\mathbf{U} = \mathbf{X}^1_{selected}\widehat{\boldsymbol{\Gamma}}^d$ and $\mathbf{V} = \mathbf{X}^2_{selected}\widehat{\boldsymbol{\Gamma}}^j$, respectively. Given $\mathbf{U}$ and $\mathbf{V}$, we compute the similarity matrix for SIDA and SIDANet in a similar fashion.

We generate the relevance network from the similarity matrix. The nodes of the graph represent variables for the pairs of views, and edges represent the correlations between pairs of variables. The function `networkPlot()` in `mvlearn` can be used to generate relevance networks. Please refer to the bottom left and right panels of Figure 2 for sample relevance network plots. Dashed and solid lines indicate negative and positive correlations respectively. Circle nodes are View 1 variables, and rectangular nodes are View 2 variables. We provide an option to tune the network through a correlation cutoff. For instance in the middle right panel of Figure 2, we only show graph for variables with correlations greater than 0.9. The plot suggest that variable V1031 from View 2 is highly correlated with variable V31 from View 1. Since there is no edge between variable V38 and variables V1037, it indicates that the correlation between these pairs of variables is smaller than 0.9.

3.2 Variable importance plots

We provide the function `VarImportancePlot()` [Top left panel of Figure 2] to visualize the weights (in absolute value) of the loadings. Since the loadings are standardized to have unit norm, a variable with larger weight contributes more to the association between the views (for SELPCCA) or to the association between the views and discrimination of classes within each view (SIDA and SIDANet). We only show the top 20 variables and their weights but one can view data matrix for all variables. In this plot, dashed and solid lines represent negative and positive correlations.

3.3 Loadings plots

We provide the function `LoadingsPlot()` [Top right panel of Figure 2] to plot discriminant and canonical correlation vectors. These graphs are useful for visualizing how selected variables from SIDA/SIDANet and SELPCCA contribute to the first and second discriminant (for SIDA and SIDANet) or canonical correlation (for SELPCCA) vectors. Variables farther from the origin and close to the first or second axis have higher impact on the first or second discriminant/canonical vectors, respectively. Variables farther from the origin and between both first and second axes have similar higher contributions to the first and second discriminant/canonical correlation vectors. In both situations, for SIDA and SIDANet, this suggests that these variables contribute more to the separation of classes and association of views. For SELPCCA, this suggests that these variables contribute more to the association between the two views. This plot can only be generated for classification and association problems with 3 or more classes (SIDA and SIDANet), or for CCA problems with two or more canonical correlation vectors requested (i.e. ncancorr > 1 for SELPCCA). The angle between two vectors also give an indication of how the two variables are correlated. In particular, vectors that are close to each other suggests that the variables have high positive correlation. Vectors that are about 90 degrees indicate that the two variables are uncorrelated. Vectors that have an angle close to 180 degrees indicate that the two variables have negative correlation.

4 Visualization of loadings and scores simultaneously

Biplots are useful for representing both loadings and discriminant scores/canonical correlation variates. We present biplots for each view and between views. In particular, we provide the function `WithinViewBiplot()` to visualize the scores and loadings for each view separately [Figure 1, Bottom left panel]. We also provide the function `BetweenViewBiplot()` to graph scores and loadings for pairs of views. The scores are the sum of scores for the two views (Refer to Section on relevance network for more explanation). Please refer to Figure 1, Bottom right panel for a sample biplot between views. In this graph, dashed red vectors represent loadings plot for the second view. And solid black vectors represent loadings plot for the first view. Please refer to section 3.3 for a brief discussion on interpreting loadings plots.

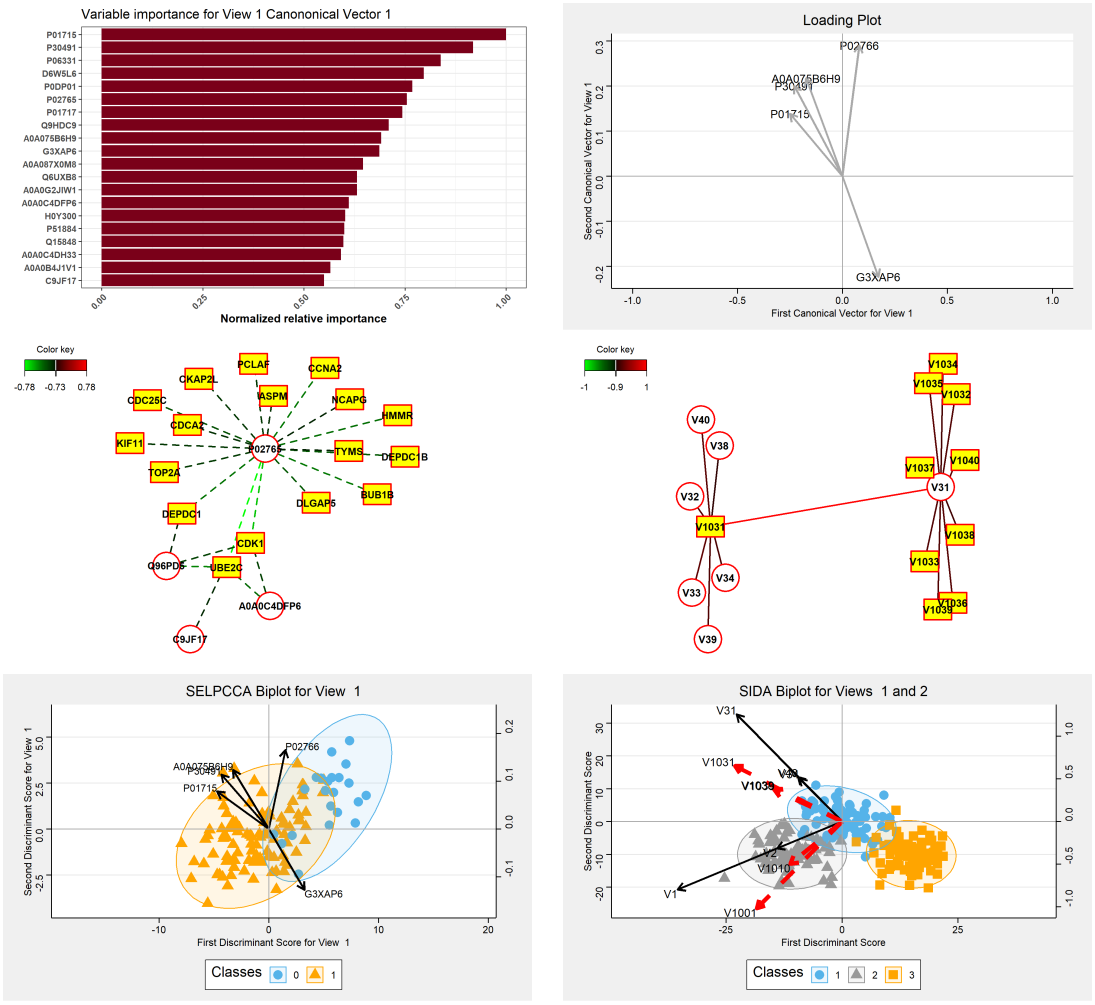


Figure 2: Sample plots for visualizing variables within and between views. *Top left panel*: variable importance plot showing the relative importance of each variable. Variables with largest absolute loadings rank high. The weights for each variable is normalized relative to the largest weight. *Top right panel*: Loading plots. We provide loading plots for each view to demonstrate the relationships between pairs of variables within each view. Variables that are farther from the origin have higher impact on the axis the vector is close to. Vectors that are close to each other suggest that the pairs of variables are correlated positively. Vectors with an angle of about 180 degrees suggest that the variables have negative correlation. Vectors with a 90 degree angle between them suggest that the two variables are not correlated. *Middle left and right panels*: Relevance network showing variable-variable connections between pairs of views with correlations > 0.73 [left panel] and > 0.9 [right panel]. Dashed and solid lines indicate negative and positive correlations, respectively. Color key gives the direction of the associations where green indicates negative correlations and red shows positive correlations. *Bottom left panel*: Within-view biplot. This plot shows the scores and loadings together for a specific view. Bottom right panel: Between-view biplot. This plot shows the scores and loadings from pairs of views together. The scores are the sum of scores for each view. Solid and dashed lines represent vectors for Views 1 and 2, respectively. In this example, variable V31 (from View 1) has a higher correlation with variables V1031 and V1038 from View 2 compared to Variable V1001 from View 2. This finding is supported by the edge between V31 and V1038 and V1031 in the network plot but no edge between V31 and V1001 given a correlation cutoff of 0.9.

References

[1] W. Zhang, C. Wendt, R. Bowler, C. P. Hersh, S. E. Safo, Robust integrative biclustering for multi-view data, Statistical methods in medical research 31 (11) (2022) 2201–2216.

[2] J. Wang, S. E. Safo, Deep ida: A deep learning method for integrative discriminant analysis of multi-view data with feature ranking–an application to covid-19 severity, ArXiv (2021).

[3] C. Lee, M. van der Schaar, A variational information bottleneck approach to multi-omics data integration, in: International Conference on Artificial Intelligence and Statistics, PMLR, 2021, pp. 1513–1521.

[4] S. E. Safo, H. Lu, Scalable randomized kernel methods for multiview data integration and prediction, arXiv preprint arXiv:2304.04692 (2023).

[5] S. E. Safo, J. Ahn, Y. Jeon, S. Jung, Sparse generalized eigenvalue problem with application to canonical correlation analysis for integrative analysis of methylation and gene expression data, Biometrics 74 (4) (2018) 1362–1371. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/biom.12886, doi:10.1111/biom.12886.
URL `https://onlinelibrary.wiley.com/doi/abs/10.1111/biom.12886`

[6] S. E. Safo, E. J. Min, L. Haine, Sparse linear discriminant analysis for multiview structured data, Biometrics n/a (n/a) (2021). arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/biom.13458, doi:https://doi.org/10.1111/biom.13458.
URL `https://onlinelibrary.wiley.com/doi/abs/10.1111/biom.13458`

[7] H. Hotelling, Relations between two sets of variables, Biometrika (1936) 312–377.

[8] A. J. Butte, P. Tamayo, D. Slonim, T. R. Golub, I. S. Kohane, Discovering functional relationships between rna expression and chemotherapeutic susceptibility using relevance networks, Proceedings of the National Academy of Sciences 97 (22) (2000) 12182–12186.

[9] I. González, K.-A. L. Cao, M. J. Davis, S. Déjean, Visualising associations between paired 'omics' data sets, BioData mining 5 (2012) 1–23.

Overview of mvlearnR

by Sandra E. Safo and Elise Palzer

2023-06-13

Installation

The package can be downloaded from GitHub at (https://github.com/lasandrall/mvlearnR). To install the package, you need to have the R-package *devtools* installed. Then use the function *install_github()* to install the package.

```
if(!"mvlearnR" %in% installed.packages()){
    library(devtools)
    devtools::install_github("lasandrall/mvlearnR")
}
library(mvlearnR)
```

Data import and prefiltering

We provide real data pertaining to COVID-19 severity and status and several simulated data to demonstrate the use of the package. Simulated data for two views and a binary outcome could be read as *data(sidaData)*, *data(selpData)*. The COVID-19 data can be imported as *data(COVID)*. This is a list with 3 entries: Proteomics, RNASeq, and Clinical data.

