R-package DFNET

Bastian Pfeifer

8/23/2021

Input data format

To perform Network Module Detection a network and node feature matrices are required. Here, we use a PPI-Network and multi-omics node features from gene expression and DNA methylation data. In addition to that, a binary vector needs to be specified reflecting the outcome class. Here, it includes the patient group survived (class 0) vs non-survived (class 1).

```
PPI      <- read.table("~/LinkedOmics/KIRC/KIDNEY_PPI.txt")
mRNA     <- read.table("~/LinkedOmics/KIRC/KIDNEY_mRNA_FEATURES.txt")
Methy    <- read.table("~/LinkedOmics/KIRC/KIDNEY_Methy_FEATURES.txt")
TARGET   <- read.table("~/LinkedOmics/KIRC/KIDNEY_SURVIVAL.txt")
```

Dimensions are:

```
dim(PPI)
```

```
## [1] 6926452       3
```

```
dim(mRNA)
```

```
## [1]   306 12029
```

```
dim(Methy)
```

```
## [1]   306 12029
```

```
dim(TARGET)
```

```
## [1]   1 306
```

The PPI Network has 6926452 edges and is organized as follows:

```
head(PPI,5)
```

```
##   protein1 protein2 combined_score
## 1     ARF5    CALM2            490
## 3     ARF5     ERN1            159
## 4     ARF5   CDKN2A            606
## 5     ARF5     P4HB            167
## 6     ARF5    STX10            267
```

The first two columns refer to the connected nodes. The last column indicate the confidence of the interaction between these nodes/protein.

The node feature matrices are organized as follows:

```
mRNA[1:5,1:5]
```

```
##                   RBL2   VDAC3   ACTN1 ATP2A1  SFRP1
## TCGA.3Z.A93Z 10.1967 10.8407 11.0698 3.0921 8.4911
## TCGA.6D.AA2E 10.4898 11.2592 11.4613 3.4214 5.9663
## TCGA.A3.3357 10.8225 11.4032 11.5370 3.0013 4.6062
## TCGA.A3.3358 11.6874 10.9420 12.8086 5.4678 5.1437
## TCGA.A3.3367 11.3013 11.0082 11.8861 4.9567 6.2678
```

where the rows are the patients and the columns are the genes/nodes. In this case, we analyze 12029 node feature values from 306 patients.

The second omic data set has the same structure.

```
Methy[1:5,1:5]
```

```
##                  RBL2   VDAC3   ACTN1  ATP2A1   SFRP1
## TCGA.3Z.A93Z -0.4897 -0.4686 -0.0063 -0.4851 -0.4156
## TCGA.6D.AA2E -0.4885 -0.4574  0.1481 -0.4799 -0.3343
## TCGA.A3.3357 -0.4854 -0.4721 -0.0540 -0.4859 -0.3052
## TCGA.A3.3358 -0.4838 -0.4339 -0.1005 -0.4778 -0.3213
## TCGA.A3.3367 -0.4890 -0.4684 -0.0493 -0.4830 -0.2719
```

Note, the rows of the multi-modal feature matrices should refer to the exact same patient.

Finally, we need the target vector specifying the survival (0) and non-survival (1) groups.

```
TARGET[1:5]
```

```
##    TCGA.3Z.A93Z TCGA.6D.AA2E TCGA.A3.3357 TCGA.A3.3358 TCGA.A3.3367
## 13            0            0            0            0            0
```

For an illustrative example on how to use DFNET we will reduce data dimension.

```
mRNA_reduced  <- mRNA[,1:200]
Methy_reduced <- Methy[,1:200]
```

Creating a DFNET graph object

The DFNET package requires the following packages

```
library(DFNET)
require(igraph)
require(ranger)
require(pROC)
```

Lets create a DFNET_graph object and keep the edges with confidence values greater the 0.95-quantile

```
DFNET_graph  <- DFNET_generate_graph_Omics(PPI, list(mRNA_reduced, Methy_reduced), TARGET, 0.95)
```

```
summary(DFNET_graph)
```

```
##                Length Class  Mode
## graph          10     igraph list
## Feature_Matrix  2     -none- list
## gene.names     54     -none- character
```

