GeneOverlap: An R package to test and visualize gene overlaps

Li Shen
Contact: li.shen@mssm.edu
or shenli.sam@gmail.com
Icahn School of Medicine at Mount Sinai
New York, New York
http://shenlab-sinai.github.io/shenlab-sinai/

October 13, 2014

Contents

1 Data preparation 1

2 Testing overlap between two gene lists 2

3 Visualizing all pairwise overlaps 4

4 Data source and processing 9

5 SessionInfo 10

1 Data preparation

Overlapping gene lists can reveal biological meanings and may lead to novel hypotheses. For example, histone modification is an important cellular mechanism that can pack and re-pack chromatin. By making the chromatin structure more dense or loose, the gene expression can be turned on or off. Tri-methylation on lysine 4 of histone H3 (H3K4me3) is associated with gene activation and its genome-wide enrichment can be mapped by using ChIP-seq experiments. Because of its activating role, if we overlap the genes that are bound by H3K4me3 with the genes that are highly expressed, we should expect a positive association. Similary, we can perform such kind of overlapping between the gene lists of different histone modifications with that of various expression groups and establish each histone modification's role in gene regulation.

Mathematically, the problem can be stated as: given a whole set I of IDs and two sets $A \in I$ and $B \in I$, and $S = A \cap B$, what is the significance of seeing S? This problem can be formulated as a hypergeometric distribution or a contigency table (which can be solved by Fisher's exact test; see *GeneOverlap* documentation). The task is so commonly seen across a biological study that it is worth being made into routines. Therefore, *GeneOverlap* is created to manipulate the data, perform the test and visualize the result of overlaps.

The *GeneOverlap* package is composed of two classes: *GeneOverlap* and *GeneOverlapMatrix*. The *GeneOverlap* class performs the testing and serves as building blocks to the *GeneOverlapMatrix* class. First, let's load the package

```
> library(GeneOverlap)
```

To use the *GeneOverlap* class, create two character vectors that represent gene names. An easy way to do this is to use `read.table("filename.txt")` to read them from text files.

As a convenience, a couple of gene lists have been compiled into the `GeneOverlap` data. Use

```
> data(GeneOverlap)
```

to load them. Now, let's see what are contained in the data: there are three objects. The `hESC.ChIPSeq.list` and `hESC.RNASeq.list` objects contain gene lists from ChIP-seq and RNA-seq experiments. The `gs.RNASeq` variable contains the number of genes in the genomic background. Refer to Section 4 for details about how they were created. Let's see how many genes are there in the gene lists.

```
> sapply(hESC.ChIPSeq.list, length)
```

```
 H3K4me3  H3K9me3 H3K27me3 H3K36me3
   13448      297     3853     4533
```

```
> sapply(hESC.RNASeq.list, length)
```

```
  Exp High Exp Medium    Exp Low
      6540       6011       8684
```

```
> gs.RNASeq
```

```
[1] 21196
```

In *GeneOverlap*, we refer to a collection of gene lists as a gene set that is represented as a named list. Here we can see that the ChIP-seq gene set contains four gene lists of different histone marks: H3K4me3, H3K9me3, H3K27me3 and H3K36me3; the RNA-seq gene set contains three gene lists of different expression levels: High, Medium and Low. Two histone marks are associated with gene activation: H3K4me3 and H3K36me3 while the other two are associated with gene repression: H3K9me3 and H3K27me3.

2 Testing overlap between two gene lists

We want to explore whether the activation mark - H3K4me3 is associated with genes that are highly expressed. First, let's construct a *GeneOverlap* object

```
> go.obj <- newGeneOverlap(hESC.ChIPSeq.list$H3K4me3,
+                          hESC.RNASeq.list$"Exp High",
+                          genome.size=gs.RNASeq)
> go.obj
```

```
GeneOverlap object:
listA size=13448
listB size=6540
Intersection size=5916
Overlap testing has not been performed yet.
```

As we can see, the *GeneOverlap* constructor has already done some basic statistics for us, such as the number of intersections. To test the statistical significance of association, we do

```
> go.obj <- testGeneOverlap(go.obj)
> go.obj
```

```
GeneOverlap object:
listA size=13448
listB size=6540
Intersection size=5916
Overlapping p-value=0e+00
Jaccard Index=0.4
```