We provide functions to filter data either using supervised or unsupervised statistical methods. The function *filter.supervised()* can be used to filter each view using three methods: *linear*, *logistic*, *t-test*, and *Kruskal-Wallis (KW)* test. P-values can be adjusted for multiple hypothesis testing. The function *filter.unsupervised()* can be used to filter each view using unsupervised methods that include: *variance* and *interquartile range (IQR)* filtering.

Results from the supervised filtering approach could be visualized via volcano plots using the function *volcanoPlot()*. The filtered or original data could be visualized via UMAP with the function *umapPlot()*.

```
#load data
data("COVIDData")

#make omics data numeric
Proteomics= apply(as.matrix(COVIDData[[1]]), 2, as.numeric)
RNASeq= apply(as.matrix(COVIDData[[2]]), 2, as.numeric)
Clinical= COVIDData[[3]]
```

```r
#Supervised filtering- Logistic regression with B-H adjusted pvalues
X=list(Proteomics, RNASeq )
Y=Clinical$DiseaseStatus.Indicator
filterOmics=filter.supervised( X,
          Y,
          method = "logistic",
          padjust=TRUE,
          adjmethod="BH",
          thresh = 0.05,
          center = FALSE,
          scale = FALSE,
          log2TransForm = FALSE,
          standardize=TRUE,
          Xtest = NULL
                        )
#> [1] "Printing top 10 results for  significant variables for View 1"
#>
Coef
#> P51884
-1.806251
#> A0A0C4DFP6;Q9NQ79-2;Q9NQ79;Q9NQ79-3;Q5T4F6
-1.985417
#> E9PEK4;P07333;P07333-2
-2.546992
#> P02765;C9JV77
-1.391253
#> P04196
-1.589441
#> P30491;P30685;P30490;P18464;P10319;P30498;P30483;P18463;P18465;P30487;P304
88;P30485  1.038614
#> Q08380
1.546364
#> D6W5L6;P07988;H0Y7V6
1.215754
#> G3XAP6;P49747;P49747-2;CON__ENSEMBL:ENSBTAP00000006074
-1.745330
#> C9JB55
-2.053021
#>
Pval
#> P51884
0.0005174311
#> A0A0C4DFP6;Q9NQ79-2;Q9NQ79;Q9NQ79-3;Q5T4F6
0.0011712642
#> E9PEK4;P07333;P07333-2
0.0011712642
#> P02765;C9JV77
```

```
0.0011712642
#> P04196
0.0011712642
#> P30491;P30685;P30490;P18464;P10319;P30498;P30483;P18463;P18465;P30487;P304
88;P30485 0.0011712642
#> Q08380
0.0011712642
#> D6W5L6;P07988;H0Y7V6
0.0012083554
#> G3XAP6;P49747;P49747-2;CON__ENSEMBL:ENSBTAP00000006074
0.0014022843
#> C9JB55
0.0017750692
#>
Keep
#> P51884
TRUE
#> A0A0C4DFP6;Q9NQ79-2;Q9NQ79;Q9NQ79-3;Q5T4F6
TRUE
#> E9PEK4;P07333;P07333-2
TRUE
#> P02765;C9JV77
TRUE
#> P04196
TRUE
#> P30491;P30685;P30490;P18464;P10319;P30498;P30483;P18463;P18465;P30487;P304
88;P30485 TRUE
#> Q08380
TRUE
#> D6W5L6;P07988;H0Y7V6
TRUE
#> G3XAP6;P49747;P49747-2;CON__ENSEMBL:ENSBTAP00000006074
TRUE
#> C9JB55
TRUE
#>
View
#> P51884
1
#> A0A0C4DFP6;Q9NQ79-2;Q9NQ79;Q9NQ79-3;Q5T4F6
1
#> E9PEK4;P07333;P07333-2
1
#> P02765;C9JV77
1
#> P04196
1
#> P30491;P30685;P30490;P18464;P10319;P30498;P30483;P18463;P18465;P30487;P304
88;P30485    1
#> Q08380
```

```
1
#> D6W5L6;P07988;H0Y7V6
1
#> G3XAP6;P49747;P49747-2;CON__ENSEMBL:ENSBTAP00000006074
1
#> C9JB55
1
#> [1] "Printing top 10 results for  significant variables for View 2"
#>              Coef         Pval Keep View
#> ASPM    2.449057 0.0001041656 TRUE    2
#> BIRC5   2.153638 0.0001041656 TRUE    2
#> BRCA2   1.736562 0.0001041656 TRUE    2
#> BUB1    2.671646 0.0001041656 TRUE    2
#> BUB1B   2.467896 0.0001041656 TRUE    2
#> CCDC150 1.941286 0.0001041656 TRUE    2
#> CCNB2   2.486528 0.0001041656 TRUE    2
#> CDC20   1.934446 0.0001041656 TRUE    2
#> CDC25C  2.575770 0.0001041656 TRUE    2
#> CDCA2   2.509729 0.0001041656 TRUE    2
```

Unsupervised method for associating data from two sources using SELP

We applied *mvlearnR* to integrate RNA sequencing and proteomics data to identify variables that maximize association between genes and proteins. For this purpose, we used the filtered data from above (i.e. *filterOmics* results). We also provide simulated data that could be used.

We use the SELPCCA method. CCA is a multivariate linear method for maximizing association between data from two sources. CCA finds weighted combinations of variables in each data that maximize the overall dependency structure between pairs of data. The function *cvselpscca()* can be used to obtain low-dimensional linear representations maximizing associations between the genes and proteins. It will allow users to select genes and proteins contributing to maximal correlation between the pairs of data. . In the example below, we obtain the first and second canonical variates for each view with the option *ncancorr=2* in *cvselpscca()*.

```
#can use simulated data
#data("selpData")
#Xdata1 <- selpData[[1]]
#Xdata2 <- selpData[[2]]

#We use the filtered data from above  and obtain first and second canonical c
orrelation vectors
#We use cross-validation to choose tuning parameters for sparsity
Xdata1 <- filterOmics$X[[1]] #proteomics
Xdata2 <- filterOmics$X[[2]] #RNASeq
```

```
mycvselpcca=cvselpscca(Xdata1, Xdata2, ncancorr=2)
#> [1] "Current iteration is 1"
#> [1] "Current iteration is 2"
#> [1] "Applying optimal tuning parameter on whole data"
#> [1] "Current Iteration Is: 1"
#> [1] "Current Iteration Is: 2"
#> [1] "Number of non-zero Xdata1 or View 1: 67 50"
#> [1] "Number of non-zero Xdata2 or View 2: 55 42"
#> [1] "Corr(Xdata1*alpha,Xdata2*beta): 0.605 0.572"
#> [1] "Sparse CCA CovStructure used is: Iden"
```

Prediction with learned low-dimensional representations from unsupervised methods

Our goal is to use the learned low-dimensional representation(s) to discriminate between those with and without COVID-19, or to predict COVID-19 severity (defined as hospital free days [HFD45]). Since SELPCCA is an unsupervised method, it can only be used to identify variables contributing to the maximum association between the two views.

If an outcome is available, one can associate the learned low-dimensional representation(s) with the outcome. We provide the function *selpscca.pred()* for this purpose. We provide the options *gaussian*, *binomial*, *poisson*, and *survival* to model continuous, categorical, count, or time-to-event data, respectively.

Application of this function using the first and second canonical variates for the genes and proteins and the outcome of COVID-19 status suggests that the first canonical variates for the proteomics and gene data are able to discriminate between those with and without COVID-19.

The function *predict()* can be used to predict out of sample data using the learned low-dimensional representations. The option *type* in *predict()* is the type of prediction required, and it follows the same definition in *predict.glm()* and *predict.coxph()* in R.

We use results from cvselp() to predict COVID-19 severity (defined as hospital free days [HFD45]). Note that HFD45 score of 0 indicates the highest severity and the patient was in the hospital after 45 days, or died before the 45-day period ended. A higher score indicates lower disease severity.

```
#Use results from cvselpcca. One can also use selpPredict() to train the mode
l by setting fitselpCCA=NULL.

Y.HFD45=as.numeric(Clinical$HFD45)
myresult.HFD45=selpscca.pred(Xdata1, Xdata2, Y.HFD45,ncancorr=2,fitselpCCA=my
cvselpcca, family="gaussian",showProgress=T)
#> Fitting SELPCCA Model
#> Fitting Prediction Model
```

```
print(summary(myresult.HFD45[["mod.fit"]]))
#>
#> Call:
#> stats::lm(formula = Y ~ ., data = selp.dat)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -30.985 -10.822   2.173  10.677  25.040
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  23.8667     1.2975  18.394  < 2e-16 ***
#> X1_1         -2.2540     0.4850  -4.647 9.02e-06 ***
#> X1_2         -1.9374     0.9898  -1.957   0.0527 .
#> X2_1          0.3811     0.3122   1.221   0.2247
#> X2_2          0.2156     0.4508   0.478   0.6334
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 14.21 on 115 degrees of freedom
#> Multiple R-squared:  0.3065, Adjusted R-squared:  0.2823
#> F-statistic:  12.7 on 4 and 115 DF,  p-value: 1.354e-08



#Predict out of sample data
newPredictions=predict(myresult.HFD45, newdata=Xdata1, newdata2=Xdata2,type="
response")

summary(newPredictions$pred.mod)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>   4.467  17.438  22.548  23.867  30.737  45.884
```

Print summary of results.

Results suggest that the first canonical variate for view 1 is significantly associated with hospital free days (p-value < 0.05)

```
print(summary(myresult.HFD45[["mod.fit"]]))
#>
#> Call:
#> stats::lm(formula = Y ~ ., data = selp.dat)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -30.985 -10.822   2.173  10.677  25.040
```

```
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  23.8667     1.2975  18.394  < 2e-16 ***
#> X1_1         -2.2540     0.4850  -4.647 9.02e-06 ***
#> X1_2         -1.9374     0.9898  -1.957   0.0527 .
#> X2_1          0.3811     0.3122   1.221   0.2247
#> X2_2          0.2156     0.4508   0.478   0.6334
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 14.21 on 115 degrees of freedom
#> Multiple R-squared:  0.3065, Adjusted R-squared:  0.2823
#> F-statistic:  12.7 on 4 and 115 DF,  p-value: 1.354e-08
```

Predict out of sample data.

Results suggest that the first canonical variate for view 1 (proteins) is significantly associated with hospital free days (p-value < 0.05), while that for view 2 (RNASeq) is marginally associated with hospital free days (p-value=0.0527).

```
newPredictions=predict(myresult.HFD45, newdata=Xdata1, newdata2=Xdata2,type="
response")

summary(newPredictions$pred.mod)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>   4.467  17.438  22.548  23.867  30.737  45.884

#mean squared error
sum((newPredictions$pred.mod-Y.HFD45)^2)/length(Y.HFD45)
#> [1] 193.6126
```

We use results from cvselp() to predict COVID-19 class membership. We obtain predicted probabilities, round these probabilities for class assignment, and obtain classification error.

```
#Use results from cvselpcca. One can also use selpPredict() to train the mode
l by setting fitselpCCA=NULL.

Y=Clinical$DiseaseStatus.Indicator
myresult=selpscca.pred(Xdata1, Xdata2, Y,fitselpCCA=mycvselpcca, family="bino
mial",showProgress=T)
#> Fitting SELPCCA Model
#> Fitting Prediction Model
```

Print summary of results.

