SMITE Vignette

Andrew D. Johnston, N. Ari Wijetunga, and John M. Greally

July 28, 2015

Abstract

This tutorial exemplifies how **SMITE** can inegrate the results from gene expression and epigenome wide studies to identify functional modules, subnetworks within a gene interaction network. Here we will take gene expression, DNA methylation and publicly available ChIP-seq results simulated from a large multi-level unpublished study, and we will generate significance-based functional modules. The main aim of our example is to find functional modules that characterize processes related to the infection of HFF cells with *Toxoplasma Gondii* when compared to uninfected controls. In the process, we detail how to set-up **SMITE**, curate data for input, use **SMITE** commands, and annotate/visualize modules.

1 Setup Environment

First, we will set our environment parameters and install the **SMITE** package. The package can be found on Github

```
options(stringsAsFactors=FALSE)
library(SMITE)
```

2 Curate Data

Before inputting epigenetic and expression data into the **SMITE** algorithms, expression, DNA methylation, and annotation tracks need to be in the proper format. Because genomics data can come in many different formats, pre-processing the data is important to establish uniformity in downstream results. Within the package, there is curated, pre-processed data available for testing as well.

Start by loading in a DNA methylation dataset(Chr, Start, Stop, effect size, p-value for each CpG). This data was generated using the HELP-tagging assay and shows the difference of mean 5'-cytosine methylation between triplicate *Toxoplasma* infected and uninfected Human Foreskin Fibroblasts (HFF). The p-value was generated using t-tests (DNA methylation ~ Infection).

To curate the DNA methylation dataset, it is necessary to remove uninformative loci (`NAs`) and p-values=0 since logarithms of p-values are employed within the code. We replace any p-values=0 with the minimum p-value in the dataset. We also remove CpGs for which the p-value is `NA`. The curated datasets available through Github have already been processed but were not included in the package because of size restrictions.

```
## Load methylation data ##
data(methylationdata)
head(methylation)
```

```
##          V1        V2        V3     V4         V5
## 7687  chr11   2160269   2160270 -0.717 0.34022689
## 95408 chr10  32793206  32793207     NA         NA
## 11891 chr10  11305429  11305430 -1.593 0.07552725
## 59902 chr10 115770766 115770767 -6.291 0.44454585
## 42342 chr11   1780702   1780703 -1.053 0.36048353
## 85425 chr11    456508    456509 -1.644 0.00000000
```

```
## Replace zeros with minimum non-zero p-value ##
methylation[, 5] <- replace(methylation[, 5], methylation[, 5] ==
    0, min(subset(methylation[, 5], methylation[, 5] != 0), na.rm = TRUE))
## Remove NAs from p-value column ##
methylation <- methylation[-which(is.na(methylation[, 5])), ]
```

Before integrating data from multiple datasets, it is **EXTREMELY** important that you choose a single gene ID platform that you will use for the entire analysis. Any form of gene ID will work as long as every dataset uses it or efforts are made to convert gene IDs. We prefer to use gene symbols to avoid downstream networks cluttered by multiple transcripts of the same gene, and we assign gene symbols to our expression dataset instead of RefSeq IDs using a convenient function. If starting from a different gene ID system, one can also convert ensemble, ensembleprot, and uniprot gene IDs to their respective gene symbols (See Manual). The `convertGeneIds` function enumerates the combinations of to and from, so if a particular annotation is desired, we can add the functionality, if requested.

```
## Load fake genes to show expression conversion ##
data(genes_for_conversiontest)
genes[, 1] <- convertGeneIds(gene_IDs = genes[, 1], ID_type = "refseq",
    ID_convert_to = "symbol")
```

Next, we load in the example expression dataset (genes, effect, p-value) generated via RNA-seq on the same HFF samples. Because of size restrictions, we could not include an unprocessed dataset. We have, however, provided an unprocessed dataset on Github for users to apply the following lines of codes to in order to better see how to pre-process their data. In the curated data, the rownames are gene symbols and the other columns include effect size (log fold change) and p-value from a negative binomial test in DESeq.