The *GeneOverlap* class formulates the problem as testing whether two variables are independent, which can be represented as a contingency table, and then uses Fisher's exact test to find the statistical significance. Here the P-value is zero, which means the overlap is highly significant. The Fisher's exact test also gives an odds ratio which represents the strength of

association. If an odds ratio is equal to or less than 1, there is no association between the two lists. If the odds ratio is much larger than 1, then the association is strong. The class also calculates the Jaccard index which measures the similarity between two lists. The Jaccard index varies between 0 and 1, with 0 meaning there is no similarity between the two and 1 meaning the two are identical. To show some more details, use the `print` function

```
> print(go.obj)
```

```
Detailed information about this GeneOverlap object:
listA size=13448, e.g. ENSG00000187634 ENSG00000188976 ENSG00000187961
listB size=6540, e.g. ENSG00000000003 ENSG00000000419 ENSG00000000460
Intersection size=5916, e.g. ENSG00000188976 ENSG00000187961 ENSG00000187608
Union size=14072, e.g. ENSG00000187634 ENSG00000188976 ENSG00000187961
Genome size=21196
# Contingency Table:
     notA  inA
notB 7124 7532
inB   624 5916
Overlapping p-value=0e+00
Odds ratio=9.0
Overlap tested using Fisher's exact test (alternative=greater)
Jaccard Index=0.4
```

which shows the odds ratio to be 9.0 and the Jaccard index to be 0.4. The `print` function also displays the contingency table which illustrates how the problem is formulated.

Further, we want to see if H3K4me3 is associated with genes that are lowly expressed. We do

```
> go.obj <- newGeneOverlap(hESC.ChIPSeq.list$H3K4me3,
+                          hESC.RNASeq.list$"Exp Low",
+                          genome.size=gs.RNASeq)
> go.obj <- testGeneOverlap(go.obj)
> print(go.obj)
```

```
Detailed information about this GeneOverlap object:
listA size=13448, e.g. ENSG00000187634 ENSG00000188976 ENSG00000187961
listB size=8684, e.g. ENSG00000000005 ENSG00000000938 ENSG00000000971
Intersection size=2583, e.g. ENSG00000187583 ENSG00000187642 ENSG00000215014
Union size=19549, e.g. ENSG00000187634 ENSG00000188976 ENSG00000187961
Genome size=21196
# Contingency Table:
     notA   inA
notB 1647 10865
inB  6101  2583
Overlapping p-value=1
Odds ratio=0.1
Overlap tested using Fisher's exact test (alternative=greater)
Jaccard Index=0.1
```

In contrast to the highly expressed genes, the P-value is now 1 with odds ratio 0.1. Therefore, H3K4me3 is not associated with gene suppression.

Once a `GeneOverlap` object is created, several accessors can be used to extract its slots. For example

```
> head(getIntersection(go.obj))
```

```
[1] "ENSG00000187583" "ENSG00000187642" "ENSG00000215014" "ENSG00000142609"
[5] "ENSG00000203301" "ENSG00000215912"
```

```
> getOddsRatio(go.obj)
```

```
[1] 0.06418943
```

```
> getContbl(go.obj)
```

```
     notA   inA
notB 1647 10865
inB  6101  2583
```

```
> getGenomeSize(go.obj)
```

```
[1] 21196
```

It is also possible to change slots "listA", "listB" and "genome.size" after object creation. For example

```
> setListA(go.obj) <- hESC.ChIPSeq.list$H3K27me3
> setListB(go.obj) <- hESC.RNASeq.list$"Exp Medium"
> go.obj
```

```
GeneOverlap object:
listA size=3853
listB size=6011
Intersection size=1549
Overlap testing has not been performed yet.
```

After any of the above slots is changed, the object is put into untested status. So we need to re-test it

```
> go.obj <- testGeneOverlap(go.obj)
> go.obj
```

```
GeneOverlap object:
listA size=3853
listB size=6011
Intersection size=1549
Overlapping p-value=2.8e-69
Jaccard Index=0.2
```

