enrichmotifpairR: An R package for finding enriched transcription factor motifs and their binding partners

Naresh Doni Jayavelu and R. David Hawkins

01/30/2022

Contents

Quick start

```
# load the package and other useful packages
library(enrichmotifpairR)
# load the example data provided with package
data("example_peaks_data")
```

Finding enriched TF motifs and their binding partners in H1-ESC at DHS peaks

```
# Finding the enriched motifs and their partners
results <- findEnrichMotifPair(
  target_data = example_peaks_data$`H1-ESC_DHS_peaks`,
  background_data = example_peaks_data$`H1-ESC_DHS_peaks_matched_background`,
  genome_ver = "hg38",
  scramble_data = FALSE,
  motif_database = "ENCODE",
  Pvalue_computation = "hyper",
  Pvalue_threshold = 0.01,
  Pvalue_adjust_method = "BH"
)
```

The enriched TF motifs are stored in the `results$motif_enrich` and their binding partners in the `results$motif_pair_enrich`.

```
# assign the results data to individual objects
enrich_motifs <- results$motif_enrich
enrich_motifs[enrich_motifs < 1e-100] <- 1e-100
enrich_motifs <- enrich_motifs %>%
    mutate(`-log10(pval_adj)` = -log10(pval_adj))
# The output of the enriched motifs top five
enrich_motifs %>% head(5) %>% arrange(desc(fold_enrich)) %>%
    select(motif_name, TF_name, tg_motif_count, bg_motif_count,
```

```
        fold_enrich, `-log10(pval_adj)`) %>%
    kable(., caption="Top 5 enriched motifs")
```

Table 1: Top 5 enriched motifs

motif_name	TF_name	tg_motif_count	bg_motif_count	fold_enrich	-log10(pval_adj)
CTCF_disc1	CTCF	2837	212	13.382075	100
SP1_disc1	SP1	1716	151	11.364238	100
NFY_disc1	NFY	1821	165	11.036364	100
RAD21_disc1	RAD21	2792	264	10.575758	100
IRF_disc1	IRF	1664	165	10.084848	100

```
enrich_motif_pairs <- results$motif_pair_enrich
enrich_motif_pairs[enrich_motif_pairs < 1e-100] <- 1e-100
enrich_motif_pairs <- enrich_motif_pairs %>%
    mutate(`-log10(pval_adj)` = -log10(pval_adj))

# The output of the enriched motif pairs, top five binding partners for **NFY**
enrich_motif_pairs %>% head(5) %>% arrange(desc(fold_enrich)) %>%
    select(motif_name_1, TF_name_1, motif_name_2, TF_name_2,
           fold_enrich, `-log10(pval_adj)`) %>%
    kable(., caption="Top 5 binding partners for NFY")
```

Table 2: Top 5 binding partners for NFY

motif_name_1	TF_name_1	motif_name_2	TF_name_2	fold_enrich	-log10(pval_adj)
NFY_disc1	NFY	SP1_known4	SP1	20.8401977	14.108859
NFY_disc1	NFY	IRF_disc4	IRF	12.0737232	15.074253
NFY_disc1	NFY	SP1_disc1	SP1	2.4215848	25.230667
NFY_disc1	NFY	NFY_known1	NFY	2.3076519	17.025177
NFY_disc1	NFY	RFX5_disc2	RFX5	1.8574959	18.174528

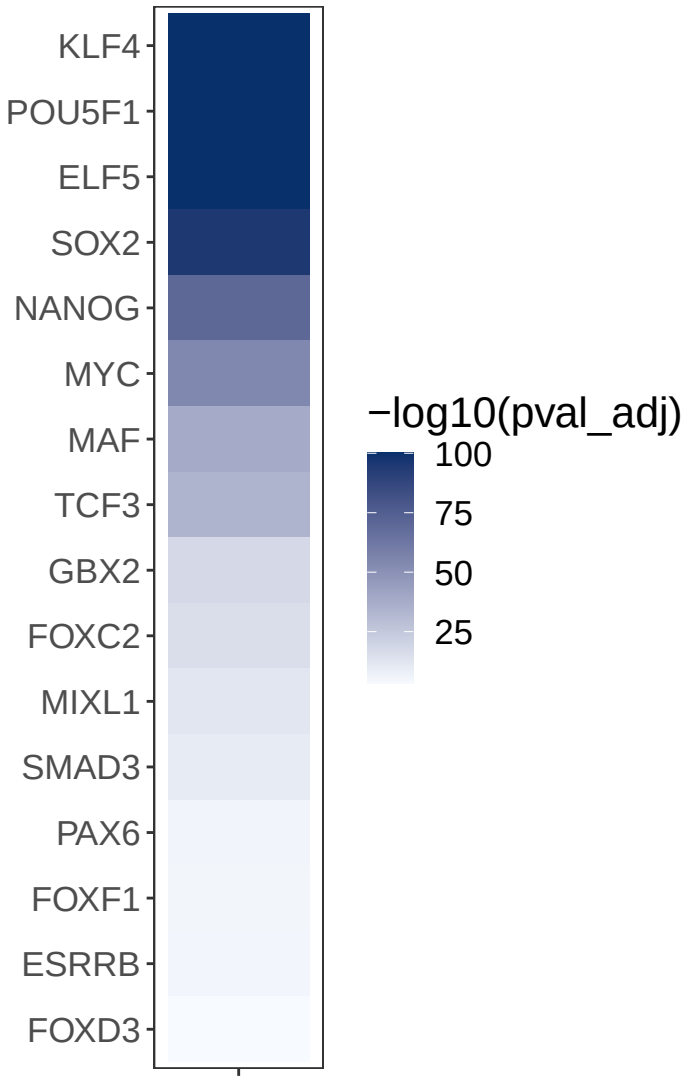
Before interpreting these results, either enriched TF motifs or their binding partners we strongly recommend to filter them based on their expression level, for instance at RPKM (FPKM) or TPM > 1. Next, we can choose selected TFs that are known to be involved based on prior knowledge or highly enriched ones. If you are interested to filter for only TF genes as defined by Lambert et al., 2018, you can do so as well.

```
# Select TFs for visualization. These are some well known TFs in
# H1-ESCs
TF_list <- c("SOX2", "NANOG", "MYC", "POU5F1", "STAT3", "TCF3", "KLF4", "SMAD1",
             "SMAD2", "SMAD3", "SMAD4", "SMAD5", "SMAD8", "TFAP2C", "KLF2",
             "KLF4", "KLF5", "ELF5", "ESRRB", "PRDM14", "DPPA4", "FOXC2",
             "FOXD3", "FOXF1", "GATA", "GBX2", "MAF", "MEF2C", "MAX", "JUN",
             "MIXL1", "NFKB", "PAX2", "PAX6", "SNAI1", "TBX6", "SUZ12")
# filter for these selected TFs
enrich_motifs_filtered <- enrich_motifs %>%
    dplyr::filter(TF_name %in% TF_list) %>%
    dplyr::distinct(TF_name, .keep_all = TRUE) %>%
    arrange(desc(`-log10(pval_adj)`))

# get the list of TFs genes
```

```
TF_df <- TF_df %>% dplyr::filter(TF_Yes_No == "Yes")

# filter TFs motifs for only TFs genes based on Lambert et al., 2018
enrich_motifs_filtered <- enrich_motifs_filtered %>%
    dplyr::filter(TF_name %in% TF_df$Name) %>%
    dplyr::distinct(TF_name, .keep_all = TRUE)
```

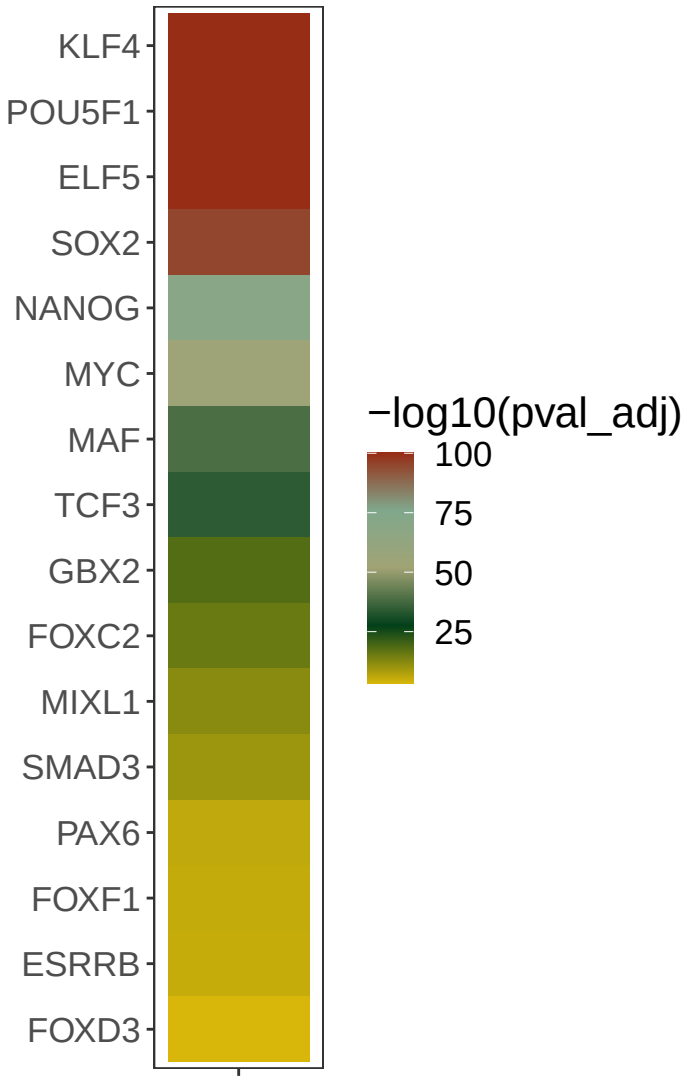
Now we can visualize the enriched motifs and enriched motif pairs using heatmaps.

```
# replace very low values so that it is easy to visualize
# enrich_motifs_filtered[enrich_motifs_filtered < 1e-100] <- 1e-100

# keep the order in plotting, by default ggplot2 sorts alphabetically
enrich_motifs_filtered <- enrich_motifs_filtered %>%
    mutate(TF_name = factor(TF_name, levels = rev(unique(TF_name))))

# choose color palette
col_pal <- brewer.pal(9,"Blues")
# enriched motifs heatmap using ggplot2
pp <- ggplot(data = enrich_motifs_filtered, aes(x = "", y = TF_name,
                                                fill = `-log10(pval_adj)`))  +
    geom_tile() + scale_fill_gradient(low = col_pal[1], high = col_pal[9]) +
    ylab("") + xlab("") + theme_bw() +
    theme(plot.background = element_blank()
    ,panel.grid.major = element_blank()
    ,panel.grid.minor = element_blank()
    ,text = element_text(size=16), axis.text.x = element_text(angle=90, vjust=0.5)
)
pp
```



