# rTRM: an R package for the identification of transcription regulatory modules (TRMs)

Diego Diez

May 19, 2021

## 1 Introduction

Transcription factors (TFs) bind to short motifs in the DNA and regulate the expression of target genes in a cell type and time dependent fashion. TFs do so by cooperating with other TFs in what it is called Transcriptional Regulatory Modules (TRMs). These TRMs contain different TFs and form a combinatorial code that explains TF specificity. We have implemented a method for the identification of TRMs that integrates information about binding locations from a single ChIP-seq experiment, computational estimation of TF binding, gene expression and protein-protein interaction (PPI) data (Diez et al. submitted; see workflow figure). rTRM implements the methods required for the integration of PPI information (step 4 in workflow). To do so, rTRM tries to identify TFs that are bound to a target TF (the one with experimental evidence- i.e. ChIP-seq data) either directly or through a bridge protein. This package has been used to identify cell-type independent and dependent TRMs associated with Stat3 functions [1]. Also, it has been used to identify TRMs in embryonic and hematopoietic stem cells as part of the publication presenting the methodology (Diez et al. submitted). Here we present the basic capabilities of rTRM with a naive example, a case study showing the identification of Sox2 related TRM in ESCs as performed in the paper describing rTRM (Diez et al. submitted), and a complete workflow in R using the PWMEnrich package for the motif enrichment step.

## 2 Minimal example

In this minimal example a dummy network is search to identify TRMs focused around a target node, *N6*, with query nodes being *N7*, *N12* and *N28*. By default *findTRM* find nodes separated a max distance of 0 (i.e. nodes directly connected). We change this with parameter *max.bridge = 1*. Because node *N28* is separated by two other nodes from the target node *N6*, it is not included in the predicted TRM. By default findTRM() returns an object of class igraph, which can be used with plotTRM(), plotTRMlegend() and other rTRM functions. However it is possible to directly obtain a graphNEL object (from the Bioconductor package graph), setting the type argument to "graphNEL". Of course it is possible to also use the igraph.to.graphNEL() function in the igraph package to transform an igraph object into a graphNEL object.

```
> # load the rTRM package
> library(rTRM)
>
> # load network example.
> load(system.file(package = "rTRM", "extra/example.rda"))
>
> # plot network
```

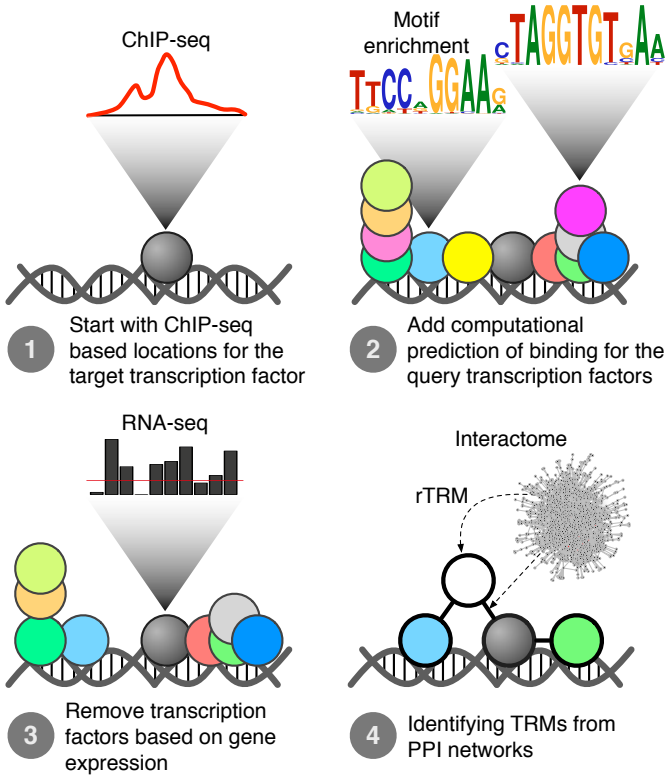


Figure 1: Workflow for the identification of TRMs. Steps 1-3 can be performed with standard Bioconductor approaches. rTRM implements a method to perform step 4.

```
> plot(g, vertex.size = 20, vertex.label.cex = 0.8, layout = layout.graphopt)
>
> # define target and query nodes:
> target = "N6"
> query = c("N7", "N12", "N28")
>
> # find TRM:
> s = findTRM(g, target = target, query = query, method = "nsa", max.bridge = 1)

removing:  4 nodes out of 8 [keeping 4 nodes]

> # annotate nodes:
> V(s)$color = "white"
> V(s)[name %in% query]$color = "steelblue2"
> V(s)[name %in% target]$color = "steelblue4"
>
> # plot:
> plot(s, vertex.size = 20, vertex.label.cex = 0.8)
```

