

# netDx use case: 4-way classification: Medulloblastoma tumour subtype

Shraddha Pai

Last updated: 20 June, 2016

## 1 Introduction

Northcott et al. (2011) identified four subtypes of primary medulloblastoma tumours based on gene expression profiles, each predictable by levels of 5 genes (Ref 1). For each tumour, we have gene-level expression data (GSE21140) from the Affymetrix Human Exon 1.0 ST array (gene-level). In this application, our goal is to classify a new tumour into one of the 4 subtypes.

The netDx workflow is shown in Figure 1. We use gene signatures identified by previous research and omit the feature selection step. Instead, GeneMANIA (GM) is used to predict tumour labels. For each of the 4 subtypes, we construct a GM database that contains genes part of the corresponding gene signature. Samples from each subtype are split into training (70%) and test (30%). One query is run for each of the four subtypes, using the training samples for the corresponding subtypes as a query. In this way, each test sample is ranked by all 4 databases. After test samples have been ranked by all four GeneMANIA queries, each set of ranks is normalized to range between 0 and 1. The patient is assigned to the class with the highest rank.

## 2 Set up

### 2.1 Set up working environment

```
# change this to a directory to which you have write access
outDir <- "~/tmp/MB"
dir.create(outDir)

numCores    <- 2L      # number of cores for parallel processing
pctTrain    <- 0.7     # fraction of samples to use for feature selection

require(netDx)
```

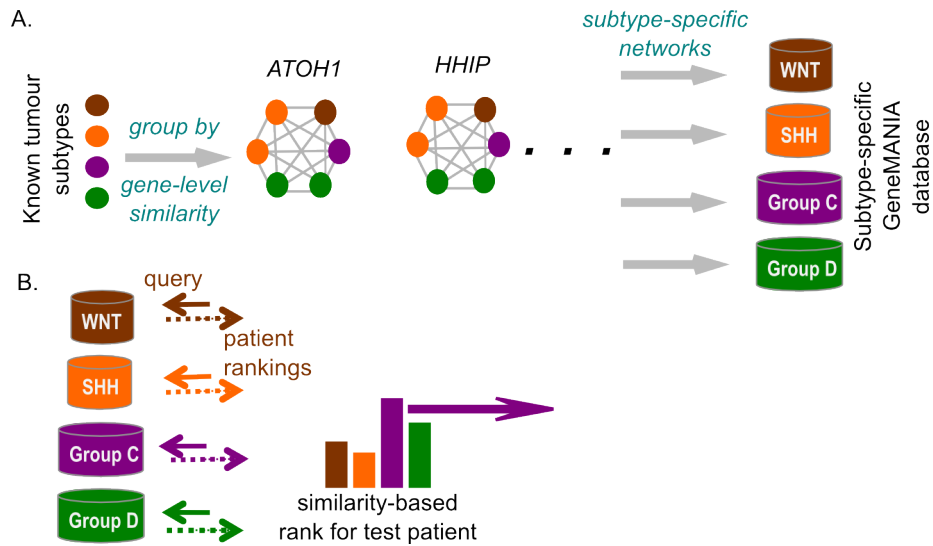


Figure 1: netDx workflow for a 4-way classification of medulloblastoma tumour from known gene signatures.

A. A GeneMANIA database is built for each subtype. The networks are at the gene-level, with a custom similarity metric (see Section 3); each subtype-specific database contains networks for the genes in the corresponding subtype signature. The networks and database contain all training and test patients.

B. A sample is classified by running four GeneMANIA queries, one per subtype. In each case, the query comprises of training samples for that subtype. The query results in subtype-specific similarity ranking. After such ranks have been obtained for all four subtypes, the sample is assigned to the class for which it has the highest rank.

```
## Loading required package: netDx
## Loading required package: bigmemory
## Loading required package: bigmemory.sri
## Loading required package: foreach
## Loading required package: doParallel
## Loading required package: iterators
## Loading required package: parallel
## Loading required package: combinat
##
## Attaching package: 'combinat'
## The following object is masked from 'package:utils':
##
##   combn
## Loading required package: GenomicRanges
```

```

## Loading required package: BiocGenerics
##
## Attaching package: 'BiocGenerics'
## The following objects are masked from 'package:parallel':
##
##   clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##   clusterExport, clusterMap, parApply, parCapply, parLapply,
##   parLapplyLB, parRapply, parSapply, parSapplyLB
## The following objects are masked from 'package:stats':
##
##   IQR, mad, xtabs
## The following objects are masked from 'package:base':
##
##   anyDuplicated, append, as.data.frame, as.vector, cbind,
##   colnames, do.call, duplicated, eval, evalq, Filter, Find, get,
##   grep, grepl, intersect, is.unsorted, lapply, lengths, Map,
##   mapply, match, mget, order, paste, pmax, pmax.int, pmin,
##   pmin.int, Position, rank, rbind, Reduce, rownames, sapply,
##   setdiff, sort, table, tapply, union, unique, unlist, unsplit
## Loading required package: S4Vectors
## Loading required package: stats4
## Loading required package: IRanges
## Loading required package: GenomeInfoDb
## Loading required package: ROCR
## Loading required package: gplots
##
## Attaching package: 'gplots'
## The following object is masked from 'package:IRanges':
##
##   space
## The following object is masked from 'package:stats':
##
##   lowess
## Loading required package: pracma
##
## Attaching package: 'pracma'
## The following object is masked from 'package:combinat':
##
##   fact
## Loading required package: RColorBrewer

require(netDx.examples)

## Loading required package: netDx.examples

