Enrichment analysis for genes in GRN motifs

Fabricio Almeida-Silva

2023-05-03

```
library(here)
```

```
## here() starts at /home/faalm/Dropbox/projects/polyploid_GRNs
```

```
library(magrene)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.2     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(BioNERO)
```

```
##
##
## Attaching package: 'BioNERO'
##
## The following object is masked from 'package:tidyr':
##
##     replace_na
```

```
library(ComplexHeatmap)
```

```
## Loading required package: grid
## ========================================
## ComplexHeatmap version 2.16.0
## Bioconductor page: http://bioconductor.org/packages/ComplexHeatmap/
## Github page: https://github.com/jokergoo/ComplexHeatmap
## Documentation: http://jokergoo.github.io/ComplexHeatmap-reference
##
## If you use it in published research, please cite either one:
## - Gu, Z. Complex Heatmap Visualization. iMeta 2022.
## - Gu, Z. Complex heatmaps reveal patterns and correlations in multidimensional
##     genomic data. Bioinformatics 2016.
##
##
## The new InteractiveComplexHeatmap package can directly export static
```

```
## complex heatmaps into an interactive Shiny app with zero effort. Have a try!
##
## This message can be suppressed by:
##   suppressPackageStartupMessages(library(ComplexHeatmap))
## ========================================
```

```
library(patchwork)

set.seed(123)
dup_palette <- c(SSD = "#1984c5", WGD = "#ffb400")
```

Overview

Here, we will describe the code to perform enrichment analyses to go deeper into the functions of genes in motifs. We will target motifs types differently:

- V and bifan motifs: enrichment for TF families using all TFs in the GRN as background;
- Lambda, delta, and bifan motifs: functional enrichment (GO and InterPro terms) using all targets in the GRN as background.

Enrichment analysis

Let's start by loading the required data.

```
# Load annotation
load(here("data", "functional_annotation.rda"))

# Load TFs
load(here("data", "tfs.rda"))

# Load GRN
load(here("products", "result_files", "grns.rda"))

## Create a list of TFs and targets for each GRN to use as background
bg_tfs <- lapply(grns, function(x) { return(unique(x$Node1)) })
bg_targets <- lapply(grns, function(x) { return(unique(x$Node2)) })
rm(grns)

# Load motifs
species <- names(functional_annotation)
files <- here(
    "products", "result_files", "motifs",
    paste0("motifs_", tolower(species), ".rda")
)
for(s in files) {
    load(s)
}

# Combine motifs for each species in a single object
motifs <- list(
    Athaliana = motifs_athaliana,
    Gmax = motifs_gmax,
    Ptrichocarpa = motifs_ptrichocarpa,
    Slycopersicum = motifs_slycopersicum,
```

```
    Vvinifera = motifs_vvinifera,
    Osativa = motifs_osativa,
    Zmays = motifs_zmays
)

rm(motifs_athaliana)
rm(motifs_gmax)
rm(motifs_ptrichocarpa)
rm(motifs_slycopersicum)
rm(motifs_vvinifera)
rm(motifs_osativa)
rm(motifs_zmays)
```