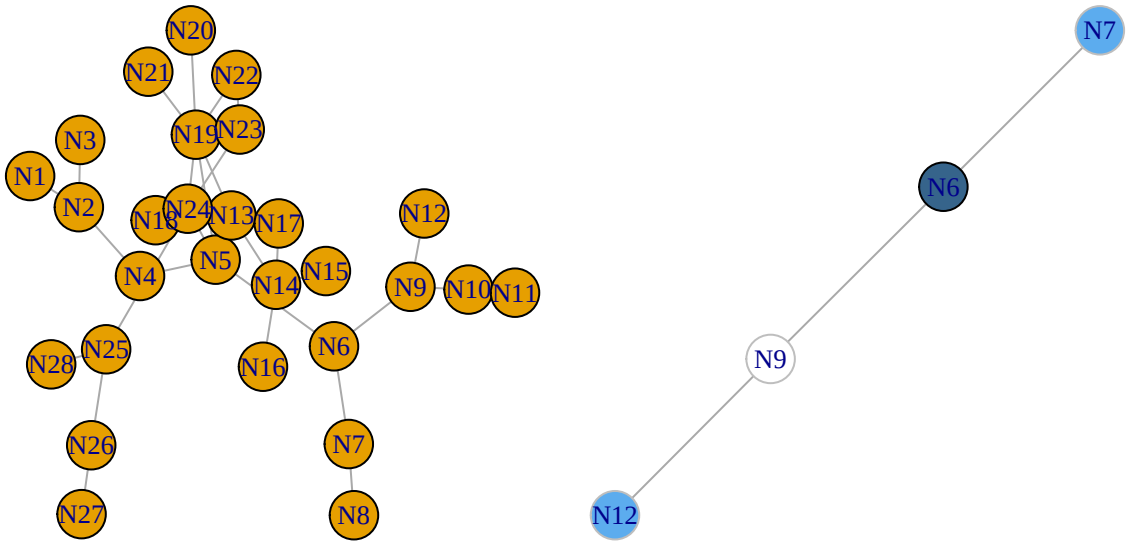


Figure 2: Identification of a TRM 1from a test network (left). In the resulting TRM (right) dark blue indicates the target node light blue are query nodes and white nodes are bridge nodes

# 3 Introduction to the rTRM package

rTRM relies on a series of optimizations. For example in the publication we used PWMs for vertebrate species compiled from different sources. This assumes the binding specificities of TFs will be conserved on all these species. Recent comparison between mouse and human PWMs suggests that this is the case for most TFs [9]. rTRM also relies on protein-protein interaction data, and so provides utilities to download data from the BioGRID database (see below). As some of these functionalities are further integrated with existing Bioconductor functionality they may be defunct in the future.

## 3.1 Database

Information about TFs, including Position Specific Weight (PWMs) matrices, mapping to Entrez Gene identifiers, orthologs in mouse and human and other annotations are stored as a SQLite database. rTRM provides a basic API for accessing the data. Below there are some examples.

To obtain PWMs:

```
> pwm = getMatrices()
> head(pwm, 1)
$MA0009.1
     1     2 3 4 5 6 7    8     9 10    11
A 0.05 0.025 1 0 0 0 0 0.00 0.025  1 0.775
C 0.70 0.025 0 0 0 0 0 0.05 0.175  0 0.125
G 0.20 0.000 0 1 1 0 1 0.00 0.700  0 0.000
T 0.05 0.950 0 0 0 1 0 0.95 0.100  0 0.100
```

To get annotations:

```
> ann = getAnnotations()
> head(ann)
```

```
  row_names   pwm_id   symbol                    family domain binding source
1         1 MA0009.1        T                         T        monomer jaspar
2         2 MA0059.1 MYC::MAX          Helix-Loop-Helix          dimer jaspar
3         3 MA0146.1      Zfx BetaBetaAlpha-zinc finger        monomer jaspar
4         4 MA0132.1     Pdx1                      Homeo        monomer jaspar
5         5 MA0162.1     Egr1 BetaBetaAlpha-zinc finger        monomer jaspar
6         6 MA0093.1     USF1          Helix-Loop-Helix        monomer jaspar
  note
1 2010
2 2010
3 2010
4 2010
5 2010
6 2010
```

To get map of TFs to genes:

```
> map = getMaps()
> head(map)
  row_names   pwm_id entrezgene organism
1         1 MA0009.1      20997    mouse
2         2 MA0059.1       4609    human
3         3 MA0059.1       4149    human
4         4 MA0146.1      22764    mouse
5         5 MA0132.1      18609    mouse
6         6 MA0162.1      13653    mouse
```

To get map of TFs to ortholog genes:

```
> o = getOrthologs(organism = "mouse")
> head(o)
  row_names entrezgene organism map_entrezgene map_organism
1       775      20997    mouse          20997        mouse
2       776       4609    human         107771        mouse
3       777       4149    human          17187        mouse
4       778      22764    mouse          22764        mouse
5       779      18609    mouse          18609        mouse
6       780      13653    mouse          13653        mouse
```

It is possible to map motif ids to entrezgene ids in the target organism (only between human and mouse). This is useful when all the information about existing PWMs is desired, as some TF binding affinities have only been studied in one organism.

```
> getOrthologFromMatrix("MA0009.1", organism = "human")
[1] "6862"
> getOrthologFromMatrix("MA0009.1", organism = "mouse")
[1] "20997"
```

## 3.2 Interactome data

rTRM requires information about protein-protein interactions (PPIs) for its predictions and includes interactome (PPI network) data from the BioGRID database [2]. Currently mouse and human interactomes are supported. The networks are provided as *igraph* format. To access the data use:

```
> # check statistics about the network.
> biogrid_mm()

Mouse PPI network data of class igraph
Number of nodes:  5315
Number of edges:  11695
Source:  The BioGRID (http://www.thebiogrid.org)
Release:  3.4.128
Downloaded:  2015-09-17
Use data(biogrid_mm) to load it

> # load mouse PPI network:
> data(biogrid_mm)
```