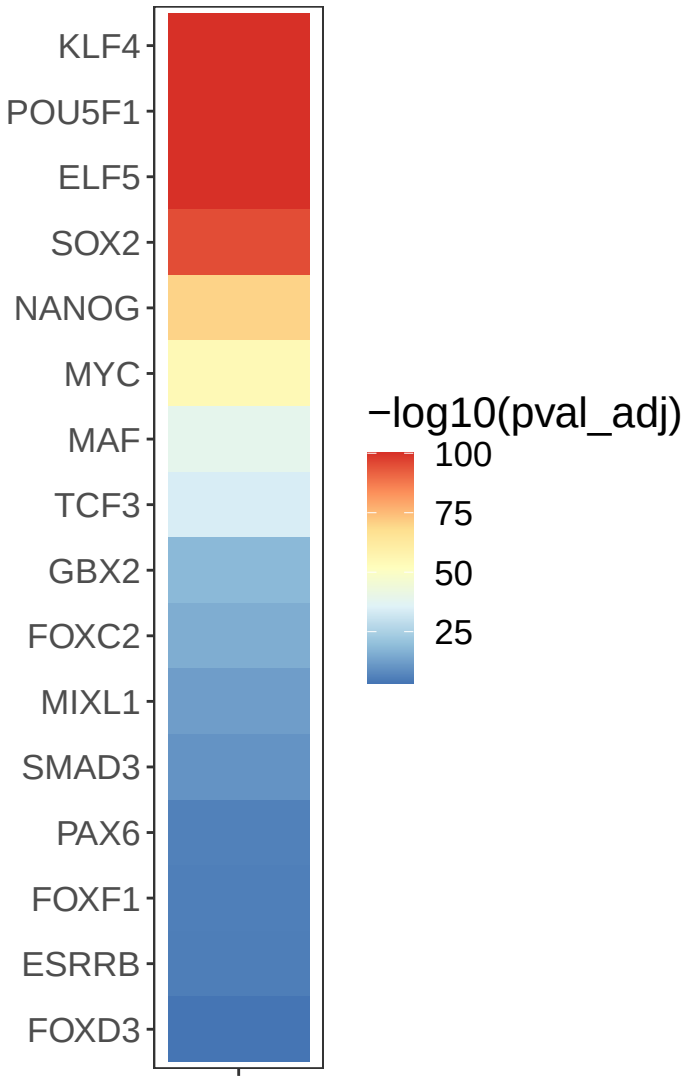
Using another color palette

```
# choose color palette from wesanderson
# Gradient color
col_pal <- wesanderson::wes_palette("Cavalcanti1", 100, type = "continuous")

# enriched motifs heaatmap using ggplot2
pp <- ggplot(data = enrich_motifs_filtered, aes(x = "", y = TF_name,
                                                fill = `-log10(pval_adj)`))  +
    geom_tile() + scale_fill_gradientn(colours = col_pal) +
    ylab("") + xlab("") + theme_bw() +
    theme(plot.background = element_blank()
    ,panel.grid.major = element_blank()
    ,panel.grid.minor = element_blank()
    ,text = element_text(size=16), axis.text.x = element_text(angle=90, vjust=0.5)
)
pp
```



Using another color palette

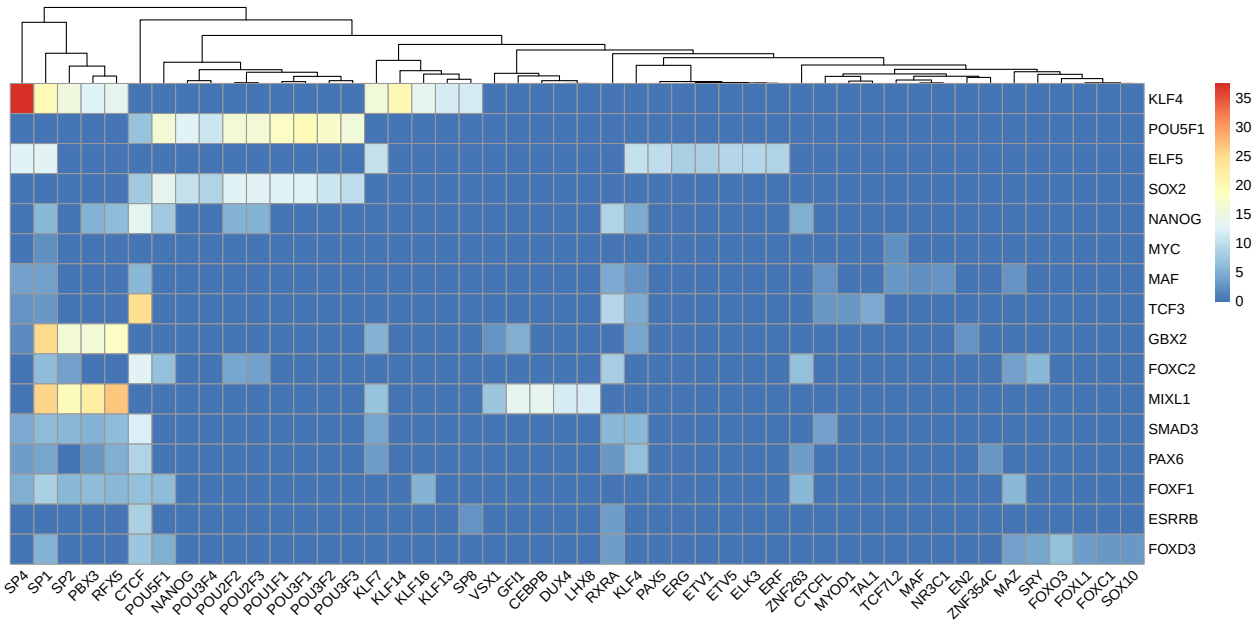
```
# choose color palette from wesanderson
# Gradient color
col_pal <- colorRampPalette(rev(brewer.pal(n = 7, name =
  "RdYlBu")))(100)

# enriched motifs heaatmap using ggplot2
pp <- ggplot(data = enrich_motifs_filtered, aes(x = "", y = TF_name,
                                                fill = `-log10(pval_adj)`))  +
    geom_tile() + scale_fill_gradientn(colours = col_pal) +
    ylab("") + xlab("") + theme_bw() +
    theme(plot.background = element_blank()
    ,panel.grid.major = element_blank()
    ,panel.grid.minor = element_blank()
    ,text = element_text(size=16), axis.text.x = element_text(angle=90, vjust=0.5)
)
pp
```



Now plot the top 10 binding partners for the above TF motifs.

```
# filter TFs motifs for only TFs genes based on Lambert et al., 2018
enrich_motif_pairs_filtered <- enrich_motif_pairs %>%
    dplyr::filter(motif_name_1 %in% enrich_motifs_filtered$motif_name) %>%
    group_by(TF_name_1, TF_name_2) %>%
    dplyr::distinct(TF_name_2, .keep_all = TRUE) %>%
    dplyr::group_by(TF_name_1) %>%
    dplyr::filter(TF_name_1 %in% TF_df$Name) %>%
    dplyr::filter(TF_name_2 %in% TF_df$Name)
# select top 10 binding partners and remove redundant motifs
enrich_motif_pairs_filtered <- enrich_motif_pairs_filtered %>%
    dplyr::filter(motif_name_1 %in% enrich_motifs_filtered$motif_name) %>%
    group_by(TF_name_1, TF_name_2) %>%
    dplyr::distinct(TF_name_2, .keep_all = TRUE) %>%
    dplyr::group_by(TF_name_1) %>% top_n(10, -pval_adj)

# choose color palette
col_pal <- brewer.pal(9,"Blues")
# enriched motif pairs heaatmap using ggplot2
```

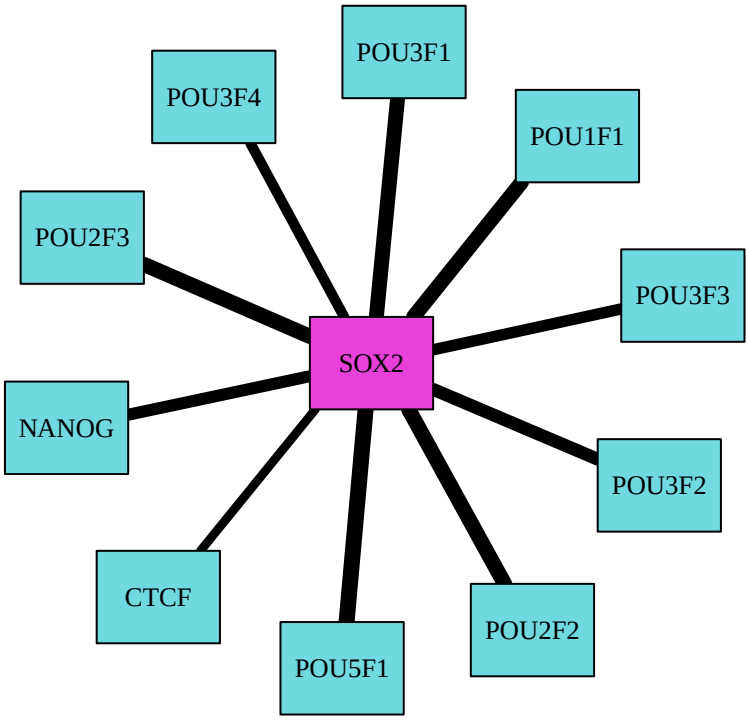
```
# pp <- ggplot(data = enrich_motif_pairs_filtered,
#              aes(x = TF_name_2, y = TF_name_1, fill = -log(pval_adj))) +
#     geom_tile() + scale_fill_gradient(low = col_pal[1], high = col_pal[9]) +
#     ylab("") + xlab("") + theme_bw() +
#     theme(plot.background = element_blank()
#     ,panel.grid.major = element_blank()
#     ,panel.grid.minor = element_blank()
#     ,text = element_text(size=16), axis.text.x = element_text(angle=90, vjust=0.5)
# )
# pp

enrich_motif_pairs_filtered_mat <- enrich_motif_pairs_filtered %>%
    select(TF_name_1, TF_name_2, `-log10(pval_adj)`) %>%
    tidyr::spread(TF_name_2, `-log10(pval_adj)`) %>%
    tibble::column_to_rownames(var = "TF_name_1") %>%
    as.matrix() %>%
    tidyr::replace_na(0)

# get the order consistent with "enrich_motifs_filtered"
idx <- match(enrich_motifs_filtered$TF_name, rownames(enrich_motif_pairs_filtered_mat))
enrich_motif_pairs_filtered_mat <- enrich_motif_pairs_filtered_mat[idx, ]
pheatmap::pheatmap(enrich_motif_pairs_filtered_mat, cluster_rows=FALSE, cluster_cols=TRUE, na_col = "gr
```