The DFNET_graph object is a list and consists of three slots. The first slot is the PPI network internally converted to a igraph object.

```
DFNET_graph$graph
```

```
## IGRAPH b2473af U--- 53 90 --
## + edges from b2473af:
```

```
##  [1] 24--33  9--33 24--33 29--33 13--33  1--23 16--26  1--23  4-- 6 14--49
## [11] 25--49 38--49 47--49 12--19 19--50 45--53 45--48  9--33 31--44 32--40
## [21] 36--40 22--52 21--52 46--52 36--40  7--20 25--49 25--38 25--47 25--42
## [31] 17--27 13--51 25--42 12--50 12--19  3--28 29--33 29--44  4-- 6  5--10
## [41] 16--26 47--49 25--47 38--47 32--40 13--18 13--33 13--51 14--49 39--43
## [51]  5--10 45--53 37--53 35--53  2--41  8--34 22--46 21--22 22--52  7--20
## [61] 39--43  3--28  3--44 45--48 31--44 29--44  3--44 21--22 21--46 21--52
## [71] 35--53 35--37 37--53 35--37 12--50 19--50 11--15  8--34 11--15 30--38
## [81] 38--49 25--38 38--47 17--27 22--46 21--46 46--52 30--38 13--18  2--41
```

The second slot is a list of feature matrices. In our case gene expression and methylation data from the same set of patients.

```
DFNET_graph$Feature_Matrix[[1]][1:5,1:5]
```

```
##                AN_1    AN_2   AN_3    AN_4   AN_5
## TCGA.3Z.A93Z 8.4911 11.7421 7.8134  9.0309 2.5894
## TCGA.6D.AA2E 5.9663 10.8787 7.7058 10.8445 2.2563
## TCGA.A3.3357 4.6062 11.2569 7.9381  9.1925 2.3231
## TCGA.A3.3358 5.1437 11.3485 7.7571  8.8595 6.3198
## TCGA.A3.3367 6.2678 11.4433 8.0568  8.9852 6.7388
```

```
DFNET_graph$Feature_Matrix[[2]][1:5,1:5]
```

```
##                 BN_1    BN_2   BN_3   BN_4   BN_5
## TCGA.3Z.A93Z -0.4156 -0.4896 0.3768 0.3365 0.3686
## TCGA.6D.AA2E -0.3343 -0.4882 0.4153 0.2964 0.3911
## TCGA.A3.3357 -0.3052 -0.4913 0.4520 0.4144 0.2489
## TCGA.A3.3358 -0.3213 -0.4869 0.4336 0.3341 0.0612
## TCGA.A3.3367 -0.2719 -0.4926 0.4449 0.3949 0.3904
```

The third slot contains the node/gene names.

```
head(DFNET_graph$gene.names,5)
```

```
## [1] "SFRP1"  "MAN1B1" "NPHP4"  "MRPS25" "MAEL"
```

A DFNET_graph object can thus be easily created, also without using the "DFNET_generate_graph_Omics" function. Note, the colnames of the feature matrices need to be as specified! A prefix letter followed with a "N" and than simply the node identifier. Node identifier should match the identifier used for the igraph network.

```
head(as_edgelist(DFNET_graph$graph, names = TRUE))
```

```
##      [,1] [,2]
## [1,]   24   33
## [2,]    9   33
## [3,]   24   33
## [4,]   29   33
## [5,]   13   33
## [6,]    1   23
```

As seen from the above table "AN_365" and "AN_3411" are connected. The same is true for the second modality "BN_365" and "BN_3411".

DFNET for Subnetwork Detection

The main function for network module detection expects the number of trees (ntrees), the number of greedy iteration (niter), and the initial size of the module as an input. The ntrees parameter specifies the number of random works initialized. The niter parameter sets the total number of greedy iterations, and the init.mtry defines the depth of the random work, and thus the initial size of the modules.

```
DFNET_object <- DFNET(DFNET_graph, ntrees=100, niter=10, init.mtry=10)
```

```
## 1  of  10  greedy steps
## 2  of  10  greedy steps
## 3  of  10  greedy steps
## 4  of  10  greedy steps
## 5  of  10  greedy steps
## 6  of  10  greedy steps
## 7  of  10  greedy steps
## 8  of  10  greedy steps
## 9  of  10  greedy steps
## 10  of  10  greedy steps
```

```
summary(DFNET_object)
```

```
##                    Length Class  Mode
## DFNET_graph           3   -none- list
## DFNET_trees        1100   -none- list
## DFNET_MODULES       100   -none- list
## DFNET_MODULES_AUC   100   -none- numeric
```

The DFNET_object contains four slots. The first slot "DFNET_graph" is the igraph object storing the network topology. The second slot "DFNET_trees" contains the Decision Trees of the Decision Forest. The third slot "DFNET_MODULES" stores the detected Network Modules, and the "DFNET_MODULES_AUC" consists of the corresponding AUC values of the modules. The accuracy of the Decision Forest Classifier can be calculated as