After gene id conversion (example shown above), we remove expression data for genes where a

gene symbol was not found (NAs). We then take the lowest (most significant) pvalue for genes that had more than 1 entry. Finally, we replace p-values of zeros with the lowest p-value in the dataset.

```
## This is just an example of how to pre-process data ##
expression <- expression[-which(is.na(expression[, 1])), ]
expression <- split(expression, expression[, 1])
expression <- lapply(expression, function(i) {
    if (nrow(as.data.frame(i)) > 1) {
        i <- i[which(i[, 3] == min(i[, 3], na.rm = TRUE))[1],
            ]
    }
    return(i)
})
expression <- do.call(rbind, expression)
expression <- expression[, -1]
expression[, 2] <- replace(expression[, 2], expression[, 2] ==
    0, min(subset(expression[, 2], expression[, 2] != 0), na.rm = TRUE))
```

```
## Load expression data ##
data(curated_expressiondata)
## View data ##
head(expression_curated)
```

```
##              effect        pval
## A1BG      0.3190600 0.835528104
## A1BG-AS1 -0.7744693 0.127105406
## A2M      -0.3604916 0.133665150
## A2M-AS1   0.3482564 0.004019176
## A2ML1     0.4232293 0.441982383
## A2MP1    -4.0973360 0.024239448
```

Then, we load in gene sequences as well as other data that we wish to include in our analysis. These files must be in bed format with the first three columns as (chr,start,stop). A gene strand and name are also required, but the downstream functions allow users to specify a column for strand and gene name.

```
## Load hg19 gene annotation BED file ##
data(hg19_genes_bed)
## Load histone modification file ##
data(histone_h3k4me1)
## View files ##
head(hg19_genes)
```

```
##     V1     V2     V3           V4        V5 V6
## 1 chr1  11873  14409    NR_046018   DDX11L1  +
## 2 chr1  14361  29370    NR_024540    WASH7P  -
```

```
## 3 chr1  34610  36081    NR_026818   FAM138A  -
## 4 chr1  34610  36081    NR_026820   FAM138F  -
## 5 chr1  69090  70008 NM_001005484     OR4F5  +
## 6 chr1 134772 140566    NR_039983 LOC729737  -
```

```
head(h3k4me1)
```

```
##     V1     V2     V3
## 1 chr1 569800 569999
## 2 chr1 724000 727199
## 3 chr1 751600 758999
## 4 chr1 760800 762199
## 5 chr1 764800 765599
## 6 chr1 766800 769399
```

3 Integrate Datasets

In `makePvalueAnnotation`, we create a `GRangesList` object where each gene is associated with a promoter region (+/- 1000bp from TSS), gene body region (TSS+1000bp to TES), and putative "enhancers" using H3K4me1 ChIP-seq peaks (+/- 5000bp from TSS). Note: These are the sizes of regions of the genome for which we are interested in calculating scores, but the user can easily alter the parameters. Additionally, the argument otherdata can be an unlimited list of bed files that will be associated with genes and then scored in functional modules. The argument other_d is the distance from the TSS that will be used to associate each otherdata dataset with a gene. If different distances are required for otherdata, then you must provide a distance for each dataset.

```
test_annotation <- makePvalueAnnotation(data = hg19_genes,
    other_data = list(h3k4me1 = h3k4me1), gene_name_col = 5,
    other_tss_distance = 5000, promoter_upstream_distance = 1000,
    promoter_downstream_distance = 1000)
```

```
## Genes are duplicated.  Removing duplicates
```

Having created a `PvalueAnnotation`, we can now load in the expression and methylation dataset. First we load the expression data. If `effect_col` and `pval_col` are not provided, the program will attempt to find them using the column names, but its probably safer to just provide the arguments.

```
test_annotation <- annotateExpression(pvalue_annotation = test_annotation,
    expr_data = expression_curated, effect_col = 1,
    pval_col = 2)
```