Results suggest that the first canonical variate for view 1 and view 2 is significantly associated with COVID-19 status (p-value < 0.05). That is, the first canonical variates for

views 1 (proteins) and 2 (genes) is able to discriminate between those with and without COVID-19.

```
print(summary(myresult[["mod.fit"]]))
#>
#> Call:
#> stats::glm(formula = factor(Y) ~ ., family = stats::binomial,
#>     data = selp.dat)
#>
#> Coefficients:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept)  9.07978    3.40891   2.664  0.00773 **
#> X1_1         1.13278    0.46940   2.413  0.01581 *
#> X1_2         0.10359    0.56277   0.184  0.85395
#> X2_1         0.93240    0.43798   2.129  0.03327 *
#> X2_2        -0.08608    0.26547  -0.324  0.74575
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for binomial family taken to be 1)
#>
#>     Null deviance: 114.339  on 119  degrees of freedom
#> Residual deviance:  16.725  on 115  degrees of freedom
#> AIC: 26.725
#>
#> Number of Fisher Scoring iterations: 10
```

Predict out of sample data.

In this example we obtain predictions based on training data but one can also use testing data.

```
newpredictions=predict(myresult, newdata=Xdata1, newdata2=Xdata2)

summary(newpredictions$pred.mod) #predictied probabilities
#>      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
#> 0.0000177 0.9639233 0.9999817 0.8166667 0.9999998 1.0000000

##Misclassification Rate
sum((round(newpredictions$pred.mod)-Y)^2)/length(Y)
#> [1] 0.05
```

SELP Predict Example with train and testing data using simulated data.

```
#Load Data
data("sidaData")
Xdata <- sidaData[[1]]
Xdata[[1]] <- Xdata[[1]][,1:500]
Xdata[[2]] <- Xdata[[2]][,1:500]
Y.simul <- sidaData[[2]]-1 #Y needs to be 0 or 1 for binomial family
```

```
Xtestdata <- sidaData[[3]]
Xtestdata[[1]] <- Xtestdata[[1]][,1:500]
Xtestdata[[2]] <- Xtestdata[[2]][,1:500]
Ytest.simul <- sidaData[[4]]-1

fit.selp <- selpscca.pred(Xdata[[1]], Xdata[[2]], Y.simul,fitselpCCA=NULL,
              family="binomial")
#> Fitting SELPCCA Model
#> [1] "Current iteration is 1"
#> [1] "Current iteration is 2"
#> [1] "Current iteration is 3"
#> [1] "Current iteration is 4"
#> [1] "Current iteration is 5"
#> [1] "Applying optimal tuning parameter on whole data"
#> [1] "Current Iteration Is: 1"
#> [1] "Current Iteration Is: 2"
#> [1] "Current Iteration Is: 3"
#> [1] "Current Iteration Is: 4"
#> [1] "Current Iteration Is: 5"
#> [1] "Number of non-zero Xdata1 or View 1: 20"
#> [1] "Number of non-zero Xdata2 or View 2: 20"
#> [1] "Corr(Xdata1*alpha,Xdata2*beta): 0.985"
#> [1] "Sparse CCA CovStructure used is: Iden"
#> Fitting Prediction Model
#> Warning: glm.fit: algorithm did not converge
#> Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

pred.fit <- predict(fit.selp, newdata=Xtestdata[[1]],
                    newdata2=Xtestdata[[2]], type = "response")

summary(pred.fit$pred.mod)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>  0.0000  0.0000  0.2066  0.4977  1.0000  1.0000

#Misclassification Rate
sum((round(pred.fit$pred.mod)-Ytest.simul)^2)/length(Ytest.simul)
#> [1] 0.015625
```

Supervised methods for associating data from two or more sources

Sparse integrative discriminant analysis (SIDA) is an integrative analysis method for jointly modeling associations between two or more views. The algorithm considers the overall association between multiview data, and the separation within each view when choosing discriminant vectors that are associated and optimally separate subjects. Thus,this method combines the advantages of linear discriminant analysis (LDA) for maximizing separation between classes in a view, and canonical correlation analysis (CCA) for maximizing correlation between two data types.

In particular, SIDA maximizes the sum of between-class separations (e.g. COVID-19 versus non-COVID-19) and the sum of squared correlations between pairs of molecular data. SIDA allows to select key variables that contribute to the maximum association of the views and separation of the classes. The method SIDANet can be used when there are prior information in the form of variable-variable connectivity.

The function *cvSIDA()* performs nfold cross-validation to select optimal tuning parameters for SIDA based on training data, and predicts training or testing class membership.

We demonstrate the SIDA method on the data pertaining to COVID-19 data. Our goal is to identify multidimensional genes and proteins that are correlated and able to discriminate between patients with and without COVID-19.

```
#can use simulated data
# data("sidaData")
# Xdata <- sidaData[[1]]
# Y <- sidaData[[2]]
# Xtestdata <- sidaData[[3]]
# Ytest <- sidaData[[4]]

#We use the filtered data from above and obtain discriminant vectors that max
imize association
#between gene and protein data while discriminating between those with and wi
thout COVID-19.
#We use cross-validation to choose variables
Xdata1 <- filterOmics$X[[1]] #proteins
Xdata2 <- filterOmics$X[[2]] #RNASeq
Y=filterOmics$Y+1  # class membership needs to be numeric, coded as 1, 2, ...
.
Xdata=list(Xdata1, Xdata2)



#Check cvsida
fit.cvsida <- cvSIDA(Xdata, Y,
                     Xtestdata = Xdata,
                     Ytest = Y, plotIt = F)
#> [1] "Getting tuning grid values"
#> [1] "Completed at time"
#> Time difference of 18.49425 secs
#> Begin 5 -folds cross-validation
#> [1] "Cross-validation completed at time"
#> Time difference of 4.656913 mins
#> [1] "Getting Results......"
#> Estimated Test Classification Error is 0.01666667
#> Estimated Train Classification Error is 0.01666667
#> Estimated Test Correlation is 0.5281469
#> Estimated Train Correlation is 0.5281469
```

```
#> Number of nonzero coefficients in view 1 is 20
#> Number of nonzero coefficients in view 2 is 72
#> [1] "Total time used is"
#> Time difference of 5.441563 mins
```

The function *cvSIDANet()* incorporates prior structural information (e.g. gene-gene connectivity) in SIDA via the use of the normalized Laplacian of graph, thus encouraging selection of predictors that are connected and behave similarly.

Covariates, if available, can be included. We provide the option to select or force covariates in the models. To select covariates (or shrink weights of some covariates to zero), set *withCov = FALSE*. To force weights of covariates to not be shruck to zero, set *withCov = TRUE*. Unlike *cvselpscca()*, *cvSIDA()* or *cvSIDANet()* is applicable to two or more views.

We demonstrate the use of *cvSIDANet()* on simulated data.

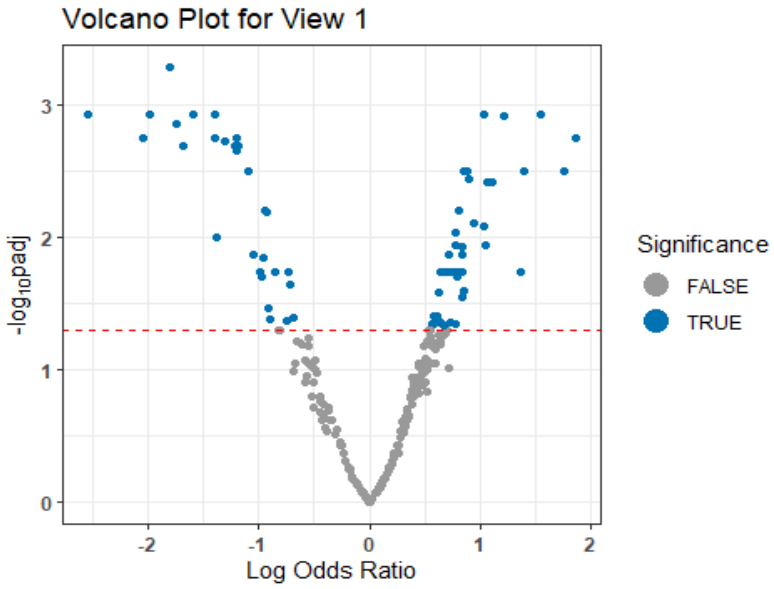
```
data("sidanetData")
Xdata <- sidanetData[[1]]
Y <- sidanetData[[2]] #class membership already coded as 1,2,...
Xtestdata <- sidanetData[[3]]
Ytest <- sidanetData[[4]] #class membership already coded as 1,2,...

#edge information
myedges=sidanetData[[5]]
myedgeweight=sidanetData[[6]]

 ##---- call cross validation
mycvsidanet=cvSIDANet(Xdata,Y,myedges,myedgeweight,withCov=FALSE,plotIt=FALSE
,Xtestdata=Xtestdata,
     Ytest=Ytest)
#> [1] "Getting tuning grid values"
#> [1] "Completed at time"
#> Time difference of 9.139924 secs
#> Begin 5 -folds cross-validation
#> [1] "Cross-validation completed at time"
#> Time difference of 6.051568 mins
#> [1] "Getting Results......"
#> Estimated Test Classification Error is 0.03333333
#> Estimated Train Classification Error is 0.04166667
#> Estimated Test Correlation is 0.8870161
#> Estimated Train Correlation is 0.8994912
#> Number of nonzero coefficients in view 1 is 20
#> Number of nonzero coefficients in view 2 is 111
#> Number of nonzero coefficients in view 1 is 40
#> Number of nonzero coefficients in view 2 is 222
#> [1] "Total time used is"
#> Time difference of 6.615679 mins
```

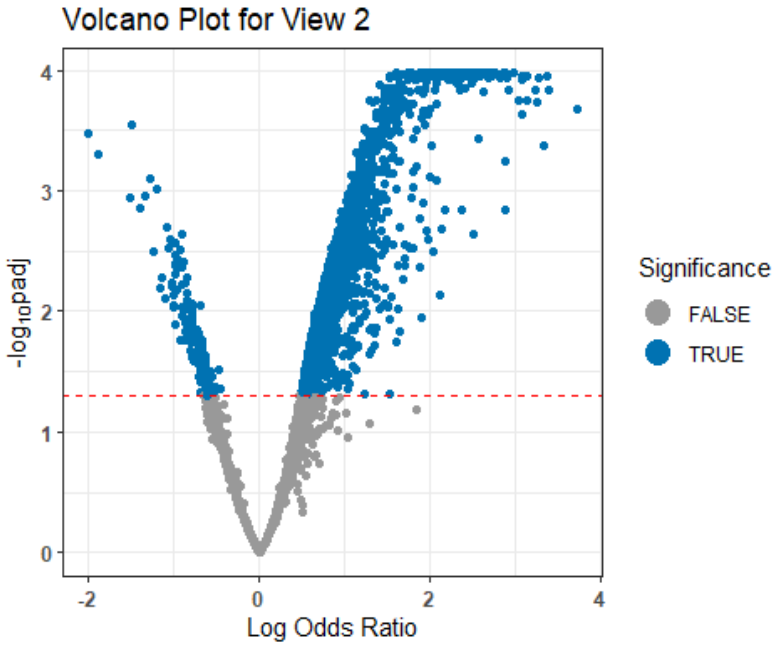
Visualizations in *mvlearnR*

Volcano Plots

Results from the supervised filtering approach for each data could be visualized via volcano plots using the function *volcanoPlot()*.

```
volcanoPlot(filterOmics)
```

