

PGSEA Example Workflow Using Expression Data from GEO

Karl Dykema

May 3, 2016

Laboratory of Computational Biology, Van Andel Research Institute

1 Get Your Data

To test for enrichment of gene sets using our package, you only need two basic items: a gene expression data set and a list of gene sets. Your expression data must either already be in ratio form, or include reference samples by which to generate a ratio on the fly. As you will see later in this example, the "reference" argument will be used because the data is not already in the form of a ratio ("experiment"/"reference").

Online gene expression data repositories represent a great wealth of knowledge and this example workflow will help guide your own analysis by using simple Bioconductor tools to exploit data from such a source, like GEO. To start off we must load the GEOquery library, and then download and parse the data with the getGEO command.

```
> library(PGSEA)
> library(GEOquery)
> library(GSEABase)
> gse <- getGEO("GSE7023", GSEMatrix=TRUE)
> #load("gse.rda")
```

Next we process the raw data from GEO to generate the phenoData for the ExpressionSet, an object that holds gene expression data within Bioconductor. See the GEOquery vignette for more information.

```
> subtype <- gsub("\\.", "_", gsub("subtype: ", "", phenoData(gse[[1]])$characteristics_ch1))
> pheno <- new("AnnotatedDataFrame", data = data.frame(subtype), varMetadata = data.frame(1))
> rownames(pheno@data) <- colnames(exprs(gse[[1]]))
> eset <- new("ExpressionSet", exprs = exprs(gse[[1]]), phenoData = pheno)
```

Next we must load the gene sets to be used, in this case we will use some included example gene sets that we were able to curate from various publications.

This list of gene sets is by no means an extensive or comprehensive list; it is merely included for example purposes. Information on the creation of each gene set has not been included, but may be available upon request.

```
> data(VAIGsc)
> details(VAIGsc[[1]])

setName: CMYC.1 down
geneIds: 27, 101, ..., 377582 (total: 89)
geneIdType: EntrezId
collectionType: ExpressionSet
setIdentifier: 185b7a75-038d-4a5e-c498-d25297d24c9d
description:
organism:
pubMedIds: 16273092
urls:
contributor: Karl Dykema <karl.dykema@vai.org>
setVersion: 0.0.1
creationDate: Fri Sep 14 12:00:24 2007
```

2 Run PGSEA

Finally, we are to the point where we can run PGSEA.

```
> pg <- PGSEA(eset, VAIGsc, ref=which(subtype=="NO"))
```

The results come back in the form of a matrix. Here is a look at a portion of it. A result of "NA" means that the test did not pass the significance threshold.

```
> pg[5:8, 5:8]
```

	GSM162152	GSM162153	GSM162154	GSM162155
HRAS.1 down	NA	NA	NA	NA
HRAS.1 up	NA	3.707952	2.960228	NA
SRC.1 down	NA	NA	NA	NA
SRC.1 up	NA	NA	NA	NA

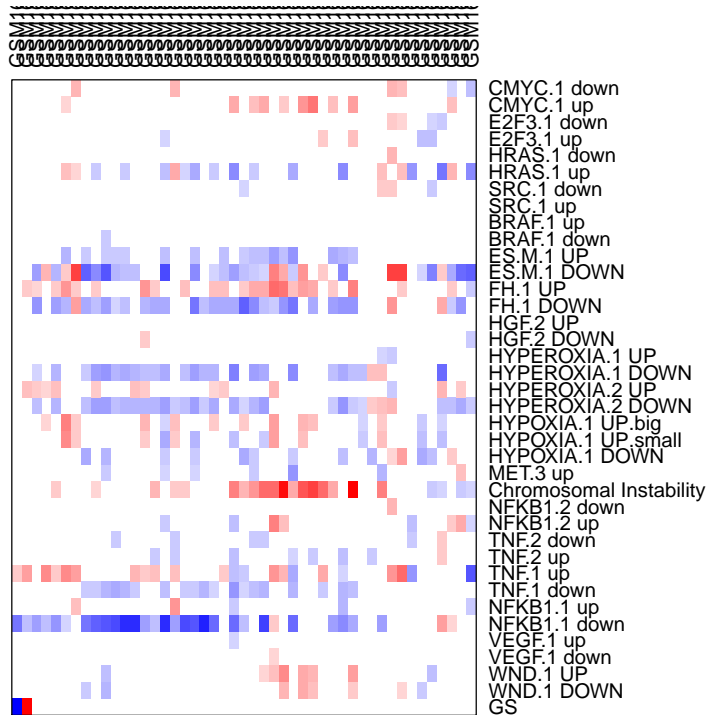
3 Visualize Results

Next we will want to create an attractive plot so that we can visually interpret our results. Looking at the range of the results helps us pick a reasonable scale for the color gradient.

```
> range(pg, finite=TRUE)

[1] -12.64043 17.62942
```

```
> smcPlot(pg, col=.rwb, scale=c(-15, 15))
```