We can also change the genome size to see how the p-value changes with it

```
> v.gs <- c(12e3, 14e3, 16e3, 18e3, 20e3)
> setNames(sapply(v.gs, function(g) {
+     setGenomeSize(go.obj) <- g
+     go.obj <- testGeneOverlap(go.obj)
+     getPval(go.obj)
+ }), v.gs)
```

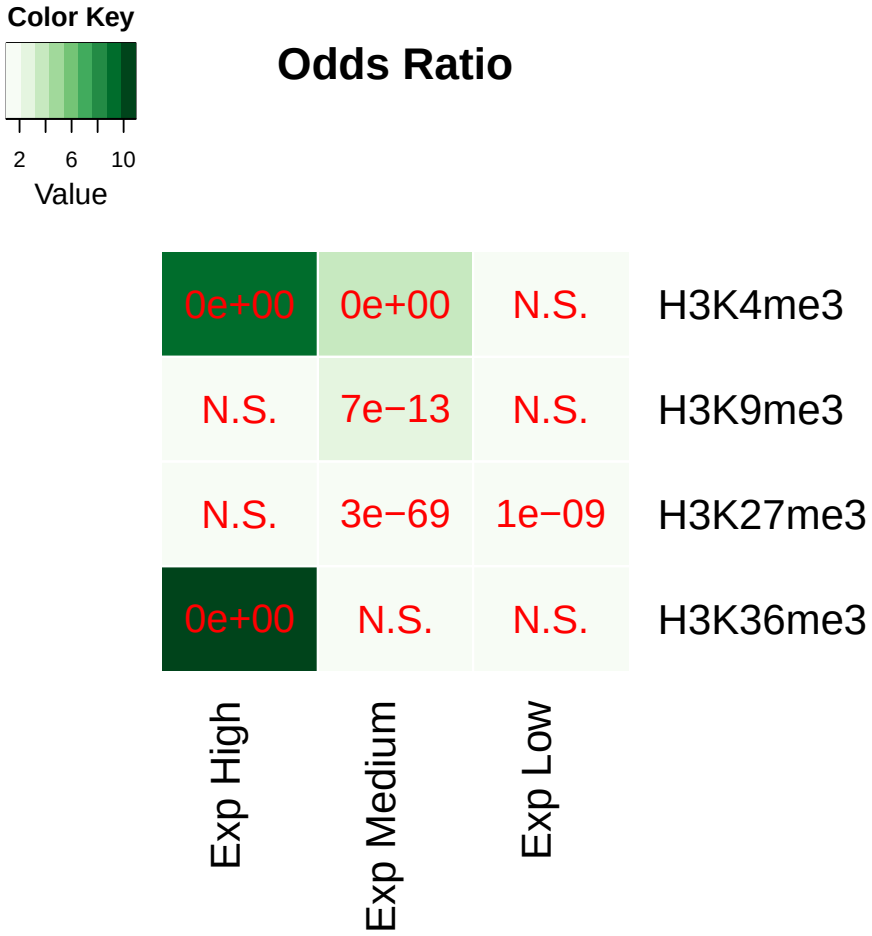
```
       12000        14000        16000        18000        20000
1.000000e+00 9.999746e-01 6.039536e-05 9.006016e-24 5.589067e-51
```

As expected, the larger the genome size, the more significant the P-value. That is because, as the "universe" grows, it becomes less and less likely for two gene lists to overlap by chance.

3 Visualizing all pairwise overlaps

When two gene sets each with one or more lists need to be compared, it would be rather inefficient to compare them manually. A matrix can be constructed, where the rows represent the lists from one set while the columns represent the lists from the other set. Each element of the matrix is a GeneOverlap object. To visualize this overlapping information altogether, a heatmap can be used. To illustrate, we want to compare all the gene lists from ChIP-seq with that from RNA-seq

```
> gom.obj <- newGOM(hESC.ChIPSeq.list, hESC.RNASeq.list,
+                   gs.RNASeq)
> drawHeatmap(gom.obj)
```



That is neat. The `newGOM` constructor creates a new *GeneOverlapMatrix* object using two named lists. Then all we need to do is to use the `drawHeatmap` function to show it. In the heatmap, the colorkey represents the odds ratios and the significant p-values are superimposed on the grids.