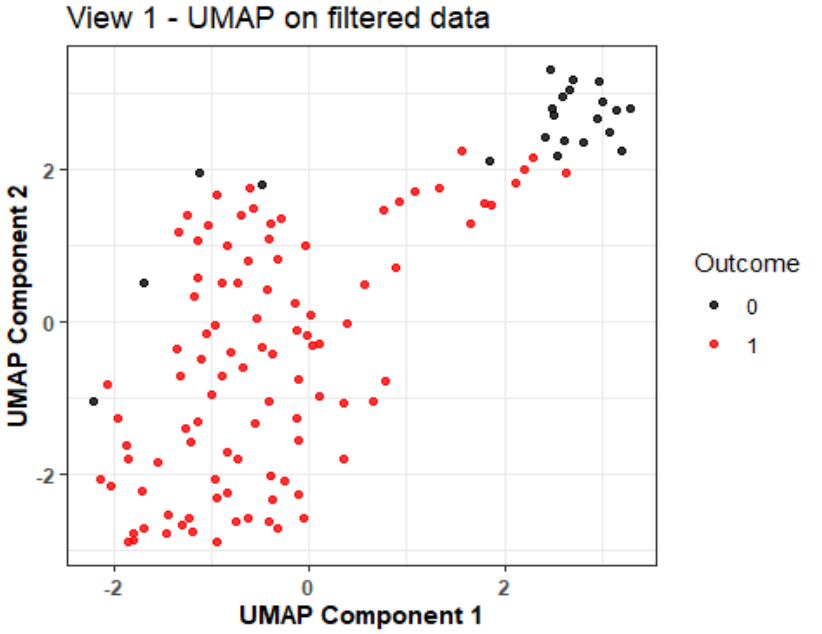


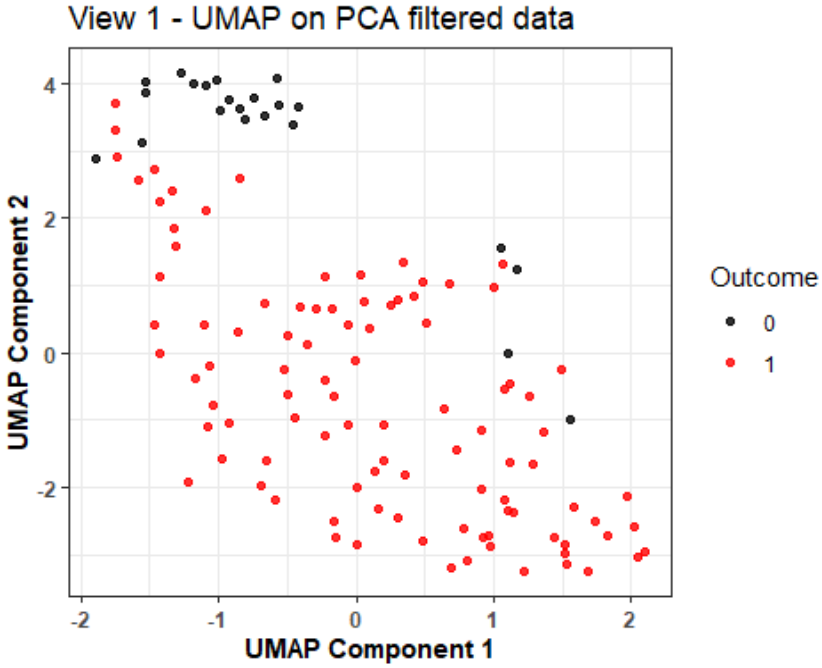


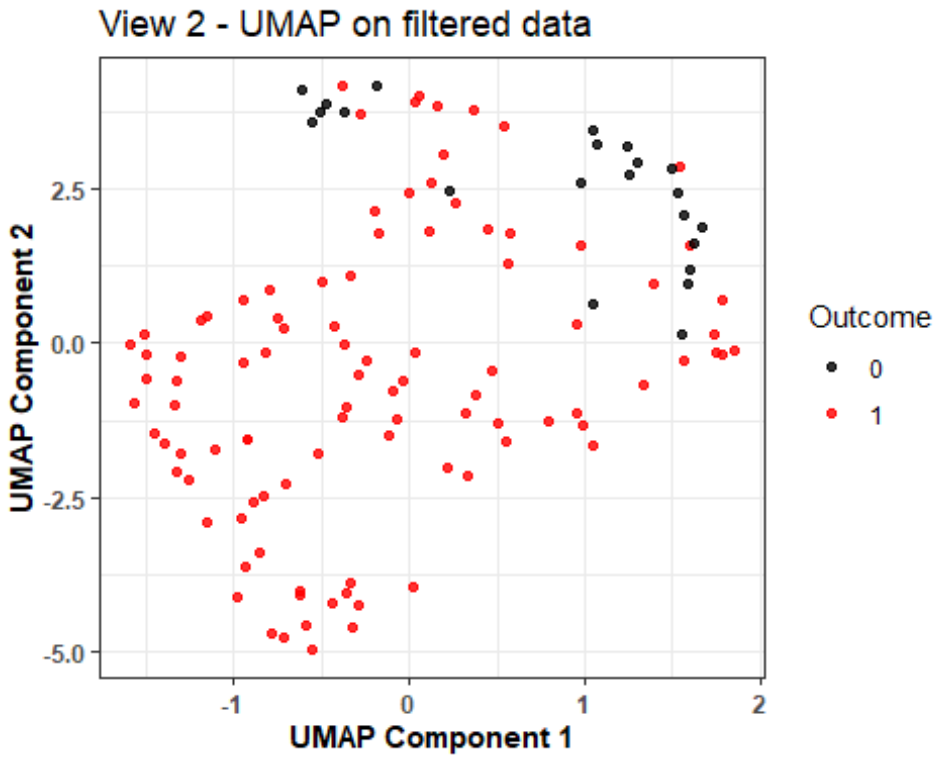
UMAP plots

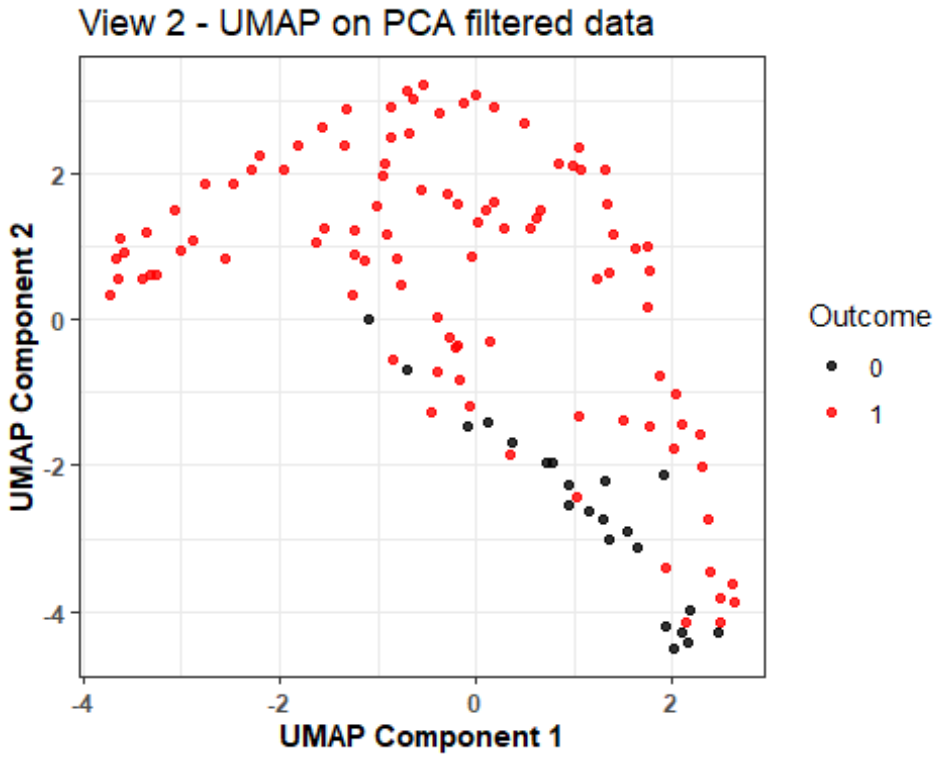
The filtered or original data could be visualized with UMAP (uniform manifold approximation and projection) with the function umapPlot(). Here, UMAP can be applied to PCA reduced data as well as the original or filtered data.

```
umapPlot(filterOmics)
```






View 2 - UMAP on filtered data
UMAP Component 2
2.5
0.0
-2.5
-5.0
-1
0
1
2
UMAP Component 1
Outcome
0
1


View 2 - UMAP on PCA filtered data
UMAP Component 2
2
0
-2
-4
-4
-2
0
2
UMAP Component 1
Outcome
0
1

