Sadanandam, et al., A colorectal cancer classification system that Associates cellular phenotype and responses to therapy.

All the R codes that were used for this manuscript are provided in this folder.

In addition, download NMFCONSENSUS. R for NMF analysis and NTPEZ.R for NTP analysis from GenePattern - Broad Institute Website http://www.broadinstitute.org/cancer/software/genepattern/.

Also, refer our previous non-peer reviewed publication in Nature Protocols - http://www.nature.com/protocolexchange/protocols/2053#/procedure

```
######################################################################
# Author: Anguraj Sadanandam
# Swiss Institute of Bioinformatics (SIB)
#
#  preprocess_affy.r
#  Preprocess the CEL files from different platforms of Affymetrix microarrays
#  before further analysis.
#  Make sure that the required CEL files are in the same directory where R
#  has been started.
#
######################################################################

#Code for preprocessing CEL files by robust multiarray (RMA) statistics

library(affy)
data <- ReadAffy()
data_rma <- rma(data)
data_exp <- exprs(data_rma)

#Note: The expression values are in log2 scale.

#Code for mapping probesets to gene symbol
library(hgu133plus2.db)
x = unlist(as.list(hgu133plus2SYMBOL)

#Note that here "hgu133plu2.db" has been used as an example for those microarrays
#done using Affymetrix GeneChip Human U133Plus 2.0 array. Install and include the #relevant library
#depending on the type of Affymetrix array used. x will contain the gene symbol corresponding each
#probe.

data_exp_1 <- cbind(x, data_exp)
write.table(data_exp_1,"hgu133plus2_genesonly.txt",
         row.names=FALSE,sep="\t",quote=FALSE)

#This will provide a table of RMA processed gene expression profile data with gene symbols. The
samples are in columns and genes are in rows.

######################################################################
#
# screenExpr_normalize.r -- Screens data from the original expression files.
#       Screens for genes with sample variance above user-specified
#       threshold. Consolidates data for duplicate genes by averaging
#       across duplicates.Then normalizes the data N(0,1) for distance
#       weighted discrimination (DWD) based merging of datasets
#
#       Author: William J. Gibb
#
# Arguments
#       exprFiles -- list of file names
#       sdCutoffs -- list of std dev thresholds corresponding to each
#               file in exprFiles
#
######################################################################

# screens for genes with sample variance
```

```
screenExpr <- function(exprFiles, sdCutoffs) {

        nFiles <- length(exprFiles);
        if(length(sdCutoffs) != nFiles) {
                stop("Number of files and number of sdCutoffs must match.\n");
        }
        exprFiles <- as.vector(exprFiles);
        sdCutoffs <- as.vector(sdCutoffs);

#       sdCutoff <- 1.2;

        for (exprFile in exprFiles) {
                sdCutoff <- sdCutoffs[exprFiles==exprFile];
                # Data set is presumed to contain non-unique geneIDs,
                #so cannot read geneIDs into rownames
                print(paste("Reading expression file: ", exprFile, " ...", sep=""));
                data.all <- read.table(file=exprFile, header=TRUE);

                print(paste("    Read expression data for ",dim(data.all)[1]," genes.",sep=""));

                # Remove Affy probes that do not map to geneIDs
                data.mapped <- data.all[toupper(data.all[,1]) != "UNMAPPED" & !is.na(data.all[,1]),];