To make it easier, let's define wrapper functions around `BioNERO::enrichment_analysis()` that perform the enrichment analysis for each species and peak.

```
# Perform SEA and add columns with info on class, mode, motif, and peak
enrichment <- function(genes, background, annot,
                       mode = NULL, motif = NULL, peak = NULL) {

    if(is(annot, "data.frame")) {
        class <- names(annot)[2]
        enrich <- BioNERO::enrichment_analysis(
            genes, background, annot, column = class
        )
        if(!is.null(enrich)) {
            enrich$Class <- class
            enrich$Mode <- mode
            enrich$Motif <- motif
            if(!is.null(peak)) { enrich$Peak <- peak }
        }

    } else {
        classes <- c("GOBP", "GOMF", "InterPro")
        enrich <- lapply(classes, function(x) {
            annot_df <- annot[[x]]
            col <- names(annot_df)[2]
            res <- BioNERO::enrichment_analysis(
                genes, background, annot_df, column = col
            )
            if(!is.null(res)) {
                res$Class <- x
                res$Mode <- mode
                res$Motif <- motif
                if(!is.null(peak)) { res$Peak <- peak }
            }
            return(res)
        })
        enrich <- Reduce(rbind, enrich)
    }
    return(enrich)
}
```

```
# Perform SEA for each species, by peak and mode
sea <- function(motifs, annot_list, tf_list, bg_targets, bg_tfs, species) {

    # Iterate through each species
    enrichment_res <- Reduce(rbind, lapply(species, function(x) {
        message("Working on species ", x)
        # Functional annotation and TFs
        annot <- annot_list[[x]]
        tf <- tf_list[[x]]

        # Background genes
        background_targets <- bg_targets[[x]]
        background_tfs <- bg_tfs[[x]]

        peaks <- names(motifs[[x]])
        # Iterate through each peak
        bypeak <- Reduce(rbind, lapply(peaks, function(y) {
            message("Peak: ", y)

            events <- c("WGD", "SSD")
            byevent <- Reduce(rbind, lapply(events, function(z) {
                message("Mode: ", z)
                # legend: x = species; y = peak; z = mode
                motif_vec <- motifs[[x]][[y]][[z]]

                # V motifs
                v_genes <- unique(gsub("->.*", "", gsub(".*<-", "", motif_vec$V)))
                v_sea <- enrichment(v_genes, background_tfs, tf, z, "V", y)

                # Bifan motifs - TF
                bifan_tf <- gsub("->.*", "", motif_vec$bifan)
                bifan_tf <- unique(unlist(strsplit(bifan_tf, ",")))
                bifan_sea_tf <- NULL
                if(length(bifan_tf) != 0) {
                    bifan_sea_tf <- enrichment(
                        bifan_tf, background_tfs, tf, z, "bifan", y
                    )
                }

                # Bifan motifs - target
                bifan_genes <- gsub(".*->", "", motif_vec$bifan)
                bifan_genes <- unique(unlist(strsplit(bifan_genes, ",")))
                bifan_sea_tar <- NULL
                if(length(bifan_genes) != 0) {
                    bifan_sea_tar <- enrichment(
                        bifan_genes, background_targets, annot, z, "bifan", y
                    )
                }

                # Lambda
                lambda_genes <- gsub("<-.*->", ",", motif_vec$lambda)
                lambda_genes <- unique(unlist(strsplit(lambda_genes, ",")))
                lambda_sea <- NULL
```

```
                if(length(lambda_genes) != 0) {
                    lambda_sea <- enrichment(
                        lambda_genes, background_targets, annot, z, "lambda", y
                    )
                }

                # Delta
                delta_genes <- gsub("<-.*->", ",", motif_vec$delta)
                delta_genes <- unique(unlist(strsplit(delta_genes, ",")))
                delta_sea <- NULL
                if(length(delta_genes) != 0) {
                    delta_sea <- enrichment(
                        delta_genes, background_targets, annot, z, "delta", y
                    )
                }

                # Combine results in a single data frame
                sea_final <- rbind(
                    v_sea, bifan_sea_tf, bifan_sea_tar
                )
                return(sea_final)
            }))

            return(byevent)
        }))
        if(!is.null(bypeak)) {
            bypeak$Species <- x
        }
        return(bypeak)
    }))
    return(enrichment_res)
}

# Broader SEA, not filtered by peak
sea_global <- function(motifs, annot_list, tf_list, bg_targets, bg_tfs, species) {

    # Iterate through each species
    enrichment_res <- Reduce(rbind, lapply(species, function(x) {
        message("Working on species ", x)
        # Functional annotation and TFs
        annot <- annot_list[[x]]
        tf <- tf_list[[x]]

        # Background genes
        background_targets <- bg_targets[[x]]
        background_tfs <- bg_tfs[[x]]

        # Create a list of motifs by mode (WGD and SSD), combining peaks
        peaks <- names(motifs[[x]])
        genes_wgd <- unlist(lapply(peaks, function(y) {
            motif_vec <- motifs[[x]][[y]]$WGD
            return(motif_vec)
        }), recursive = FALSE)
```

```
genes_wgd <- sapply(unique(names(genes_wgd)), function(x) {
    return(unname(unlist(genes_wgd[names(genes_wgd) == x])))
}, simplify=FALSE)

genes_ssd <- unlist(lapply(peaks, function(y) {
    motif_vec <- motifs[[x]][[y]]$SSD
    return(motif_vec)
}), recursive = FALSE)
genes_ssd <- sapply(unique(names(genes_ssd)), function(x) {
    return(unname(unlist(genes_ssd[names(genes_ssd) == x])))
}, simplify=FALSE)
mlist <- list(WGD = genes_wgd, SSD = genes_ssd)

# Iterate through `mlist` and perform SEA by mode
events <- c("WGD", "SSD")
byevent <- Reduce(rbind, lapply(events, function(z) {
    message("Mode: ", z)
    motif_vec <- mlist[[z]]

    # V motifs
    v_genes <- unique(gsub("->.*", "", gsub(".*<-", "", motif_vec$V)))
    v_sea <- enrichment(v_genes, background_tfs, tf, z, "V", y)

    # Bifan motifs - TF
    bifan_tf <- gsub("->.*", "", motif_vec$bifan)
    bifan_tf <- unique(unlist(strsplit(bifan_tf, ",")))
    bifan_sea_tf <- NULL
    if(length(bifan_tf) != 0) {
        bifan_sea_tf <- enrichment(
            bifan_tf, background_tfs, tf, z, "bifan", y
        )
    }

    # Bifan motifs - target
    bifan_genes <- gsub(".*->", "", motif_vec$bifan)
    bifan_genes <- unique(unlist(strsplit(bifan_genes, ",")))
    bifan_sea_tar <- NULL
    if(length(bifan_genes) != 0) {
        bifan_sea_tar <- enrichment(
            bifan_genes, background_targets, annot, z, "bifan", y
        )
    }

    # Lambda
    lambda_genes <- gsub("<-.*->", ",", motif_vec$lambda)
    lambda_genes <- unique(unlist(strsplit(lambda_genes, ",")))
    lambda_sea <- NULL
    if(length(lambda_genes) != 0) {
        lambda_sea <- enrichment(
            lambda_genes, background_targets, annot, z, "lambda", y
        )
    }
```

```
            # Delta
            delta_genes <- gsub("<-.*->", ",", motif_vec$delta)
            delta_genes <- unique(unlist(strsplit(delta_genes, ",")))
            delta_sea <- NULL
            if(length(delta_genes) != 0) {
                delta_sea <- enrichment(
                    delta_genes, background_targets, annot, z, "delta", y
                )
            }

            # Combine results in a single data frame
            sea_final <- rbind(
                v_sea, bifan_sea_tf, bifan_sea_tar
            )
            return(sea_final)
        }))

        return(byevent)
    }))

    if(!is.null(enrichment_res)) {
        enrichment_res$Species <- x
    }
    return(enrichment_res)
}
```