To load in modification data, we have provided a function that can be used multiple times, once for each modifcation data type (e.g. DNA methylation, DNA hydroxymethylation). This is accomplished by using the `mod_type` character string, which can be any word. Please do

not use an underscore or names that are nested in one another (e.g. methylation and methyl) as this will cause erratic behavior when string splitting column names downstream. For each loaded modification, when `mod_corr=TRUE` (DEFAULT);set as `FALSE` to reduce computation time) the function will determine a correlation structure, adjust the p-values and combine the p-values using the method specified (here, Stouffer's method (DEFAULT)). In addition, we use a Monte Carlo Method (MCM) of random sampling of the combined scores to determine a FDR like p-value which will used as the p-value/score in downstream analysis. When combining p-values using any method (see companion paper or `?annotateModification` for more details), p-values will be combined over the gene promoters (DEFAULT), gene bodies (DEFAULT), and over any provided other datasets. Stouffer's method allows optional weights to be given, which we define here has `"distance"` (weighting the effect and p-value by distance from the TSS), `"pval"` (weighting the effect by the signifcance of the effect), or anything other text (unweighted). Note: Depending on the amount of data, this step can take roughly 10 minutes per `mod_type`.

```
test_annotation <- annotateModification(test_annotation,
    methylation, weight_by_method = "Stouffer",
    weight_by = c(promoter = "distance", body = "distance",
        h3k4me1 = "distance"), verbose = TRUE,
    mod_corr = FALSE, mod_type = "methylation")
```

We can view the loaded data using the following functions:

```
## See expression data ##
head(extractExpression(test_annotation))
## See the uncombined p-values for a specific or all
## modType(s) ##
extractModification(test_annotation, mod_type = "methylation")
## See the combined p-value data.frame ##
head(extractModSummary(test_annotation))
```

4 Adjusting Values and Scoring

Having loaded gene expression and DNA methylation data into the `PvalueAnnotation`, we now bring all of the data together into an object class called a `PvalueObject`. The `PvalueObject` we will create has a slot within the `PvalueAnnotation` called `score_data` accessed through `slot(PvalueAnnotation, "scoredata")` or one of the accessor functions that we have set up like `SMITEextractScores`. At this step we specify an *a priori* argument that will be used in scoring downstream `effect_directions`. We must specify an `effect_directions` element for each modification-context pairing that we wish to score. It should reflect whether you want your modification-context (e.g. DNA methylation in gene promoters) to have a pre-specified relationship with expression so that if the opposite relationship is observed the score will be penalized. Users can specify either `"increase"`, `"decrease"` and `"bidirectional"` (for no prespecifed relationship). This information will