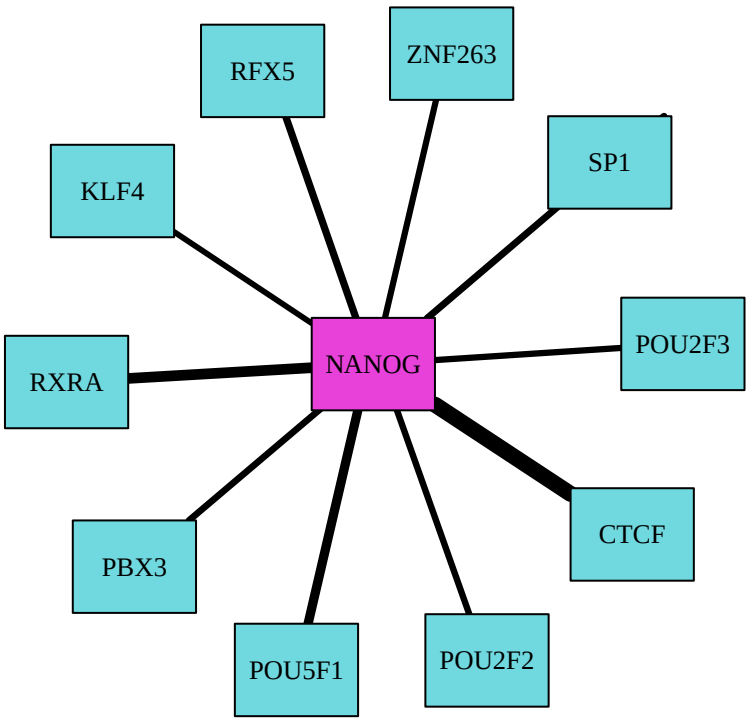
Now we can make colorful networks for select TFs of interest using the `igraph` package

```
# create a custom function for plotting networks
plot_network <- function(enrich_motif_pairs_filtered, TF_name = TF_name,
                         color_TF = "#70d9e0", color_bind_TF = "#e841da"){
    edges <- enrich_motif_pairs_filtered %>%
        dplyr::filter(TF_name_1 == TF_name) %>%
    dplyr::mutate(sig = -log(pval_adj)) %>%
    dplyr::mutate(sig = sig*8/max(sig)) %>% as.data.frame() %>%
    dplyr::select(TF_name_1, TF_name_2, sig)
nodes <- data.frame(
  name=unique(c(edges$TF_name_1, edges$TF_name_2)),
```

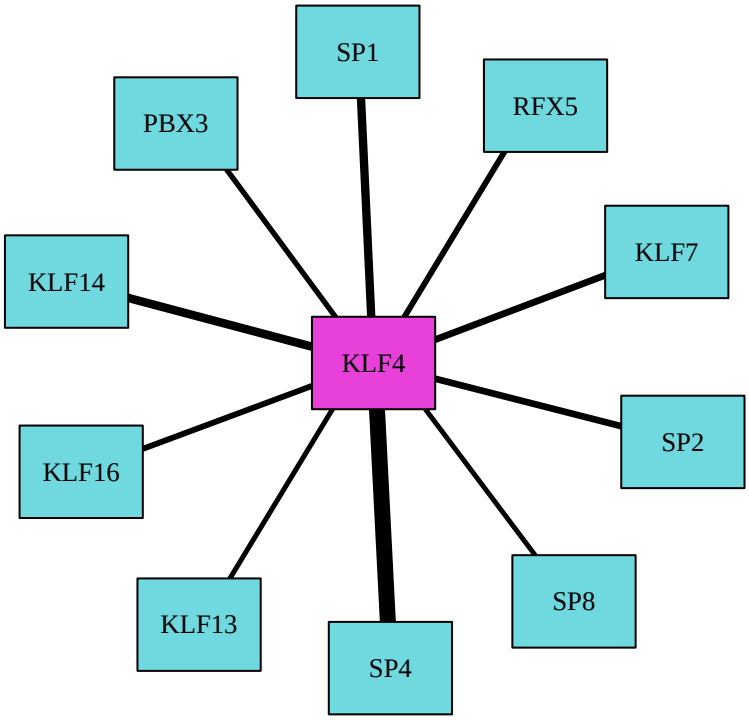
```
  role=c(rep("TF",1),rep("partner", nrow(edges)))
)
# Turn it into igraph object
network <- igraph::graph_from_data_frame(d=edges, vertices=nodes,
                                         directed=FALSE)
# Make a palette of 3 colors
library(RColorBrewer)
# col  <- brewer.pal(3, "Set1")[1:2]
col <- c(color_TF, color_bind_TF)
# Create a vector of color
my_color <- col[as.numeric(as.factor(igraph::V(network)$role))]
# plotting the network
plot(network, vertex.color=my_color, vertex.shape = c("rectangle"),
     vertex.size = 40, vertex.size2 = 30, vertex.label.cex=0.8,
     vertex.label.color="black", edge.width=igraph::E(network)$sig,
     edge.color="black")
}
# select TF "SOX2" and extract its connections
plot_network(enrich_motif_pairs_filtered, TF_name = "SOX2",
             color_TF = "#70d9e0", color_bind_TF = "#e841da")
```




```
# select TF "NANOG" and extract its connections
plot_network(enrich_motif_pairs_filtered, TF_name = "NANOG",
             color_TF = "#70d9e0", color_bind_TF = "#e841da")
```




```
# select TF "KLF4" and extract its connections
plot_network(enrich_motif_pairs_filtered, TF_name = "KLF4",
             color_TF = "#70d9e0", color_bind_TF = "#e841da")
```



Additional example use cases:

Example use case 1: Genomic regions from two conditions

Here, we have genomic regions from two conditions, for instance, cells treated vs untreated. To demonstrate this example, we obtained the ATAC-seq data in Th17 cells treated with a stimulus and untreated Th17 cells. We can find TFs and their binding partner TFs specifically enriched in stimulated cells relative to untreated cells. To do so, we need to provide ATAC-seq peaks from stimulated cells as the input set and ATAC-seq peaks from untreated cells as the control set in the `enrichmotifpairR` package.

```
# Finding the enriched motifs and their partners
results <- findEnrichMotifPair(
  target_data = example_peaks_data$Th17_stimulated_ATAC_seq_peaks,
  background_data = example_peaks_data$Th17_ATAC_seq_peaks,
  genome_ver = "hg19",
  scramble_data = FALSE,
  motif_database = "ENCODE",
  Pvalue_computation = "hyper",
  Pvalue_threshold = 0.05,
  Pvalue_adjust_method = "BH"
)
```

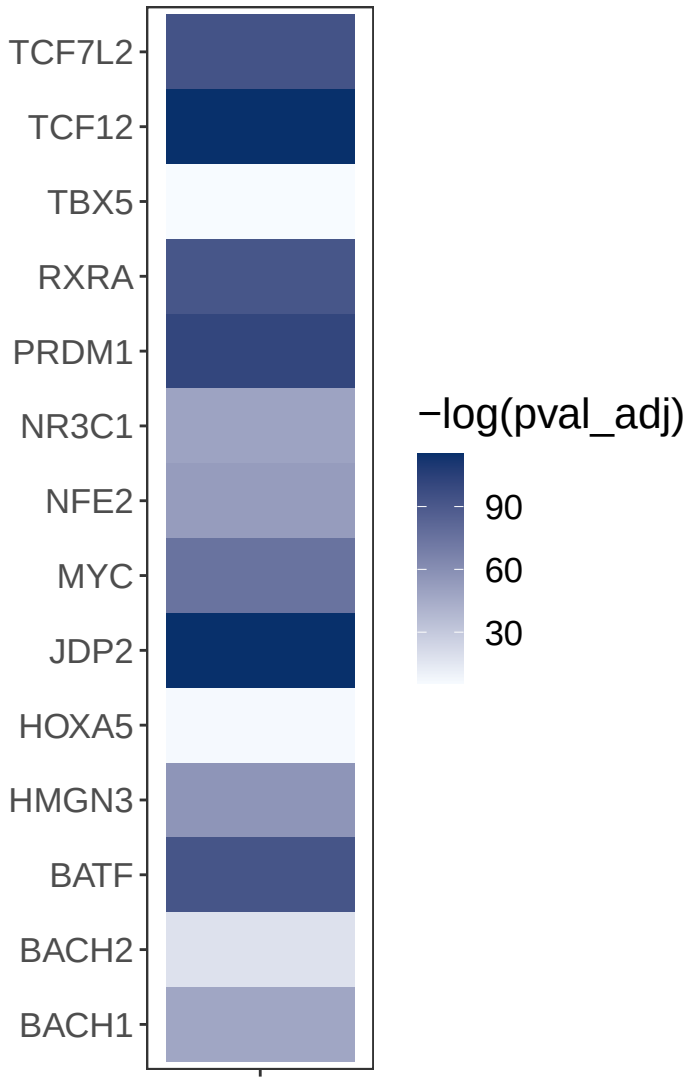
Assign the resulting data to individual objects and filter motifs for only TFs genes. TF genes are defined by Lambert et al., 2018.

```
# assign the results data to individual objects
enrich_motifs <- results$motif_enrich
enrich_motif_pairs <- results$motif_pair_enrich
# get the list of TFs genes
TF_df <- TF_df %>% dplyr::filter(TF_Yes_No == "Yes")
# filter TFs motifs for only TFs genes based on Lambert et al., 2018
enrich_motifs_filtered <- enrich_motifs %>%
    dplyr::filter(TF_name %in% TF_df$Name) %>%
    dplyr::distinct(TF_name, .keep_all = TRUE)
enrich_motif_pairs_filtered <- enrich_motif_pairs %>%
    dplyr::filter(motif_name_1 %in% enrich_motifs_filtered$motif_name) %>%
    group_by(TF_name_1, TF_name_2) %>%
    dplyr::distinct(TF_name_2, .keep_all = TRUE) %>%
    dplyr::group_by(TF_name_1) %>%
    dplyr::filter(TF_name_1 %in% TF_df$Name) %>%
    dplyr::filter(TF_name_2 %in% TF_df$Name)
```

