MetaGxBreast: a package for breast cancer gene expression analysis

Deena M.A. Gendoo[1,2], Natchar Ratanasirigulchai[1], Gregory Chen[2], Levi Waldron[3,4], and Benjamin Haibe-Kains[*1,2]

[1]Bioinformatics and Computational Genomics Laboratory, Princess Margaret Cancer Center, University Health Network, Toronto, Ontario, Canada
[2]Department of Medical Biophysics, University of Toronto, Toronto, Canada
[3]Department of Biostatistics and Computational Biology, Dana-Farber Cancer Institute, Boston, MA, USA
[4]Department of Biostatistics, Harvard School of Public Health, Boston, MA, USA

November 12, 2015

Contents

1 Installing and Loading the Package

The MetaGxBreast package is a compendium of Breast Cancer datasets. The package is publicly available and can be installed from CRAN into R version 2.13.0 or higher.

To install the MetaGxBreast package from CRAN:

```
knitr::opts_chunk$set(eval=TRUE,cache=TRUE)
install.packages("MetaGxBreast")
```

To install the MetaGxBreast package locally, first download the source from CRAN and then run the following in the terminal:

```
R CMD INSTALL MetaGxBreast_2.3.tar.gz
```

Please also load other libraries needed:

*benjamin.haibe.kains@utoronto.ca

```
library(xtable)
library(knitr)
library(tools)
```

2 Loading Datasets

First we load the MetaGxBreast package into the workspace.

To load the packages into R, please use the following commands:

```
library(MetaGxBreast)
source(system.file("extdata", "patientselection.config", package="MetaGxBreast"))
min.number.of.genes <- 0
rm(remove.duplicates)
source(system.file("extdata", "createEsetList.R", package="MetaGxBreast"))
esets.mapped <- esets
```

This will load 39 expression datasets, with patients selected according to the default settings in the patientselection.config file. Users can modify the file to filter and annotate gene expression datasets and individual samples within them based on the following criteria:

Datasets: Conduct probe-gene mapping to select for the 'best' probe (default = TRUE)

Datasets: Retain only genes that are common across all platforms loaded (default = FALSE)

Datasets: Retain studies with a minimum sample size (default = 40)

Datasets: Retain studies with a minimum umber of genes (default = 1000)

Datasets: Retain studies with a minimum number of survival events

Datasets: Remove duplicate samples (default = TRUE)

Datasets: Rescale genes to Z-scores (default = FALSE)

Samples: Ensure specific patient metadata is not missing

Samples: Filter samples by sample type (tumour, healthy, etc)

3 Obtaining Sample Counts in Datasets

To obtain the number of samples per dataset, run the following:

```
numSamples <- NULL
for(i in 1:length(esets.mapped)){
        numSamples <- c(numSamples, length(sampleNames(esets.mapped[[i]])))
}

SampleNumberSummary <- data.frame(NumberOfSamples = numSamples, row.names = names(esets.mapped))
SampleNumberSummary <- rbind(SampleNumberSummary, sum(SampleNumberSummary[,"NumberOfSamples"]))
rownames(SampleNumberSummary)[nrow(SampleNumberSummary)] <- "Total"

knitr::kable(SampleNumberSummary,digits = 2)
```

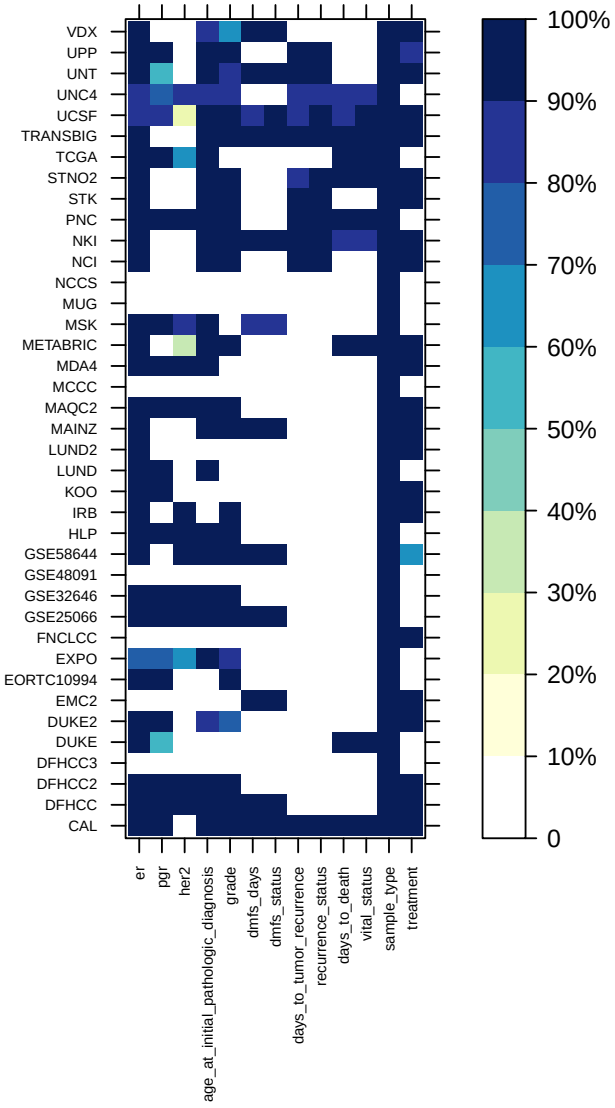
	NumberOfSamples
CAL	118
DFHCC	115
DFHCC2	84
DFHCC3	40
DUKE	171
DUKE2	160
EMC2	204
EORTC10994	49
EXPO	353
FNCLCC	150
GSE25066	508
GSE32646	115
GSE48091	623
GSE58644	321
HLP	53
IRB	129
KOO	88
LUND	143
LUND2	105
MAINZ	200
MAQC2	230
MCCC	75
MDA4	129
METABRIC	2136
MSK	99
MUG	152
NCCS	183
NCI	99
NKI	337
PNC	92
STK	159
STNO2	118
TCGA	1073
TRANSBIG	198
UCSF	162
UNC4	305
UNT	133
UPP	251
VDX	344
Total	10004