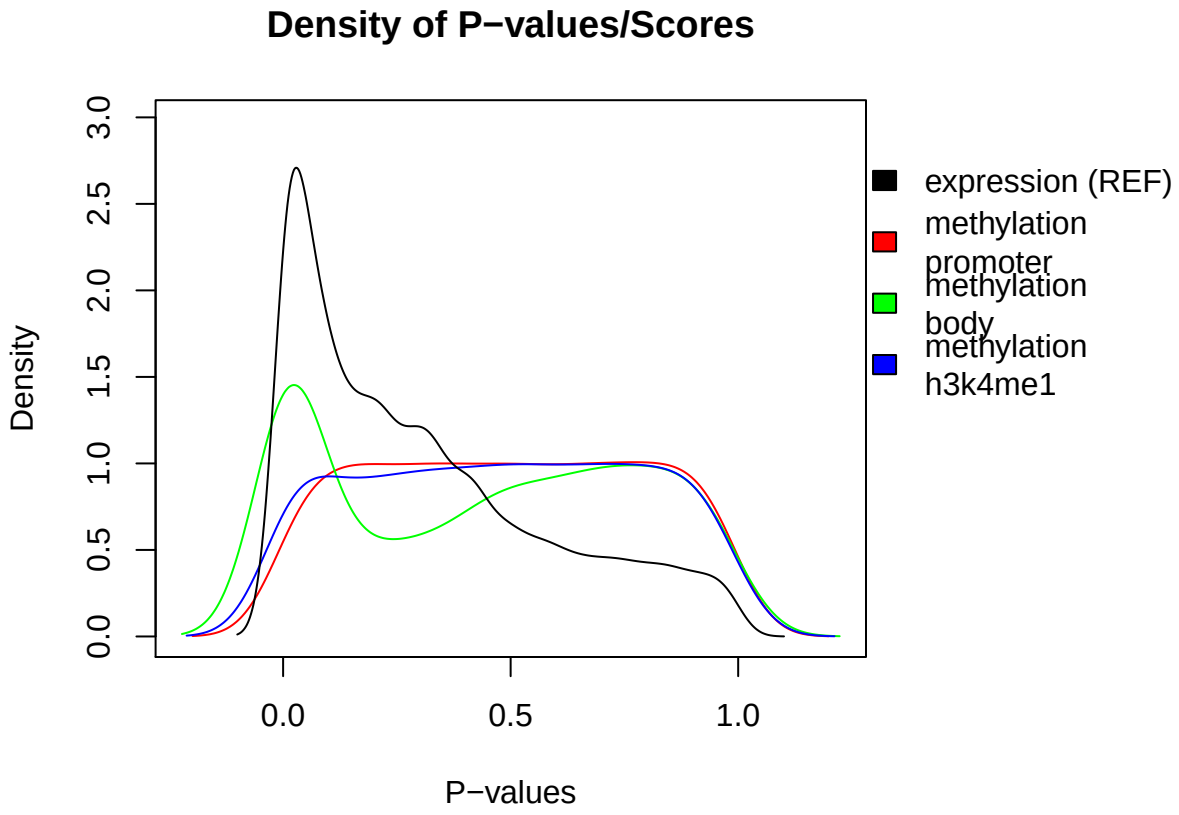
be stored and will not be used until the `SMITEscorePval` function is used.

```
test_annotation <- makePvalueObject(test_annotation,
    effect_directions = c(methylation_promoter = "decrease",
        methylation_body = "decrease",
        methylation_h3k4me1 = "bidirectional"))
```

P-values/scores that are combined and then randomized will have a different range than gene expression, or one another, which can skew the identified modules toward a particular modification-context pairing. Using the `plotDensityPval` function we view the distribution and then normalize using the `normalizePval` function if necessary. Using the (DEFAULT) `rescale` procedure, p-values are logit transformed before rescaling resulting in a shifting of an approximately normal distribution and no effect on the order of p-values within a modification-context pairing.

