

MetaGxOvarian: a package for ovarian cancer gene expression analysis

Deena M.A. Gendoo^{1,2}, Natchar Ratanasirigulchai¹, Michael Zon¹, Gregory Chen², Levi Waldron^{3,4}, and Benjamin Haibe-Kains^{*1,2}

¹Bioinformatics and Computational Genomics Laboratory, Princess Margaret Cancer Center, University Health Network, Toronto, Ontario, Canada

²Department of Medical Biophysics, University of Toronto, Toronto, Canada

³Department of Biostatistics and Computational Biology, Dana-Farber Cancer Institute, Boston, MA, USA

⁴Department of Biostatistics, Harvard School of Public Health, Boston, MA, USA

February 16, 2018

Contents

| | | |
|----------|--|----------|
| 1 | Installing the Package | 2 |
| 2 | Loading Datasets | 2 |
| 3 | Obtaining Sample Counts in Datasets | 3 |
| 4 | Assess Phenotype Data | 4 |
| 5 | Session Info | 6 |

*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 number 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 <- 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 | 204.00 |
| GSE6008 | 103.00 |
| GSE6822 | 66.00 |
| GSE8842 | 83.00 |
| GSE9891 | 285.00 |
| GSE12418 | 54.00 |
| GSE12470 | 53.00 |
| GSE13876 | 157.00 |
| GSE14764 | 80.00 |
| GSE17260 | 110.00 |
| GSE18520 | 63.00 |
| GSE20565 | 140.00 |
| GSE26193 | 107.00 |
| GSE26712 | 195.00 |
| GSE30009 | 103.00 |
| GSE30161 | 58.00 |
| GSE32062 | 260.00 |
| GSE32063 | 40.00 |
| GSE44104 | 60.00 |
| GSE49997 | 204.00 |
| GSE51088 | 172.00 |
| PMID15897565 | 63.00 |
| PMID17290060 | 117.00 |
| PMID19318476 | 42.00 |
| TCGAOVARIAN | 578.00 |
| Total | 3526.00 |

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.

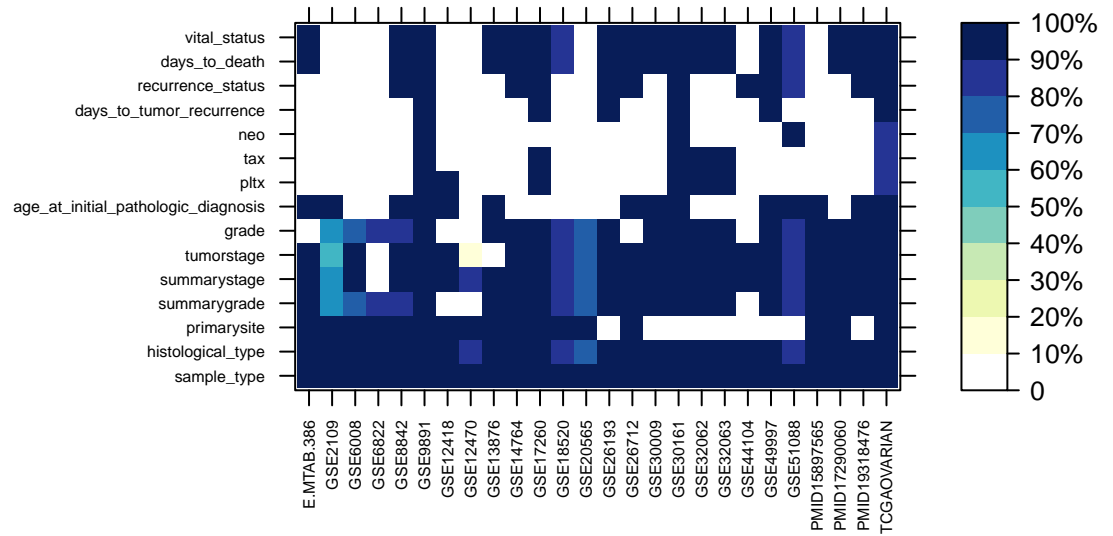
```
> #pData Variables
> pDataID <- c("sample_type", "histological_type", "primarysite", "summarygrade",
+             "summarystage", "tumorstage", "grade",
+             "age_at_initial_pathologic_diagnosis", "pltx", "tax",
+             "neo", "days_to_tumor_recurrence", "recurrence_status",
```

```

+           "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)))
>

```

Available Clinical Annotation



5 Session Info

- R Under development (unstable) (2018-01-26 r74169), x86_64-apple-darwin15.6.0
- Locale: C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

- Running under: OS X El Capitan 10.11.6
- Matrix products: default
- BLAS:
/Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRblas.0.dylib
- LAPACK:
/Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRlapack.dylib
- Base packages: base, datasets, grDevices, graphics, methods, parallel, stats, utils
- Other packages: AnnotationHub 2.11.2, Biobase 2.39.2, BiocGenerics 0.25.3, ExperimentHub 1.5.1, MetaGxOvarian 0.99.7, impute 1.53.0, lattice 0.20-35, xtable 1.8-2
- Loaded via a namespace (and not attached): AnnotationDbi 1.41.4, BiocInstaller 1.29.4, DBI 0.7, IRanges 2.13.26, R6 2.2.2, RSQLite 2.0, Rcpp 0.12.15, S4Vectors 0.17.32, bit 1.1-12, bit64 0.9-7, blob 1.1.0, compiler 3.5.0, curl 3.1, digest 0.6.15, grid 3.5.0, htmltools 0.3.6, httpuv 1.3.5, httr 1.3.1, interactiveDisplayBase 1.17.0, memoise 1.1.0, mime 0.5, pillar 1.1.0, pkgconfig 2.0.1, rlang 0.1.6, shiny 1.0.5, stats4 3.5.0, tibble 1.4.2, tools 3.5.0, yaml 2.1.16