feseR: Combining efficiently multiple feature selection methods in a R-workflow

Enrique Audain and Yasset Perez-Riverol

Octuber 2017

Introduction

feseR provides funcionalities to combine multiple Feature Selection (FS) methods to analyze high-dimensional omics data in R environment. The different feature selection steps can be classificated in: Univariate (Correlation filter and Gain Information), Multivariate (Principal Component Analysis and Matrix Correlation based) and Recursive Feature Elimination (wrapped up with a Machine Learning algorithm). The goal is to assemble the different steps in an efficient workflow to perform feature selection task in the context of classification and regression problems. The package includes also several example dataset.

Available dataset

We provide some example dataset (Transcriptomics and Proteomics) with the package. Some general description of the data are listed bellow:

- TNBC (Label-free deep proteome analysis of 44 (samples and technical replicates) human breast specimens)
- GSE5325 (Analysis of breast cancer tumor samples using 2-color cDNA microarrays)
- GSE48760 (Transcriptomics analysis of left ventricles of mouse subjected to an isoproterenol challenge)

Note: Datasets are expected to be a matrix with features in columns and samples in rows.

Examples

Preparing your data

```
library(feseR)

 # loading example data (TNBC)
 data(TNBC)

 # getting features
 features <- TNBC[,-ncol(TNBC)]

 # getting class variable (expected last column)
 class <- TNBC[,ncol(TNBC)]

 # pre-filtering
 # keep only features (cols) with maximal missing rate 0.25 across samples (rows)
 features <- filterMissingnessRate(features, max_missing_rate = 0.25)
```

```
## Found 6628 features with max missing rate 0.25
```

```
## Kept 6628 features out of 15524. Percent of removed features: 57.3%
```

```
  # impute missing values
  features <- imputeMatrix(features, method = "mean")
```

```
## Imputing matrix with method: mean
```

```
  # Scale data features. These transformations coerce the original predictors
  # to have zero mean and standard deviation equal one.
  features <- scale(features, center=TRUE, scale=TRUE)
```

Univariate filter examples

```
  # filtering by correlation
  output <- filter.corr(features = features, class = class, mincorr = 0.3)
```

```
## Kept 3059 features out of 6628. Number of removed features (Univariate correlation filter): 53.85%
```

```
  # filtering by gain information
  output <- filter.gain.inf(features = features, class = class, zero.gain.out = TRUE)
```

```
## Kept 950 features out of 6628. Number of removed features (Gain Information filter): 85.67%
```

Multivariate filter examples

```
  # filtering by matrix correlation (cutoff 0.75)
  output <- filter.matrix.corr(features = features, maxcorr = 0.75)
```

```
## Kept 2754 features out of 6628. Number of removed features (Matrix correlation filter) 58.45%
```

```
  # data dimension reduction using PCA (return only PCs explaining 95% of the variance)
  output <- filter.pca(features = features, cum.var.cutoff = .95)
```

```
## Features reduced to: 33 Principal components
```

Combining Feature Selection methods

This function allows to combine multiple feature selection methods in a workflow

```
  # combining filter univariate corr., multivariate matrix corr. and
  # recursive feature elimination wrapped with random forest
   results <- combineFS(features = features, class = class,
                        univariate = 'corr', mincorr = 0.3,
                        multivariate = 'mcorr', maxcorr = 0.75,
                        wrapper = 'rfe.rf', number.cv = 10,
                        group.sizes = seq(1,100,10),
                        verbose = F, extfolds = 10)


   # getting the metrics from the training process
   training_results <- results$training

   # getting the metrics from the testing process
   testing_results <- results$testing
```

Results from the training phase

Table 1: Best model metrics from 10-folds cross-validation resampling.