Now, let's perform the SEA for each species.

```
annot_list <- functional_annotation
tf_list <- tfs

# Performing SEA for each species
sea_ath <- sea(motifs, annot_list, tf_list, bg_targets, bg_tfs, "Athaliana")
sea_gma <- sea(motifs, annot_list, tf_list, bg_targets, bg_tfs, "Gmax")
sea_ptr <- sea(motifs, annot_list, tf_list, bg_targets, bg_tfs, "Ptrichocarpa")
sea_sly <- sea(motifs, annot_list, tf_list, bg_targets, bg_tfs, "Slycopersicum")
sea_vvi <- sea(motifs, annot_list, tf_list, bg_targets, bg_tfs, "Vvinifera")
sea_osa <- sea(motifs, annot_list, tf_list, bg_targets, bg_tfs, "Osativa")
sea_zma <- sea(motifs, annot_list, tf_list, bg_targets, bg_tfs, "Zmays")

sea_by_peak <- rbind(sea_ath, sea_gma, sea_sly, sea_vvi, sea_osa, sea_zma)
sea_by_peak$GeneID <- NULL
readr::write_tsv(
    sea_by_peak,
    file = here("products", "tables", "motif_functional_enrichment_by_peak.tsv")
)
```

To conclude, let's perform another SEA, but not by peak. Here, we will only perform the SEA for WGD- and SSD- derived gene pairs in motifs.

```
# Perform global SEA
seag_ath <- sea_global(motifs, annot_list, tf_list, bg_targets, bg_tfs, "Athaliana")
seag_gma <- sea_global(motifs, annot_list, tf_list, bg_targets, bg_tfs, "Gmax")
seag_ptr <- sea_global(motifs, annot_list, tf_list, bg_targets, bg_tfs, "Ptrichocarpa")
seag_sly <- sea_global(motifs, annot_list, tf_list, bg_targets, bg_tfs, "Slycopersicum")
```