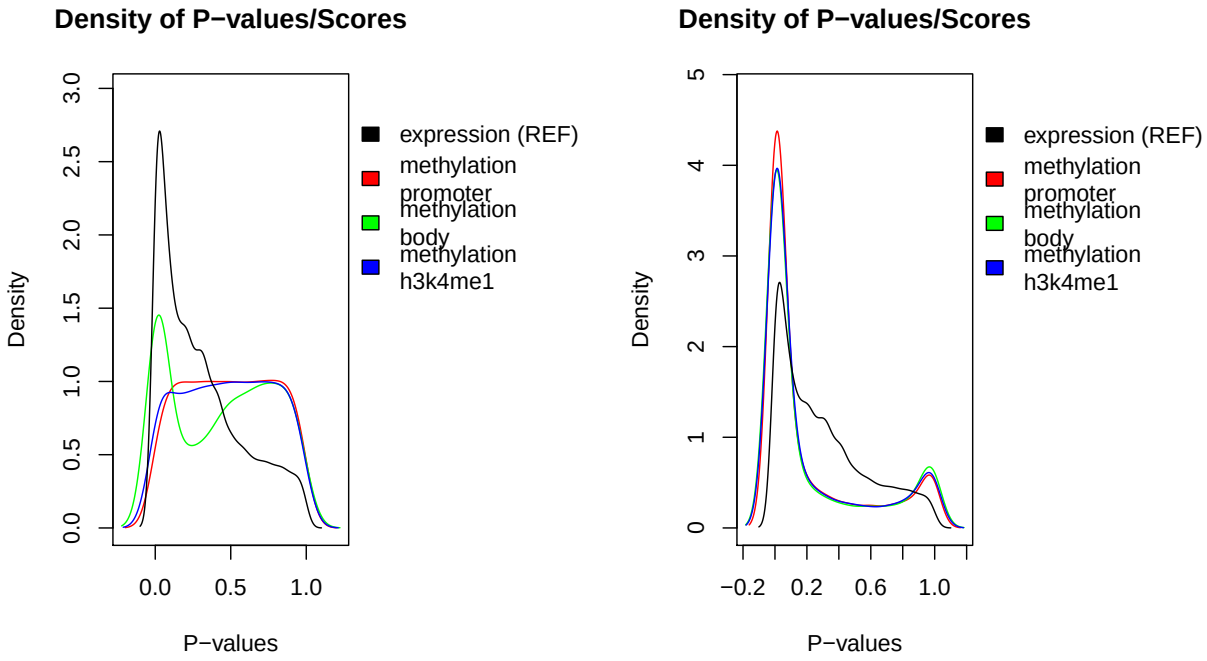
```
## Plot density of p-values ##
plotDensityPval(pvalue_annotation = test_annotation,
    ref = "expression")
```

```
## Plotting:  methylation_promoter
## Plotting:  methylation_body
## Plotting:  methylation_h3k4me1
```



```
## Normalize the p-values to the
## range of expression ##
test_annotation <- normalizePval(test_annotation,
    ref = "expression", method = "rescale")
```

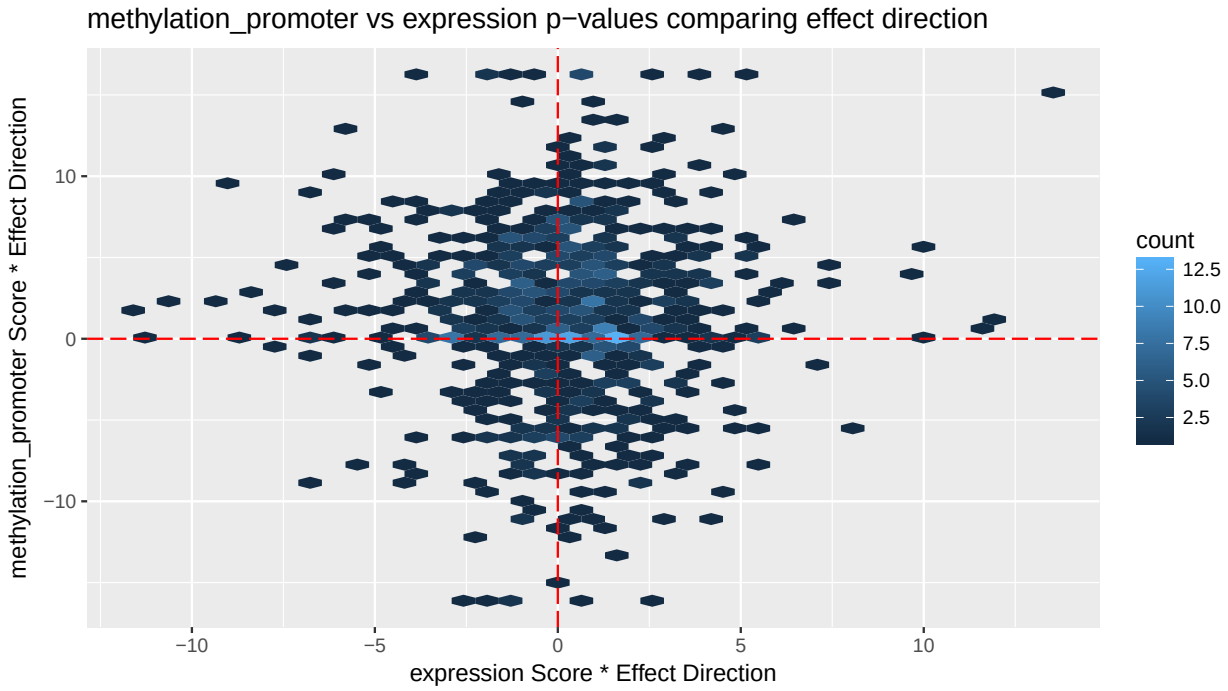
```
## Plotting:  methylation_promoter
## Plotting:  methylation_body
## Plotting:  methylation_h3k4me1
## Plotting:  methylation_promoter
## Plotting:  methylation_body
## Plotting:  methylation_h3k4me1
```