DFNET Edge Importance Scores

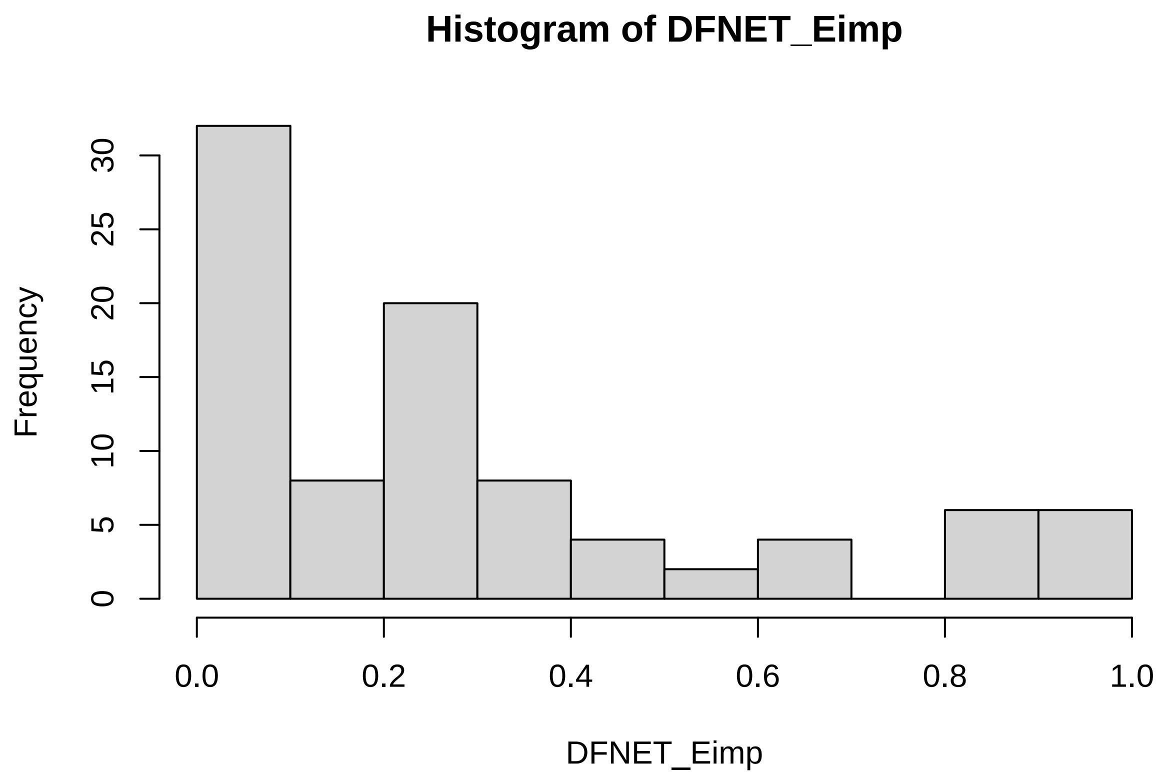
Edge Importance Scores can be calculated with the following function

```
DFNET_Eimp   <- DFNET_Edge_Importance(DFNET_graph, DFNET_object)
```

```
length(DFNET_Eimp)
```

```
## [1] 90
```

```
hist(DFNET_Eimp)
```



DFNET Detected Modules

The detected modules can be retrieved via the "DFNET_modules" function

```
DFNET_mod    <- DFNET_modules(DFNET_graph, DFNET_object, DFNET_Eimp)
```

```
head(DFNET_mod)
```

```
##                  Module EDGE_IMP       AUC      IMP
## 1006       25 38 47 49 2.712617 0.6304140 3.343031
## 1059    25 30 38 47 49 2.329420 0.6194231 2.948843
## 1003    14 25 38 47 49 2.313275 0.6232245 2.936499
## 1037    25 38 42 47 49 2.266540 0.6281941 2.894734
## 1039 14 25 30 38 47 49 2.060501 0.6520753 2.712576
## 1016 25 30 38 42 47 49 2.021556 0.6547443 2.676300
```

The modules are ranked by their importance (last column). Note, node ids are shown, but the actual node names can be retrieved from "DFNET_graph$gene.names". The rownames of the returned data matrix point to the tree identifier. We recall, the last ntree trees of the forest are the selected modules. Each of these modules is reflected by a decision tree. Lets access the decision tree which is associated with the top-ranked module.

```
topTree <- as.numeric(rownames(DFNET_mod))[1]
DFNET_object$DFNET_trees[[topTree]]
```

```
## Ranger result
##
## Call:
```

```
##  ranger(dependent.variable.name = "target", data = MM_DATA, split.select.weights = WEIGHTS/sum(WEIGH
##
## Type:                             Classification
## Number of trees:                  1
## Sample size:                      306
## Number of independent variables:  8
## Mtry:                             8
## Target node size:                 1
## Variable importance mode:         impurity
## Splitrule:                        gini
## OOB prediction error:             33.64 %
```