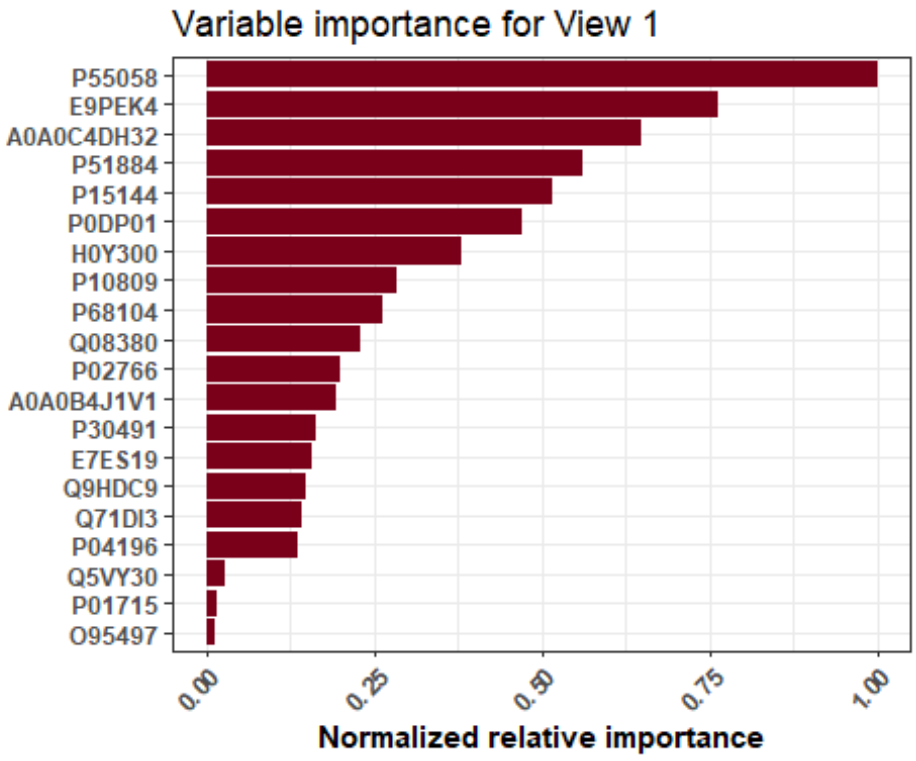
Variable Importance Plots

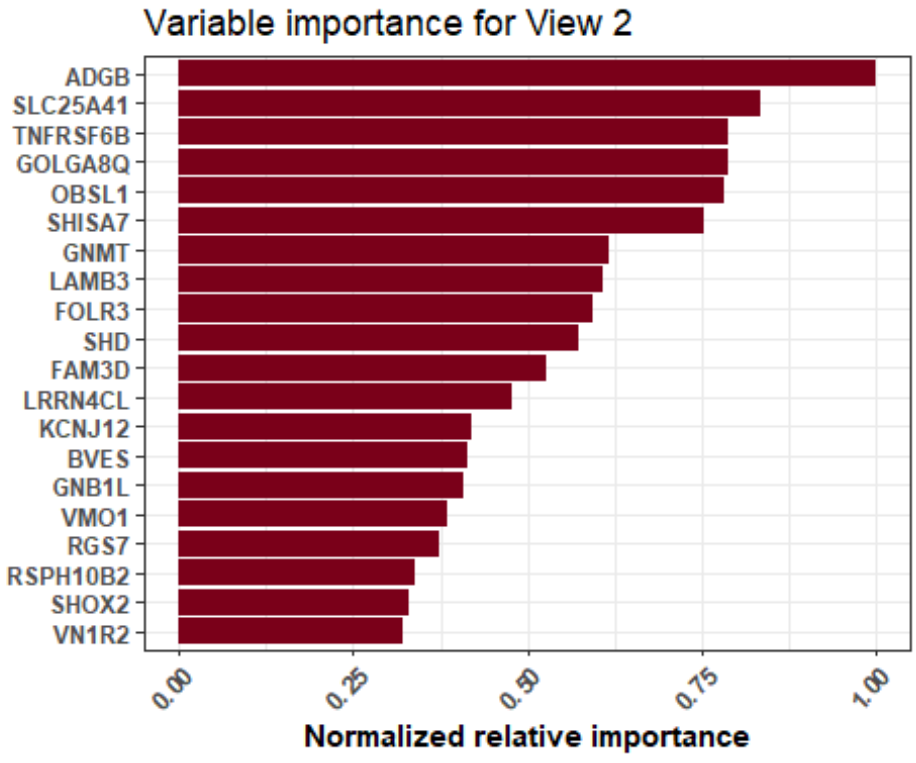
We provide the function *VarImportancePlot()* to visualize the weights (in absolute value) of the low-dimensional loadings. Since these loadings are standardized to have unit norm, a variable with larger weight contributes more to the association between the views (for SELPCCA) or to the association between the views and discrimination of classes within each view (for SIDA and SIDANet). We only show the top 20 variables and their weights but one can view data matrix for all variables.

Variable importance tables and plots from the supervised integrative analysis method, SIDA

```
VarImportancePlot(fit.cvsida)
#>    View Variable Name Loading Absolute Loading Normalized Relative Importa
nce
#> 1     1        P55058  -0.546            0.546                          1.
000
#> 2     1        E9PEK4  -0.417            0.417                          0.
764
#> 3     1    A0A0C4DH32   0.355            0.355                          0.
649
#> 4     1        P51884  -0.307            0.307                          0.
563
#> 5     1        P15144  -0.282            0.282                          0.
517
#> 6     1        P0DP01   0.258            0.258                          0.
472
#> 7     1        H0Y300   0.207            0.207                          0.
379
#> 8     1        P10809   0.156            0.156                          0.
286
#> 9     1        P68104   0.143            0.143                          0.
263
#> 10    1        Q08380   0.126            0.126                          0.
231
#> 11    1        P02766  -0.109            0.109                          0.
199
#> 12    1    A0A0B4J1V1   0.106            0.106                          0.
195
#> 13    1        P30491   0.090            0.090                          0.
165
#> 14    1        E7ES19  -0.087            0.087                          0.
159
#> 15    1        Q9HDC9   0.081            0.081                          0.
149
#> 16    1        Q71DI3   0.078            0.078                          0.
142
#> 17    1        P04196  -0.075            0.075                          0.
```

```
137
#> 18    1        Q5VY30  -0.016            0.016                          0.
028
#> 19    1        P01715   0.008            0.008                          0.
015
#> 20    1        O95497  -0.008            0.008                          0.
014
#>    View Variable Name Loading Absolute Loading Normalized Relative Importa
nce
#> 1     2          ADGB  -0.345            0.345                          1.
000
#> 2     2      SLC25A41  -0.288            0.288                          0.
835
#> 3     2      TNFRSF6B   0.273            0.273                          0.
790
#> 4     2       GOLGA8Q  -0.272            0.272                          0.
788
#> 5     2         OBSL1   0.270            0.270                          0.
781
#> 6     2         SHISA7  -0.260            0.260                          0.
754
#> 7     2          GNMT  -0.214            0.214                          0.
619
#> 8     2         LAMB3   0.210            0.210                          0.
608
#> 9     2         FOLR3   0.205            0.205                          0.
594
#> 10    2           SHD   0.198            0.198                          0.
574
#> 11    2         FAM3D   0.182            0.182                          0.
528
#> 12    2       LRRN4CL  -0.165            0.165                          0.
478
#> 13    2        KCNJ12  -0.145            0.145                          0.
420
#> 14    2          BVES   0.143            0.143                          0.
415
#> 15    2         GNB1L   0.141            0.141                          0.
409
#> 16    2          VMO1   0.133            0.133                          0.
384
#> 17    2          RGS7  -0.129            0.129                          0.
375
#> 18    2      RSPH10B2   0.117            0.117                          0.
338
#> 19    2         SHOX2   0.113            0.113                          0.
329
#> 20    2         VN1R2  -0.111            0.111                          0.
321
```