To retrieve information from a *GeneOverlapMatrix* object, two important accessors called `getMatrix` and `getNestedList` can be used. The `getMatrix` accessor gets information such as p-values as a matrix, for example

```
> getMatrix(gom.obj, name="pval")
```

```
          Exp High   Exp Medium      Exp Low
H3K4me3  0.0000000 0.000000e+00 1.000000e+00
H3K9me3  0.9996654 7.071538e-13 9.999497e-01
H3K27me3 1.0000000 2.797009e-69 1.061420e-09
H3K36me3 0.0000000 9.999998e-01 1.000000e+00
```

or the odds ratios

```
> getMatrix(gom.obj, "odds.ratio")
```

```
           Exp High Exp Medium    Exp Low
H3K4me3   8.9672775  3.8589147 0.06418943
H3K9me3   0.6366248  2.3460391 0.62254173
H3K27me3  0.3338513  1.9408115 1.24113618
H3K36me3 10.9219512  0.8265542 0.02145576
```

The getNestedList accessor can get gene lists for each comparison as a nested list: the outer list represents the columns and the inner list represents the rows

```
> inter.nl <- getNestedList(gom.obj, name="intersection")
> str(inter.nl)
```

```
List of 3
 $ Exp High  :List of 4
  ..$ H3K4me3 : chr [1:5916] "ENSG00000188976" "ENSG00000187961" "ENSG00000187608" "ENSG00000188157" ...
  ..$ H3K9me3 : chr [1:66] "ENSG00000157184" "ENSG00000117133" "ENSG00000162694" "ENSG00000178104" ...
  ..$ H3K27me3: chr [1:574] "ENSG00000188976" "ENSG00000188157" "ENSG00000162576" "ENSG00000228594" ...
  ..$ H3K36me3: chr [1:3290] "ENSG00000188976" "ENSG00000188157" "ENSG00000160087" "ENSG00000127054" ...
 $ Exp Medium:List of 4
  ..$ H3K4me3 : chr [1:4985] "ENSG00000187634" "ENSG00000188290" "ENSG00000131591" "ENSG00000186891" ...
  ..$ H3K9me3 : chr [1:142] "ENSG00000143786" "ENSG00000197472" "ENSG00000135747" "ENSG00000196418" ...
  ..$ H3K27me3: chr [1:1549] "ENSG00000187634" "ENSG00000188290" "ENSG00000131591" "ENSG00000186891" ...
  ..$ H3K36me3: chr [1:1151] "ENSG00000157933" "ENSG00000116198" "ENSG00000171680" "ENSG00000162413" ...
 $ Exp Low   :List of 4
  ..$ H3K4me3 : chr [1:2583] "ENSG00000187583" "ENSG00000187642" "ENSG00000215014" "ENSG00000142609" ...
  ..$ H3K9me3 : chr [1:90] "ENSG00000215912" "ENSG00000232423" "ENSG00000204501" "ENSG00000157358" ...
  ..$ H3K27me3: chr [1:1745] "ENSG00000187583" "ENSG00000237330" "ENSG00000197921" "ENSG00000142611" ...
  ..$ H3K36me3: chr [1:101] "ENSG00000171819" "ENSG00000116663" "ENSG00000187144" "ENSG00000215902" ...
```

Another important accessor is the method "[" that allows one to retrieve *GeneOverlap* objects in a matrix-like fashion. For example,

```
> go.k4.high <- gom.obj[1, 1]
> go.k4.high
```

```
GeneOverlap object:
listA size=13448
listB size=6540
Intersection size=5916
Overlapping p-value=0e+00
Jaccard Index=0.4
```