Lets have a closer look

```
Nodes        <- as.numeric(strsplit(DFNET_mod[1,1]," ")[[1]])
DFNET_graph$gene.names[Nodes]
```

```
## [1] "ARPC2" "RAB5C" "IGF2R" "SYT1"
```

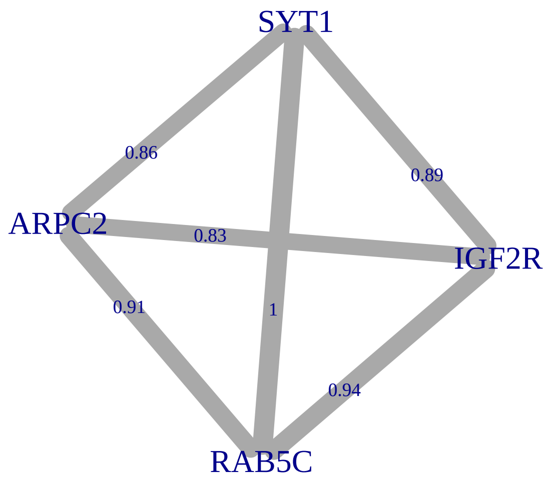
The module is reflected by the following edges and nodes

```
Top_Module <- DFNET_get_module(Nodes, DFNET_graph, DFNET_Eimp)
head(Top_Module)
```

```
##   Node1 Node2          EDGE_IMP
## 1 ARPC2 RAB5C 0.906769470301354
## 2 ARPC2 IGF2R 0.834077367820124
## 3 RAB5C IGF2R  0.93795044588996
## 4 ARPC2  SYT1 0.855806195985849
## 5 RAB5C  SYT1                 1
## 6 IGF2R  SYT1 0.890629757135708
```

We can visualize this subgraph using the function “DFNET_plot_module”.

```
require(igraph)
DFNET_plot_module(Nodes, DFNET_graph, DFNET_Eimp)
```