The amount of available PPI data increases rapidly so it is desirable to have a way to access the newest data conveniently. rTRM includes support for direct download and processing of PPI data from the BioGRID database. The PPI network is stored as an *igraph* object that can be readily used with rTRM or stored for later use. Below there is an example of the BioGRID database update procedure.

```
> # obtain dataset.
> db = getBiogridData()  # retrieves latest release.
> # db = getBiogridData('3.2.96') # to get a specific release.
>
> # check release:
> db$release
> db$data
>
> # process PPI data for different organisms (currently supported human and mouse):
> biogrid_hs = processBiogrid(db, org = "human")
> biogrid_mm = processBiogrid(db, org = "mouse")
```

PPI data from other databases could be used as long as it is formatted as an *igraph* object with the 'name' attribute containing entrezgene identifiers and the 'label' attribute containing the symbol.

### 3.2.1 Using PSICQUIC package to obtain protein-protein interactions

One possibility available from Bioconductor 2.13 is to use the package PSIQUIC to obtain PPI data. PSICQUIC provides access to different databases of PPIs, including BioGRID and STRINGS, and databases of cellular networks like KEGG or Reactome. For exaple, to obtain the human BioGRID data (note is named BioGrid at PSICQUIC):

```
> library(PSICQUIC)
> psicquic <- PSICQUIC()
> providers(psicquic)
>
> # obtain BioGrid human PPIs (as data.frame):
> tbl = interactions(psicquic, species = "9606", provider = "BioGrid")
>
> # the target and source node information needs to be polished (i.e. must be
> # Entrez gene id only)
> biogrid_hs = data.frame(source = tbl$A, target = tbl$B)
> biogrid_hs$source = sub(".*locuslink:(.*)\\|BIOGRID:.*", "\\1", biogrid_hs$source)
```

```
> biogrid_hs$target = sub(".*locuslink:(.*)\\|BIOGRID:.*", "\\1", biogrid_hs$target)
>
> # create graph.
> library(igraph)
> biogrid_hs = graph.data.frame(biogrid_hs, directed = FALSE)
> biogrid_hs = simplify(biogrid_hs)
>
> # annotate with symbols.
> library(org.Hs.eg.db)
> V(biogrid_hs)$label = select(org.Hs.eg.db, keys = V(biogrid_hs)$name, columns = c("SYMBOL"))$SYMBOL
```

Facilities will be added to automatically process the data from PSICQUIC to produce igraph objects suitable for rTRM.

# 4 Case study: TRM associated with Sox2 in embryonic stem cells (ESCs)

Sox2 is a TF involved in the determination and maintainance of pluripotency in embryonic stem cells (ESCs). Sox2 forms a transcriptional regulatory module with Nanog and Pou5f1 (Oct4), and together determine ESCs phenotype. Other TFs important to this process are Erssb and Klf4. In this case study we want to identify TRMs associated with Sox2. ChIP-seq data for Sox2 was obtained from Chen et al. (Cell 2008) [3] and motif enrichment analysis performed with HOMER [4], followed by matching against our library of PWMs using TOMTOM [5]. The starting dataset is the TOMTOM output file with the motifs enriched in the Sox2 binding regions.

```
> # read motif enrichment results.
> motif_file = system.file("extra/sox2_motif_list.rda", package = "rTRM")
> load(motif_file)
> length(motif_list)
[1] 177
> head(motif_list)
[1] "MA0039.2" "MA0071.1" "MA0075.1" "MA0077.1" "MA0078.1" "MA0112.2"
```

First, we read the motifs and convert them into gene identifiers (i.e. Entrez Gene identifier). To do this we use the function **getOrthologFromMatrix**, which takes a list of motif identifiers and the target organism as parameters. The function returns a list with the Entrez Gene ids.

```
> # get the corresponding gene.
> tfs_list = getOrthologFromMatrix(motif_list, organism = "mouse")
> tfs_list = unique(unlist(tfs_list, use.names = FALSE))
> length(tfs_list)
[1] 98
> head(tfs_list)
[1] "18609" "20682" "13983" "20665" "18291" "18227"
```

Next, we need a list of genes expressed in ESC. For this, the dataset was obtained from GEO (GSE27708; [6]) and processed using the custom CDFs from the BrainArray project [7] and the **rma** function from the package *affy* [8]. Genes expressed were determined by removing all genes with log2 expression ¡ 5 in all samples.

```
> # load expression data.
> eg_esc_file = system.file("extra/ESC-expressed.txt", package = "rTRM")
> eg_esc = scan(eg_esc_file, what = "")
> length(eg_esc)
[1] 8734
> head(eg_esc)
[1] "100008567" "100017"    "100019"    "100037258" "100038489" "100039781"
> tfs_list_esc = tfs_list[tfs_list %in% eg_esc]
> length(tfs_list_esc)
[1] 22
> head(tfs_list_esc)
[1] "26380" "18999" "20674" "16600" "26379" "13984"
```