Plot the heatmap of enriched TF motifs

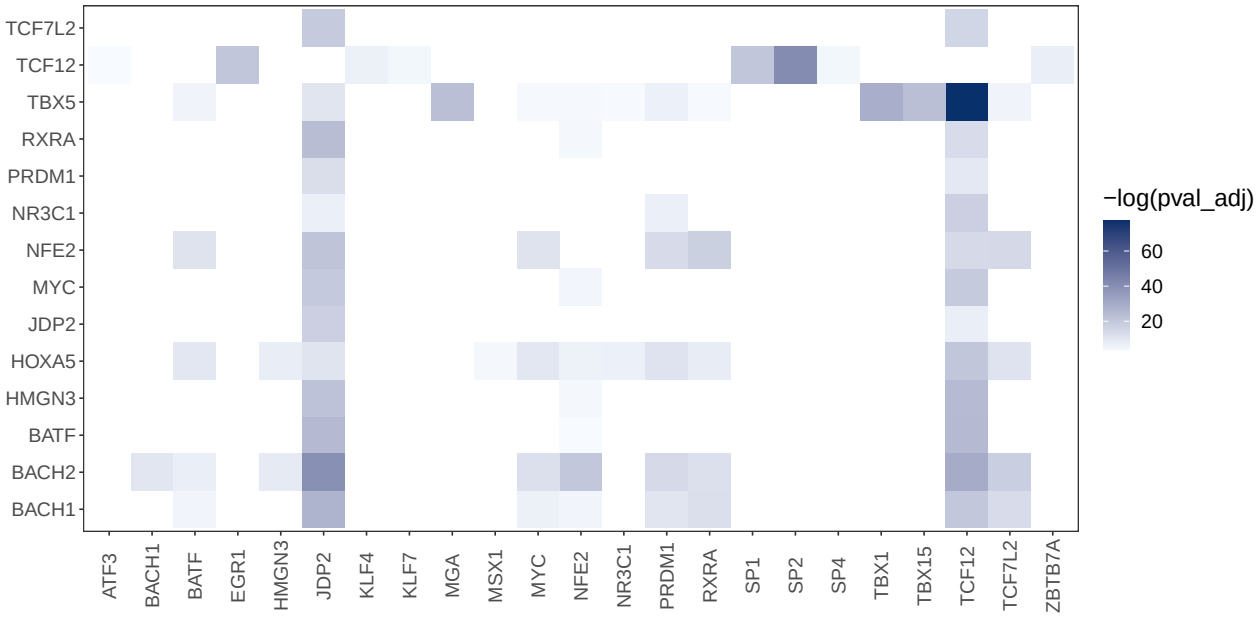
```
# Select TFs for visualization , here we have only 14 TFs and plotting them all
enrich_motifs_filtered[enrich_motifs_filtered < 1e-50] <- 1e-50

# choose color palette
col_pal <- brewer.pal(9,"Blues")
# enriched motifs heatmap using ggplot2
pp <- ggplot(data = enrich_motifs_filtered, aes(x = "", y = TF_name,
                                                fill = -log(pval_adj)))  +
    geom_tile() + scale_fill_gradient(low = col_pal[1], high = col_pal[9]) +
    ylab("") + xlab("") + theme_bw() +
    theme(plot.background = element_blank()
    ,panel.grid.major = element_blank()
```

```
    ,panel.grid.minor = element_blank()
    ,text = element_text(size=16), axis.text.x = element_text(angle=90, vjust=0.5)
)
pp
```



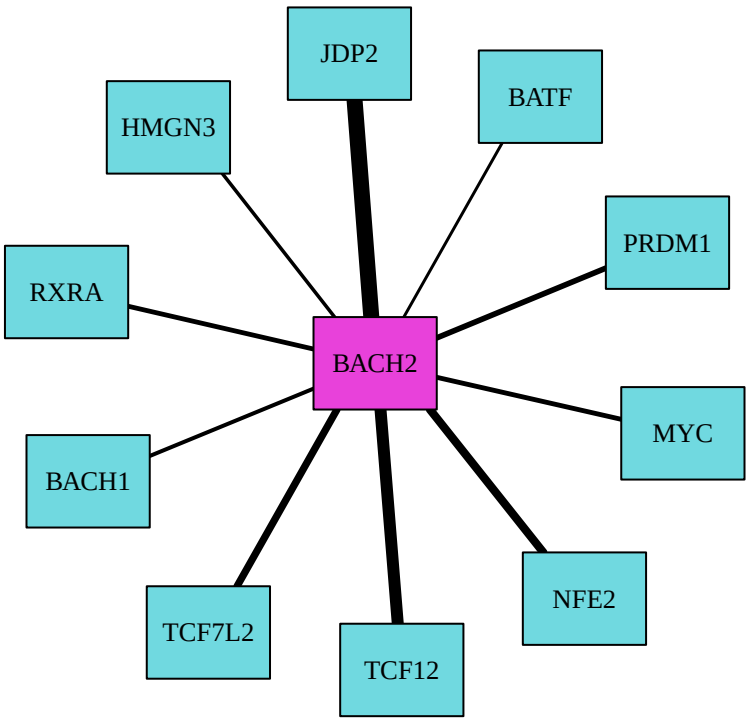
Now plot the binding partners for the above TF motifs

```
# choose color palette
col_pal <- brewer.pal(9,"Blues")
# enriched motif pairs heaatmap using ggplot2
pp <- ggplot(data = enrich_motif_pairs_filtered,
             aes(x = TF_name_2, y = TF_name_1, fill = -log(pval_adj))) +
    geom_tile() + scale_fill_gradient(low = col_pal[1], high = col_pal[9]) +
    ylab("") + xlab("") + theme_bw() +
    theme(plot.background = element_blank()
    ,panel.grid.major = element_blank()
    ,panel.grid.minor = element_blank()
    ,text = element_text(size=16), axis.text.x = element_text(angle=90, vjust=0.5)
)
pp
```

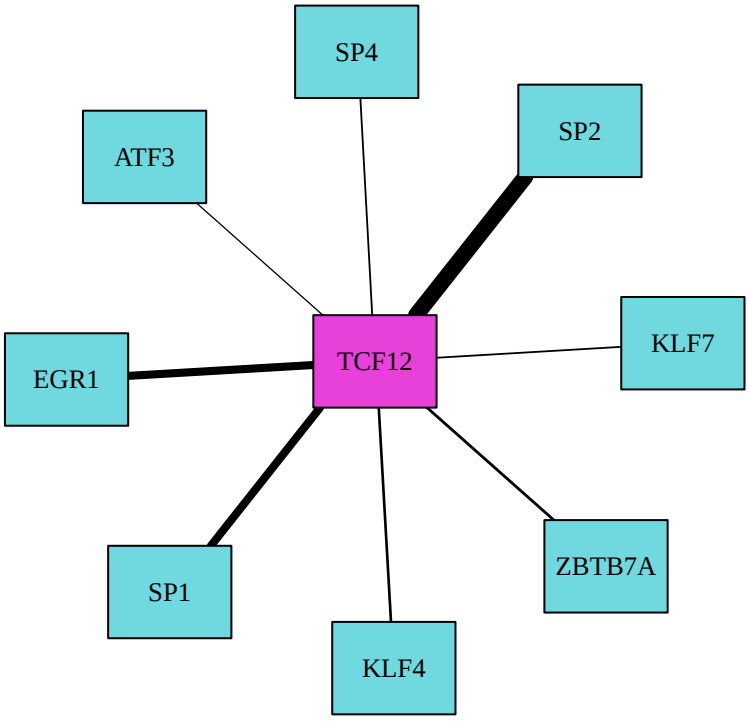


Now we can make colorful networks for select TFs of interest using the `igraph` package.

```
# create a custom function for plotting networks
plot_network <- function(enrich_motif_pairs_filtered, TF_name = TF_name,
                         color_TF = "#70d9e0", color_bind_TF = "#e841da"){
    edges <- enrich_motif_pairs_filtered %>%
        dplyr::filter(TF_name_1 == TF_name) %>%
    dplyr::mutate(sig = -log(pval_adj)) %>%
    dplyr::mutate(sig = sig*8/max(sig)) %>% as.data.frame() %>%
    dplyr::select(TF_name_1, TF_name_2, sig)
nodes <- data.frame(
  name=unique(c(edges$TF_name_1, edges$TF_name_2)),
  role=c(rep("TF",1),rep("partner", nrow(edges)))
)
# Turn it into igraph object
network <- igraph::graph_from_data_frame(d=edges, vertices=nodes,
                                         directed=FALSE)
# Make a palette of 3 colors
library(RColorBrewer)
# col  <- brewer.pal(3, "Set1")[1:2]
col <- c(color_TF, color_bind_TF)
# Create a vector of color
my_color <- col[as.numeric(as.factor(igraph::V(network)$role))]
# plotting the network
plot(network, vertex.color=my_color, vertex.shape = c("rectangle"),
     vertex.size = 40, vertex.size2 = 30, vertex.label.cex=0.8,
     vertex.label.color="black", edge.width=igraph::E(network)$sig,
     edge.color="black")
}
# select TF "BACH2" and extract its connections
plot_network(enrich_motif_pairs_filtered, TF_name = "BACH2",
             color_TF = "#70d9e0", color_bind_TF = "#e841da")
```

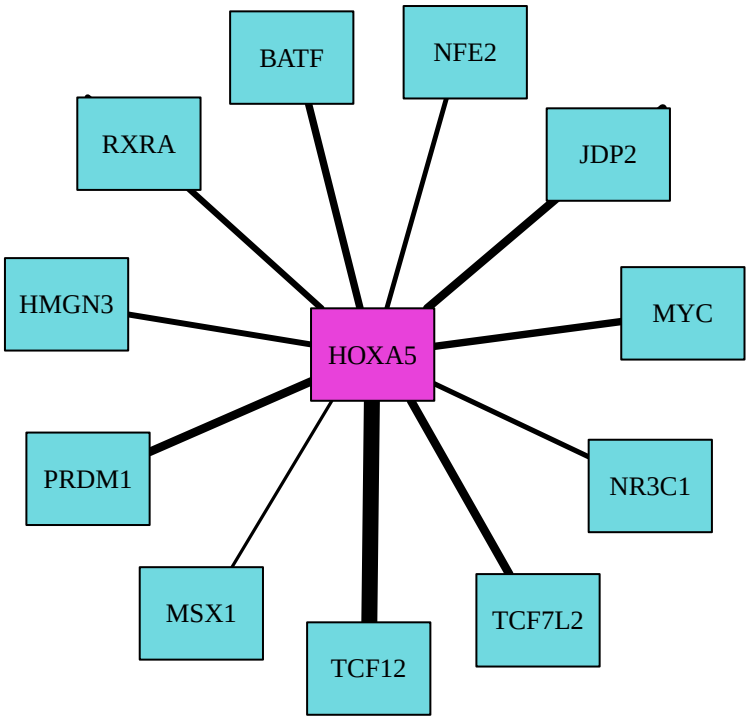



```
# select TF "TCF12" and extract its connections
plot_network(enrich_motif_pairs_filtered, TF_name = "TCF12",
             color_TF = "#70d9e0", color_bind_TF = "#e841da")
```




```
# select TF "HOXA5" and extract its connections
plot_network(enrich_motif_pairs_filtered, TF_name = "HOXA5",
```

```
color_TF = "#70d9e0", color_bind_TF = "#e841da")
```



Example use case 2: Genomic regions from two conditions

Here, we have genomic regions from two conditions, for instance, cells differentiating from one cell fate to another. To demonstrate this example, we obtained ATAC-seq data in Th0 cells (activated T cells) moving to Th1. We can find TFs and their binding partner TFs specifically enriched in Th1 cells relative to Th0 cells. To do so, we need to provide ATAC-seq peaks from Th1 cells as the input set and ATAC-seq peaks from Th0 cells as the control set in the `enrichmotifpairR` package. In this case, we are selecting motifs from the `CISBP` database.

```
# Finding the enriched motifs and their partners
results <- findEnrichMotifPair(
  target_data = example_peaks_data$Th1_ATAC_seq_peaks,
  background_data = example_peaks_data$Th0_ATAC_seq_peaks,
  genome_ver = "hg19",
  scramble_data = FALSE,
  motif_database = "CISBP",
  Pvalue_computation = "hyper",
  Pvalue_threshold = 0.01,
  Pvalue_adjust_method = "BH"
)
```