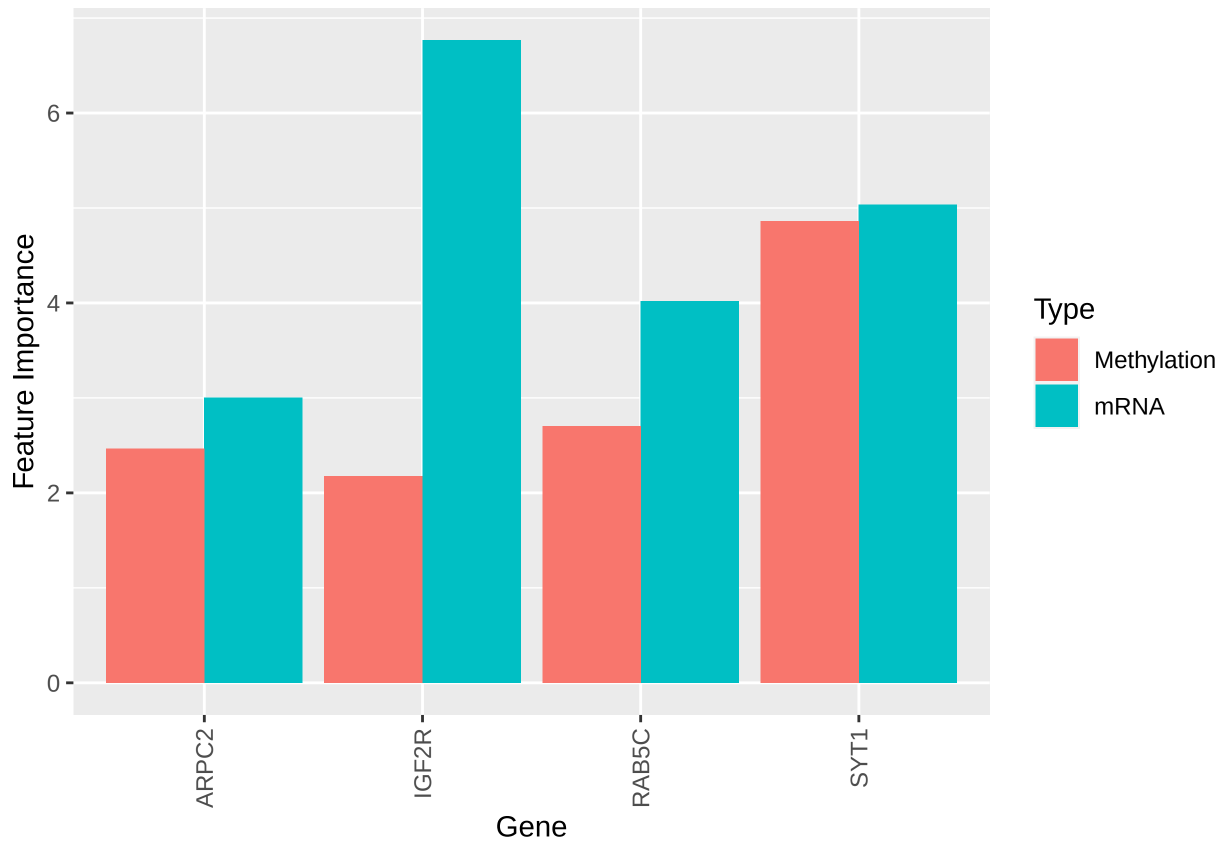
DFNET Node Feature importance scores

The feature importances of the nodes of that module can be calculated as

```
DFNET_Fimp   <- DFNET_calc_feature_importance(Nodes, DFNET_object, DFNET_graph)
DFNET_Fimp
```

```
##          ARPC2    RAB5C    IGF2R     SYT1
## omic1 3.002046 4.021766 6.766200 5.036018
## omic2 2.467372 2.701812 2.178533 4.861801
```

```
## GGPLOT
library(ggplot2)
library(reshape)

RES1 <- cbind(colnames(DFNET_Fimp),DFNET_Fimp[1,])
RES2 <- cbind(colnames(DFNET_Fimp),DFNET_Fimp[2,])
RES1 <- cbind(RES1,"mRNA")
RES2 <- cbind(RES2,"Methylation")

RES <- rbind(RES1,RES2)
rownames(RES) <- NULL
colnames(RES) <- c("Gene","IMP","Type")
RES     <- as.data.frame(RES)
RES$IMP <- as.numeric(RES$IMP)

p <- ggplot(RES, aes(fill=Type, y=IMP, x=Gene)) +
    geom_bar(position="dodge", stat="identity") +
    ylab("Feature Importance") +
    theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))

plot(p)
```



DFNET as a machine learning classifier

DFNET can not only be used as a feature module selector, but as tree-based ML model. Lets split the data into a train (80%) and test set (20%).

```
# Create TRAIN set ---------------------------------- #
DFNET_graph_train <- DFNET_graph
## 80% of the sample size
smp_size  <- floor(0.80 * nrow(DFNET_graph$Feature_Matrix[[1]]))
train_ids <- sample(seq_len(nrow(DFNET_graph$Feature_Matrix[[1]])), size = smp_size)
for(xx in 1:length(DFNET_graph_train$Feature_Matrix)){
    DFNET_graph_train$Feature_Matrix[[xx]] <- DFNET_graph$Feature_Matrix[[xx]][train_ids,]
}
table(DFNET_graph_train$Feature_Matrix[[1]][,"target"])
```

```
##
##   0   1
## 161  83
```

```
# Create TEST set ------------------------------------- #
DFNET_graph_test  <- DFNET_graph
test_ids <- (1:nrow(DFNET_graph$Feature_Matrix[[1]]))[-train_ids]
for(xx in 1:length(DFNET_graph_test$Feature_Matrix)){
    DFNET_graph_test$Feature_Matrix[[xx]] <- DFNET_graph$Feature_Matrix[[xx]][test_ids,]
}
table(DFNET_graph_test$Feature_Matrix[[1]][,"target"])
```

```
##
##  0  1
## 42 20
```

Now, lets build the DFNET classifier and check its performance

```
# Perform DFNET
DFNET_object <- DFNET(DFNET_graph_train, ntrees=100, niter=10, init.mtry=20)

# PREDICTION
DFNET_pred   <- DFNET_predict(DFNET_object, DFNET_graph_test, n.last.trees = 100)
```

```
head(DFNET_pred, 10)
```

```
## TCGA.A3.3357 TCGA.A3.3367 TCGA.AK.3434 TCGA.AK.3453 TCGA.AK.3454 TCGA.AK.3461
##            0            0            0            0            0            0
## TCGA.B0.4693 TCGA.B0.4713 TCGA.B0.4810 TCGA.B0.4817
##            0            1            0            1
```

To evaluate the performance on the test data the package caret and e1071 needs to be installed.

```
require(caret)
```

```
## Loading required package: caret
```

```
## Loading required package: lattice
```

```
# PERFORMANCE
target       <- as.factor(DFNET_graph_test$Feature_Matrix[[1]][,"target"])
DFNET_perf   <- DFNET_performance(DFNET_pred, target)
```

```
## Loading required package: e1071
```

```
DFNET_perf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 39 12
##          1  3  8
##
##                Accuracy : 0.7581
##                  95% CI : (0.6326, 0.8578)
##     No Information Rate : 0.6774
##     P-Value [Acc > NIR] : 0.10892
##
##                   Kappa : 0.3725
##
##  Mcnemar's Test P-Value : 0.03887
##
```

```
##                Precision : 0.7273
##                   Recall : 0.4000
##                       F1 : 0.5161
##               Prevalence : 0.3226
##           Detection Rate : 0.1290
##     Detection Prevalence : 0.1774
##        Balanced Accuracy : 0.6643
##
##         'Positive' Class : 1
##
```

```
DFNET_perf$byClass
```

