Hypoxia Classifier Tutorial

Laura Puente-Santamaria

Step 1: Load decision trees and gene expression matrix

```
load("Tree_collection.RData")
load("GeneExpression_example.RData")
head(gene_expression)
##
              [,1]
             1.000
## A1BG
## A1BG-AS1 5.635
             2.000
## A1CF
## A2M
            45.972
## A2M-AS1
             5.000
## A2ML1
             2.243
```

This gene expression matrix is the salmon output corresponding to GSM2390150, an RNA-seq of HUVEC cells grown in hypoxia for 8h.

Step 2: Ranking percentile.

```
rank_percentile <- matrix( 100*rank( gene_expression ) / length( gene_expression ) )
rownames( rank_percentile ) <- rownames( gene_expression )
colnames( rank_percentile ) <- "Sample 1"
head(rank_percentile)</pre>
```

```
## A1BG 39.94070
## A1BG-AS1 50.75651
## A1CF 44.06828
## A2M 65.71069
## A2M-AS1 50.03055
## A2ML1 45.44474
```

We rank genes from the least to the most expressed, with 100 being the most expressed gene in the sample, and 0 the least. The trees generated with rpart take as input a data frame with rows for samples and columns for variables:

```
rank_percentile <- data.frame( t( rank_percentile ) )
rank_percentile[,1:5]</pre>
```

```
## A1BG A1BG.AS1 A1CF A2M A2M.AS1 ## Sample 1 39.9407 50.75651 44.06828 65.71069 50.03055
```

Step 3: Classify the sample.

Lets use tree #125 as our classifier:

```
decisionTree <- fullTreeCollection[[ 125 ]]
prediction <- predict( decisionTree, rank_percentile )
prediction</pre>
```

```
## N H
## Sample 1 0.07352941 0.9264706
```

The resulting matrix, **prediction**, has two columns with the probabilities of the sample to be normoxic (first column) or hypoxic (second column).

Handling missing data

Decision trees generated with rpart require that new data to classify includes all variables used to build the model, in our case, the following 20 genes: ADM, ANKRD37, BHLHE40, BNIP3, BNIP3L, C4orf3, C8orf58, CCNG2, EGLN1, GPR146, KDM3A, MAFF, MIR210HG, NARF, NDRG1, P4HA1, PFKFB3, SLC2A1, TCAF2, and ZNF395.

Occassionally, one or more of these genes will not be present in the expression data for several reasons: a gene might not be included in the reference transcriptome, or those genes not detected in the sample were filtered out. As an example, using a table that is missing data on "ZNF395", the function predict will throw an error:

We can solve this issue adding an extra column with NA values to our gene expression table:

```
rank_missing$ZNF395 <- as.numeric( NA )
predict( decisionTree, rank_missing )</pre>
```

```
## N H
## Sample 1 0.07352941 0.9264706
```