                print(paste("    Mapped data to ",dim(data.mapped)[1]," genes IDs.",sep=""));

                # Restrict analysis to genes with sample var above threshold
                nLastCol <- dim(data.mapped)[2];
                print("Removing genes with low sample variance...");
                sampleVar <- apply(data.mapped[,2:nLastCol],1,var, na.rm=TRUE);
            # compute sample variance
                idxHighVar <- sampleVar > sdCutoff^2;
                data.highVar <- data.mapped[idxHighVar,]
                sampleVar.highVar <- sampleVar[idxHighVar];
                print(paste("    Found ",dim(data.highVar)[1]," genes exceeding SD threshold of
                              ",sdCutoff,".",sep=""));

                nGenesHighVar <- dim(data.highVar)[1];
                # Cannot have replicate gene names for clustering; choose gene with largest
                #sample variance
                genesUnique <- as.vector(unique(data.highVar[,1]));
                nGenesUnique <- length(genesUnique);
                nSamples <- dim(data.highVar)[2]-1;             # first col is still gene name
                data <- array(dim=c(nGenesUnique,nLastCol));
                data[,1] <- genesUnique;
                colnames(data) <- colnames(data.highVar);
                print("Removing duplicate genes (selecting for max standard deviation)...");

                for (gene in genesUnique) {
                        # index/indices of all genes matching gene (detect duplicates)
                        idxGenes <- seq(along=1:nGenesHighVar)[data.highVar[,1]==gene];

                        data.slice <- data.highVar[idxGenes,2:nLastCol];
                        if (length(idxGenes)>1) {
                                idxMaxVar <- which.max(sampleVar.highVar[idxGenes]);
                       # find dupls with max var
                                data.slice <- data.slice[idxMaxVar,];
                            }
                        data[data[,1]==gene,2:nLastCol] <- as.matrix(data.slice);
                }
                print(paste("    ",nGenesUnique," unique genes IDs remain.",sep=""));
                outFile <- paste(strsplit(exprFile,".txt"),"_sd",sdCutoff,".txt",sep="");
                write.table(file=outFile, data, quote=FALSE, row.names=FALSE, sep="\t");
                print(paste("Screened data written to file: ", outFile, sep=""));
                print("");
                rm(data.all,data.mapped,data.highVar,data);
        }
```

```
# Post analysis after gene selection
# Normalization N(0,1) for DWD based merging of datasets

data<-read.delim(outFile,stringsAsFactors=FALSE)
dataUnique<-matrix(as.numeric(unlist(data[,2:dim(data)[2]])),nrow=dim(data)[1],ncol=(dim(data)[2]-1))
rownames(dataUnique)<-data[,1]
colnames(dataUnique)<-colnames(data)[2:dim(data)[2]]

library(impute)

imp<-impute.knn(dataUnique)

dataUnique<-imp$data

# column normalization
 colMax <- apply(dataUnique,2,max, na.rm=TRUE);
            colMin <- apply(dataUnique,2,min, na.rm=TRUE);
            scaleFactor <- 1.0/(colMax-colMin);
            for (i in 1:dim(dataUnique)[2]) {
                dataUnique[,i] <- (dataUnique[,i]-colMin[i])*scaleFactor[i];
            }

# row median centering

 rowMed <- apply(dataUnique,1,median, na.rm=TRUE);
 dataUnique<-dataUnique-rowMed

outFile1<- paste(strsplit(exprFile,".txt"),"_sd",sdCutoff,"_col_Norm_row_Med",".txt",sep="");
        dataUnique1<-cbind(data[,1],dataUnique)
        colnames(dataUnique1)[1]<-"Genes"

        write.table(file=outFile1, dataUnique1, quote=FALSE, row.names=FALSE, sep="\t");
      }