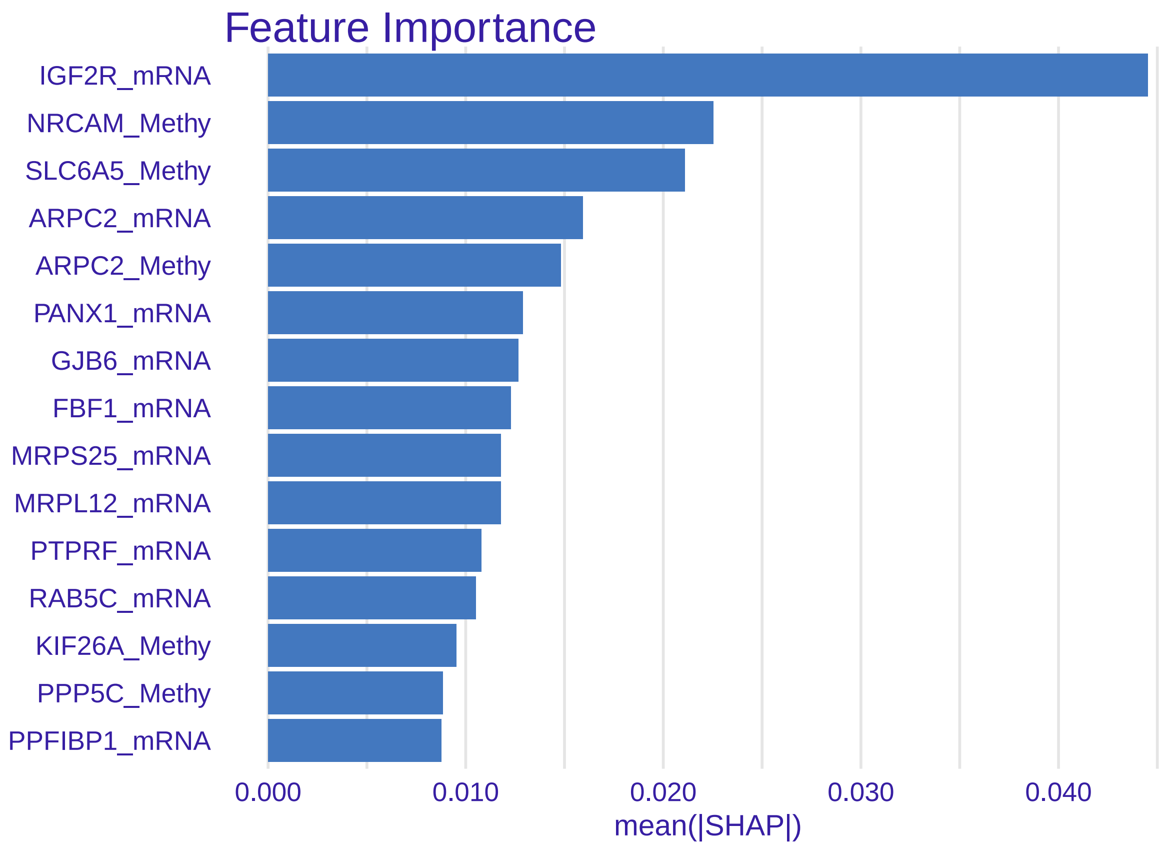
```
##          Sensitivity          Specificity       Pos Pred Value
##            0.4000000            0.9285714            0.7272727
##       Neg Pred Value            Precision               Recall
##            0.7647059            0.7272727            0.4000000
##                   F1           Prevalence       Detection Rate
##            0.5161290            0.3225806            0.1290323
## Detection Prevalence    Balanced Accuracy
##            0.1774194            0.6642857
```