We can compare p-values for different features using the `plotCompareScores` function, with the hope that it may reveal interesting patterns within the data.

```
plotCompareScores(test_annotation, x_name = "expression",
    y_name = "methylation_promoter")
```

```
## Warning: Removed 20228 rows containing non-finite values (stat_binhex).
```



Scoring the genes is the final step before finding modules, and after scoring, the high scoring genes can be extracted using the `SMITEhighScores` function. When using `SMITEscorePval`, users can provide an optional weighting vector that will allow prioritization of certain modfication-context pairings. We included this feature because we find that a researcher often has a specific goal (e.g. finding DNA methylation that is significantly different at enhancer), and rather than allowing researchers to selectively choose results that support their hypothesis, it may be beneficial to define a numeric quantity that can be reported at the time of publication and reproduced. We provide the `SMITEreport` function for text dump of all defined parameters.

```
# score with all four features contributing
test_annotation <- scorePval(test_annotation,
    weights = c(methylation_promoter = 0.3, methylation_body = 0.1,
        expression = 0.3, methylation_h3k4me1 = 0.3))
```

5 Visualize Modules

Now that we have generated weighted significance values from different modifications that relate to each gene, we can visualize modules using interactome data and module-building packages, such as `BioNet`. Specifically, following the example of `Epimods` we use igraph's spinglass algorithm to determine the best modules.

First, we load the desired interactome, in our case from `REACTOME` and run the Spin-glass algorithm. We can then run a `goseq` like approach on our modules to annotate them using pathways, like `KEGG`.