```

Load the example data

```
# Load the Medulloblastoma dataset
data(MBblastoma)
```

## 2.2 Define gene signatures

```
# subtypes and genes predictive of these. From Table 1 of PMC3306784
# http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3306784/table/Tab1/
groupSig <- list(
  WNT=c("WIF1", "TNC", "GAD", "DKK2", "EMX2"),
  SHH=c("PDLIM3", "EYA1", "HHIP", "ATOH1", "SFRP1"),
  Group3=c("IMPG2", "GABRA5", "EGFL11", "NRL", "MAB21L2", "NPR3"),
  Group4=c("KCNA1", "EOMES", "KHDRBS2", "RBM24", "UNC5D", "OAS1")
)
```

## 2.3 Split samples into train and test

For this example we manually separate samples from each of the 4 subtypes into 70% training and 30% test

```
# -----
# split dataset into train/test in group-wise manner. Note that we
# include patients not in any particular group because they serve
# as negatives
TT_STATUS <- character(nrow(MB.pheno))
for (g in unique(MB.pheno$STATUS)) {
  idx <- which(MB.pheno$STATUS %in% g)
  status <- rep("TEST", length(idx))
  status[1:(floor(pctTrain * length(idx)))] <- "TRAIN"
  TT_STATUS[idx] <- sample(status, replace=FALSE) # scramble
}
MB.pheno <- cbind(MB.pheno, TT_STATUS=TT_STATUS)
```

## 3 Build predictor for each subtype

In this example, we skip feature selection and move straight to predicting subtypes for test tumours, based on similarity to training samples.

```
MB.pheno_train <- subset(MB.pheno, TT_STATUS %in% "TRAIN")
```

For each subtype, we build a GeneMANIA database consisting of features for that subtype. Each database contains all 103 samples. Features are at the levels of genes. The similarity measure is a custom-defined one, here seen in the `geneSim()` function.

```

# custom similarity measure
geneSim <- function(x) {
  if (nrow(x)>=1) x <- x[1,]
  nm <- colnames(x)
  x <- as.numeric(x)
  n <- length(x)
  rngX <- max(x)-min(x)

  out <- matrix(NA,nrow=n,ncol=n);
  # weight between i and j is
  # wt(i,j) = 1 - (abs(g[i]-g[j])/(max(g)-min(g)))
  # where g is the eMB.expression vector for each gene
  for (j in 1:n) out[,j] <- 1-(abs((x-x[j])/rngX))
  rownames(out) <- nm; colnames(out)<- nm
  out
}

```

For each subtype, the loop combines the following steps:

- `makePSN_NamedMatrix()`: Function that creates the patient networks from the expression vector. Here we supply the custom similarity-computing function via the `customFunc` argument. The interaction networks are written in the directory `netDir`.
- `GM_createDB()`: All the input networks in `netDir` are integrated into a single GeneMANIA database.
- `GM_writeQueryFile()`: Write a GeneMANIA query file; the GeneMANIA query will obtain a similarity ranking for test patients
- `runGeneMANIA()`: Runs a GeneMANIA query
- `GM_getQueryROC()`: Process the patient ranking output file (the `.prank` file)

```

# directories with group-specific predictors
predRes <- list()
## for each subtype
for (g in names(groupSig)){
  pDir <- sprintf("%s/%s",outDir,g)
  if (file.exists(pDir)) unlink(pDir)
  dir.create(pDir)

  # each gene has its own PSN
  sigNets <- list()
  for (g2 in groupSig[[g]]) sigNets[[g2]] <- g2
}

