A Short Introduction to the OmicsMarkeR Package

Charles Determan Jr.
cdetermanjr@hotmail.com

November 4, 2013

1 Introduction

The OmicsMarkeR package contains functions to streamline the analysis of 'omics' level datasets with the objective to classify groups and determine the most important features. OmicsMarkeR loads packages as needed and assumes that they are installed. I will provide a short tutorial using the both synthetic datasets created by internal functions as well as the 'Sonar' dataset. Install OmicMarkeR using

```
> # currently in development
> # install.packages("OmicsMarkeR", dependencies = c("Imports"))
```

to ensure that all the needed packages are installed.

You may also install the developmental version using

```
> library(devtools)
> install_github("OmicsMarkeR", username = "cdeterman")
```

2 Example

OmicsMarkeR has a few simplified functions that attempt to streamline the classification and feature selection process including the addition of stability metrics. We will first generate a synthetic dataset. This includes three functions that can be used to create multivariate datasets that can mimic specific omics examples. This can include a null dataset via `nvar = 50` and `nsamp = 100`.

The `create.corr.matrix` function induces correlations to the datasets. The `create.discr.matrix` function induces variables to be discriminate between groups. The number of groups can be specified with `num.groups`.

```
> library("OmicsMarkeR")
> set.seed(123)
> dat.discr <- create.discr.matrix(
+   create.corr.matrix(
+     create.random.matrix(nvar = 50,
+                          nsamp = 100,
+                          st.dev = 1,
+                          perturb = 0.2)),
+   D = 10
+   )$discr.mat
```

To avoid confusion in the coding, one can isolate the variables and classes from the newly created synthetic dataset. These two objects are then used in the `fs.stability` function. I can then choose which algorithm(s) to apply e.g. `method = c("plsda", "rf")`, the number of top important features `f = 20`, the number of bootstrap reptititions for stability analysis `k = 3`, the number of k-fold cross-validations `k.folds = 10` and if I would like to see the progress output `verbose = TRUE`.

```
> vars <- dat.discr[,1:(ncol(dat.discr)-1)]
> groups <- dat.discr[,ncol(dat.discr)]
> results <- fs.stability(vars,
+                         groups,
+                         method = c("plsda", "rf"),
+                         f = 10,
+                         k = 3,
+                         k.folds = 10,
+                         verbose = FALSE)

[1] "plsda complete"
[1] "rf complete"

Aggregating results
Selecting tuning parameters
[1] "plsda complete"

Aggregating results
Selecting tuning parameters
[1] "rf complete"

Aggregating results
Selecting tuning parameters
Calculating Model Performance Statistics
```

If I would like to see the performance metrics, I can simply use the `performance.metrics` function. This will provide a concise data.frame of confusion matrix and ROC statistics. Additionally, the Robustness-Performance Trade-off value is provided with the results `RPT`.

```
> performance.metrics(results)

Model Performance Statistics

                       plsda         rf
Accuracy           0.88333333 0.96666667
Kappa              0.76666667 0.93333333
ROC.AUC            0.92000000 1.00000000
Sensitivity        0.93333333 1.00000000
Specificity        0.83333333 0.93333333
Pos Pred Value     0.84747475 0.94444444
Neg Pred Value     0.93333333 1.00000000
Accuracy SD        0.07582875 0.07582875
Kappa SD           0.15165751 0.15165751
ROC.AUC SD         0.06480741 0.06480741
Sensitivity SD     0.08164966 0.08164966
Specificity SD     0.09831921 0.09831921
Pos Pred Value SD 0.08817435 0.08817435
Neg Pred Value SD 0.08164966 0.08164966

> results$RPT

    plsda        rf
0.7933472 0.3715877
```