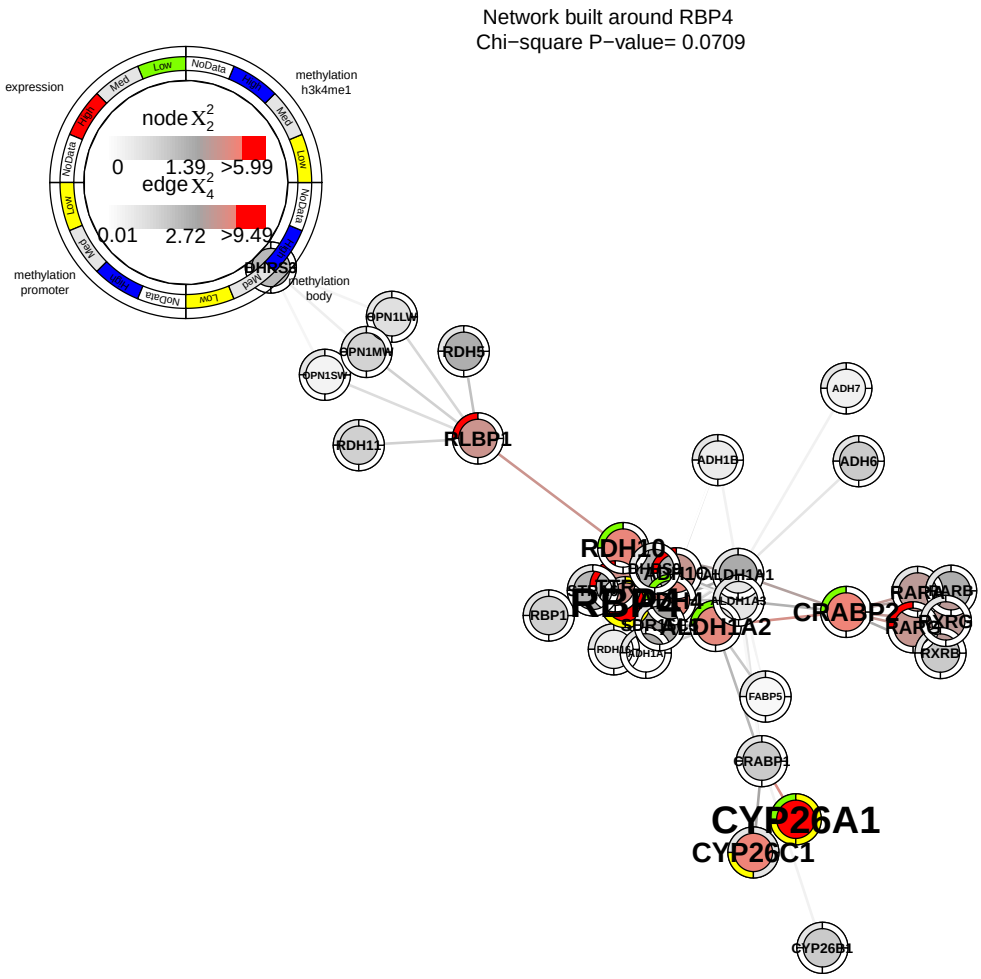
```
## load REACTOME ##
load(system.file("data", "Reactome.Symbol.Igraph.rda",
    package = "SMITE"))
## Run Spin-glass ##
test_annotation <- runSpinglass(test_annotation,
    network = REACTOME, maxsize = 50, num_iterations = 1000)
## Run goseq on individual modules to determine
## bias an annotate functions ##
test_annotation <- runGOseq(test_annotation, coverage = read.table(system.file("extdata
    "hg19_symbol_hpaii.sites.inbodyand2kbupstream.bed.gz",
    package = "SMITE")), type = "kegg")
```

We can use keywords such as "cell cycle" to find within the goseq output which modules we are most interested. We can then use `SMITEplotModule` to visualize the module.

```
## search goseq output for keywords ##
searchGOseq(test_annotation, search_string = "cell cycle")
```