Next, we load the PPI network and filter out potential degree outliers and proteins not expressed in the paired expression data.

```
> # load and process PPI data.
> biogrid_mm()

Mouse PPI network data of class igraph
Number of nodes:  5315
Number of edges:  11695
Source:  The BioGRID (http://www.thebiogrid.org)
Release:  3.4.128
Downloaded:  2015-09-17
Use data(biogrid_mm) to load it

> data(biogrid_mm)
> ppi = biogrid_mm
> vcount(ppi)
[1] 5315
> ecount(ppi)
[1] 11695
> # remove outliers.
> f = c("Ubc", "Sumo1", "Sumo2", "Sumo3")
> f = select(org.Mm.eg.db, keys = f, columns = "ENTREZID", keytype = "SYMBOL")$ENTREZID

'select()' returned 1:1 mapping between keys and columns

> f
[1] "22190"  "22218"  "170930" "20610"
> ppi = removeVertices(ppi, f)
> vcount(ppi)
[1] 4984
> ecount(ppi)
[1] 11081
> # filter by expression.
> ppi_esc = induced.subgraph(ppi, V(ppi)[name %in% eg_esc])
> vcount(ppi_esc)
[1] 3109
> ecount(ppi_esc)
[1] 4889
> # ensure a single component.
```

```
> ppi_esc = getLargestComp(ppi_esc)
> vcount(ppi_esc)
[1] 2576
> ecount(ppi_esc)
[1] 4851
```

To identify TRMs we define a target TF (the one the ChIP-seq data comes from) and some query TFs (the ones with enriched binding sites in the neighborhood of the target TF).

```
> # define target.
> target = select(org.Mm.eg.db, keys = "Sox2", columns = "ENTREZID", keytype = "SYMBOL")$ENTREZID

'select()' returned 1:1 mapping between keys and columns

> target
[1] "20674"
> # find TRM.
> s = findTRM(ppi_esc, target, tfs_list_esc, method = "nsa", max.bridge = 1)

11 query nodes NOT FOUND in network-- removed
removing:  320 nodes out of 328 [keeping 8 nodes]

> vcount(s)
[1] 8
> ecount(s)
[1] 15
```

Finally, we layout the network using a customized concentric layout and plot the network and the legend.

```
> # generate layout (order by cluster, then label)
> cl = getConcentricList(s, target, tfs_list_esc)
> l = layout.concentric(s, cl, order = "label")
>
> # plot TRM.
> plotTRM(s, layout = l, vertex.cex = 15, label.cex = 0.8)

Warning in if (class(layout) == "function") l = layout(g) else l = layout:  the condition
has length > 1 and only the first element will be used

> plotTRMlegend(s, title = "ESC Sox2 TRM", cex = 0.8)
```

# 5 A complete workflow in R

In this section we will identify Sox2 TRMs using a workflow performed completely in R. For this the MotifDb package will be used to obtain the information about PWMs, and PWMEnrich package for identifying enriched motifs. PWMEnrich requires the computation of background models and the enrichment analysis *per se*, which are computational intensive. Therefore these steps were not run during the compilation of this vignette.

The first step is to retrieve a set of PWMs. Here we will use the MotifDb package available in Bioconductor. We will use only mouse PWMs (i.e. PWMs for the target organism). It could be possible to use matrices from other species but then the user has to obtain the orthologs in the target organism (e.g. using getOrthologsFromBiomart() in rTRM or using the Biomart package directly).