######################################################################
#
# Silhouette.r - Evaluates cluster statistically using Silhouette
#                statistics
#
#
######################################################################

# Change file name accordingly - gene expresssion matrix
dwd<-read.delim
("DWD_std_output_GSE13294__GSE1433_default_data_cdt_NMF_ordered.txt",stringsAsFactors=FALSE)
col<-dim(dwd)[2]
row<-dim(dwd)[1]
dwd_m<-matrix(as.numeric(unlist(dwd[,2:col])),nrow=row,ncol=(col-1))
colnames(dwd_m)<-colnames(dwd)[2:col]
rownames(dwd_m)<-dwd[,1]

# provide the cluster information file with tab delimited samples names and cluster identity as a
text file
con<-read.delim("consensus.k.5.gct", stringsAsFactors=FALSE)
dim(con)

# matches the samples names from the expression matrix file to that of the cluster file
m<-match(con[,1],colnames(dwd_m))
w<-which(!is.na(m))
m1<-m[w]
dwd_m_1<-dwd_m[,m1]

library(cluster)

si <- silhouette(as.numeric(con[w,2]), dist(t(dwd_m_1), "euclidean"))

si_1<-matrix(as.numeric(si),nrow=(col-1),ncol=3)
```

```
si_1<-cbind(con[w,],si_1)

# provide the silhouette output file name accordingly
write.table(si_1,"si_o.txt", row.names=FALSE,sep="\t",quote=FALSE)

si_c<-si[,1]

or<-order(si_c)

si_c<-si_c[or]
si_c[si_c==5]="blue"
si_c[si_c==1]="green"
si_c[si_c==2]="yellow"
si_c[si_c==3]="red"
si_c[si_c==4]="magenta"

# provides silhouette plot
plot(si,col = si_c)

# selects those samples that are statistically valid
ww<-which(si_1[,5] >= 0)

length(ww)

dwd_2<-cbind(dwd[,1],dwd_m_1[,ww])

# provide output file name accordingly
write.table(dwd_2,"DWD_std_output_GSE13294__GSE1433_default_data_cdt_NMF_ordered-silhouettee-
selected-samples.txt", row.names=FALSE,sep="\t",quote=FALSE)



######################################################################
#
# SAM_analysis.R -- Performs SAM analysis
# This analysis was performed after silhouette based selection of
# core 387 samples
#
######################################################################

data<-read.delim
("DWD_std_output_GSE13294__GSE1433_default_data_cdt_NMF_ordered_silhouette_selected_samples.txt",
stringsAsFactors=FALSE)

dim(data)

data_m<-matrix(as.numeric(unlist(data[,2:388])),nrow=1262,ncol=387)

rownames(data_m)<-data[,1]

colnames(data_m)<-colnames(data)[2:388]

dim(data_m)

con<-read.delim("consensus.k.5.gct.txt", stringsAsFactors=FALSE)
dim(con)

m<-match(con[,1],colnames(data_m))

w<-which(!is.na(m))

data_m_1<-data_m[,m[w]]

dim(data_m_1)

library(siggenes)
```

```
sam.out<-sam(data_m_1,con[w,2],rand=123,gene.names=rownames(data_m_1))



######################################################################
#
#  PAM.R - Code for prediction analysis of microarrays (PAM)
#  PAM was used after SAM analysis and mappping SAM genes
#
######################################################################

# reading the data
dwd<-read.delim
("DWD_std_output_GSE13294__GSE1433_default_data_cdt_NMF_ordered_silhouette_selected_samples_SAM_genes
_sd0.txt", stringsAsFactors=FALSE)

 dim(dwd)

# converting to a numeric matrix
 dwd_m<-matrix(as.numeric(unlist(dwd[,2:388])),nrow=786,ncol=387)
 colnames(dwd_m)<-colnames(dwd)[2:388]
 rownames(dwd_m)<-dwd[,1]

# reading sample identity or class
lab<-read.delim("label_silhouette_1.txt",stringsAsFactors=FALSE)

#matching the sample names and sample identity
m<-match(lab[,1],colnames(dwd_m))
dwd_2<-dwd_m[,m]

dim(dwd_2)
head(colnames(dwd_2))
head(lab[,1])

#running PAM analysis
library(pamr)
mydata <- list(x=dwd_2,y=factor(lab[,2]), geneid=rownames(dwd_2),genenames=rownames(dwd_2))

#training the data
