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 a 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)

Examples

Preparing your data

```r
library(feseR)

 # loading example data (TNBC)
 data(TNBC)

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

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

 # getting only those features (i.e. genes/proteins)
 # with expression values for all instances (samples)
 features <- features[ , colSums(is.na(features)) == 0]

 # 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 filters

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

```
## Number of removed features (Univariate correlation filter): 55.16 %
```

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

```
## Number of removed features (Gain Information filter): 17.45 %
```

Multivariate filters

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

```
## Number of removed features (Matrix correlation filter) 43.44 %
```

```r
# reducing data using PCA (return Principal Components holding 95% of variance)
output <- filter.pca(features = features, cum.var.cutoff = .95)
```

```
## Features reduced to: 31 Principal components
```

Combining Feature Selection methods

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

```r
# 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(1,100,5),
                      extfolds = 20)


 # 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	0.5133	0.2865	0.2644	0.2324
2	0.62	0.4241	0.3353	0.4355
3	0.7817	0.6554	0.2681	0.3999
4	0.8317	0.7579	0.2561	0.348
5	0.7567	0.7155	0.3281	0.3545
6	0.7583	0.6548	0.283	0.372
7	0.7033	0.6476	0.3183	0.3448
8	0.8517	0.7487	0.1775	0.3262
9	0.7483	0.706	0.3076	0.3208
10	0.785	0.6534	0.2323	0.3663
15	0.7817	0.7417	0.3157	0.3344
20	0.7867	0.75	0.3425	0.3727
25	0.9267	0.8889	0.1235	0.1816
30	0.7933	0.75	0.3118	0.3333
35	0.7933	0.75	0.3118	0.3333
40	0.8133	0.775	0.3186	0.3426
45	0.8133	0.775	0.3186	0.3426
50	0.8467	0.825	0.319	0.3344
55	0.8467	0.825	0.319	0.3344
60	0.8467	0.825	0.319	0.3344
65	0.8467	0.825	0.319	0.3344
70	0.8467	0.825	0.319	0.3344
75	0.8467	0.825	0.319	0.3344
80	0.8133	0.775	0.3186	0.3426
85	0.8467	0.825	0.319	0.3344
90	0.7967	0.7583	0.3313	0.361
95	0.8467	0.825	0.319	0.3344
100	0.8467	0.825	0.319	0.3344
757	0.8167	0.7833	0.3375	0.3689

Results from the testing phase

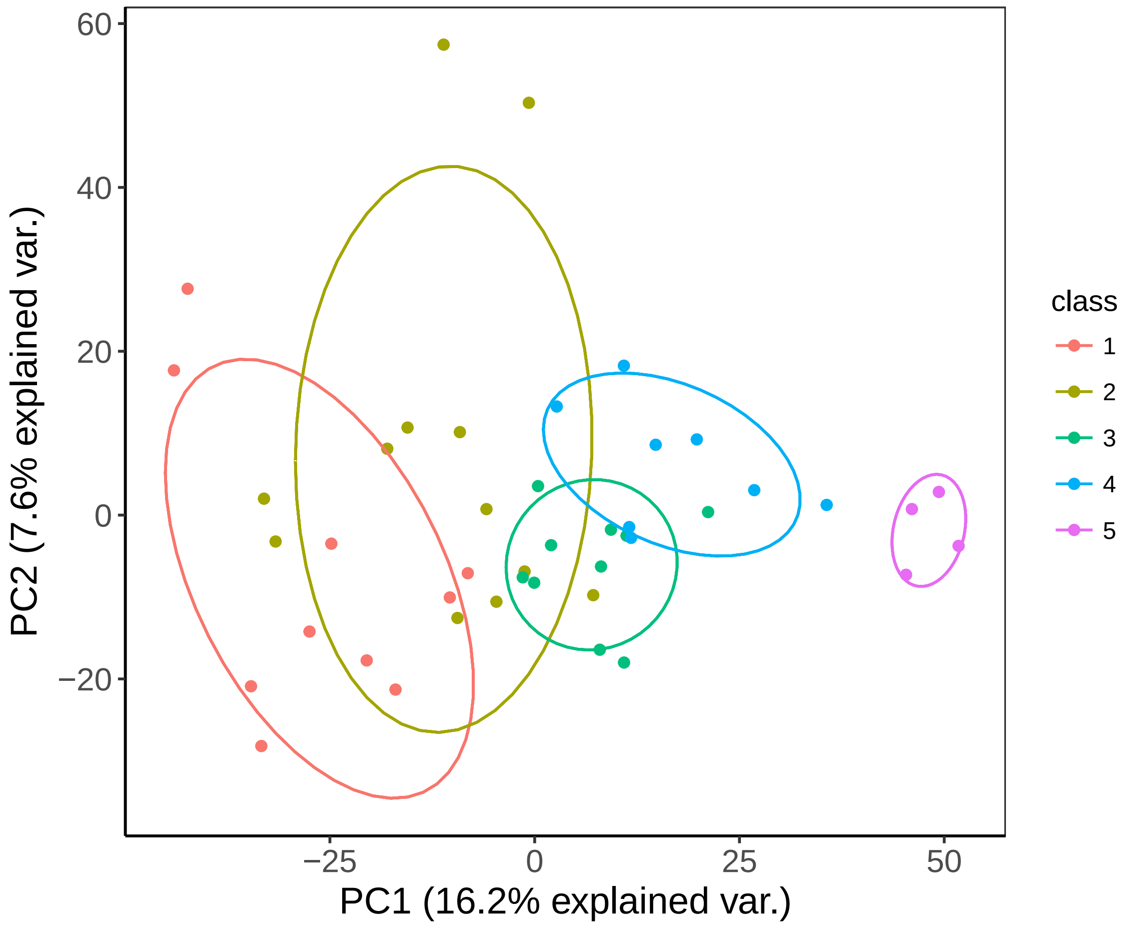
Table 2: Classification metrics from twenty class-balanced and randomized runs.