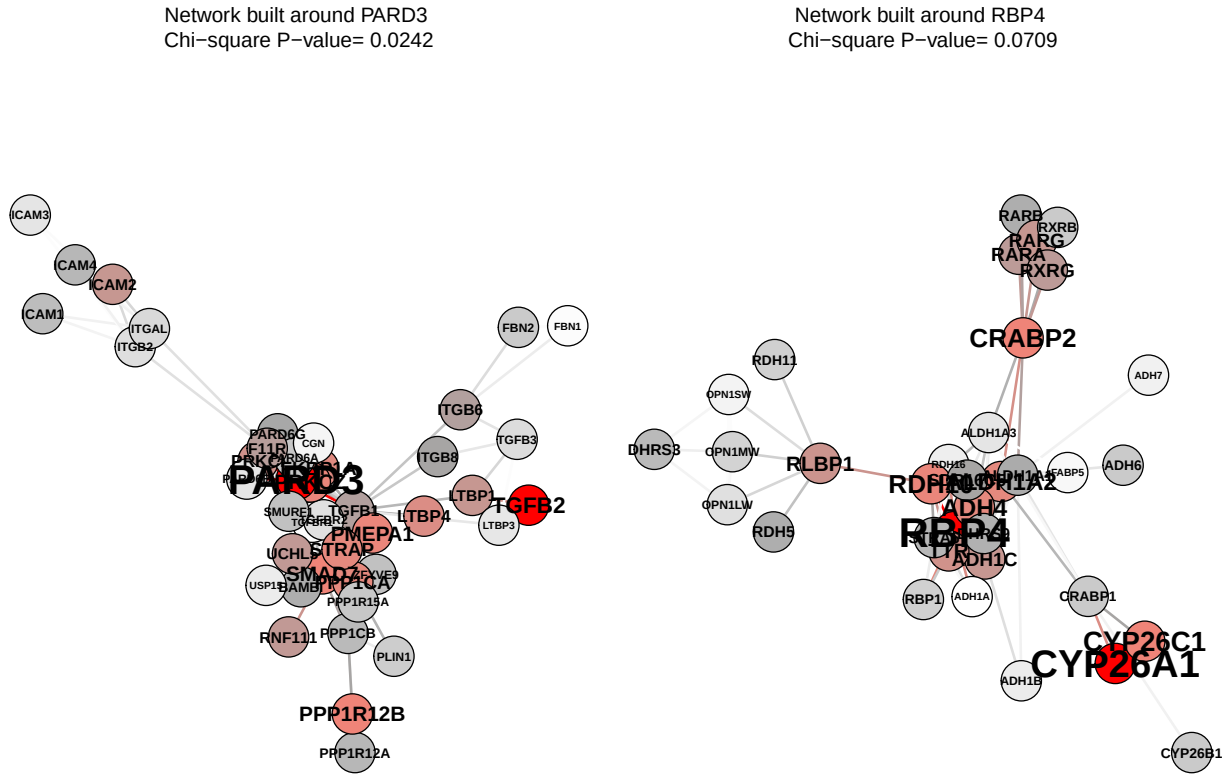
```
##   epimod_name epimod_position_pval       term rank_of_term total_terms
## 1       PARD3           4 / 0.0242 Cell cycle           28          30
```

```
## Draw a network ##
plotModule(test_annotation, which_network = 6, layout = "fr",
    label_scale = TRUE, compare_plot = FALSE)
```

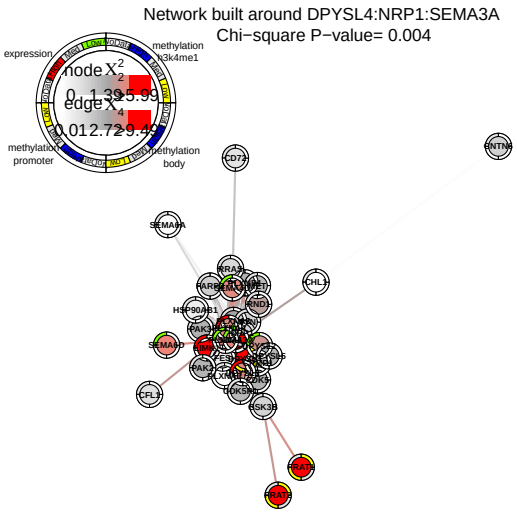



```
## Compare two networks ##
plotModule(test_annotation, which_network = c(4, 6), layout = "fr",
    label_scale = TRUE, compare_plot = TRUE)

## Press key to go to next plot
```




```
## Draw a network with goseq analysis ##
plotModule(test_annotation, which_network = 1, layout = "fr",
    goseq = TRUE, label_scale = FALSE)
```



Num Genes	Enriched Pathway/Term
21	Axon guidance
4	T cell receptor signaling pathway
3	Renal cell carcinoma
5	Regulation of actin cytoskeleton
5	Focal adhesion
3	Adherens junction
3	ErbB signaling pathway
3	Alzheimer's disease