DFNET Tree-based shapley values

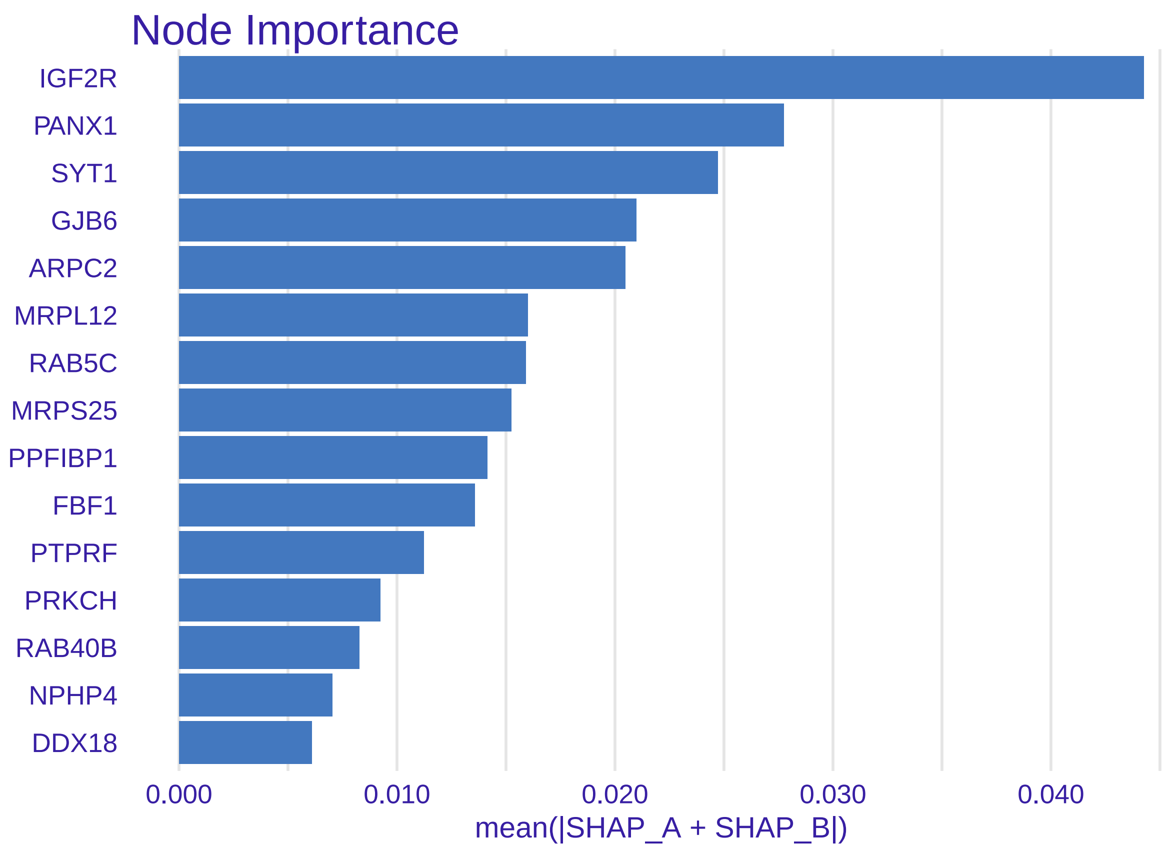
Explanations of the prediction can be obtained using tree-based shapley values. The R-package treeshap needs to be installed from GitHub.

```
#------------------------------
# install.packages("devtools")
# devtools::install_github("ModelOriented/treeshap")
require(treeshap)

# Explanations
forest_shap <- DFNET_explain(DFNET_object, DFNET_graph_test, n.last.trees=100)
sv <- forest_shap$shaps
```