4 Assess Phenotype Data

We can also obtain a summary of the phenotype data (pData) for each expression dataset. Here, we assess the proportion of samples in every datasets that contain a specific pData variable.

```
#pData Variables
pDataID <- c("er","pgr", "her2", "age_at_initial_pathologic_diagnosis",
             "grade", "dmfs_days", "dmfs_status", "days_to_tumor_recurrence",
             "recurrence_status", "days_to_death", "vital_status", "sample_type", "treatment")

pDataPercentSummaryTable <- NULL
```

```
pDataSummaryNumbersTable <- NULL
for(e in 1:length(esets.mapped)){
  message(e)
        eset <- esets.mapped[[e]]
        pDataPercentSummary <- NULL
        pDataSummaryNumbers <- NULL
        for(p in 1:length(pDataID)){
                pDataSummaryNumbers <- c(pDataSummaryNumbers,
                                         sum(!is.na(pData(eset)[,pDataID[p]])))
                pDataPercentSummary <- c(pDataPercentSummary,
                                         (sum(!is.na(pData(eset)[,pDataID[p]]))/nrow(pData(eset)))*100)

        }
        if(e == 1){
                pDataSummaryNumbersTable <- data.frame(test = pDataSummaryNumbers)
                pDataPercentSummaryTable <- data.frame(test = pDataPercentSummary)
        } else {
                pDataPercentSummaryTable <- cbind(pDataPercentSummaryTable,pDataPercentSummary)
                pDataSummaryNumbersTable <- cbind(pDataSummaryNumbersTable, pDataSummaryNumbers)
        }
}
rownames(pDataSummaryNumbersTable) <- pDataID
rownames(pDataPercentSummaryTable) <- pDataID
colnames(pDataSummaryNumbersTable) <- names(esets.mapped)
colnames(pDataPercentSummaryTable) <- names(esets.mapped)

pDataSummaryNumbersTable <- rbind(pDataSummaryNumbersTable, SampleNumberSummary["Total",])
rownames(pDataSummaryNumbersTable)[nrow(pDataSummaryNumbersTable)] <- "Total"


# Generate a heatmap representation of the pData
pDataPercentSummaryTable<-t(pDataPercentSummaryTable)
pDataPercentSummaryTable<-cbind(Name=(rownames(pDataPercentSummaryTable)),pDataPercentSummaryTable)

nba<-pDataPercentSummaryTable
gradient_colors = c("#ffffff","#ffffd9","#edf8b1","#c7e9b4",
                    "#7fcdbb","#41b6c4","#1d91c0","#225ea8","#253494","#081d58")

library(lattice)
nbamat<-as.matrix(nba)
rownames(nbamat)<-nbamat[,1]
nbamat<-nbamat[,-1]
Interval<-as.numeric(c(10,20,30,40,50,60,70,80,90,100))

levelplot(t(nbamat),col.regions=gradient_colors,main="Available Clinical Annotation",
          scales=list(x=list(rot=90, cex=0.5), y= list(cex=0.5),key=list(cex=0.2)),
          at=seq(from=0,to=100,length=10),cex=0.2, ylab="", xlab="", lattice.options=list(),
          colorkey=list(at=as.numeric(factor(c(seq(from=0, to=100, by=10)))),
                        labels=as.character(c( "0","10%","20%","30%", "40%","50%",
                                               "60%", "70%", "80%","90%", "100%"),
                                            cex=0.2,font=1,col="brown",height=1, width=1.4),
                        col=(gradient_colors)))
```



5 Session Info

```
toLatex(sessionInfo())
```

- R version 3.2.0 Patched (2015-05-20 r68389), `x86_64-apple-darwin10.8.0`
- Locale: `C/en_CA.UTF-8/en_CA.UTF-8/C/en_CA.UTF-8/en_CA.UTF-8`
- Base packages: base, datasets, grDevices, graphics, methods, parallel, stats, utils
- Other packages: Biobase 2.28.0, BiocGenerics 0.14.0, MetaGxBreast 0.99.0, genefilter 1.50.0, lattice 0.20-33, logging 0.7-103, survival 2.38-1
- Loaded via a namespace (and not attached): AnnotationDbi 1.30.1, DBI 0.3.1, GenomeInfoDb 1.4.0, IRanges 2.2.2, RSQLite 1.0.0, S4Vectors 0.6.0, XML 3.98-1.2, annotate 1.46.0, evaluate 0.7, formatR 1.2,

grid 3.2.0, highr 0.5, knitr 1.10.5, magrittr 1.5, splines 3.2.0, stats4 3.2.0, stringi 0.5-5, stringr 1.0.0, tools 3.2.0, xtable 1.7-4