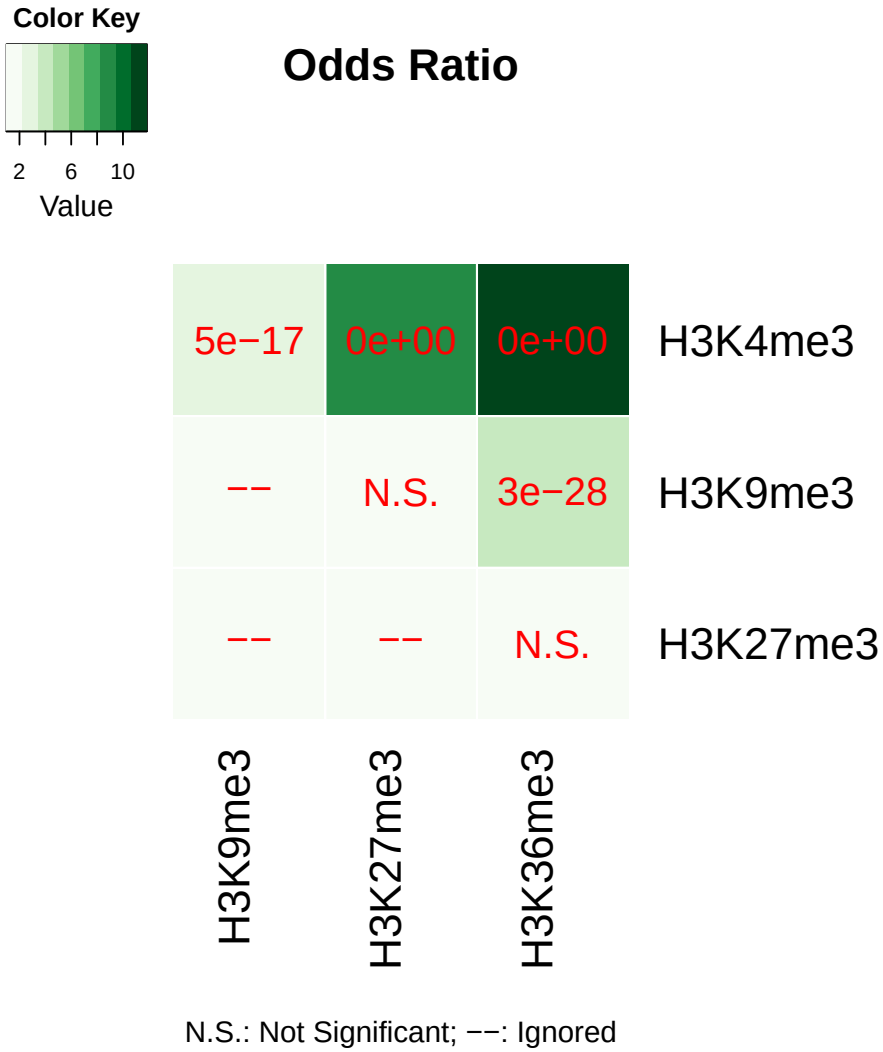
gets the *GeneOverlap* object that represents the comparison between H3K4me3 and highly expressed genes. It is also possible to get *GeneOverlap* objects using labels, such as

```
> gom.obj["H3K9me3", "Exp Medium"]
```

```
GeneOverlap object:
listA size=297
listB size=6011
Intersection size=142
Overlapping p-value=7.1e-13
Jaccard Index=0.0
```

GeneOverlapMatrix can also perform self-comparison on one gene set. For example, if we want to know how the ChIP-seq gene lists associate with each other, we can do