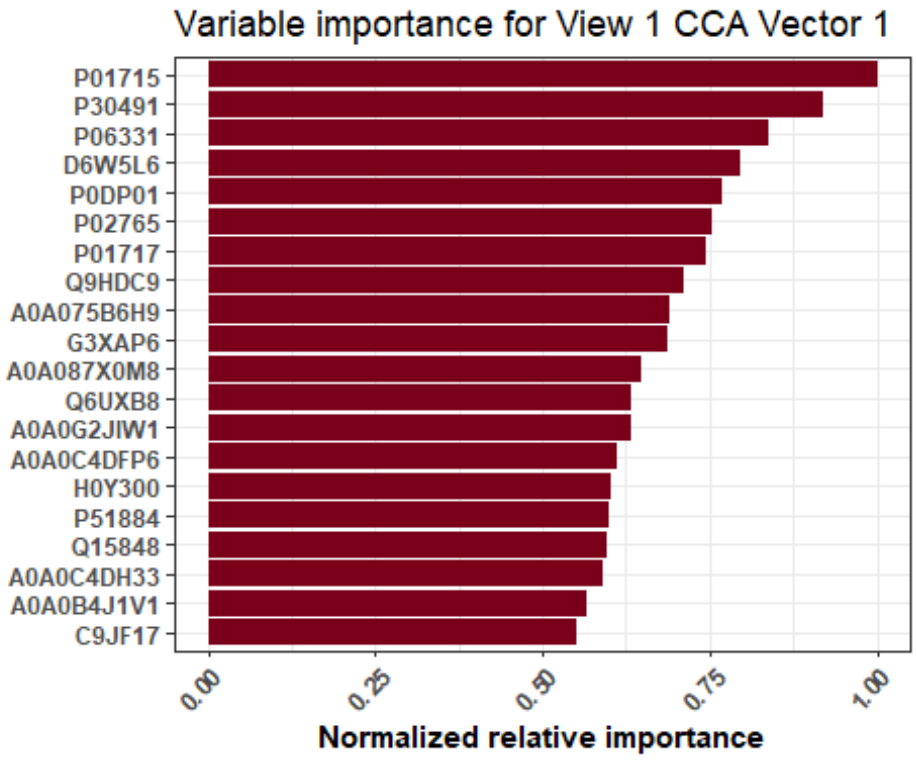

```
#> NULL
```

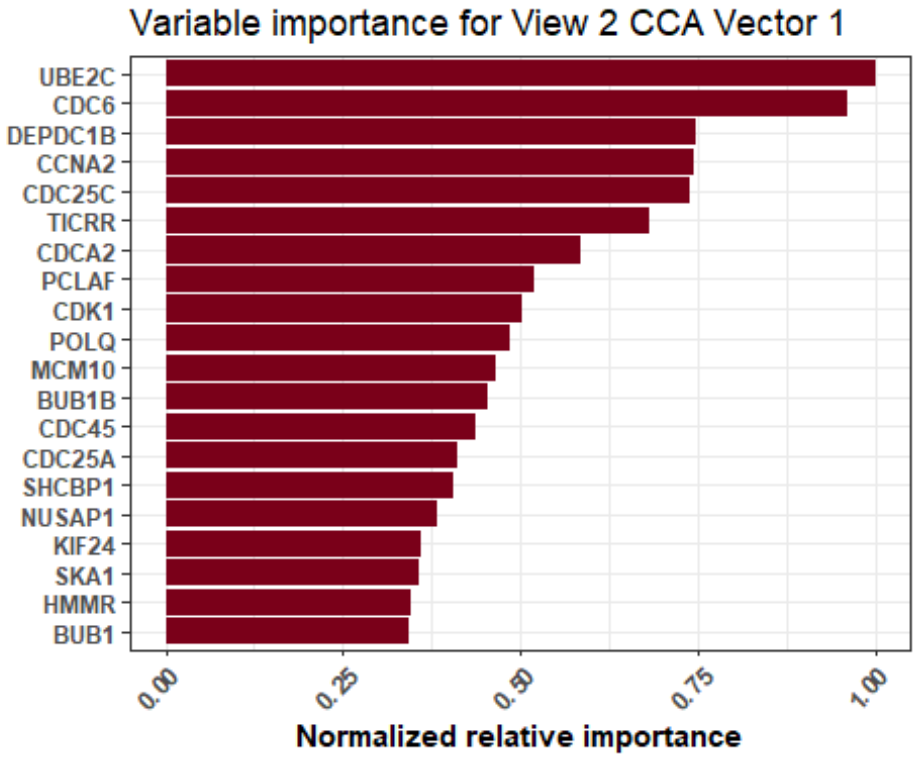
Here's the variable importance tables and plots from SELPCCA (the unsupervised integrative analysis method)

```
VarImportancePlot(mycvselpcca)
#>    View 1 Canonical Correlation Vector
#> 1                                    1
#> 2                                    1
#> 3                                    1
#> 4                                    1
#> 5                                    1
#> 6                                    1
#> 7                                    1
#> 8                                    1
#> 9                                    1
#> 10                                   1
#>                                                                        V
ariable Name
#> 1
P01715
#> 2  P30491;P30685;P30490;P18464;P10319;P30498;P30483;P18463;P18465;P30487;P
30488;P30485
#> 3
P06331
#> 4                                                                D6W5L6;P
07988;H0Y7V6
#> 5
P0DP01
#> 6                                                                       P
02765;C9JV77
#> 7
P01717
#> 8                                                              Q9HDC9;H0Y
512;Q9HDC9-2
#> 9
A0A075B6H9
#> 10                            G3XAP6;P49747;P49747-2;CON__ENSEMBL:ENSBTA
P00000006074
#>    Loading Absolute Loading Normalized Relative Importance
#> 1    0.250            0.250                          1.000
#> 2    0.229            0.229                          0.918
#> 3    0.209            0.209                          0.839
#> 4    0.199            0.199                          0.796
#> 5    0.192            0.192                          0.768
#> 6   -0.188            0.188                          0.755
#> 7    0.186            0.186                          0.743
#> 8    0.177            0.177                          0.710
#> 9    0.173            0.173                          0.691
#> 10  -0.172            0.172                          0.687
#>    View 1 Canonical Correlation Vector
#> 75                                   2
#> 76                                   2
#> 77                                   2
```

```
#> 78                                   2
#> 79                                   2
#> 80                                   2
#> 81                                   2
#> 82                                   2
#> 83                                   2
#> 84                                   2
#>                                                                            V
ariable Name
#> 75                                                                         Q
5VY30;P02753
#> 76                                                            P02766;A0A087WT5
9;A0A087WV45
#> 77                             A6XND1;A6XND0;P17936;P17936-2;H0Y5K2;B3KWK7;H
0Y485;C9JMX4
#> 78
Q9BXR6
#> 79
P06727
#> 80                                                                       Q96
PD5;Q96PD5-2
#> 81                                G3XAP6;P49747;P49747-2;CON__ENSEMBL:ENSBTA
P00000006074
#> 82
A0A075B6H9
#> 83                                                                         C
9JF17;P05090
#> 84 P30491;P30685;P30490;P18464;P10319;P30498;P30483;P18463;P18465;P30487;P
30488;P30485
#>    Loading Absolute Loading Normalized Relative Importance
#> 75  -0.312            0.312                          1.000
#> 76  -0.289            0.289                          0.927
#> 77  -0.269            0.269                          0.863
#> 78   0.256            0.256                          0.823
#> 79  -0.247            0.247                          0.792
#> 80  -0.243            0.243                          0.779
#> 81   0.226            0.226                          0.724
#> 82  -0.219            0.219                          0.701
#> 83  -0.212            0.212                          0.679
#> 84  -0.202            0.202                          0.649
#>    View 2 Canonical Vector Variable Name Loading Absolute Loading
#> 1                        1         UBE2C   0.351            0.351
#> 2                        1          CDC6   0.338            0.338
#> 3                        1       DEPDC1B   0.263            0.263
#> 4                        1         CCNA2   0.262            0.262
#> 5                        1        CDC25C   0.259            0.259
#> 6                        1         TICRR   0.240            0.240
#> 7                        1         CDCA2   0.206            0.206
#> 8                        1         PCLAF   0.182            0.182
#> 9                        1          CDK1   0.176            0.176
```

```
#> 10                         1          POLQ   0.171             0.171
#>    Normalized Relative Importance
#> 1                           1.000
#> 2                           0.961
#> 3                           0.748
#> 4                           0.745
#> 5                           0.738
#> 6                           0.683
#> 7                           0.586
#> 8                           0.519
#> 9                           0.501
#> 10                          0.486
#>      View 2 Canonical Vector Variable Name Loading Absolute Loading
#> 3083                       2         UPK3A  -0.373             0.373
#> 3084                       2         GPR35  -0.360             0.360
#> 3085                       2           IL4  -0.342             0.342
#> 3086                       2         SARDH  -0.299             0.299
#> 3087                       2   ARPIN-AP3S2  -0.257             0.257
#> 3088                       2        SLC2A6  -0.251             0.251
#> 3089                       2         OLFM4   0.221             0.221
#> 3090                       2         ASGR1  -0.203             0.203
#> 3091                       2          RIN1  -0.192             0.192
#> 3092                       2       CEACAM6   0.150             0.150
#>      Normalized Relative Importance
#> 3083                          1.000
#> 3084                          0.966
#> 3085                          0.916
#> 3086                          0.801
#> 3087                          0.688
#> 3088                          0.674
#> 3089                          0.592
#> 3090                          0.545
#> 3091                          0.515
#> 3092                          0.402
```


Variable importance for View 1 CCA Vector 1
P01715
P30491
P06331
D6W5L6
P0DP01
P02765
P01717
Q9HDC9
A0A075B6H9
G3XAP6
A0A087X0M8
Q6UXB8
A0A0G2JIW1
A0A0C4DFP6
H0Y300
P51884
Q15848
A0A0C4DH33
A0A0B4J1V1
C9JF17
0.00
0.25
0.50
0.75
1.00
Normalized relative importance

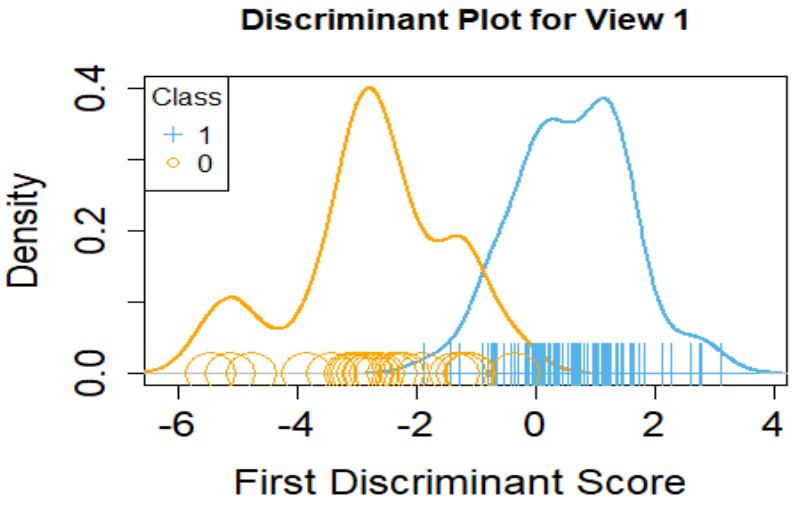

Variable importance for View 2 CCA Vector 1
UBE2C
CDC6
DEPDC1B
CCNA2
CDC25C
TICRR
CDCA2
PCLAF
CDK1
POLQ
MCM10
BUB1B
CDC45
CDC25A
SHCBP1
NUSAP1
KIF24
SKA1
HMMR
BUB1
0.00
0.25
0.50
0.75
1.00
Normalized relative importance

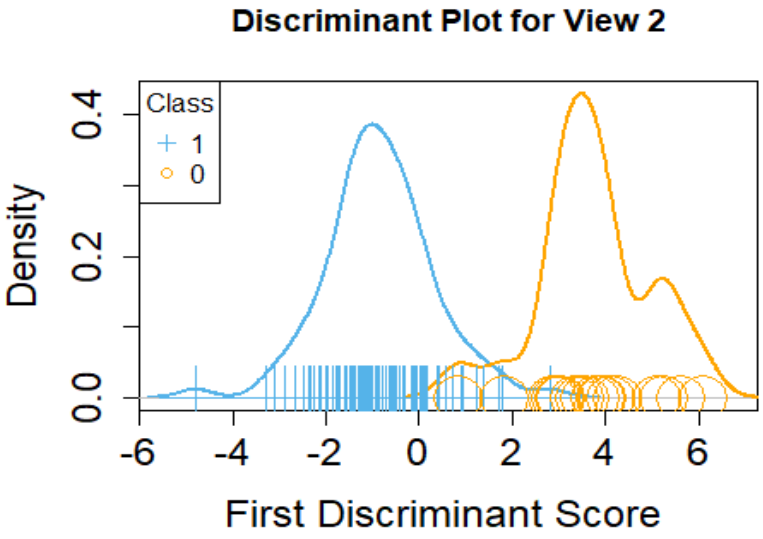
```
#> NULL
```

Discriminant Plots

We provide the function *DiscriminantPlots()* to plot the discriminant vectors for visualizing class separation. Here's an example plot with two classes.

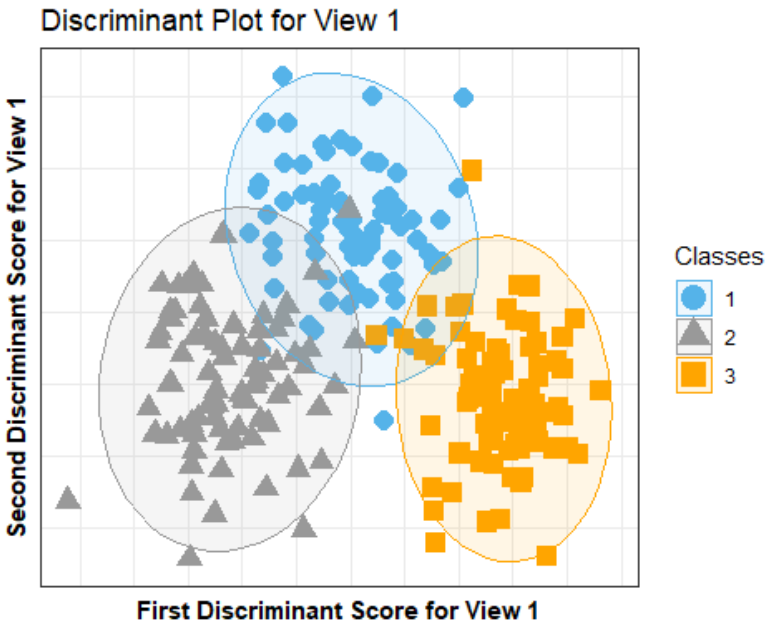
```
Xdata1 <- filterOmics$X[[1]]
Xdata2 <- filterOmics$X[[2]]
Y=filterOmics$Y  # class membership needs to be numeric, coded as 1, 2, ....
Xdata=list(Xdata1, Xdata2)


DiscriminantPlots(Xdata, Y, fit.cvsida$hatalpha)
```