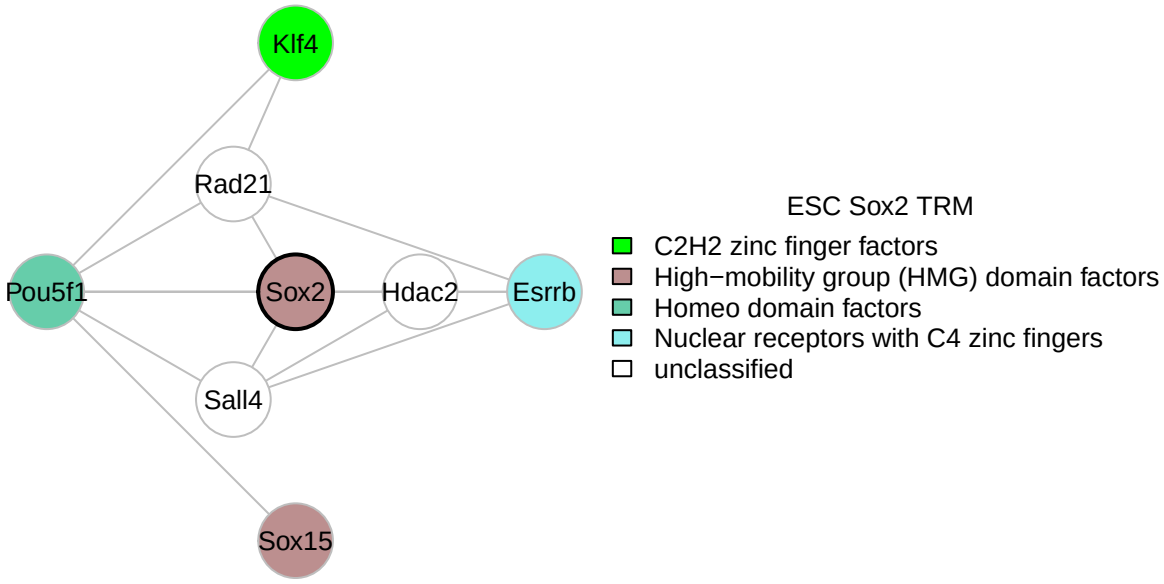


Figure 3: Sox2 specific TRM in ESCs.

```
> library(rTRM)
> library(BSgenome.Mmusculus.UCSC.mm8.masked)  # Sox2 peaks found against mm8

Loading required package:  BSgenome
Loading required package:  GenomeInfoDb
Loading required package:  GenomicRanges
Loading required package:  Biostrings
Loading required package:  XVector

Attaching package:  'Biostrings'
The following object is masked from 'package:base':
      strsplit
Loading required package:  rtracklayer

Attaching package:  'rtracklayer'
The following object is masked from 'package:igraph':
      blocks
Loading required package:  BSgenome.Mmusculus.UCSC.mm8

> library(PWMEnrich)

Loading required package:  grid

Attaching package:  'grid'
The following object is masked from 'package:Biostrings':
      pattern
Warning:  replacing previous import 'Biostrings::pattern' by 'grid::pattern' when loading
'PWMEnrich'

> registerCoresPWMEnrich(1)  # register number of cores for parallelization in PWMEnrich
> library(MotifDb)
```

```
See system.file("LICENSE", package="MotifDb") for use restrictions.

> # select mouse PWMs:
> sel.mm = values(MotifDb)$organism %in% c("Mmusculus")
> pwm.mm = MotifDb[sel.mm]
```

The matrices need to be passed as counts, that is the PFM need to be converted to counts. The easiest way is to multiply by 100 and round the results. We also need to convert it to integer.

```
> # generate logn background model of PWMs:
> p = as.list(pwm.mm)
> p = lapply(p, function(x) round(x * 100))
> p = lapply(p, function(x) t(apply(x, 1, as.integer)))
```

With the PFMs we compute the background model using the makeBackground() function from the PWMEnrich package, which returns the corresponding PWMs. This requires a list with the PFMs as counts, the organisms to obtain the sequences to compute the background and the type of background model (here "logn" model is used).

```
> pwm_logn = makeBackground(p, Mmusculus, type = "logn")
```

Next we read the peak information from the Sox2 Chip-seq data. This is the original coordinates obtained from Chen et al. (Cell 2008), which were obtained for Mmusculus mm8 genome. The function getSequencesFromGenome() is an utility wrapper to getSeq() that facilitates appending a label to the sequences' ids. PWMEnrich requires sequences the same size or longer to the motifs so we check what is the largest motif and filter the sequences accordingly.

```
> sox2_bed = read.table(system.file("extra/ESC_Sox2_peaks.txt", package = "rTRM"))
>
> colnames(sox2_bed) = c("chr", "start", "end")
>
> sox2_seq = getSequencesFromGenome(sox2_bed, Mmusculus, append.id = "Sox2")
> # PWMEnrich throws an error if the sequences are shorter than the motifs so we
> # filter those sequences.
> min.width = max(sapply(p, ncol))
> sox2_seq_filter = sox2_seq[width(sox2_seq) >= min.width]
```

Next, enrichment is computed with the sequences and the PWMs with the background model as parameters.

```
> # find enrichment:
> sox2_enr = motifEnrichment(sox2_seq_filter, pwms = pwm_logn, group.only = TRUE)
```

Next, retrieve the enriched motifs by choosing an appropriate cutoff. Here a raw.score of ¿ 5 is used. Then, using the annotations in the MotifDb dataset, we can obtain the Entrezgene ids associated with the enriched TF motifs.

```
> res = groupReport(sox2_enr)
>
> plot(density(res$raw.score), main = "", log = "x", xlab = "log(raw.score)")

Warning in xy.coords(x, y, xlabel, ylabel, log):  2 x values <= 0 omitted from logarithmic
plot
```

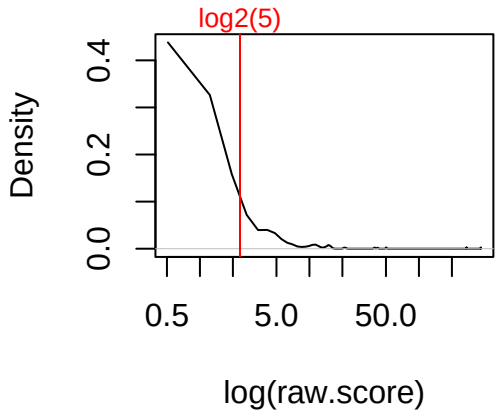


Figure 4: Density of log2(raw.score) for group. The selected cutoff is indicated with a red line.

```
> abline(v = log2(5), col = "red")
> mtext(text = "log2(5)", at = log2(5), side = 3, cex = 0.8, col = "red")
>
> res.gene = unique(values(MotifDb[res$id[res$raw.score > 5]])$geneId)
> res.gene = unique(na.omit(res.gene))
```