```

```

# create patient networks using train & test samples
# networks are limited to signature genes for this subtype
idx      <- which(MB.xpr_names %in% groupSig[[g]])
cat(sprintf("Subtype : %s { %s } => %i measures\n",
           g, paste(groupSig[[g]],collapse=","), length(idx)))
netDir   <- sprintf("%s/networks",pDir)
if (!file.exists(netDir)) unlink(netDir)

# this function call creates the PSN
netList  <- makePSN_NamedMatrix(MB.xpr[idx,], MB.xpr_names[idx],
                               sigNets,netDir,
                               simMetric="custom",customFunc=geneSim,
                               verbose=TRUE)

# create a GeneMANIA database out of these networks
dbDir   <- GM_createDB(netDir, MB.pheno$ID, pDir)

# run a query using training samples for this subtype.
# get ranking for all patients in the database by running GeneMANIA
trainSamps <- MB.pheno$ID[which(MB.pheno$TT_STATUS %in% "TRAIN"
                               & MB.pheno$STATUS %in% g)]
qFile     <- sprintf("%s/query.txt", pDir)
GM_writeQueryFile(trainSamps, "all", nrow(MB.pheno),qFile)
resFile  <- runGeneMANIA(dbDir$dbDir, qFile, pDir)

# compute ROC curve for each predictor
predRes[[g]] <- GM_getQueryROC(sprintf("%s.PRANK",resFile),
                              MB.pheno, g)
}

## Subtype : WNT { WIF1,TNC,GAD,DKK2,EMX2 } => 4 measures
## Got 4 networks
## * Creating placeholder files
## * Populating database files, recoding identifiers
## * Build GeneMANIA index
## * Build GeneMANIA cache
## * Cleanup[1] "java -d64 -Xmx6G -cp /Library/Frameworks/R.framework/Versions/3.2/Resource
## * Attempt 1 : query.txt
## Subtype : SHH { PDLIM3,EYA1,HHIP,ATOH1,SFRP1 } => 5 measures
## Got 5 networks
## * Creating placeholder files
## * Populating database files, recoding identifiers
## * Build GeneMANIA index
## * Build GeneMANIA cache
## * Cleanup[1] "java -d64 -Xmx6G -cp /Library/Frameworks/R.framework/Versions/3.2/Resource

```

```

## * Attempt 1 : query.txt
## Subtype : Group3 { IMPG2,GABRA5,EGFL11,NRL,MAB21L2,NPR3 } => 6 measures
## Got 5 networks
## * Creating placeholder files
## * Populating database files, recoding identifiers
## * Build GeneMANIA index
## * Build GeneMANIA cache
## * Cleanup[1] "java -d64 -Xmx6G -cp /Library/Frameworks/R.framework/Versions/3.2/Resour
## * Attempt 1 : query.txt
## Subtype : Group4 { KCNA1,EOMES,KHDRBS2,RBM24,UNC5D,OAS1 } => 6 measures
## Got 6 networks
## * Creating placeholder files
## * Populating database files, recoding identifiers
## * Build GeneMANIA index
## * Build GeneMANIA cache
## * Cleanup[1] "java -d64 -Xmx6G -cp /Library/Frameworks/R.framework/Versions/3.2/Resour
## * Attempt 1 : query.txt

```

## 4 Evaluate performance

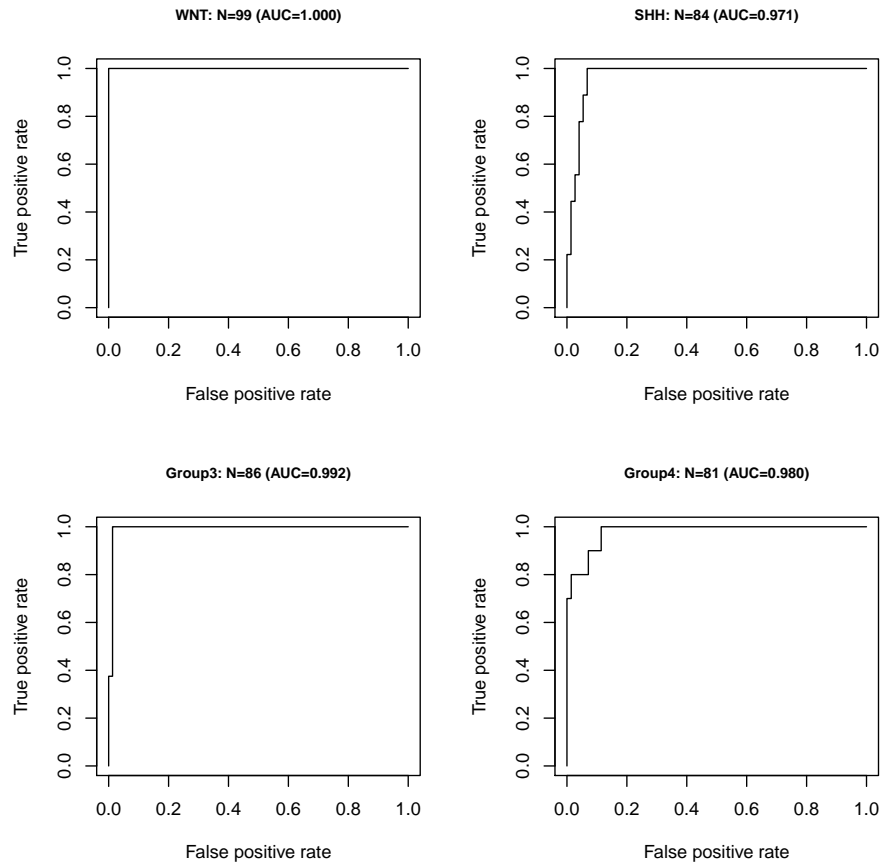
Once we have run GeneMANIA on all 4 subclasses, the patient gets assigned to the class for which it has the highest ranking.