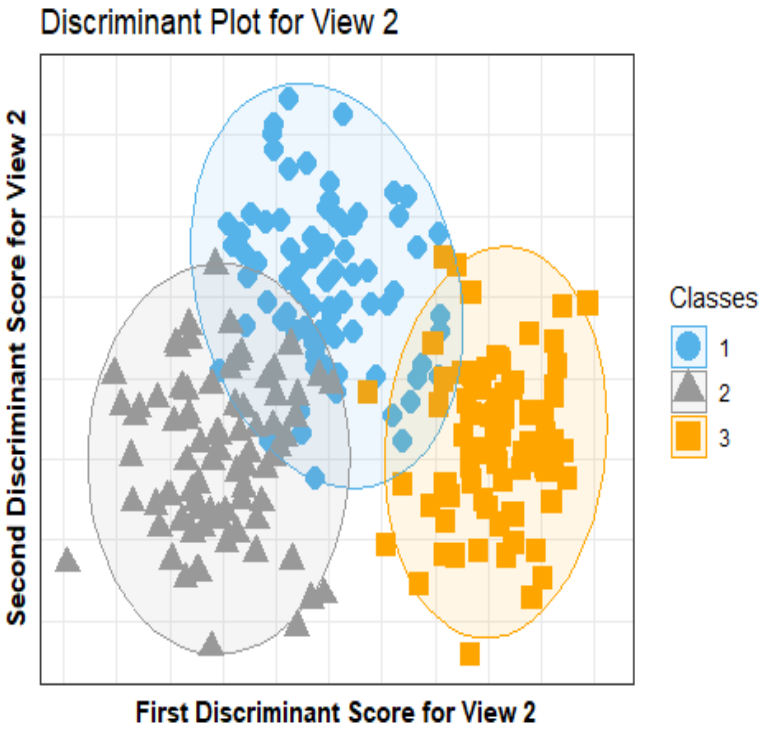




And here's an example plot with three classes

```
Xdata <- sidanetData[[1]]
Y <- sidanetData[[2]]

DiscriminantPlots(Xdata, Y, mycvsidanet$hatalpha)
```

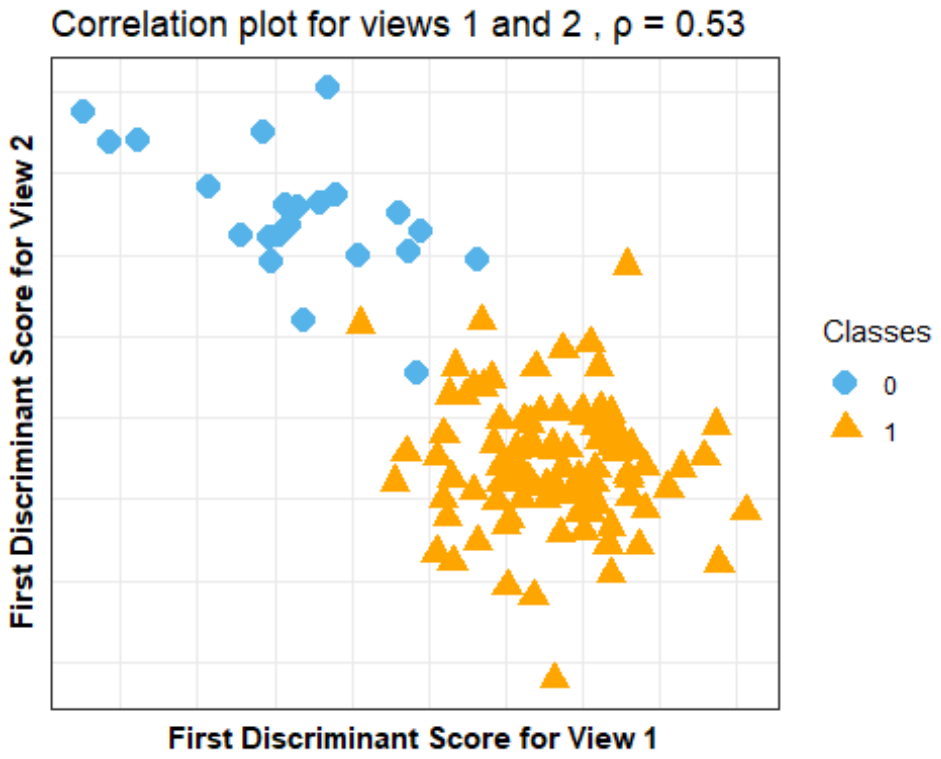




Correlation Plots

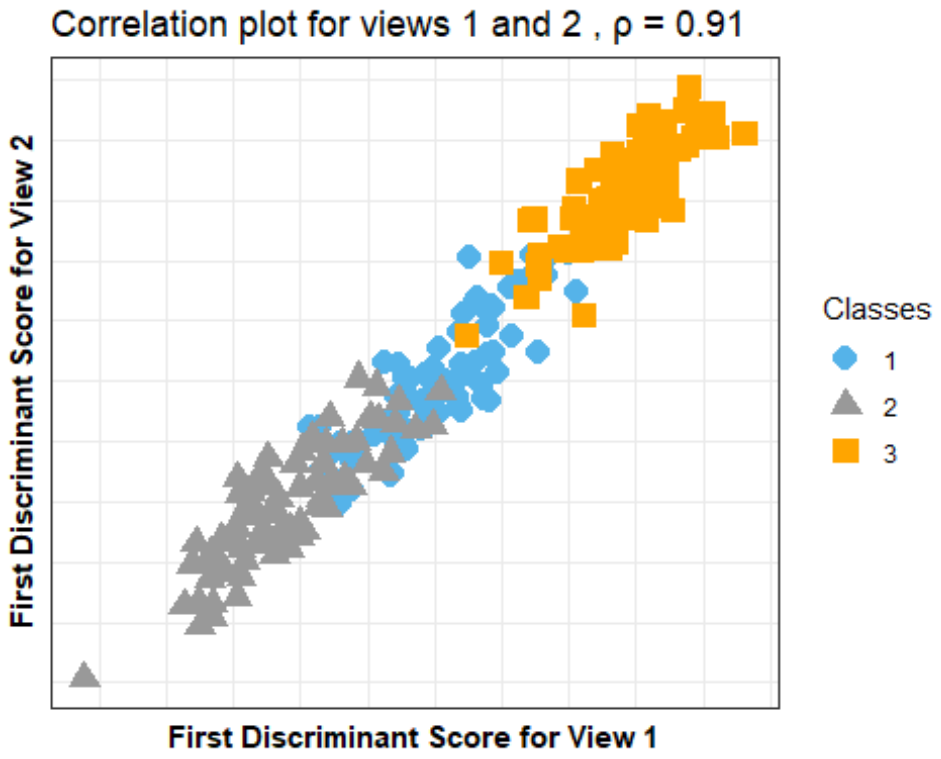
We also provide the function *CorrelationPlots()* for visualizing correlations between estimated discriminant vectors. Here's an example graph with two classes and views moderately correlated. Correlation estimate is given in absolute value.

```
Xdata1 <- filterOmics$X[[1]]
Xdata2 <- filterOmics$X[[2]]
Y=filterOmics$Y
Xdata=list(Xdata1, Xdata2)

CorrelationPlots(Xdata, Ytest=Y, fit.cvsida$hatalpha)
```



And here's an example plot with three classes and high correlation between pairs of views

```
Xdata <- sidanetData[[1]]
Y <- sidanetData[[2]]

CorrelationPlots(Xdata, Y, mycvsidanet$hatalpha)
```



Relevance Network Plot

We provide the function *networkPlot()* to visualize associations of selected variables between pairs of views. We estimate pairwise similarity matrix using low-dimensional representations of our sparse integrative analysis methods (selpcca, sida, sidanet). We create bipartite graph (bigraph) where variables or nodes from one view are connected to variables or nodes from another view. We construct the bigraph from a pairwise similarity matrix obtained from the outputs of our integrative analysis methods. We estimate the similarity score between a pair of selected variables from two views by calculating the inner product of each selected variable and the sum of canonical variates (for SELPCCA) or discriminant vectors (for SIDA, SIDANet) for the pairs of views. The entries in the similarity matrix is a robust approximation of the Pearson correlation between pairs of variables and the two views under consideration. This network graph has potential to shed light on the complex associations between pairs of views.

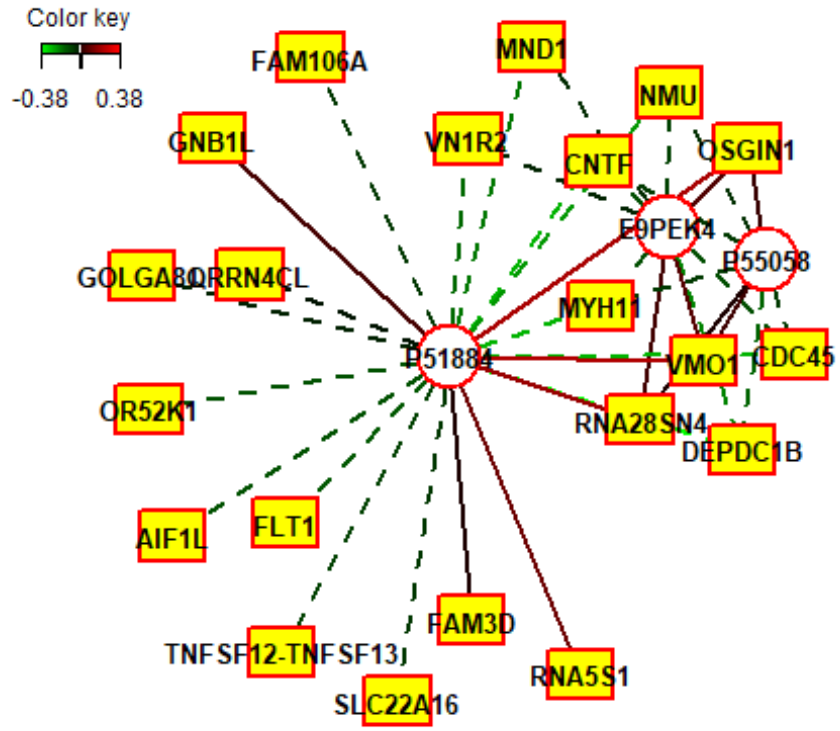
Variable pairs with high similarity score may be of interest. The relevance of the associations can be explored by changing the cutoff. This can also be used to reduce the size of the graph, for dense network.

By default, dashed lines indicate negative associations and solid lines represent positive associations.

Here's the relevance network for results using SIDA. We set the correlation cutoff to 0.25 for a less dense network.