Then proceed with the same steps as in the Use Case example shown in the previous section. The resulting TRM is similar ( 85% of edges shared) to the one in the Use Case, which used HOMER for motif enrichment. Differences may be to different approaches to determine the background. HOMER uses random sets of sequences with similar composition to the ChIP-seq peaks provided to generate the background. For PWMEnrich we generated a background using promoter sequences, defined as 2000 bp upstream of the transcription start site (TSS) of all genes in the genome. Generally, using different strategies for enrichment will tend to produce slightly different TRMs.

```
> data(biogrid_mm)
> ppi = biogrid_mm
> vcount(ppi)
[1] 5315
> ecount(ppi)
[1] 11695
> f = c("Ubc", "Sumo1", "Sumo2", "Sumo3")
> f = select(org.Mm.eg.db, keys = f, columns = "ENTREZID", keytype = "SYMBOL")$ENTREZID

'select()' returned 1:1 mapping between keys and columns

> ppi = removeVertices(ppi, f)
> vcount(ppi)
[1] 4984
> ecount(ppi)
[1] 11081
```

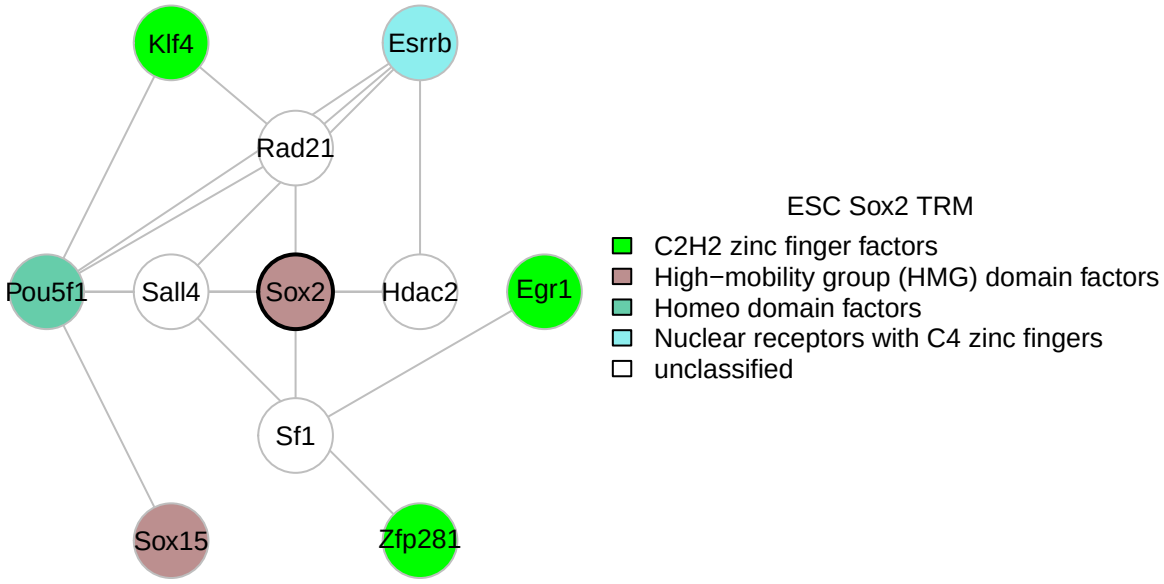


Figure 5: Sox2 TRM identified using PWMEnrich for the motif enrichment.

```
> # filter by expression.
> eg_esc = scan(system.file("extra/ESC-expressed.txt", package = "rTRM"), what = "")
> ppi_esc = induced.subgraph(ppi, V(ppi)[name %in% eg_esc])
> vcount(ppi_esc)
[1] 3109
> ecount(ppi_esc)
[1] 4889
> # ensure a single component.
> ppi_esc = getLargestComp(ppi_esc)
> vcount(ppi_esc)
[1] 2576
> ecount(ppi_esc)
[1] 4851
> sox2.gene = select(org.Mm.eg.db, keys = "Sox2", columns = "ENTREZID", keytype = "SYMBOL")$ENTREZID

'select()' returned 1:1 mapping between keys and columns

> sox2_trm = findTRM(ppi_esc, target = sox2.gene, query = res.gene)

18 query nodes NOT FOUND in network-- removed
removing:  346 nodes out of 357 [keeping 11 nodes]

> cl = getConcentricList(sox2_trm, t = sox2.gene, e = res.gene)
> l = layout.concentric(sox2_trm, concentric = cl, order = "label")
> plotTRM(sox2_trm, layout = l, vertex.cex = 15, label.cex = 0.8)

Warning in if (class(layout) == "function") l = layout(g) else l = layout:  the condition
has length > 1 and only the first element will be used

> plotTRMlegend(sox2_trm, title = "ESC Sox2 TRM", cex = 0.8)
```

We next compare the similarity between the TRM identified using motifs enriched as identified

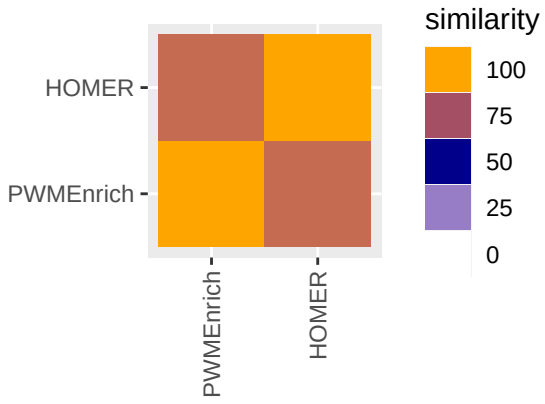


Figure 6: Similarity between the TRMs predicted using motif enrichment information from PWMEnrich and HOMER.