Run	Variables	Accuracy	Kappa	AccuracyPValue
1	100	1	1	2.216e-07
2	80	1	1	2.216e-07
3	35	0.6923	0.6119	0.004876
4	75	0.9231	0.9008	6.703e-06
5	60	0.8462	0.8045	9.42e-05
6	85	1	1	2.216e-07
7	80	0.8462	0.803	9.42e-05
8	95	1	1	2.216e-07
9	757	0.9231	0.8984	6.703e-06
10	100	1	1	2.216e-07
11	95	0.6923	0.6148	0.004876
12	100	0.6154	0.5038	0.02132
13	85	0.9231	0.9015	6.703e-06
14	90	0.9231	0.8992	6.703e-06
15	757	1	1	2.216e-07
16	25	1	1	2.216e-07
17	70	1	1	2.216e-07
18	70	0.7692	0.7	0.000816
19	50	1	1	2.216e-07
20	95	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).

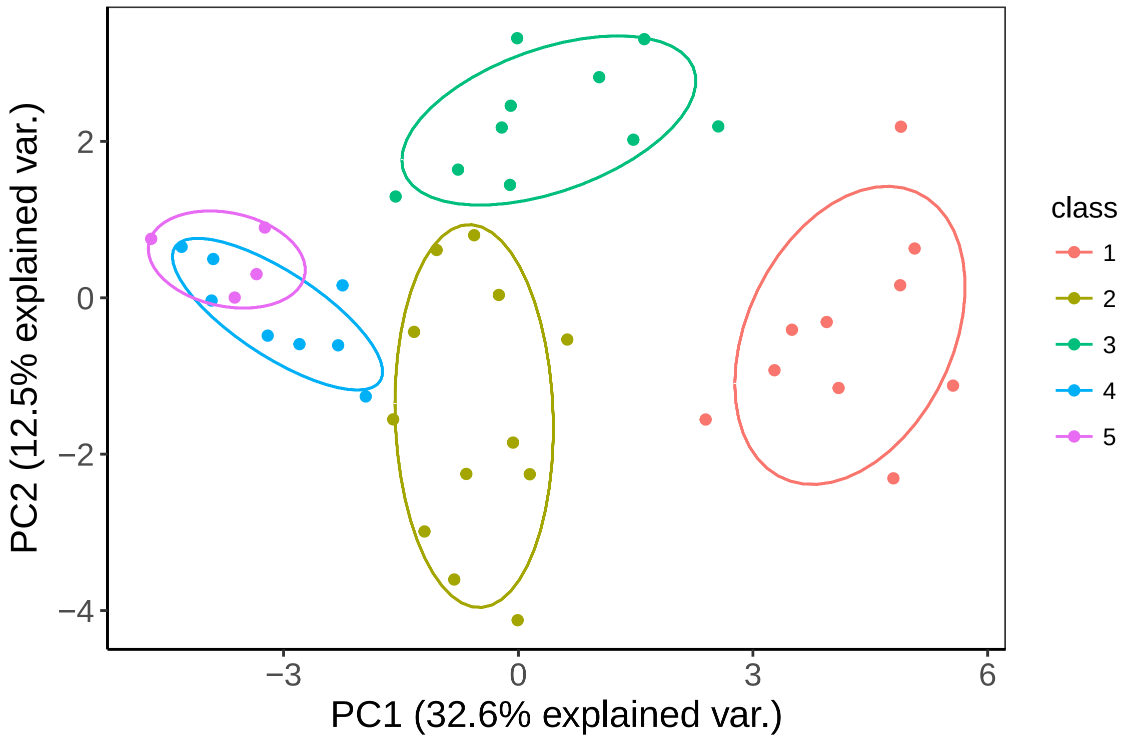
```
# plot PCA (PC1 vs. PC2)
plot_pca(features = features, class = class, list.plot = FALSE)
```