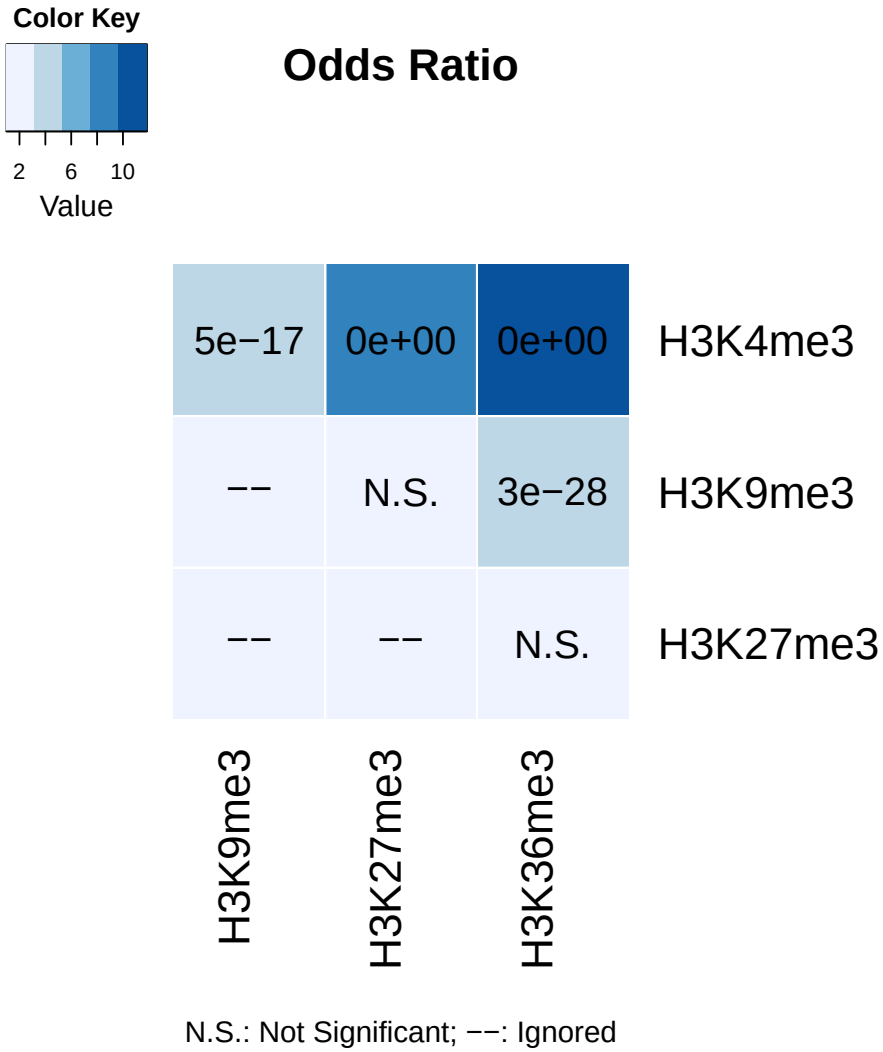
```
> gom.self <- newGOM(hESC.ChIPSeq.list,
+                    genome.size=gs.RNASeq)
> drawHeatmap(gom.self)
```



Only the upper triangular matrix is used.

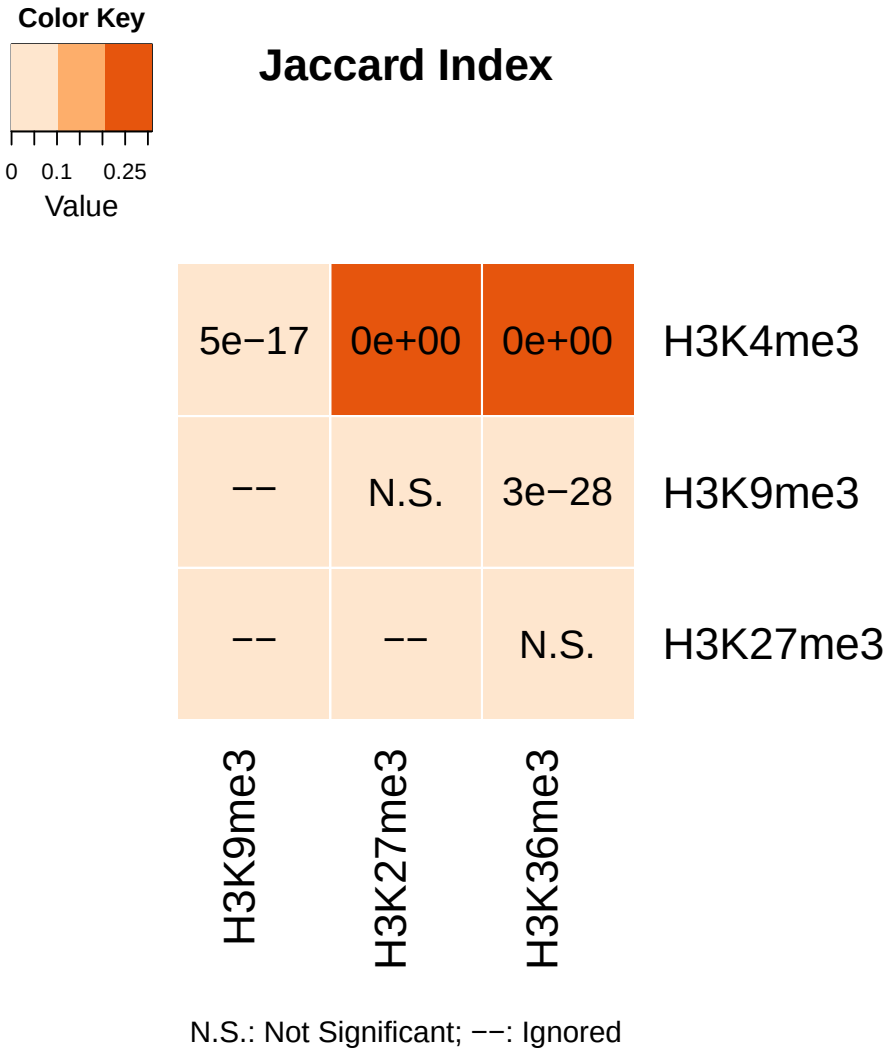
It is also possible to change the number of colors and the colors used for the heatmap. For example

```
> drawHeatmap(gom.self, ncolused=5, grid.col="Blues", note.col="black")
```