with HOMER and those identified with PWMEnrich. As shown in the heatmap, both methods return similar results.

```
> m = getSimilarityMatrix(list(PWMEnrich = sox2_trm, HOMER = s))
> m
          PWMEnrich     HOMER
PWMEnrich 100.00000  83.33333
HOMER      83.33333 100.00000
> d = as.data.frame.table(m)
> g = ggplot(d, aes(x = Var1, y = Var2, fill = Freq))
> g + geom_tile() + scale_fill_gradient2(limit = c(0, 100), low = "white", mid = "darkblue",
+     high = "orange", guide = guide_legend("similarity", reverse = TRUE), midpoint = 50) +
+     xlab(NULL) + ylab(NULL) + theme(aspect.ratio = 1, axis.text.x = element_text(angle = 90,
+     vjust = 0.5, hjust = 1))
```

# 6 Ploting parameters

The most important parameter determining the appearance of your network will be the layout. When networks contain many nodes and edges are difficult to interpret. rTRM implements two igraph layouts that try improve the visualization and interpretation of the identified TRMs. The layout *layout.concentric* is a circular layout with multiple concentric layers that places the target TFs in the center, the enriched (or query) TFs in the outer circle and the bridge TFs in the middle circle. Another layut is *layout.arc* that tries to mimic the layout presented in the rTRM description (Fig. 1). In this case all nodes are plotted in a liner layout, with the targets in the center, and the enriched (query) nodes at each side. Those enriched nodes connected directly to any of the target nodes are placed in the left side. Those connected through a bridge node are placed in the right side, with the bridge node placed in between. The following figure compares the concentric layout obtained in the previous section with a layout using the *layout.arc* function.

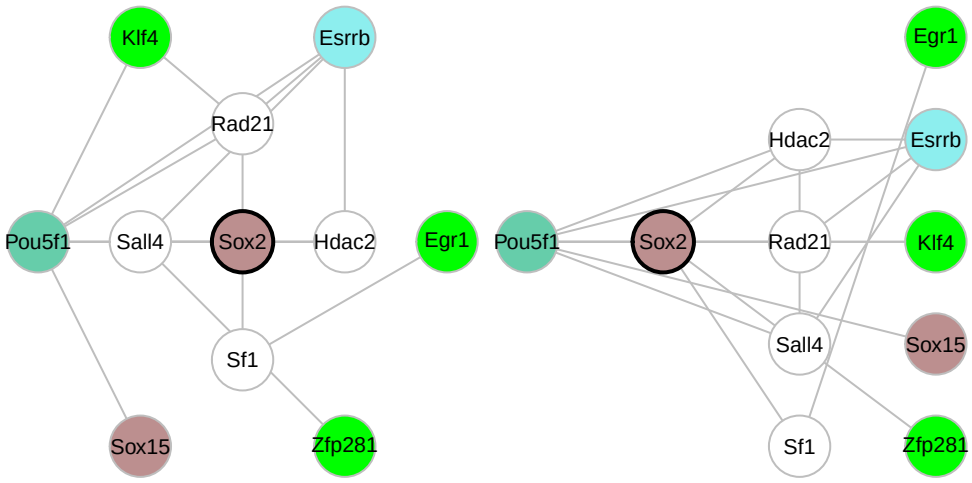


Figure 7: Sox2 TRM obtained with PWMEnrich workflow and layout.concentric is shown in the left. Same TRM with layout.arc is shown in the right.

```
> plotTRM(sox2_trm, layout = l, vertex.cex = 15, label.cex = 0.7)

Warning in if (class(layout) == "function") l = layout(g) else l = layout:  the condition
has length > 1 and only the first element will be used

> l = layout.arc(sox2_trm, target = sox2.gene, query = res.gene)
> plotTRM(sox2_trm, layout = l, vertex.cex = 15, label.cex = 0.7)

Warning in if (class(layout) == "function") l = layout(g) else l = layout:  the condition
has length > 1 and only the first element will be used
```

# 7 Citation

If you use *rTRM* in your research please include the following reference:

```
> citation(package = "rTRM")

To cite rTRM in publications use:

  Diego Diez, Andrew P. Hutchins and Diego Miranda-Saavedra. Systematic
  identification of transcriptional regulatory modules from
  protein-protein interaction networks. Nucleic Acids Research, 2014.
  42 (1) e6.

A BibTeX entry for LaTeX users is

  @Article{,
    title = {Systematic identification of transcriptional regulatory modules from protein-protein int
    author = {Diego Diez and Andrew P. Hutchins and Diego Miranda-Saavedra},
    year = {2014},
    journal = {Nucleic Acids Research},
```

```
    volume = {42},
    number = {1},
    pages = {e6-e6},
    doi = {10.1093/nar/gkt913},
  }
```

