OncoSig-RF Overview

Written by: Joshua Broyde (2/23/2018)

OncoSig-RF is an algorithm for determinging novel sets of proteins that support the activity of an oncogene or tumor suppressor (i.e. Oncoprotein-Centric Map, or OC-map). Given a gold standard and an input molecular interaction network, the algorithm uses a random forest classifier to discover novel members of the OC-Map.

OncoSig-RF Input and Options

OncoSig-RF Requires 2 input files, 1.a network of interactions and 2.A gold standard of pathway members (i.e. members of the OC-Map) to train on.

The network is a tab delimited dataframe of 3 columns, Gene A, Gene B and the strength of the interaction. This can range from 0 to infinite, but should not be a negative number. Note that the dataframe must be redundant, so that the interaction is represented twice, between A and B and B and A.

The other required inputs are as follows: The maximum number of iterations for monte-carlo cross validation. (Default=50) The fraction of the gold standard to train on at eact iteration. (Default=.5) The number of trees to to create in each forest in the random forest classifier. (Default=50) The number of proteins in the negative standard for each training in the random forest (default=same number as picked for the gold standard). This is the balance option. (Default=1) Note the the "balance" option picks the same *number* of proteins, not percentage, so if there are 200 proteins in the gold standard, by default 50% are picked then at each.

To run OncoSig-RF is from the bash command line:

```
source location/to/OncoSig-RF/runOncoSig-RFscript_wrapper.sh
    location/input_network.txt
```

OncoSig-RF Example

An example script for running OncoSig-RF is included in the "runOncoSig-RFscript.R" script. We will now run OncoSig-RF step-by step from that script to discover novel members of Kras-regulated pathways. We will run OncoSig-RF using `test_network.txt` in the `test` directory.

In this example `test_network.txt` is the network file, `gold_standard.txt` is the gold standard. There are 50 iterations, .5 of the gold standard is used for training for each iteration, there are 75 trees in each iteration, and the balance equals 1, which means that the same number of negative proteins are taken at each iteration for training. If balance were equal to 2, twice as many negative proteins would be samples at each iteration.

OncoSig-RF uses the randomForest and MASS package as well as internal functions:

```
library(randomForest)
library(MASS)
library(Matrix)
library('getopt')
#Change this depending on where the functions are located
source("location/of/OncoSig-RF/functions.R")
```

If you do not have randomForest of MASS, then first install them:

```
install.packages("randomForest")
install.packages("MASS")
```

Get the location the network and Gold Standard:

```
Network_location="Test/test_network.txt"
Gold_Standard_location="Test/gold_standard.txt"
```

Read in the network and Gold Standard and the other paramters.The network must be tab delimited. The first two columns are the names, and third column is the strength of the interaction:

```
arg <- commandArgs(trailingOnly = TRUE)
args=as.vector(arg);
Network_location=args[[1]]
message("Network location: ", Network_location, sep="" )
Gold_Standard_location=args[[2]]
message("Gold_Standard_location: ",Gold_Standard_location,sep="")
max_iterations=args[[3]]
max_iterations=as.numeric(max_iterations)
Fraction_Gold_sample=args[[4]]
Fraction_Gold_sample=as.numeric(Fraction_Gold_sample)
ntrees=args[[5]]
ntrees=as.numeric(ntrees)
balance=as.numeric(args[7])
```

The Network matrix looks like this. Note that it is symmetric:

```
Q13131_PREPPI   P14625  1.111887e+03
P14625_PREPPI   Q13131  1.111887e+03
P37058_PREPPI   P15428  1.502400e+03
P15428_PREPPI   P37058  1.502400e+03
Q8IY84_PREPPI   Q9Y3S1  7.255526e+02
```

```
Q9Y3S1_PREPPI   Q8IY84  7.255526e+02
Q13315_PREPPI   Q96T68  2.535267e+03
Q96T68_PREPPI   Q13315  2.535267e+03
P27348_PREPPI   O75385  1.084084e+04
O75385_PREPPI   P27348  1.084084e+04
```

In this particular example, only PREPPI protein-protein interactions are represented. However, other interaction types may be included as well.

Next, we will convert the network list (e.g. an adjacency list) to an adjacency matrix. Note that this may take a few minutes if the network is very large.

```
Network=read.delim(Network_location,header=F)
Network$V1=as.character(Network$V1)
Network$V2=as.character(Network$V2)
Network$V3=as.numeric(Network$V3)
Network=as.matrix(Network)
Gold_Standard=read.delim(Gold_Standard_location,header=F)
Gold_Standard$V1=as.character(Gold_Standard$V1)
```

This converts the network to a matrix. Inputes non-interactions as 0. Network[,3]=as.numeric(Network[,3]) Network_matrix=listToMatrix(Network)

The Network_matrix looks like this. Zero indicates no edge between nodes:

```
   Q13131_PREPPI P14625_PREPPI P37058_PREPPI P15428_PREPPI
P14625      1111.887     25258.640         0.000           0.0
Q13131      8911.691      1111.887         0.000           0.0
P15428         0.000         0.000      1502.400           0.0
P37058         0.000         0.000      2079.157        1502.4
```

Convert Matrix to Dataframe for future steps Network_matrix_df=as.data.frame(Network_matrix)

Remove members of the gold standard that are not present in the network.

```
Gold_Standard_in_Network_names=intersect(rownames(Network_matrix_df),Gold_Standard$V1)
```

Retrieve the negative set (i.e. all proteins not in the gold standard): Negative_Set_names=setdiff(rownames(Network_matrix_df),Gold_Standard_in_Network_names)

Next, run the random forest classifier. The Random forest classifier will train on a portion of the gold standard and a sample of negative standard of the same size. A variant of repeated random sub-sampling validation is used to train the classifier. To do this, a fraction of the gold standard is randomly sampled from the data, and a random sample of the negative set of the same size is also sampled. A random forest is created with a number of trees. To generate new predictions, the score of each protein is predicted only with the random forests that were not used to train it. In the example script, the Kras gold standard has 250 members, so each random forest will be trained on 250 (125 + 125) proteins total.

If the set that you are using has a very small number of proteins in in (e.g. 3-30),
I recommend using a larger fraction of the gold standard and more iterations.

```
Query_output_results=runOncoSig-RF(Network_matrix_df,Gold_Standard_in_Network_names,max_iter
    = max_iterations, Fraction_Gold_sample
    =Fraction_Gold_sample,ntrees = ntrees, balance = balance)
Query_output_results_scores=Query_output_results[[1]]
write.table(Query_output_results_scores,file="OncoSig-RF_results.txt",row.names
    = TRUE, col.names=FALSE,quote = FALSE,sep="\t")
save(Query_output_results,Query_output_results_scores,Gold_Standard_location,Network_locatio
    file="OncoSig-RF_objects.R")
```

Now evaluate performance using a ROC Curve:

```
library(ROCR)
Query_output_results_scores=Query_output_results[[1]]
#See how good the performance is:
Query_output_results_scores$label=0
Query_output_results_scores[Gold_Standard_in_Network_names,2]=1
pred=prediction(Query_output_results_scores$Score,Query_output_results_scores$label)
pdf("Performance.pdf",height=5,width=5)
perf=performance(pred,measure = "tpr", x.measure = "fpr")
plot(perf,col='red') #Plot the ROC curve
abline(a=0,b=1);
```