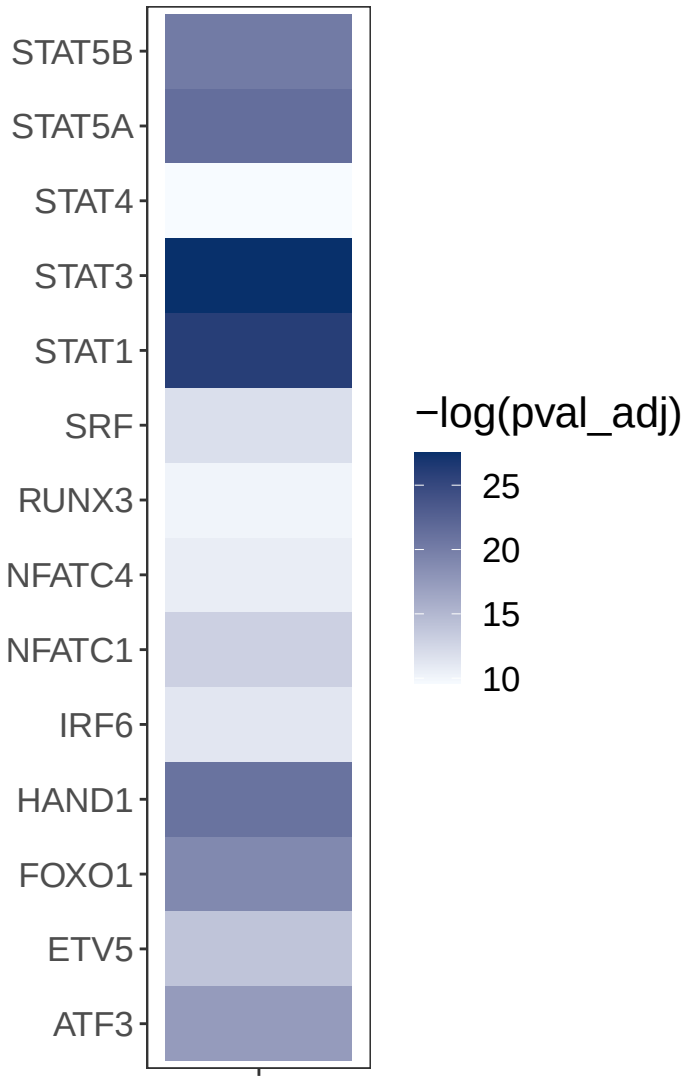
Assign the resulting data to individual objects and filter motifs for only TFs genes. TF genes are defined by Lambert et al., 2018.

```
# assign the results data to individual objects
enrich_motifs <- results$motif_enrich
enrich_motif_pairs <- results$motif_pair_enrich

TF_list <- unique(sort(c("STAT1", "STAT3", "STAT4", "STAT5A", "STAT5B", "ATF3",
                         "JUN", "JUNB", "FOXO1", "HAND1", "NFATC1", "NFATC2",
```

```
                        "NFATC3", "NFATC4", "SRF", "RUNX3", "IRF1", "IRF6",
                        "ETV5")))

# filter for these selected TFs
enrich_motifs_filtered <- enrich_motifs %>%
    dplyr::filter(TF_name %in% TF_list) %>%
    dplyr::distinct(TF_name, .keep_all = TRUE)

# get the list of TFs genes
TF_df <- TF_df %>% dplyr::filter(TF_Yes_No == "Yes")

# filter TFs motifs for only TFs genes based on Lambert et al., 2018
enrich_motifs_filtered <- enrich_motifs_filtered %>%
    dplyr::filter(TF_name %in% TF_df$Name) %>%
    dplyr::distinct(TF_name, .keep_all = TRUE)

enrich_motif_pairs_filtered <- enrich_motif_pairs %>%
    dplyr::filter(motif_name_1 %in% enrich_motifs_filtered$motif_name) %>%
    group_by(TF_name_1, TF_name_2) %>%
    dplyr::distinct(TF_name_2, .keep_all = TRUE) %>%
    dplyr::group_by(TF_name_1) %>%
    dplyr::filter(TF_name_1 %in% TF_df$Name) %>%
    dplyr::filter(TF_name_2 %in% TF_df$Name)

# select top 15 binding partners and remove redundant motifs
enrich_motif_pairs_filtered <- enrich_motif_pairs_filtered %>%
    dplyr::filter(motif_name_1 %in% enrich_motifs_filtered$motif_name) %>%
    group_by(TF_name_1, TF_name_2) %>%
    dplyr::distinct(TF_name_2, .keep_all = TRUE) %>%
    dplyr::group_by(TF_name_1) %>% top_n(15, -pval_adj)
```

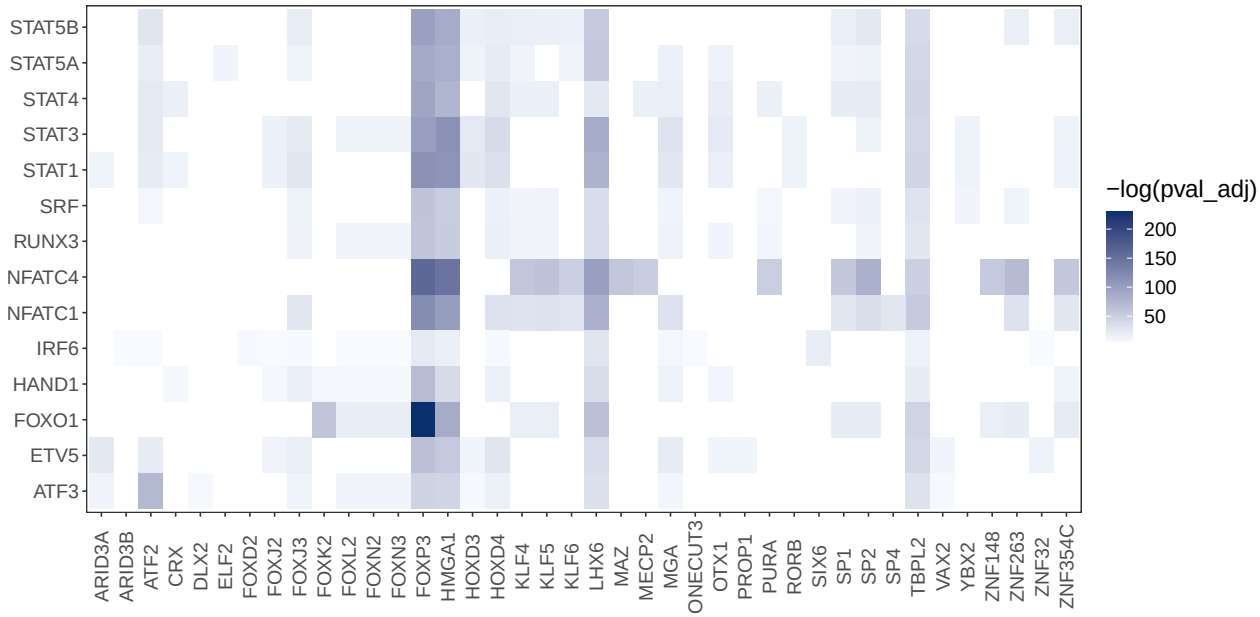
Plot the heatmap of enriched TF motifs

```
# choose color palette
col_pal <- brewer.pal(9,"Blues")
# enriched motifs heatmap using ggplot2
pp <- ggplot(data = enrich_motifs_filtered, aes(x = "", y = TF_name,
                                                fill = -log(pval_adj)))  +
    geom_tile() + scale_fill_gradient(low = col_pal[1], high = col_pal[9]) +
    ylab("") + xlab("") + theme_bw() +
    theme(plot.background = element_blank()
    ,panel.grid.major = element_blank()
    ,panel.grid.minor = element_blank()
    ,text = element_text(size=16), axis.text.x = element_text(angle=90, vjust=0.5)
)
pp
```



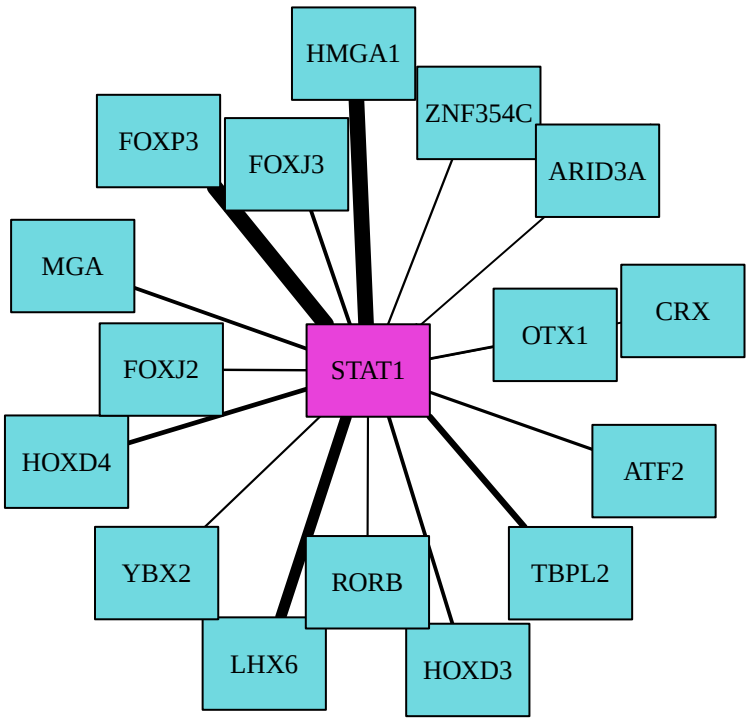
Now plot the binding partners for the above TF motifs.

```
# replace very low values so that it is easy to visualize
enrich_motif_pairs_filtered[enrich_motif_pairs_filtered < 1e-100] <- 1e-100
# choose color palette
col_pal <- brewer.pal(9,"Blues")
# enriched motif pairs heatmap using ggplot2
pp <- ggplot(data = enrich_motif_pairs_filtered,
             aes(x = TF_name_2, y = TF_name_1, fill = -log(pval_adj))) +
    geom_tile() + scale_fill_gradient(low = col_pal[1], high = col_pal[9]) +
    ylab("") + xlab("") + theme_bw() +
    theme(plot.background = element_blank()
    ,panel.grid.major = element_blank()
    ,panel.grid.minor = element_blank()
    ,text = element_text(size=16), axis.text.x = element_text(angle=90, vjust=0.5)
)
pp
```

