MetaGxOvarian: a package for ovarian cancer gene expression analysis

Michael Zon[1], 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

May 20, 2021

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 3.5.0 or higher.

To install the MetaGxOvarian package from Bioconductor:

```
> if (!requireNamespace("BiocManager", quietly = TRUE))
+     install.packages("BiocManager")
> BiocManager::install("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. Users can modify the parameters of the function to restrict datasets that do not meet certain criteria for loading. Some example parameters are shown below:

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

Datasets: Retain studies with a minimum sample size (default = 0)

Datasets: Retain studies with a minimum umber of genes (default = 0)

Datasets: Retain studies with a minimum number of survival events (default = 0)

Datasets: Remove duplicate samples (default = TRUE)

3 Obtaining Sample Counts in Datasets

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

```
> numSamples <- vapply(seq_along(esets), FUN=function(i, esets){
+   length(sampleNames(esets[[i]]))
+   }, numeric(1), esets=esets)
```

```
> 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)
```

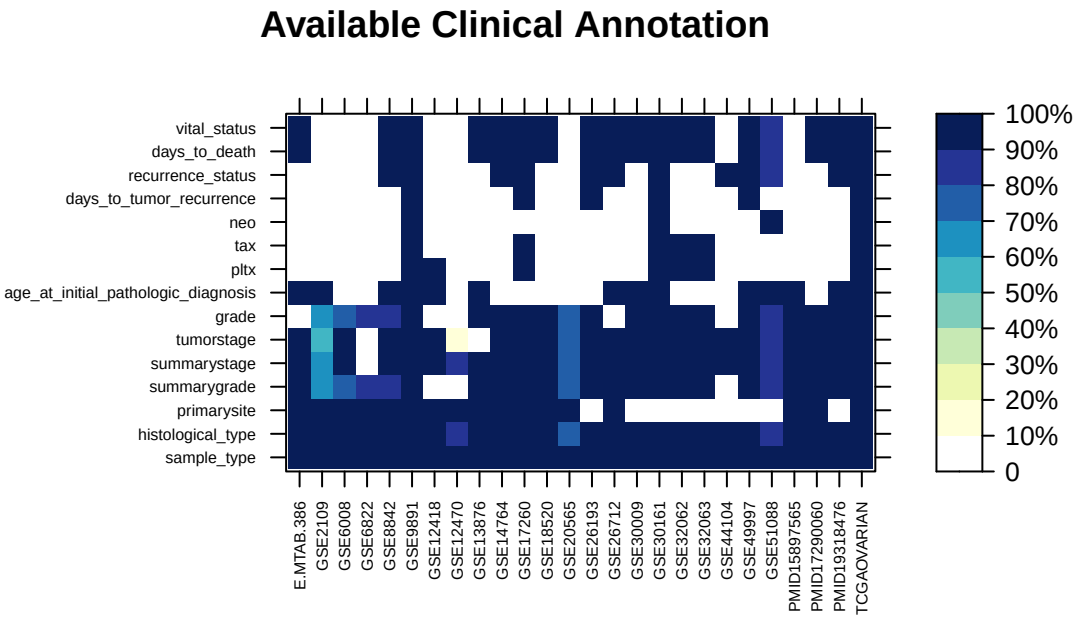
	NumberOfSamples
E.MTAB.386	129.00
GSE2109	202.00
GSE6008	101.00
GSE6822	62.00
GSE8842	83.00
GSE9891	276.00
GSE12418	54.00
GSE12470	49.00
GSE13876	157.00
GSE14764	79.00
GSE17260	110.00
GSE18520	59.00
GSE20565	135.00
GSE26193	14.00
GSE26712	191.00
GSE30009	103.00
GSE30161	58.00
GSE32062	257.00
GSE32063	40.00
GSE44104	47.00
GSE49997	204.00
GSE51088	172.00
PMID15897565	63.00
PMID17290060	117.00
PMID19318476	42.00
TCGAOVARIAN	536.00
Total	3340.00

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
> pDataSummaryNumbersList = lapply(esets, function(x)
+   vapply(pDataID, function(y) sum(!is.na(pData(x)[,y])), numeric(1)))
> pDataPercentSummaryList = lapply(esets, function(x)
+   vapply(pDataID, function(y)
+     sum(!is.na(pData(x)[,y]))/nrow(pData(x)), numeric(1))*100)
> pDataSummaryNumbersTable = sapply(pDataSummaryNumbersList, function(x) x)
> pDataPercentSummaryTable = sapply(pDataPercentSummaryList, function(x) x)
> 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(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 version 4.1.0 (2021-05-18), `x86_64-pc-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_US.UTF-8`, `LC_COLLATE=C`, `LC_MONETARY=en_US.UTF-8`, `LC_MESSAGES=en_US.UTF-8`, `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: Ubuntu 20.04.2 LTS
- Matrix products: default
- BLAS: /home/biocbuild/bbs-3.13-bioc/R/lib/libRblas.so
- LAPACK: /home/biocbuild/bbs-3.13-bioc/R/lib/libRlapack.so
- Base packages: base, datasets, grDevices, graphics, methods, parallel, stats, stats4, utils
- Other packages: AnnotationHub 3.0.0, Biobase 2.52.0, BiocFileCache 2.0.0, BiocGenerics 0.38.0, ExperimentHub 2.0.0, GenomeInfoDb 1.28.0, GenomicRanges 1.44.0, IRanges 2.26.0, MatrixGenerics 1.4.0, MetaGxOvarian 1.12.0, S4Vectors 0.30.0, SummarizedExperiment 1.22.0, dbplyr 2.1.1, impute 1.66.0, lattice 0.20-44, matrixStats 0.58.0, xtable 1.8-4
- Loaded via a namespace (and not attached): AnnotationDbi 1.54.0, BiocManager 1.30.15, BiocVersion 3.13.1, Biostrings 2.60.0, DBI 1.1.1, DelayedArray 0.18.0, GenomeInfoDbData 1.2.6, KEGGREST 1.32.0, Matrix 1.3-3, R6 2.5.0, RCurl 1.98-1.3, RSQLite 2.2.7, Rcpp 1.0.6, XVector 0.32.0, assertthat 0.2.1, bit 4.0.4, bit64 4.0.5, bitops 1.0-7, blob 1.2.1, cachem 1.0.5, compiler 4.1.0, crayon 1.4.1, curl 4.3.1, digest 0.6.27, dplyr 1.0.6, ellipsis 0.3.2, fansi 0.4.2, fastmap 1.1.0, filelock 1.0.2, generics 0.1.0, glue 1.4.2, grid 4.1.0, htmltools 0.5.1.1, httpuv 1.6.1, httr 1.4.2, interactiveDisplayBase 1.30.0, later 1.2.0, lifecycle 1.0.0, magrittr 2.0.1, memoise 2.0.0, mime 0.10, pillar 1.6.1, pkgconfig 2.0.3, png 0.1-7, promises 1.2.0.1, purrr 0.3.4, rappdirs 0.3.3, rlang 0.4.11, rstudioapi 0.13, shiny 1.6.0, tibble 3.1.2, tidyselect 1.1.1, tools 4.1.0, utf8 1.2.1, vctrs 0.3.8, withr 2.4.2, yaml 2.2.1, zlibbioc 1.38.0