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

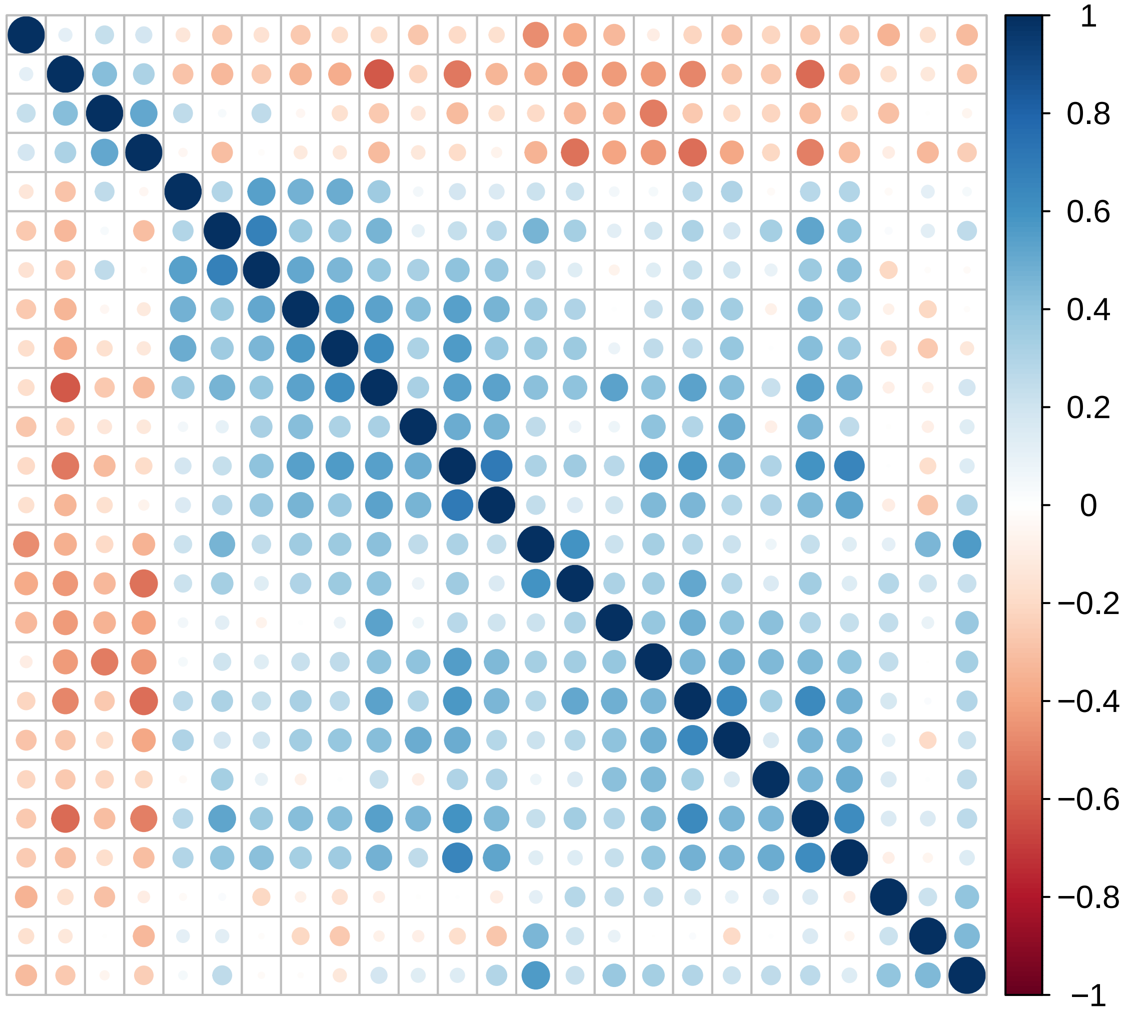
```
# getting the filtered matrix
filtered.features <- features[,results$opt.variables]

# 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')
```


1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1