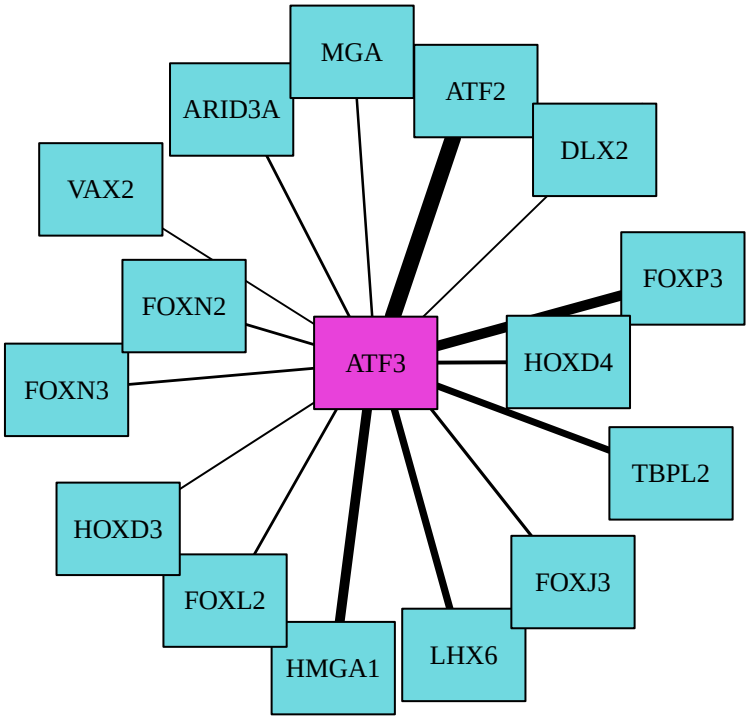


Now we can make colorful networks for select TFs of interest using the igraph package.

```
# create a custom function for plotting networks
plot_network <- function(enrich_motif_pairs_filtered, TF_name = TF_name,
                         color_TF = "#70d9e0", color_bind_TF = "#e841da"){
    edges <- enrich_motif_pairs_filtered %>%
        dplyr::filter(TF_name_1 == TF_name) %>%
    dplyr::mutate(sig = -log(pval_adj)) %>%
    dplyr::mutate(sig = sig*8/max(sig)) %>% as.data.frame() %>%
    dplyr::select(TF_name_1, TF_name_2, sig)
nodes <- data.frame(
  name=unique(c(edges$TF_name_1, edges$TF_name_2)),
  role=c(rep("TF",1),rep("partner", nrow(edges)))
)
# Turn it into igraph object
network <- igraph::graph_from_data_frame(d=edges, vertices=nodes,
                                         directed=FALSE)
# Make a palette of 3 colors
library(RColorBrewer)
# col  <- brewer.pal(3, "Set1")[1:2]
col <- c(color_TF, color_bind_TF)
# Create a vector of color
my_color <- col[as.numeric(as.factor(igraph::V(network)$role))]
# plotting the network
plot(network, vertex.color=my_color, vertex.shape = c("rectangle"),
     vertex.size = 40, vertex.size2 = 30, vertex.label.cex=0.8,
     vertex.label.color="black", edge.width=igraph::E(network)$sig,
     edge.color="black")
}
# select TF "STAT1" and extract its connections
plot_network(enrich_motif_pairs_filtered, TF_name = "STAT1",
             color_TF = "#70d9e0", color_bind_TF = "#e841da")
```

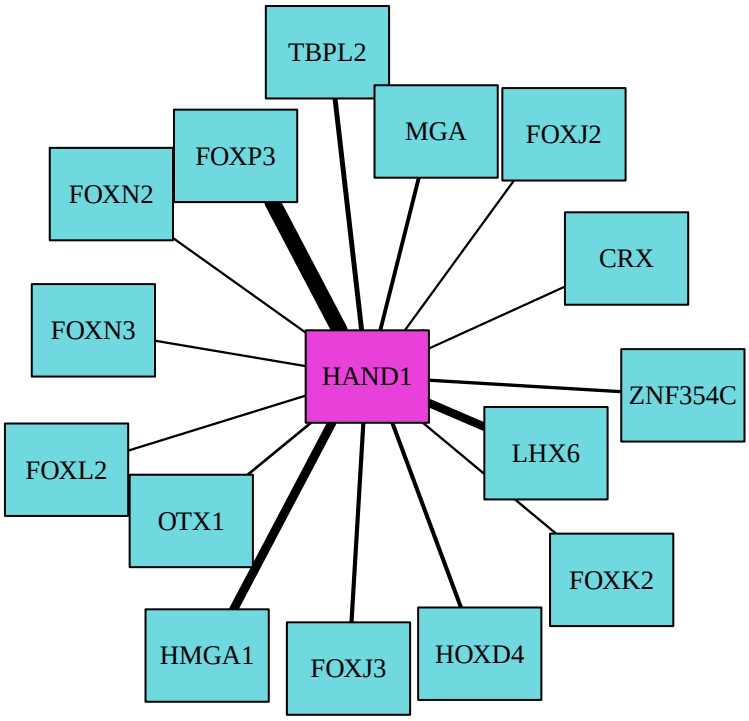



```
# select TF "ATF3" and extract its connections
plot_network(enrich_motif_pairs_filtered, TF_name = "ATF3",
             color_TF = "#70d9e0", color_bind_TF = "#e841da")
```




```
# select TF "HAND1" and extract its connections
plot_network(enrich_motif_pairs_filtered, TF_name = "HAND1",
```

```
color_TF = "#70d9e0", color_bind_TF = "#e841da")
```



Example use case 3: Genomic regions from two chromatin states

Here, we have genomic regions from two chromatin states, for instance, active genomic regions vs repressive genomic regions. To demonstrate this example, we obtained H3K27ac peaks representing active regions and H3K27me3 peaks representing repressive genomic regions in CD8+ cells. We can find TFs and their binding partner TFs specifically enriched at active genomic regions relative to repressive regions. To do so, we need to provide H3K27ac peaks as the input set and H3K27me3 peaks as the control set in the `enrichmotifpairR` package. In this case, we are selecting motifs from the `JASPAR-CORE` database.

```
# Finding the enriched motifs and their partners
results <- findEnrichMotifPair(
  target_data = example_peaks_data$`CD8+_H3K27ac_peaks`,
  background_data = example_peaks_data$`CD8+_H3K27me3_peaks`,
  genome_ver = "hg38",
  scramble_data = FALSE,
  motif_database = "JASPAR_CORE",
  Pvalue_computation = "hyper",
  Pvalue_threshold = 0.01,
  Pvalue_adjust_method = "BH"
)
```

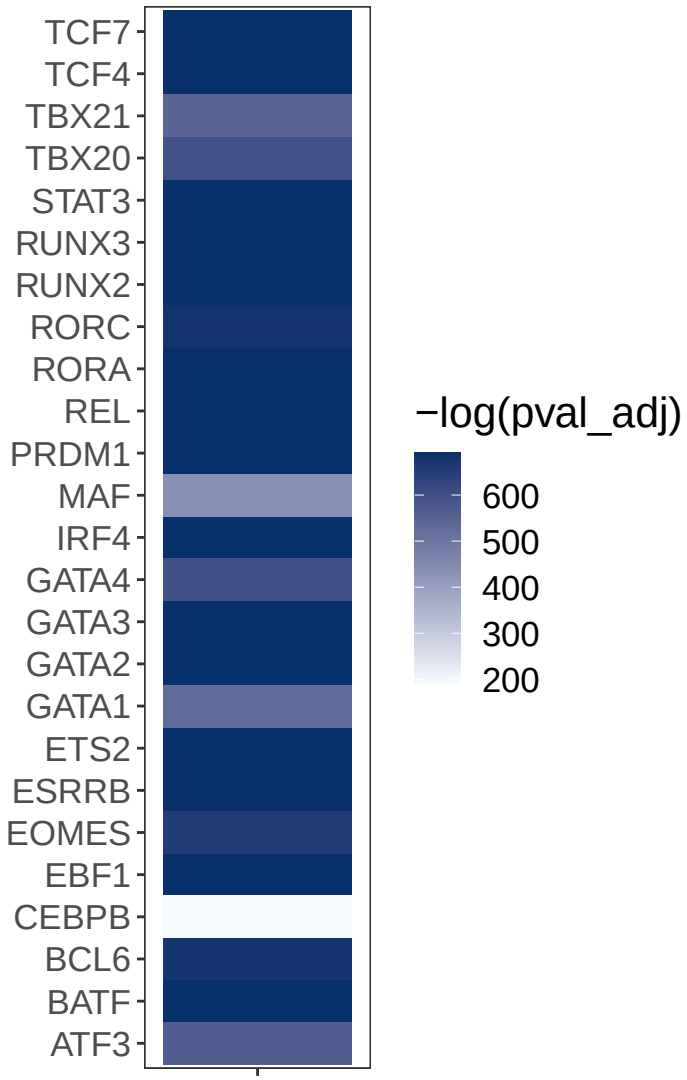
Assign the resulting data to individual objects and filter motifs for only TFs genes. TF genes are defined by Lambert et al., 2018.

```
# assign the results data to individual objects
enrich_motifs <- results$motif_enrich
enrich_motif_pairs <- results$motif_pair_enrich

TF_list <- unique(sort(c("PRDM1", "TBX21", "TBX20", "EOMES", "BLIMP1", "ESRRB",
                         "ID2", "ID3", "CEBPB", "STAT3", "STAT5", "STAT6",
```

```
                        "ATF3", "FOXO1", "RUNX1", "RUNX2", "GATA1", "GATA2",
                        "GATA3", "GATA4", "BCL6", "BATF", "RORA", "RORC",
                        "IRF4", "REL", "TCF1", "TCF7", "IRF4", "TCF4", "EBF1",
                        "MAF", "ETS2", "RUNX3")))

# filter for these selected TFs
enrich_motifs_filtered <- enrich_motifs %>%
    dplyr::filter(TF_name %in% TF_list) %>%
    dplyr::distinct(TF_name, .keep_all = TRUE)

# get the list of TFs genes
TF_df <- TF_df %>% dplyr::filter(TF_Yes_No == "Yes")

# filter TFs motifs for only TFs genes based on Lambert et al., 2018
enrich_motifs_filtered <- enrich_motifs_filtered %>%
    dplyr::filter(TF_name %in% TF_df$Name) %>%
    dplyr::distinct(TF_name, .keep_all = TRUE)

enrich_motif_pairs_filtered <- enrich_motif_pairs %>%
    dplyr::filter(motif_name_1 %in% enrich_motifs_filtered$motif_name) %>%
    group_by(TF_name_1, TF_name_2) %>%
    dplyr::distinct(TF_name_2, .keep_all = TRUE) %>%
    dplyr::group_by(TF_name_1) %>%
    dplyr::filter(TF_name_1 %in% TF_df$Name) %>%
    dplyr::filter(TF_name_2 %in% TF_df$Name)

# select top 10 binding partners and remove redundant motifs
enrich_motif_pairs_filtered <- enrich_motif_pairs_filtered %>%
    dplyr::filter(motif_name_1 %in% enrich_motifs_filtered$motif_name) %>%
    group_by(TF_name_1, TF_name_2) %>%
    dplyr::distinct(TF_name_2, .keep_all = TRUE) %>%
    dplyr::group_by(TF_name_1) %>% top_n(10, -pval_adj)
```