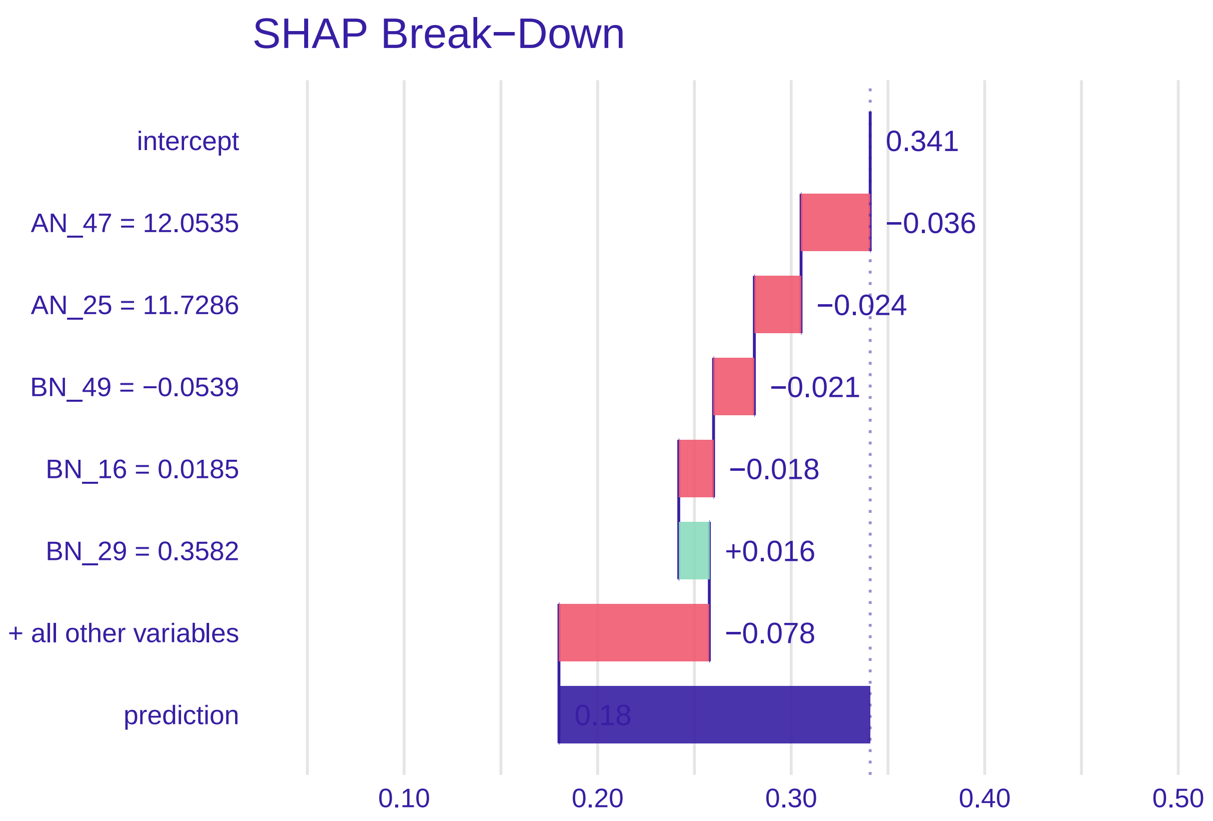
```
require(ggplot2)

global_sv <- colMeans(abs(sv))

# convert to gene names
NN  <- names(global_sv)
gene_names <- DFNET_graph_test$gene.names
NN2 <- strsplit(NN, "_")
NN3 <- sapply(NN2,function(x){return(x[1])})
NN4 <- gsub("AN", "mRNA", NN3)
NN4 <- gsub("BN", "Methy", NN4)
NN5 <- paste(gene_names, NN4, sep="_")
names(global_sv) <- NN5

df <- data.frame(variable = factor(names(global_sv)), importance = as.vector(global_sv))
df$variable <- reorder(df$variable, df$importance)
df <- df[order(df$importance, decreasing = TRUE)[1:15], ]
p <- ggplot(df, aes(x = variable, y = importance)) +
  geom_bar(stat = "identity", fill = colors_discrete_drwhy(1)) +
  coord_flip() +
  theme_drwhy_vertical() +
  ylab("mean(|SHAP|)") + xlab("") +
  labs(title = "Feature Importance") +
  scale_y_continuous(labels = scales::comma) +
  theme(legend.position = "none")
p
```



```
# join mm features into node importance
variable_count <- dim(sv)[2]/2
sv_join <- sv[,1:variable_count] + sv[,(variable_count+1):(2*variable_count)]
colnames(sv_join) <- as.numeric(lapply(
  strsplit(colnames(sv_join), "_"), function(x) ifelse(length(x[-1]) == 0, NA, x[-1])
))
global_sv_joined <- colMeans(abs(sv_join))
names(global_sv_joined) <- DFNET_graph_test$gene.names[-length(DFNET_graph_test$gene.names)]
df_joined <- data.frame(
  variable = factor(names(global_sv_joined)),
  importance = as.vector(global_sv_joined)
)
df_joined$variable <- reorder(df_joined$variable, df_joined$importance)
df_joined <- df_joined[order(df_joined$importance, decreasing = TRUE)[1:15], ]

p_joined <- ggplot(df_joined, aes(x = variable, y = importance)) +
  geom_bar(stat = "identity", fill = colors_discrete_drwhy(1)) +
  coord_flip() +
  theme_drwhy_vertical() +
  ylab("mean(|SHAP_A + SHAP_B|)") + xlab("") +
  labs(title = "Node Importance") +
  scale_y_continuous(labels = scales::comma) +
  theme(legend.position = "none")
p_joined
```



```
# local explanation for patient 20
treeshap::plot_contribution(forest_shap, 20)
```


SHAP Break−Down
intercept
AN_47 = 12.0535
AN_25 = 11.7286
BN_49 = −0.0539
BN_16 = 0.0185
BN_29 = 0.3582
+ all other variables
prediction
0.341
−0.036
−0.024
−0.021
−0.018
+0.016
−0.078
0.18
0.10
0.20
0.30
0.40
0.50