Next with the addition of a few different arguments we can make this plot look much nicer.

```
> smcPlot(pg, factor(subtype), col=.rwb, scale=c(-15, 15), margins=c(1, 1, 6, 9), show.grid=
```



4 Analyze Your Results

One might visually attempt to determine significant differences between our subtypes, but a computational approach is more robust. One simple way to further mine these results is to return an unfiltered matrix from PGSEA and then use a linear modeling approach. In this example we will highlight the differences between Papillary RCC Type 2b and normal renal tissue.

```
> pgNF <- PGSEA(eset, VAlgsc, ref=which(subtype=="NO"), p.value=NA)
> library(limma)
> design <- model.matrix(~ -1+factor(subtype))
> colnames(design) <- names(table(subtype))
> fit <- lmFit(pgNF, design)
> contrast.matrix <- makeContrasts(P2B-NO , levels=design)
> fit <- contrasts.fit(fit, contrast.matrix)
> fit <- eBayes(fit)
> topTable(fit, n=10)[,c("logFC", "t", "adj.P.Val")]
```

	logFC	t	adj.P.Val
Chromosomal Instability	9.107348	7.795340	2.179072e-08
FH.1 UP	5.111695	6.434137	1.176181e-06
ES.M.1 UP	-3.417913	-6.169907	1.953786e-06

TNF.1 down	-2.783328	-4.845456	1.165604e-04
CMYC.1 up	3.539790	4.818909	1.165604e-04
FH.1 DOWN	-5.053123	-4.352914	4.470032e-04
TNF.2 up	-2.179557	-3.578635	4.265204e-03
WND.1 UP	2.695910	3.473171	5.101648e-03
NFKB1.1 down	-3.784797	-2.670381	3.858354e-02
HGF.2 UP	1.087091	2.660417	3.858354e-02

Now that we have figured out the significantly different gene sets, we can grab the rownames from "topTable" and insert it into our plotting function.

```
> smcPlot(pg[as.numeric(rownames(topTable(fit, n=10)))], factor(subtype, levels=c("P1", "P2B")))
```



Biological interpretation of these results is tricky, and can only end up as "good" as the gene sets and gene expression data used. We have some confidence in most of these signatures, and in this case, a few of the results are worth of note.

The "FH.1" gene sets were created from gene expression data generated from fumarate-hydratase (gene symbol: "FH") deficient uterine fibroids (PMID: 16319128). Using these signatures in PGSEA attempted to detect the same pattern of altered expression as was seen in the fibroids. "FH" is known to be involved with aggressive renal papillary cancers (PMIDs: 17895761, 17392716,

17270241, etc.) so this result is expected. While the involvement of "FH" and renal cancer was already well-established, the association with "c-MYC" was unreported, so it was an interesting research topic for us. For more information see: "Detection of DNA copy number changes and oncogenic signaling abnormalities from gene expression data reveals MYC activation in high-grade papillary renal cell carcinoma." (Cancer Res. 2007 Apr 1;67(7):3171-6. PMID: 17409424)

5 Using Large Databases of GeneSets with PGSEA

Instead of using our included example gene sets, one may wish to go another route and use a large database such as "GO". We have included a helper function to quickly generate gene sets from this database.

```
> gos <- go2smc()
> pg <- PGSEA(eset, gos, ref=which(subtype=="NO"))
> pgNF <- PGSEA(eset, gos, ref=which(subtype=="NO"), p.value=NA)
> #load("ABC.rda")
> design <- model.matrix(~ -1+factor(subtype))
> colnames(design) <- names(table(subtype))
> fit <- lmFit(pgNF, design)
> contrast.matrix <- makeContrasts(P2B-P1,levels=design)
> fit <- contrasts.fit(fit, contrast.matrix)
> fit <- eBayes(fit)
```

The linear model approach above will highlight the differences between subtypes P2B and P1 for all go terms. Below, we plot the results and restrict the subtypes shown to those two that we are currently interested in.

```
> smcPlot(pg[as.numeric(rownames(topTable(fit,n=30,resort.by="logFC"))),], factor(subtype,1e
```

GS	P1									
	P2B									