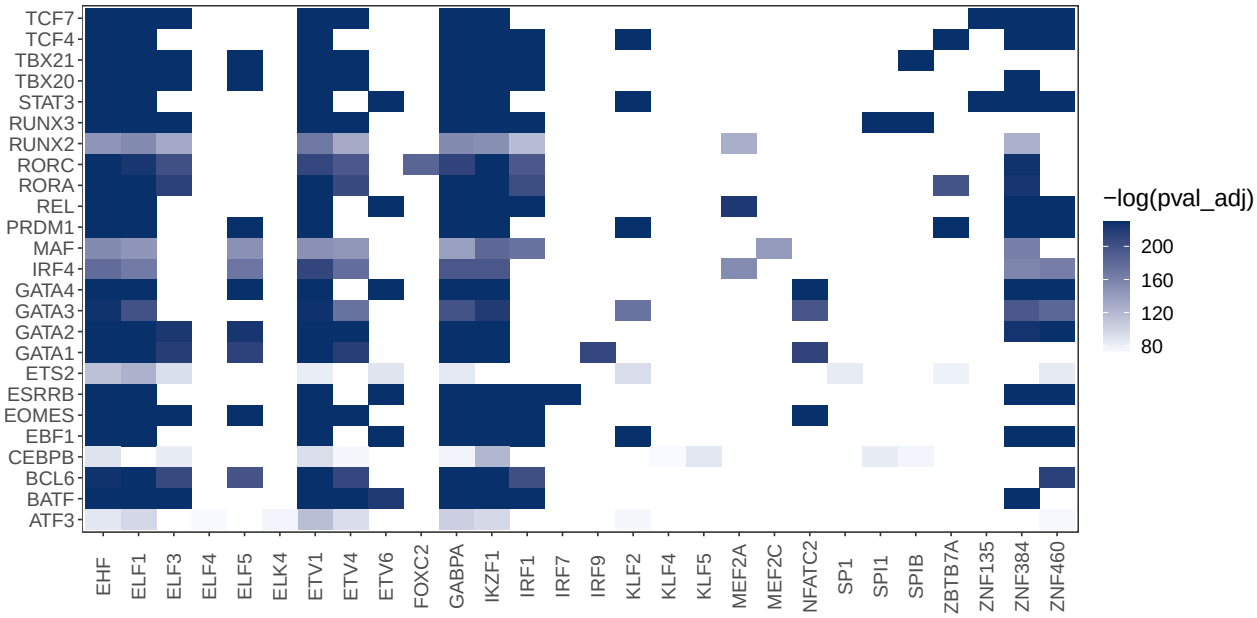
Plot the heatmap of enriched TF motifs

```
# replace very low values so that it is easy to visualize
enrich_motifs_filtered[enrich_motifs_filtered < 1e-300] <- 1e-300
# choose color palette
col_pal <- brewer.pal(9,"Blues")
# enriched motifs heatmap using ggplot2
pp <- ggplot(data = enrich_motifs_filtered, aes(x = "", y = TF_name,
                                                fill = -log(pval_adj)))  +
    geom_tile() + scale_fill_gradient(low = col_pal[1], high = col_pal[9]) +
    ylab("") + xlab("") + theme_bw() +
    theme(plot.background = element_blank()
    ,panel.grid.major = element_blank()
    ,panel.grid.minor = element_blank()
    ,text = element_text(size=16), axis.text.x = element_text(angle=90, vjust=0.5)
)
pp
```



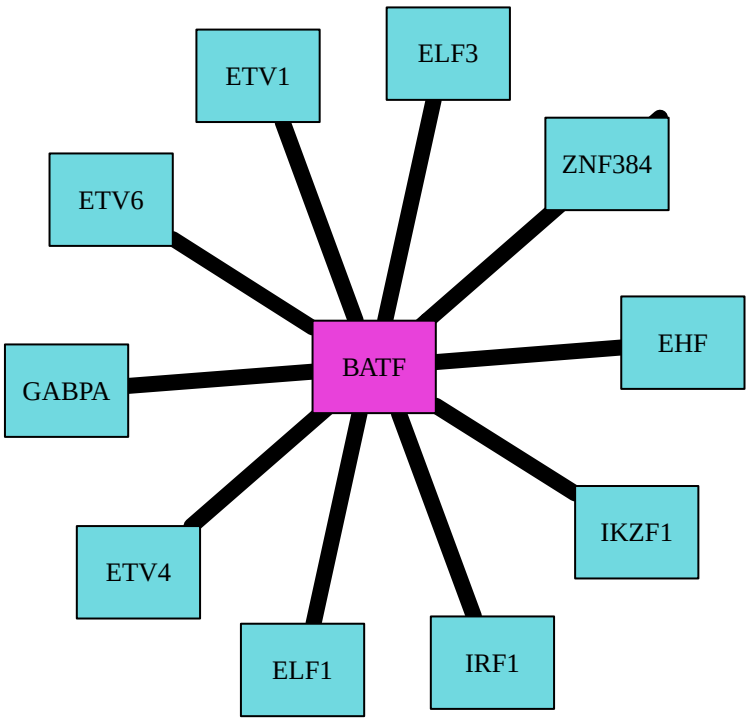
Now plot the binding partners for the above TF motifs

```
# replace very low values so that it is easy to visualize
enrich_motif_pairs_filtered[enrich_motif_pairs_filtered < 1e-100] <- 1e-100
# choose color palette
col_pal <- brewer.pal(9,"Blues")
# enriched motif pairs heaatmap using ggplot2
pp <- ggplot(data = enrich_motif_pairs_filtered,
             aes(x = TF_name_2, y = TF_name_1, fill = -log(pval_adj))) +
    geom_tile() + scale_fill_gradient(low = col_pal[1], high = col_pal[9]) +
    ylab("") + xlab("") + theme_bw() +
    theme(plot.background = element_blank()
    ,panel.grid.major = element_blank()
    ,panel.grid.minor = element_blank()
    ,text = element_text(size=16), axis.text.x = element_text(angle=90, vjust=0.5)
)
pp
```

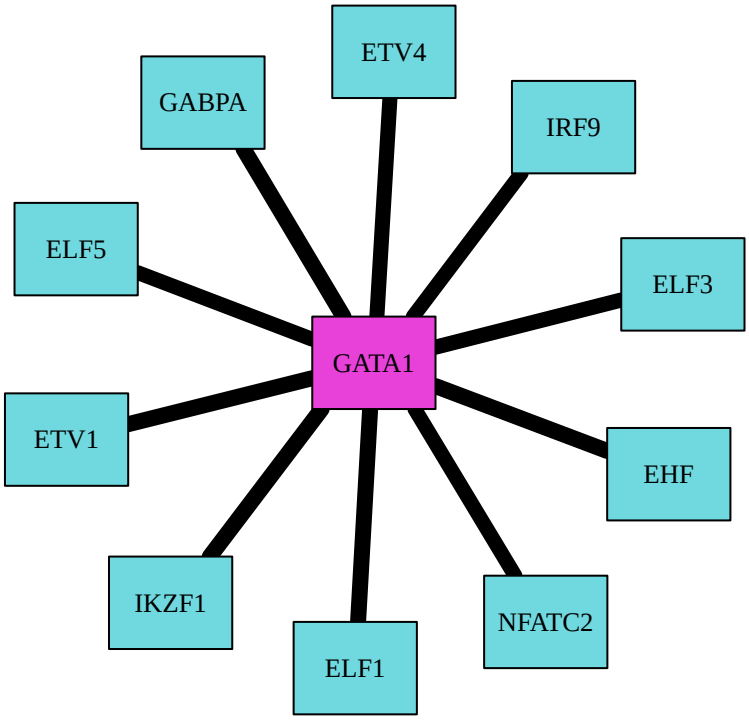


Now we can make colorful networks for select TFs of interest using the igraph package.

```
# create a custom function for plotting networks
plot_network <- function(enrich_motif_pairs_filtered, TF_name = TF_name,
                         color_TF = "#70d9e0", color_bind_TF = "#e841da"){
    edges <- enrich_motif_pairs_filtered %>%
        dplyr::filter(TF_name_1 == TF_name) %>%
    dplyr::mutate(sig = -log(pval_adj)) %>%
    dplyr::mutate(sig = sig*8/max(sig)) %>% as.data.frame() %>%
    dplyr::select(TF_name_1, TF_name_2, sig)
nodes <- data.frame(
  name=unique(c(edges$TF_name_1, edges$TF_name_2)),
  role=c(rep("TF",1),rep("partner", nrow(edges)))
)
# Turn it into igraph object
network <- igraph::graph_from_data_frame(d=edges, vertices=nodes,
                                         directed=FALSE)
# Make a palette of 3 colors
library(RColorBrewer)
# col  <- brewer.pal(3, "Set1")[1:2]
col <- c(color_TF, color_bind_TF)
# Create a vector of color
my_color <- col[as.numeric(as.factor(igraph::V(network)$role))]
# plotting the network
plot(network, vertex.color=my_color, vertex.shape = c("rectangle"),
     vertex.size = 40, vertex.size2 = 30, vertex.label.cex=0.8,
     vertex.label.color="black", edge.width=igraph::E(network)$sig,
     edge.color="black")
}

# select TF "BATF" and extract its connections
plot_network(enrich_motif_pairs_filtered, TF_name = "BATF",
             color_TF = "#70d9e0", color_bind_TF = "#e841da")
```

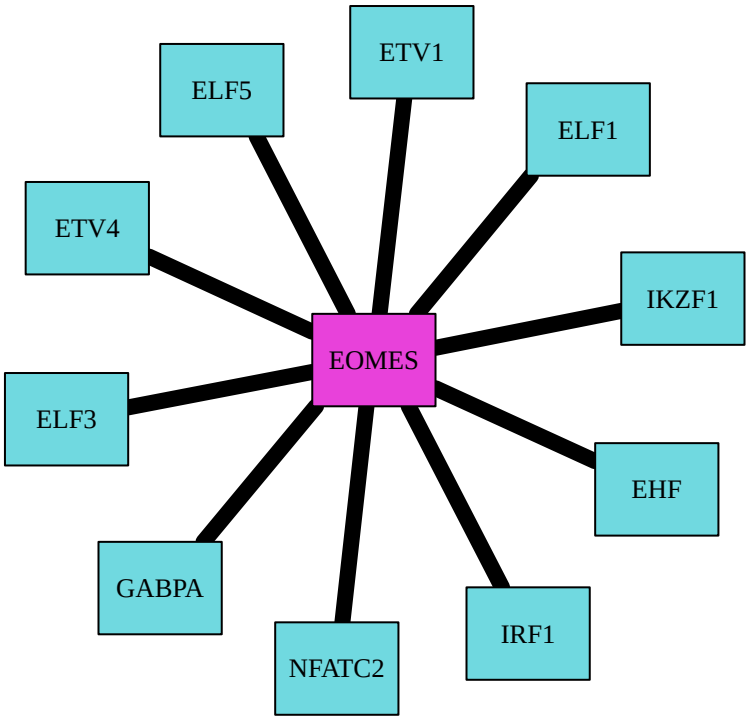



```
# select TF "GATA1" and extract its connections
plot_network(enrich_motif_pairs_filtered, TF_name = "GATA1",
             color_TF = "#70d9e0", color_bind_TF = "#e841da")
```




```
# select TF "EOMES" and extract its connections
plot_network(enrich_motif_pairs_filtered, TF_name = "EOMES",
```

```
                color_TF = "#70d9e0", color_bind_TF = "#e841da")
```



Special use cases:

Finding enriched TF motif pairs from Jolma et al.,2015.

In the paper Jolma et al.,2015, the authors report PWM for pairs of TFs and if you want to find enrichment of these TF pairs in the input peaks one can use the function `findEnrichMotifPair_Jolma2015`.