Variables	Accuracy	Kappa	AccuracySD	KappaSD
1	0.4333	0.2694	0.2144	0.2286
11	0.5333	0.3714	0.267	0.3569
21	0.6417	0.5266	0.2915	0.3745
31	0.7	0.5833	0.2123	0.3068
41	0.65	0.531	0.1916	0.25
51	0.6833	0.5667	0.1561	0.2108
61	0.8167	0.7167	0.2108	0.3518
71	0.7083	0.5738	0.2196	0.3435
81	0.7917	0.7167	0.2013	0.2727
91	0.7667	0.6833	0.3235	0.4335
930	0.7417	0.6597	0.2058	0.274

Results from the testing phase

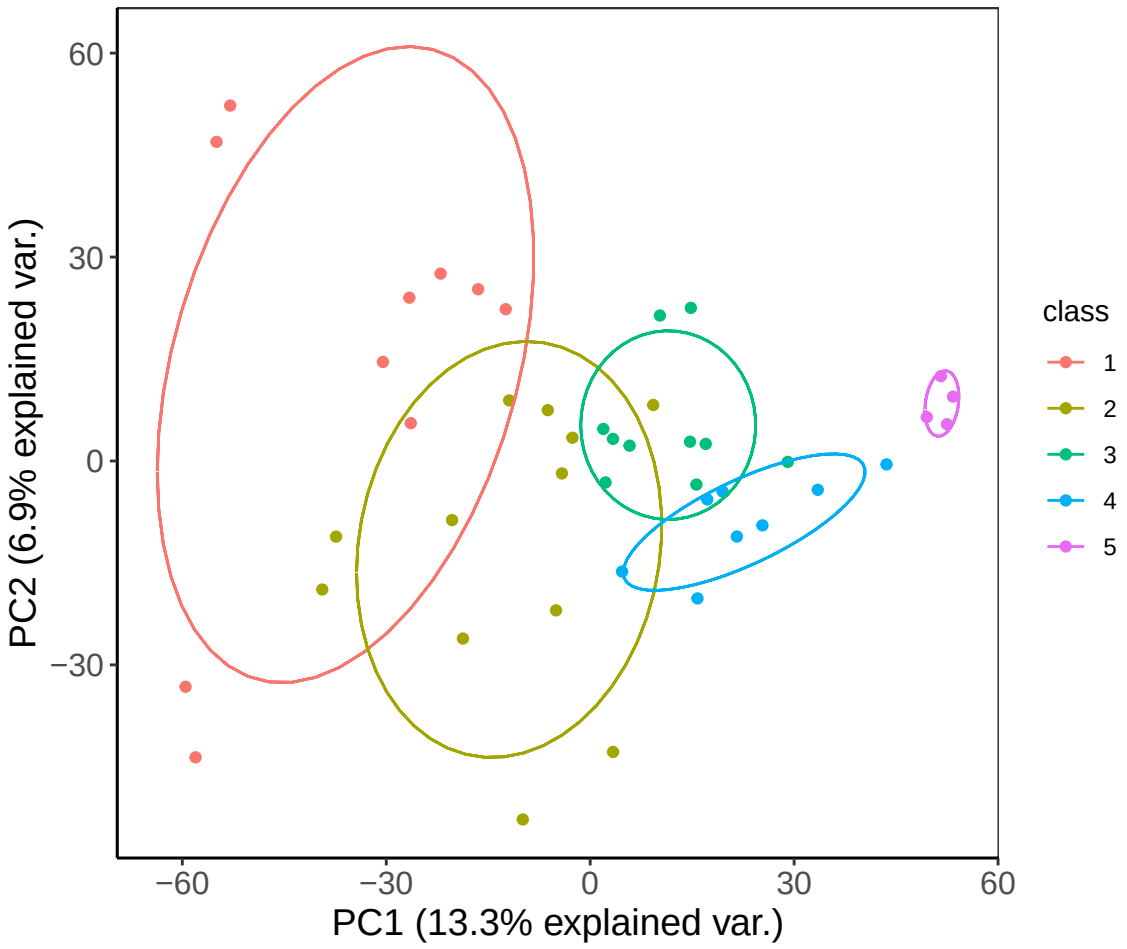
Table 2: Classification metrics from ten class-balanced and randomized runs.