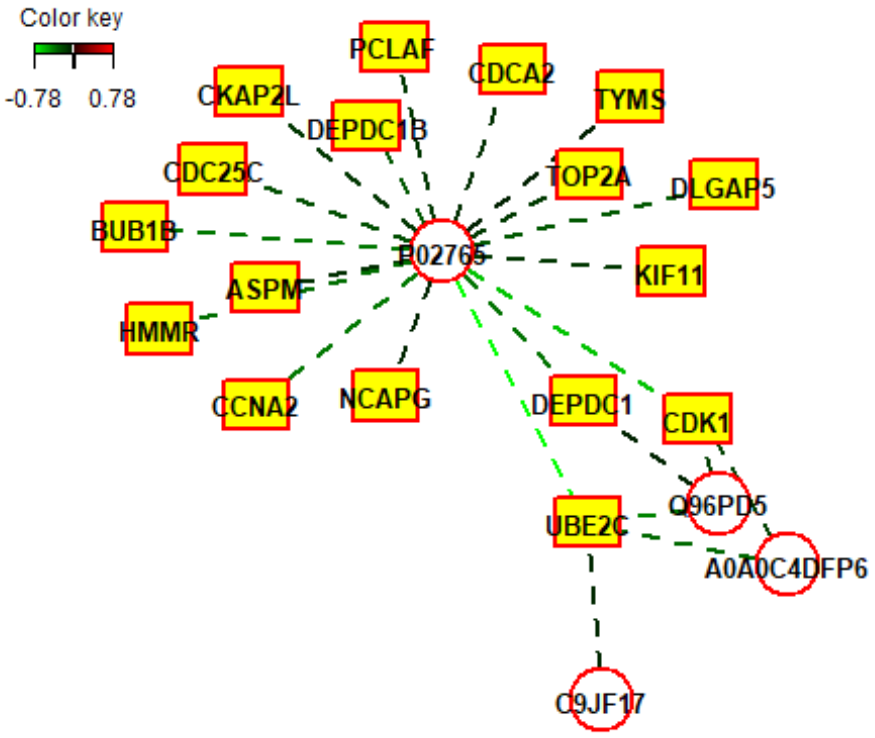
Results suggest that some of the highly ranked proteins (e.g. P55058, E9PEK4, P51884) are moderately associated with highly-ranked genes (GOLGA8Q, FAM3D),

```
networkPlot(fit.cvsida,cutoff=.25)
```



Here's the relevance network for results using SELPCCA. We set the correlation cutoff to 0.73 for a less dense network.

```
networkPlot(mycvselpcca, cutoff=0.73)
```



Loadings Plots

We provide the function *LoadingsPlot()* to plot discriminant and canonical correlation vectors. These graphs are useful for visualizing how selected variables from SIDA/SIDANet and SELPCCA contribute to the first and second discriminant (for SIDA and SIDANet) or canonical correlation (for SELPCCA) vectors.

Variables farther from the origin and close to the first or second axis have higher impact on the first or second discriminant/canonical vectors, respectively.

Variables farther from the origin and between both first and second axes have similar higher contributions to the first and second discriminant/canonical correlation vectors.

In both situations, for SIDA and SIDANet, this suggests that these variables contribute more to the separation of classes and association of views.

For SELPCCA, this suggests that these variables contribute more to the association between the two views. This plot can only be generated for classification and association problems with 3 or more classes (SIDA and SIDANet), or for CCA problems with two or more canonical correlation vectors requested (i.e. ncancorr > 1 for SELPCCA).

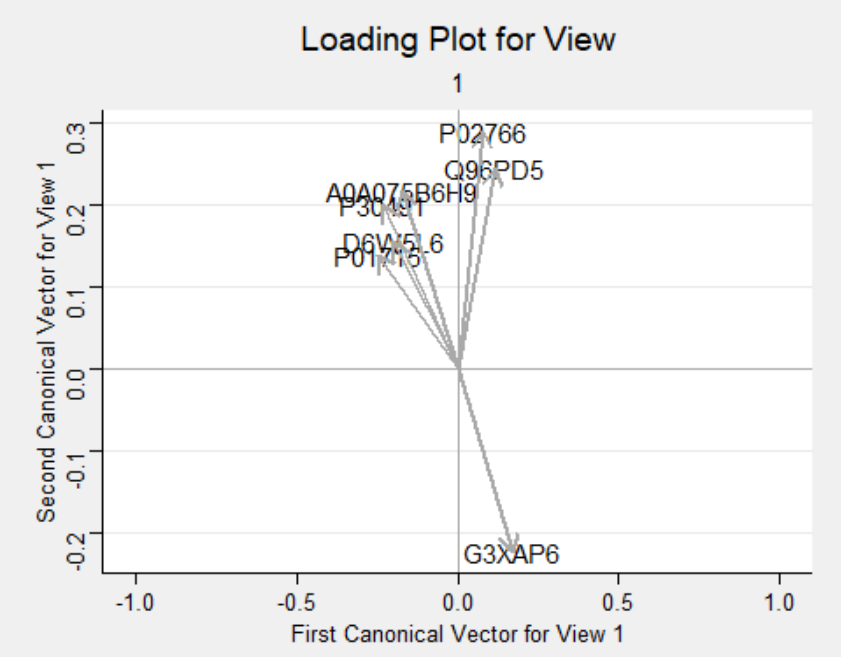
The angle between two vectors also give an indication of how the two variables are correlated. In particular, vectors that are close to each other suggests that the variables have high positive correlation. Vectors that are about 90 degrees indicate that the two

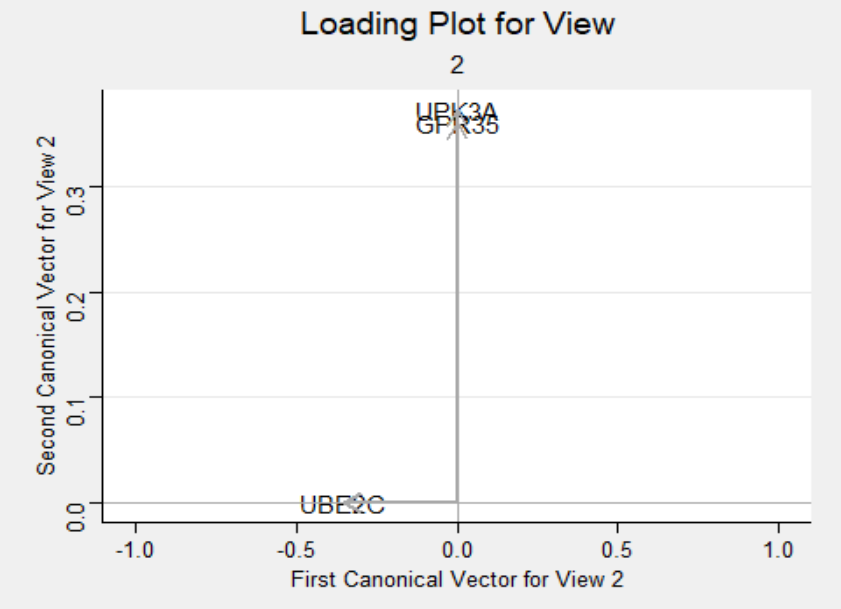
variables are uncorrelated. Vectors that have an angle close to 180 degrees indicate that the two variables have negative correlation.

In order not to clutter the graph, we provide the option *keep.loadings* for the number of variables with highest absolute weights to show on the graph for each view.

Here's the loadings plot for results from SELPCCA In this example, we show the top 7 variables with largest absolute weights on the plot for view 1 and top 3 for view 2

```
LoadingsPlots(mycvselpcca,keep.loadings = c(7,3))
```



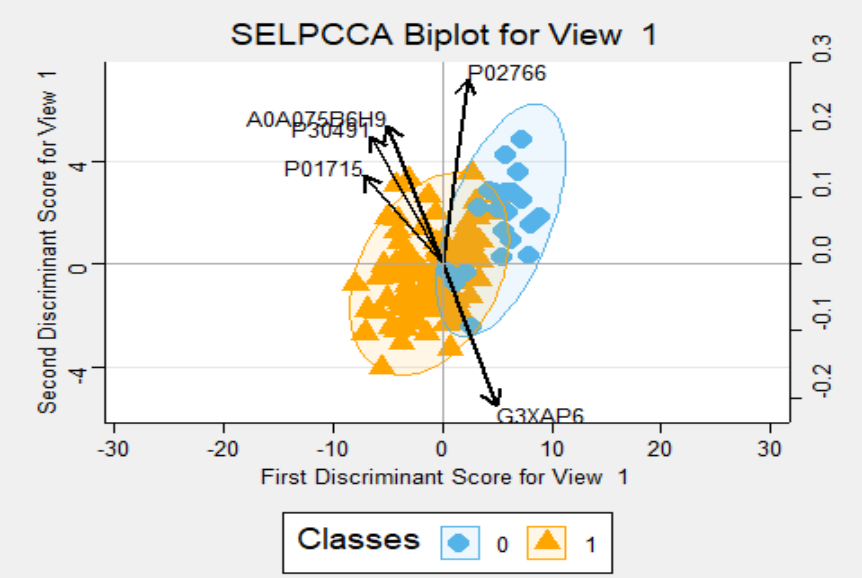


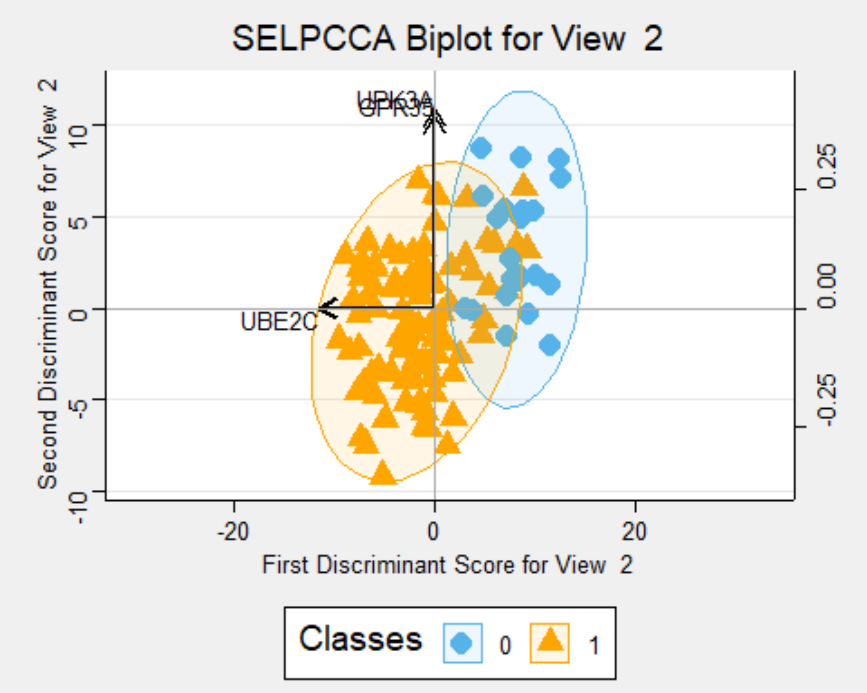
Biplots

Biplots are useful for representing both loadings plot and discriminant scores/canonical correlation variates.

We provide the function *WithinViewBiplot()* to visualize the scores and loadings for each view separately.

```
Y=Clinical$DiseaseStatus.Indicator
WithinViewBiplot(mycvselpcca,Y,keep.loadings = c(5,3))
```





Now, instead of visualizing biplots separately for each view, we provide the function *BetweenViewBiplot()* to graph scores and loadings for pairs of views.

In this graph, dashed red vectors represent loadings plot for the second view. And solid black vectors represent loadings plot for the first view.

The scores are the sum of scores for the two views.

Here's an example Biplot for the COVID-19 application, where we show the top 3 variables for view 1 and the top 3 for view 2.

```
Y=Clinical$DiseaseStatus.Indicator


BetweenViewBiplot(mycvselpcca,Y,keep.loadings = c(3,3))
```