Take a look at the ROC curves.

```

par(mfrow=c(2,2))
tmp <- sapply(names(predRes), function(nm){
  x <- predRes[[nm]]
  plot(x$roc,
       main=sprintf("%s: N=%i (AUC=%1.3f)",
                    nm,length(x$roc@x.values[[1]]),x$auc),
       cex.main=0.8)
})

```



```
save(predRes, file=sprintf("%s/predictions.Rdata", outDir))
```

Finally, predict the class of each test sample

```
predClass      <- GM_OneVAll_getClass(predRes)
## *** 66 rows have an NA prediction
testSamps     <- merge(x=MB.pheno, y=predClass, by="ID")
```

Compute class match accuracy

```
rightClass <- testSamps$STATUS == testSamps$PRED_CLASS
numCor <- sum(rightClass); ln <- nrow(testSamps)
cat(sprintf("Overall classifier accuracy = %i of %i (%i%%)",
            numCor, ln, round((numCor/ln)*100)))
```



```
## Overall classifier accuracy = 29 of 37 (78%)
```

Examine class-specific accuracy:

```
for (g in names(groupSig)){
  idx <- which(testSamps$STATUS %in% g)
  numCor <- sum(rightClass[idx]); ln <- length(idx)
  cat(sprintf("\t%s = %i of %i (%i%%)\n",
             g, numCor, ln, round((numCor/ln)*100)))
}

## WNT = 3 of 3 (100%)
## SHH = 9 of 9 (100%)
## Group3 = 8 of 8 (100%)
## Group4 = 9 of 10 (90%)
```

## 5 sessionInfo

```
sessionInfo()

## R version 3.2.4 (2016-03-10)
## Platform: x86_64-apple-darwin13.4.0 (64-bit)
## Running under: OS X 10.10.5 (Yosemite)
##
## locale:
## [1] en_CA.UTF-8/en_CA.UTF-8/en_CA.UTF-8/C/en_CA.UTF-8/en_CA.UTF-8
##
## attached base packages:
## [1] stats4    parallel  stats      graphics  grDevices  utils      datasets
## [8] methods   base
##
## other attached packages:
## [1] netDx.examples_0.0.0.9000 netDx_0.9
## [3] RColorBrewer_1.1-2        pracma_1.8.8
## [5] ROCR_1.0-7                gplots_2.17.0
## [7] GenomicRanges_1.22.4     GenomeInfoDb_1.6.3
## [9] IRanges_2.4.8            S4Vectors_0.8.11
## [11] BiocGenerics_0.16.1      combinat_0.0-8
## [13] doParallel_1.0.10        iterators_1.0.8
## [15] foreach_1.4.3            bigmemory_4.5.18
## [17] bigmemory.sri_0.1.3      knitr_1.13
##
## loaded via a namespace (and not attached):
```

```
## [1] Rcpp_0.12.3      XVector_0.10.0    magrittr_1.5
## [4] zlibbioc_1.16.0   highr_0.6         plyr_1.8.3
## [7] stringr_1.0.0     caTools_1.17.1   tools_3.2.4
## [10] KernSmooth_2.23-15 gtools_3.5.0     reshape2_1.4.1
## [13] formatR_1.4       bitops_1.0-6     codetools_0.2-14
## [16] evaluate_0.9      gdata_2.17.0     stringi_1.1.1
## [19] compiler_3.2.4
```

## 6 References

1. Northcott PA et al. (2011). J Clin Oncol. 29 (11):1408.