MetaGxOvarian: a package for ovarian cancer gene expression analysis

Deena M.A. Gendoo[1,2], Natchar Ratanasirigulchai[1], Michael Zon[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

February 3, 2018

Contents

[*]benjamin.haibe.kains@utoronto.ca

1 Installing the Package

The MetaGxOvarian package is a compendium of Ovarian Cancer datasets. The package is publicly available and can be installed from Bioconductor into R version 3.4.1 or higher.

To install the MetaGxOvarian package from Bioconductor:

```
> source("http://bioconductor.org/biocLite.R")
> biocLite("MetaGxOvarian")
```

2 Loading Datasets

First we load the MetaGxOvarian package into the workspace.

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

```
> library(MetaGxOvarian)
> esets = MetaGxOvarian::loadOvarianEsets()[[1]]
```

This will load 26 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)){
+         numSamples <- c(numSamples, length(sampleNames(esets[[i]])))
+ }
> SampleNumberSummaryAll <- data.frame(NumberOfSamples = numSamples,
+                                      row.names = names(esets))
> total <- sum(SampleNumberSummaryAll[,"NumberOfSamples"])
> SampleNumberSummaryAll <- rbind(SampleNumberSummaryAll, total)
> rownames(SampleNumberSummaryAll)[nrow(SampleNumberSummaryAll)] <- "Total"
> require(xtable)
> print(xtable(SampleNumberSummaryAll,digits = 2), floating = FALSE)
```

	NumberOfSamples
E.MTAB.386	129
GSE2109	204
GSE6008	103
GSE6822	66
GSE8842	83
GSE9891	285
GSE12418	54
GSE12470	53
GSE13876	157
GSE14764	80
GSE17260	110
GSE18520	63
GSE20565	140
GSE26193	107
GSE26712	195
GSE30009	103
GSE30161	58
GSE32062	260
GSE32063	40
GSE44104	60
GSE49997	204
GSE51088	172
PMID15897565	63
PMID17290060	117
PMID19318476	42
TCGAOVARIAN	578
Total	3526

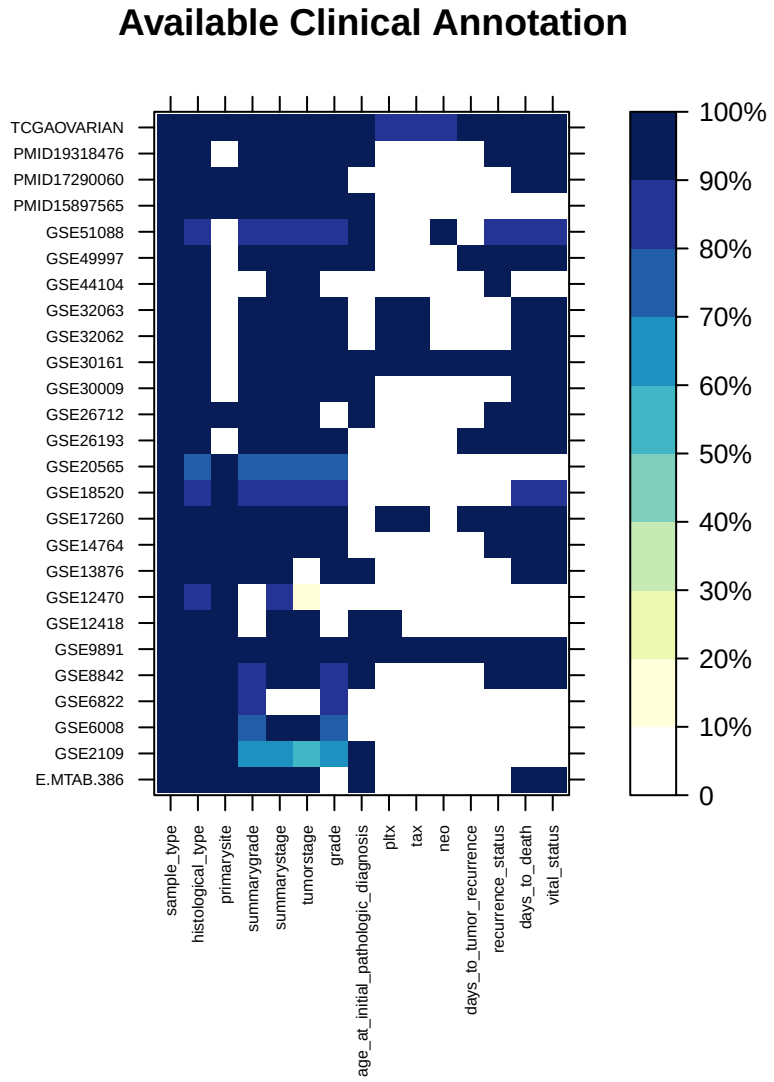
4 Assess Phenotype Data

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("sample_type", "histological_type", "primarysite", "summarygrade",
+              "summarystage", "tumorstage", "grade",
+              "age_at_initial_pathologic_diagnosis", "pltx", "tax",
+              "neo", "days_to_tumor_recurrence", "recurrence_status",
```

```
+              "days_to_death", "vital_status")
> pDataPercentSummaryTable <- NULL
> pDataSummaryNumbersTable <- NULL
> for(e in 1:length(esets)){
+         eset <- esets[[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)
> colnames(pDataPercentSummaryTable) <- names(esets)
> pDataSummaryNumbersTable <- rbind(pDataSummaryNumbersTable, 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

- R Under development (unstable) (2018-01-20 r74146), `x86_64-pc-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_US.UTF-8`, `LC_COLLATE=en_US.UTF-8`,

LC_MONETARY=en_US.UTF-8, LC_MESSAGES=C,
LC_PAPER=en_US.UTF-8, LC_NAME=C, LC_ADDRESS=C,
LC_TELEPHONE=C, LC_MEASUREMENT=en_US.UTF-8,
LC_IDENTIFICATION=C

- Running under: Debian GNU/Linux 9 (stretch)
- Matrix products: default
- BLAS: /usr/lib/openblas-base/libblas.so.3
- LAPACK: /usr/lib/libopenblasp-r0.2.19.so
- Base packages: base, datasets, graphics, grDevices, methods, parallel, stats, utils
- Other packages: AnnotationHub 2.11.2, Biobase 2.39.2, BiocGenerics 0.25.1, ExperimentHub 1.5.1, impute 1.53.0, lattice 0.20-35, logging 0.7-103, MetaGxOvarian 0.99.1, xtable 1.8-2
- Loaded via a namespace (and not attached): AnnotationDbi 1.41.4, BiocInstaller 1.29.4, bit 1.1-12, bit64 0.9-7, blob 1.1.0, compiler 3.5.0, curl 3.1, DBI 0.7, digest 0.6.15, grid 3.5.0, htmltools 0.3.6, httpuv 1.3.5, httr 1.3.1, interactiveDisplayBase 1.17.0, IRanges 2.13.17, memoise 1.1.0, mime 0.5, pillar 1.1.0, pkgconfig 2.0.1, R6 2.2.2, Rcpp 0.12.15, rlang 0.1.6, RSQLite 2.0, S4Vectors 0.17.29, shiny 1.0.5, stats4 3.5.0, tibble 1.4.2, tools 3.5.0, yaml 2.1.16