# 8 Session Information

```
> sessionInfo()
R version 4.1.0 beta (2021-05-03 r80259)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 20.04.2 LTS

Matrix products: default
BLAS:   /home/biocbuild/bbs-3.14-bioc/R/lib/libRblas.so
LAPACK: /home/biocbuild/bbs-3.14-bioc/R/lib/libRlapack.so

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_GB              LC_COLLATE=C
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
 [1] grid      parallel  stats4    stats     graphics  grDevices utils
 [8] datasets  methods   base

other attached packages:
 [1] MotifDb_1.35.0
 [2] PWMEnrich_4.29.0
 [3] BSgenome.Mmusculus.UCSC.mm8.masked_1.3.99
 [4] BSgenome.Mmusculus.UCSC.mm8_1.4.0
 [5] BSgenome_1.61.0
 [6] rtracklayer_1.53.0
 [7] Biostrings_2.61.0
 [8] XVector_0.33.0
 [9] GenomicRanges_1.45.0
[10] GenomeInfoDb_1.29.0
[11] org.Mm.eg.db_3.13.0
[12] AnnotationDbi_1.55.0
[13] IRanges_2.27.0
[14] S4Vectors_0.31.0
[15] Biobase_2.53.0
[16] BiocGenerics_0.39.0
[17] rTRM_1.31.0
[18] igraph_1.2.6
[19] knitr_1.33
```

```
[20] ggplot2_3.3.3

loaded via a namespace (and not attached):
 [1] MatrixGenerics_1.5.0        httr_1.4.2
 [3] bit64_4.0.5                 gtools_3.8.2
 [5] assertthat_0.2.1            highr_0.9
 [7] blob_1.2.1                  GenomeInfoDbData_1.2.6
 [9] Rsamtools_2.9.0             yaml_2.2.1
[11] pillar_1.6.1                RSQLite_2.2.7
[13] lattice_0.20-44             glue_1.4.2
[15] digest_0.6.27               colorspace_2.0-1
[17] Matrix_1.3-3                XML_3.99-0.6
[19] pkgconfig_2.0.3             zlibbioc_1.39.0
[21] purrr_0.3.4                 scales_1.1.1
[23] gdata_2.18.0                BiocParallel_1.27.0
[25] tibble_3.1.2                KEGGREST_1.33.0
[27] farver_2.1.0                generics_0.1.0
[29] ellipsis_0.3.2              cachem_1.0.5
[31] withr_2.4.2                 SummarizedExperiment_1.23.0
[33] splitstackshape_1.4.8       magrittr_2.0.1
[35] crayon_1.4.1                memoise_2.0.0
[37] evaluate_0.14               fansi_0.4.2
[39] data.table_1.14.0           tools_4.1.0
[41] BiocIO_1.3.0                formatR_1.9
[43] lifecycle_1.0.0             matrixStats_0.58.0
[45] stringr_1.4.0               munsell_0.5.0
[47] DelayedArray_0.19.0         compiler_4.1.0
[49] evd_2.3-3                   rlang_0.4.11
[51] RCurl_1.98-1.3              rjson_0.2.20
[53] labeling_0.4.2              bitops_1.0-7
[55] restfulr_0.0.13             gtable_0.3.0
[57] DBI_1.1.1                   R6_2.5.0
[59] GenomicAlignments_1.29.0    seqLogo_1.59.0
[61] dplyr_1.0.6                 fastmap_1.1.0
[63] bit_4.0.4                   utf8_1.2.1
[65] stringi_1.6.2               Rcpp_1.0.6
[67] vctrs_0.3.8                 png_0.1-7
[69] tidyselect_1.1.1            xfun_0.23
```

# References


[1] Hutchins, A. P., D. Diez, et al. (2013). "Distinct transcriptional regulatory modules underlie STAT3's cell type-independent and cell type-specific functions." Nucleic Acids Res.

[2] Stark, C., B. J. Breitkreutz, et al. (2011). "The BioGRID Interaction Database: 2011 update." Nucleic Acids Res 39(Database issue): D698-704.

[3] Chen, X., H. Xu, et al. (2008). "Integration of external signaling pathways with the core transcriptional network in embryonic stem cells." Cell 133(6): 1106-1117.

[4] Heinz, S., C. Benner, et al. (2010). "Simple combinations of lineage-determining transcription factors prime cis-regulatory elements required for macrophage and B cell identities." Mol Cell 38(4): 576-589.

[5] Gupta, S., J. A. Stamatoyannopoulos, et al. (2007). "Quantifying similarity between motifs." Genome Biol 8(2): R24.

[6] Ho, L., E. L. Miller, et al. (2011). "esBAF facilitates pluripotency by conditioning the genome for LIF/STAT3 signalling and by regulating polycomb function." Nat Cell Biol 13(8): 903-913.

[7] Dai, M., P. Wang, et al. (2005). "Evolving gene/transcript definitions significantly alter the interpretation of GeneChip data." Nucleic Acids Res 33(20): e175.

[8] Gautier, L., L. Cope, et al. (2004). "affy–analysis of Affymetrix GeneChip data at the probe level." Bioinformatics 20(3): 307-315.

[9] Jolma, A., Yan, J., Whitington, T., Toivonen, J., Nitta, K. R., Rastas, P., et al. (2013). DNA-Binding Specificities of Human Transcription Factors. Cell, 152(1-2), 327–339.