Run	Variables	Accuracy	Kappa	AccuracyPValue
1	888	1	1	2.216e-07
2	51	0.8462	0.803	9.42e-05
3	91	0.7692	0.6929	0.000816
4	61	0.8462	0.803	9.42e-05
5	61	1	1	2.216e-07
6	839	0.9231	0.9008	6.703e-06
7	886	1	1	2.216e-07
8	853	1	1	2.216e-07
9	91	0.8462	0.806	9.42e-05
10	41	0.9231	0.9008	6.703e-06

Visualizing the Feature Selection process

- Groups distribution on the first two Principal Components (PC1 and PC2) from the original data (without apply any Feature Selection method).

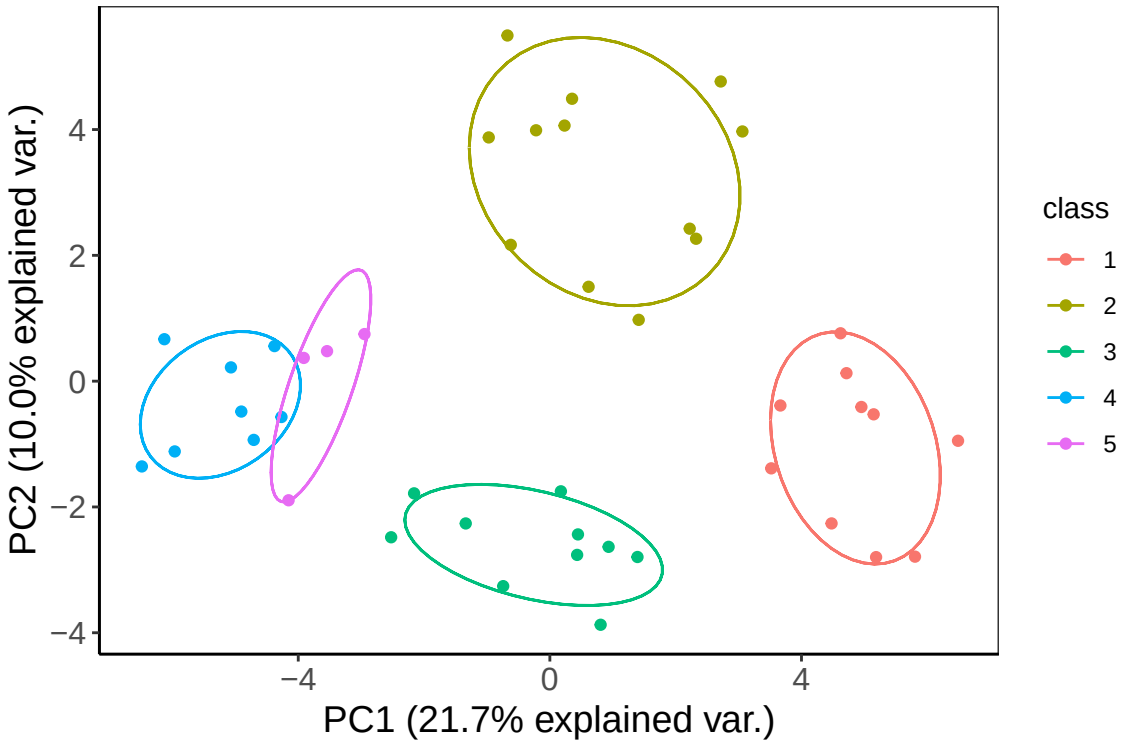
```
# plot PCA (PC1 vs. PC2)
plot_pca(features = features, class = class, list.plot = FALSE)
```



- Groups distribution on the first two Principal Components (PC1 and PC2) after to apply the Feature Selection workflow.

```
# getting the filtered matrix
filtered.features <- features[,results$opt.variables]

# plot PCA (PC1 vs. PC2)
plot_pca(features = filtered.features, class = class, list.plot = FALSE)
```



- Plotting the correlation matrix of final features.

```
# plot correlation matrix
plot_corr(features = filtered.features, corr.method = 'pearson')
```