or to show the Jaccard index values instead

```
> drawHeatmap(gom.self, what="Jaccard", ncolused=3, grid.col="Oranges",
+             note.col="black")
```



4 Data source and processing

The experimental data used here were downloaded from the ENCODE [ENCODE Consortium,] project's website. Both ChIP-seq and RNA-seq samples were derived from the human embryonic stem cells (hESC). The raw read files were aligned to the reference genome using Bowtie [Langmead et al., 2009] and Tophat [Trapnell et al., 2009].

Cufflinks [Trapnell et al., 2010] was used to estimate the gene expression levels from the RNA-seq data. Only protein coding genes were retained for further analysis. The genes were filtered by FPKM status and only the genes with "OK" status were kept. This left us with 21,196 coding genes whose FPKM values were reliabled estimated. The genes were then separated into three groups: high (FPKM>10), medium (FPKM>1 and <=10) and low (FPKM<=1). The duplicated gene IDs within each group were removed before testing.

For ChIP-seq, one replicate from IP treatment and one replicate from input control were used. Peak calling was performed using MACS2 (v2.0.10) with default parameter values. The peak lists then went through region annotation using the diffReps package [Shen et al., 2013]. After that, the genes that had peaks on genebody and promoter regions were extracted. The genes were further filtered using the RNA-seq gene list obtained above.

5 SessionInfo

```
> sessionInfo()

R Under development (unstable) (2014-10-07 r66723)
Platform: x86_64-unknown-linux-gnu (64-bit)

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C               LC_TIME=en_US.UTF-8
 [4] LC_COLLATE=C               LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C                  LC_ADDRESS=C
[10] LC_TELEPHONE=C             LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] GeneOverlap_1.3.0

loaded via a namespace (and not attached):
[1] BiocStyle_1.5.0    KernSmooth_2.23-13 RColorBrewer_1.0-5 bitops_1.0-6
[5] caTools_1.17.1     gdata_2.13.3       gplots_2.14.2      gtools_3.4.1
[9] tools_3.2.0
```

References

[ENCODE Consortium,] ENCODE Consortium. The encyclopedia of dna elements (http://encodeproject.org/encode/).

[Langmead et al., 2009] Langmead, B., Trapnell, C., Pop, M., and Salzberg, S. (2009). Ultrafast and memory-efficient alignment of short dna sequences to the human genome. *Genome Biology*, 10(3):R25.

[Shen et al., 2013] Shen, L., Shao, N.-Y., Liu, X., Maze, I., Feng, J., and Nestler, E. J. (2013). diffreps: Detecting differential chromatin modification sites from chip-seq data with biological replicates. *PLoS ONE*, 8(6):e65598.

[Trapnell et al., 2009] Trapnell, C., Pachter, L., and Salzberg, S. L. (2009). Tophat: discovering splice junctions with rna-seq. *Bioinformatics*, 25(9):1105–1111.

[Trapnell et al., 2010] Trapnell, C., Williams, B. A., Pertea, G., Mortazavi, A., Kwan, G., van Baren, M. J., Salzberg, S. L., Wold, B. J., and Pachter, L. (2010). Transcript assembly and quantification by rna-seq reveals unannotated transcripts and isoform switching during cell differentiation. *Nat Biotech*, 28(5):511–515. 10.1038/nbt.1621.