mytrain <-pamr.train(mydata)

#leave 10 out cross validation (LOCV)
mycv <- pamr.cv(mytrain,mydata, nfold=10)

#plot to check different thresholds
pamr.plotcv(mycv)

#confusion matrix to check the prediction error
pamr.confusion(mycv, threshold=2)

#provides gene list and their scores for each class
pamr.listgenes(mytrain, mydata, threshold=2)



######################################################################
# Author - Adam Olshen, UCSF
# Survival_Analysis.R - Perform survival analysis
#
######################################################################

#Updated 05/07/12

library(survival)

dat <- read.delim("patient-information-class5-silhouette-
selected_class_checked_olshen.txt",header=TRUE,sep="\t",as.is=TRUE)
dat.dfs <- Surv(dat$dfs,dat$censor)
```

```
dat.dfs.subtypes <- coxph(dat.dfs~as.factor(dat$Subtypes))
dat.dfs.stage <- coxph(dat.dfs~as.factor(dat$Duke.stage))
dat.location <- rep(NA,nrow(dat))
dat.location[which(dat$"X.Sample_characteristics_ch1"=="Location: Colon")] <- TRUE
dat.location[which(dat$"X.Sample_characteristics_ch1"=="Location: Left")] <- TRUE
dat.location[which(dat$"X.Sample_characteristics_ch1"=="Location: Right")] <- TRUE
dat.location[which(dat$"X.Sample_characteristics_ch1"=="Location: Rectum")] <- FALSE
dat.dfs.location <- coxph(dat.dfs~dat.location)
dat.dfs.gender <- coxph(dat.dfs~as.factor(dat$Gender))
dat.dfs.age <- coxph(dat.dfs~dat$Age)
dat.radiation <- rep(NA,nrow(dat))
dat.radiation[grep("Y",dat$Adjxrt)] <- TRUE
dat.radiation[grep("N",dat$Adjxrt)] <- FALSE
dat.dfs.radiation <- coxph(dat.dfs~dat.radiation)
dat.chemo <- rep(NA,nrow(dat))
dat.chemo[grep("Y",dat$Adjctx)] <- TRUE
dat.chemo[grep("N",dat$Adjctx)] <- FALSE
dat.dfs.chemo <- coxph(dat.dfs~dat.chemo)
dat.treated <- rep(NA,nrow(dat))
dat.treated[dat.radiation|dat.chemo] <- TRUE
dat.treated[!dat.radiation & !dat.chemo] <- FALSE
dat.dfs.treated <- coxph(dat.dfs~dat.treated)

dat2 <- dat[!dat.treated,]
dat2.dfs <- Surv(dat2$dfs,dat2$censor)
#Does not converge!!!
dat2.dfs.subtypes <- coxph(dat2.dfs~as.factor(dat2$Subtypes))
dat2.dfs.stage <- coxph(dat2.dfs~as.factor(dat2$Duke.stage))
dat2.location <- rep(NA,nrow(dat2))
dat2.location[which(dat2$"X.Sample_characteristics_ch1"=="Location: Colon")] <- TRUE
dat2.location[which(dat2$"X.Sample_characteristics_ch1"=="Location: Left")] <- TRUE
dat2.location[which(dat2$"X.Sample_characteristics_ch1"=="Location: Right")] <- TRUE
dat2.location[which(dat2$"X.Sample_characteristics_ch1"=="Location: Rectum")] <- FALSE
dat2.dfs.location <- coxph(dat2.dfs~dat2.location)
dat2.dfs.gender <- coxph(dat2.dfs~as.factor(dat2$Gender))
dat2.dfs.age <- coxph(dat2.dfs~dat2$Age)

#Revised 03/15/13

model.subtype.<- coxph(dat.dfs~as.factor(dat$Subtypes))
model.treatment <- coxph(dat.dfs~dat.treated)
model.stage <- coxph(dat.dfs~as.factor(dat$Duke))
model.subtype.treatment <- coxph(dat.dfs~as.factor(dat$Subtypes)+dat.treated)
model.subtype.stage <- coxph(dat.dfs~as.factor(dat$Subtypes)+as.factor(dat$Duke))
model.subtype.treatment <- coxph(dat.dfs~as.factor(dat$Subtypes)+dat.treated)
model.subtype.treatment.interaction <- coxph(dat.dfs~as.factor(dat$Subtypes)*dat.treated)

model.subtype.treated <- coxph(dat.dfs[dat.treated]~as.factor(dat$Subtypes)[dat.treated])
model.subtype.nottreated <- coxph(dat.dfs[!dat.treated]~as.factor(dat$Subtypes)[!dat.treated])

model.subtype.treated <- survdiff(dat.dfs[dat.treated]~as.factor(dat$Subtypes)[dat.treated]+strata
(dat$Duke.stage)[dat.treated])
model.subtype.untreated <- survdiff(dat.dfs[!dat.treated]~as.factor(dat$Subtypes)[!dat.treated]
+strata(dat$Duke.stage)[!dat.treated])
```