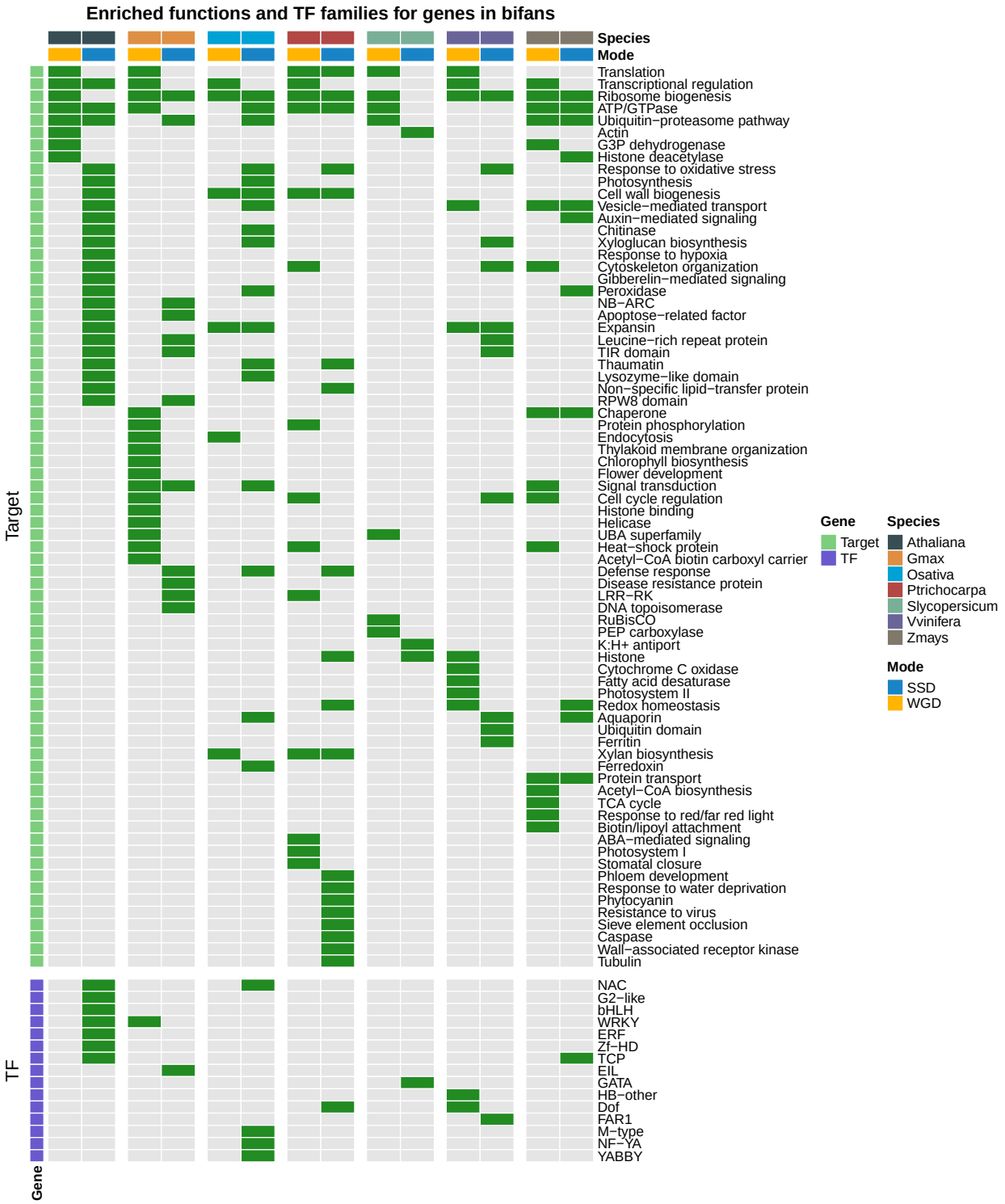
```
seag_vvi <- sea_global(motifs, annot_list, tf_list, bg_targets, bg_tfs, "Vvinifera")
seag_osa <- sea_global(motifs, annot_list, tf_list, bg_targets, bg_tfs, "Osativa")
seag_zma <- sea_global(motifs, annot_list, tf_list, bg_targets, bg_tfs, "Zmays")

# Combine results into a single data frame
sea_combined <- rbind(
    seag_ath %>% mutate(Species = "Athaliana"),
    seag_gma %>% mutate(Species = "Gmax"),
    seag_sly %>% mutate(Species = "Slycopersicum"),
    seag_vvi %>% mutate(Species = "Vvinifera"),
    seag_osa %>% mutate(Species = "Osativa"),
    seag_zma %>% mutate(Species = "Zmays"),
    seag_ptr %>% mutate(Species = "Ptrichocarpa")
)
sea_combined$GeneID <- NULL
sea_combined$Peak <- NULL
readr::write_tsv(
    sea_combined,
    file = here("products", "tables", "motif_functional_enrichment_by_mode.tsv")
)
```