Here, we demonstrate this functionality by using genomic regions from two conditions, for instance, cells differentiating from one cell fate to another. We obtained ATAC-seq data in Th0 cells (activated T cells) moving to Th2. We can find enriched TF pairs in Th2 cells relative to Th0 cells. To do so, we need to provide ATAC-seq peaks from Th2 cells as the input set and ATAC-seq peaks from Th0 cells as the control set in the `enrichmotifpairR` package.

```
# Finding the enriched motif pairs
results <- findEnrichMotifPair_Jolma2015(
  target_data = example_peaks_data$Th2_ATAC_seq_peaks,
  background_data = example_peaks_data$Th0_ATAC_seq_peaks,
  genome_ver = "hg19",
  scramble_data = FALSE,
  Pvalue_computation = "hyper",
  Pvalue_threshold = 0.01,
  Pvalue_adjust_method = "BH"
)
```

The enriched TF motif pairs are stored in the `results`.

```
enrich_motif_pairs <- results
# The output of the enriched motif pairs
enrich_motif_pairs[, 2:7] %>% head(10) %>%
    kable(., caption="Top 10 motif pairs")
```

Table 3: Top 10 motif pairs

TF_pair_name	TF_name_1	TF_name_2	fold_enrich	pval	pval_adj
ETV2_FOXI1	ETV2	FOXI1	1.2572627	0	0
GCM1_NHLH1	GCM1	NHLH1	1.2008419	0	0
FLI1_FOXI1	FLI1	FOXI1	1.2438104	0	0
GCM1_FIGLA	GCM1	FIGLA	1.1798084	0	0
GCM1_NHLH1	GCM1	NHLH1	1.1969086	0	0
GCM1_FIGLA	GCM1	FIGLA	1.2093465	0	0
FOXO1_ELK1	FOXO1	ELK1	1.2203088	0	0
FOXJ2_ELF1	FOXJ2	ELF1	1.2086310	0	0
ERF_FOXI1	ERF	FOXI1	1.2259755	0	0
E2F3_EOMES	E2F3	EOMES	1.3086354	0	0

```
# remove the duplicate motif pairs
enrich_motif_pairs_filtered <- enrich_motif_pairs %>%
    group_by(TF_name_1, TF_name_2) %>%
    dplyr::distinct(TF_name_1, TF_name_2, .keep_all = TRUE)
enrich_motif_pairs_filtered[, 2:7] %>% head(10) %>%
    kable(., caption="Top 10 motif pairs after removing duplicates")
```

Table 4: Top 10 motif pairs after removing duplicates

TF_pair_name	TF_name_1	TF_name_2	fold_enrich	pval	pval_adj
ETV2_FOXI1	ETV2	FOXI1	1.2572627	0	0
GCM1_NHLH1	GCM1	NHLH1	1.2008419	0	0
FLI1_FOXI1	FLI1	FOXI1	1.2438104	0	0
GCM1_FIGLA	GCM1	FIGLA	1.1798084	0	0
FOXO1_ELK1	FOXO1	ELK1	1.2203088	0	0
FOXJ2_ELF1	FOXJ2	ELF1	1.2086310	0	0
ERF_FOXI1	ERF	FOXI1	1.2259755	0	0
E2F3_EOMES	E2F3	EOMES	1.3086354	0	0
ETV5_FIGLA	ETV5	FIGLA	1.2211229	0	0
FOXO1_ELF1	FOXO1	ELF1	1.2113878	0	0

Directly finding all possible enriched TF motif pairs without first looking for individual enriched motifs.

If users are interested in finding all possible enriched TF motif pairs without first looking for individual enriched motifs, one can use the function `findEnrichMotifPairAll`.

Here, we demonstrate this functionality by using genomic regions from two conditions, for instance, cells differentiating from one cell fate to another. We obtained ATAC-seq data in Th0 cells (activated T cells) moving to Th1. We can find all enriched TFs pairs in Th1 cells relative to Th0 cells. To do so, we need to provide ATAC-seq peaks from Th1 cells as the input set and ATAC-seq peaks from Th0 cells as the control set in the `enrichmotifpairR` package. In this case, we are selecting motifs from `JASPAR-UNVALIDATED` database.

```
# Finding the enriched motifs and their partners
results <- findEnrichMotifPairAll(
  target_data = example_peaks_data$Th1_ATAC_seq_peaks,
  background_data = example_peaks_data$Th0_ATAC_seq_peaks,
  genome_ver = "hg19",
```

```
  scramble_data = FALSE,
  motif_database = "JASPAR_UNVALIDATED",
  Pvalue_computation = "hyper",
  Pvalue_threshold = 0.01,
  Pvalue_adjust_method = "BH"
)
```

The enriched TF motif pairs are stored in the `results`. These pairs contain duplicates (redundant entries) and should be removed. To do so one can make use of the function `removeDuplicateTFPairs`. Also filter for only TF genes. TF genes are defined by Lambert et al., 2018.

```
# assign the results data to individual objects
enrich_motif_pairs <- results
# remove duplicate pairs
removeDuplicateTFPairs <- function(data = data){
    data <- data %>% dplyr::arrange(pval_adj) %>%
        dplyr::mutate(key = paste0(pmin(TF_name_1, TF_name_2),
                                   pmax(TF_name_1, TF_name_2), sep = "")) %>%
        dplyr::distinct(key, .keep_all = TRUE) %>%
        dplyr::filter(TF_name_1 != TF_name_2) %>%
        dplyr::select(-key)
    return(data)
}
enrich_motif_pairs <- removeDuplicateTFPairs(data = enrich_motif_pairs)
# get the list of TFs genes
TF_df <- TF_df %>% dplyr::filter(TF_Yes_No == "Yes")
# filter TFs motifs for only TFs genes based on Lambert et al., 2018
enrich_motif_pairs_filtered <- enrich_motif_pairs %>%
    dplyr::distinct(TF_name_1, TF_name_2, .keep_all = TRUE) %>%
    dplyr::filter(TF_name_1 %in% TF_df$Name) %>%
    dplyr::filter(TF_name_2 %in% TF_df$Name) %>%
    dplyr::filter(pval_adj < 1e-05)
```

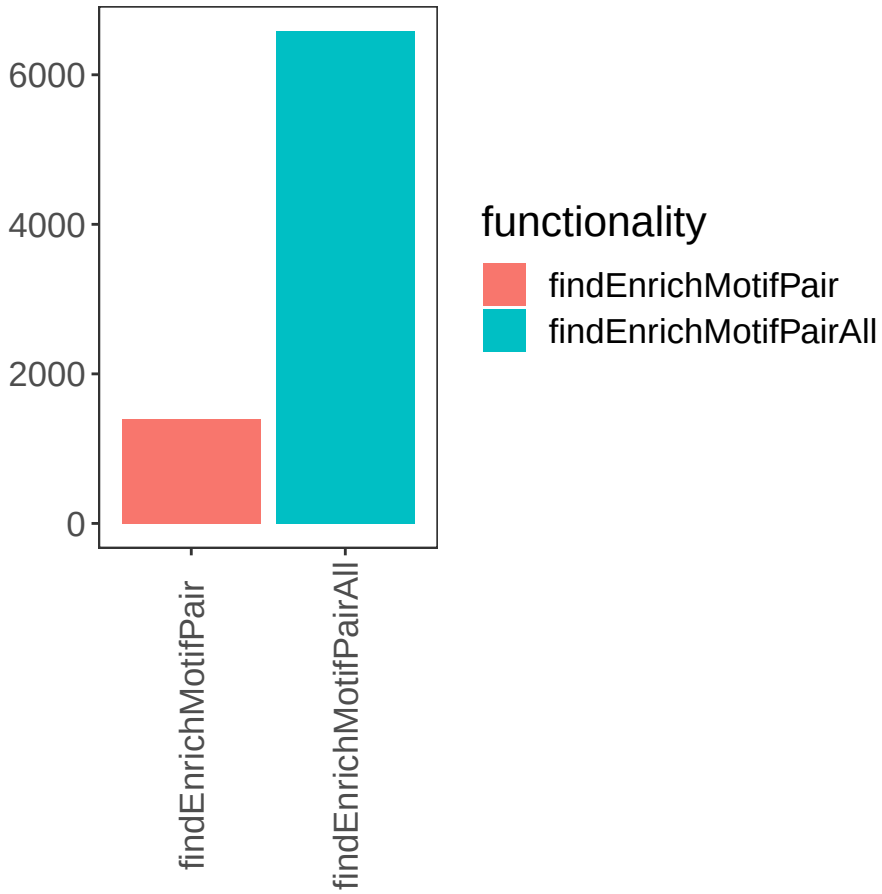
Compare these TF motif pairs with the motif pairs from the functionality of `findEnrichMotifPair`.

```
# Finding the enriched motifs and their partners
results <- findEnrichMotifPair(
  target_data = example_peaks_data$Th1_ATAC_seq_peaks,
  background_data = example_peaks_data$Th0_ATAC_seq_peaks,
  genome_ver = "hg19",
  scramble_data = FALSE,
  motif_database = "JASPAR_UNVALIDATED",
  Pvalue_computation = "hyper",
  Pvalue_threshold = 0.01,
  Pvalue_adjust_method = "BH"
)

# assign the results data to individual objects
enrich_motif_pairs_2 <- results$motif_pair_enrich
# remove duplicate pairs
removeDuplicateTFPairs <- function(data = data){
    data <- data %>% dplyr::arrange(pval_adj) %>%
        dplyr::mutate(key = paste0(pmin(TF_name_1, TF_name_2),
                                   pmax(TF_name_1, TF_name_2), sep = "")) %>%
        dplyr::distinct(key, .keep_all = TRUE) %>%
        dplyr::filter(TF_name_1 != TF_name_2) %>%
```

```
        dplyr::select(-key)
    return(data)
}
enrich_motif_pairs_2 <- removeDuplicateTFPairs(data = enrich_motif_pairs_2)
# get the list of TFs genes
TF_df <- TF_df %>% dplyr::filter(TF_Yes_No == "Yes")
# filter TFs motifs for only TFs genes based on Lambert et al., 2018
enrich_motif_pairs_filtered_2 <- enrich_motif_pairs_2 %>%
    dplyr::distinct(TF_name_1, TF_name_2, .keep_all = TRUE) %>%
    dplyr::filter(TF_name_1 %in% TF_df$Name) %>%
    dplyr::filter(TF_name_2 %in% TF_df$Name) %>%
    dplyr::filter(pval_adj < 1e-05)
```

Comparison plot

```
# create data frame to plot the number of TF pairs comparison plot
data <- data.frame(functionality = c("findEnrichMotifPair", "findEnrichMotifPairAll"), value = c(nrow(en

# bar plot
pp <- ggplot(data = data,
             aes(x = functionality, y = value, fill = functionality)) +
    geom_bar(stat = "identity") +
    ylab("") + xlab("") + theme_bw() +
    theme(plot.background = element_blank()
    ,panel.grid.major = element_blank()
    ,panel.grid.minor = element_blank()
    ,text = element_text(size=16), axis.text.x = element_text(angle=90, vjust=0.5)
)
pp
```



As you can see, `findEnrichMotifPair` determines relatively fewer enriched motif pairs than `findEnrichMotifPairAll`, as the former finds only the pairs associated with individually enriched motifs.