After manual inspection of the enrichment results, I created the table *summarized_functional_enrichment_of_motifs.csv*, which contains summarized enrichment results for visual exploration. One important thing to observe is that the only class of motifs with enriched functions/TF families was bifans.

To conclude, let's visualize enrichment results as a presence/absence heatmap.

```
# Load data
sum_enrich <- readr::read_csv(
    here("products", "tables", "summarized_functional_enrichment_of_motifs.csv"),
    show_col_types = FALSE
)

# Turn long data frame to gene/absence matrix
pav <- sum_enrich %>%
    mutate(present = 1) %>%
    mutate(cols = str_c(Species, "_", Mode)) %>%
    select(Class, present, cols) %>%
    pivot_wider(names_from = cols, values_from = present) %>%
    column_to_rownames("Class") %>%
    as.matrix()
pav[is.na(pav)] <- 0

# Create row annotation, column annotation, and annotation colors
## Column annotation: species, mode
annotation_col <- data.frame(
    row.names = colnames(pav),
    Mode = gsub(".*_", "", colnames(pav)),
    Species = gsub("_.*", "", colnames(pav))
)

## Row annotation: gene type (TF or target)
c <- sum_enrich %>%
    dplyr::select(Class, Motif) %>%
    distinct() %>%
```

```
    as.data.frame()

annotation_row <- data.frame(
    row.names = c$Class,
    Gene = c$Motif
)
annotation_row$Gene <- gsub("Targets", "Target", annotation_row$Gene)

## Annotation colors
annotation_colors <- list(
    Mode = dup_palette,
    Species = c(
        Athaliana = "#374E55FF", Gmax = "#DF8F44FF", Osativa = "#00A1D5FF",
        Ptrichocarpa = "#B24745FF", Slycopersicum = "#79AF97FF",
        Vvinifera = "#6A6599FF", Zmays = "#80796BFF"
    ),
    Gene = c(TF = "slateblue3", Target = "palegreen3")
)

# Plot heatmap
p_heatmap <- ComplexHeatmap::pheatmap(
    pav, name = "Presence/absence",
    main = "Enriched functions and TF families for genes in bifans",
    cluster_rows = FALSE, cluster_cols = FALSE,
    annotation_col = annotation_col,
    annotation_row = annotation_row,
    labels_col = rep("", ncol(pav)),
    annotation_colors = annotation_colors,
    column_split = annotation_col$Species,
    row_split = annotation_row$Gene,
    row_gap = unit(3, "mm"),
    column_gap = unit(3, "mm"),
    color = c("grey90", "forestgreen"), border_color = "white",
    legend = FALSE
)
p_heatmap
```



By looking at the figure, we can see some patterns:

- SSD-derived genes in bifans are enriched in stress-related genes. These include genes involved in response to oxidative stress, response to hypoxia, and different classes of pathogenesis-related proteins, such as chitinases and lysozymes, peroxidases, and thaumatins. There are also leucine-rich repeat receptor kinases, TIR domains, and wall-associated kinases.

- Overall, only SSD-derived genes were enriched in particular TF families. Although WGD-derived tend to be retained more often than SSD-derived genes, WGD-derived genes in bifans were not enriched in any specific TF family.
- SSD-derived genes are enriched in stress-related transcription factor families, such as NAC, G2-like, bHLH, WRKY, and ERF. This is in line with previous observations that stress-related TFs tend to be duplicated by tandem duplications. This mechanism is likely adaptive, as genes in tandem tend to be co-regulated. Then, keeping stress-related genes in tandem arrays would ensure that they are coordinately activated.
- WGD-derived genes in bifans are enriched in processes such as translation, flower development, transcriptional regulation, histone modifications, photosynthesis-related processes (e.g., chlorophyll biosynthesis, thylakoid membrane organization, RuBisCO, and photosystem I formation), cell cycle regulation, and carbohydrate and lipid metabolism.

```
# Save heatmap as PDF
p_heatmap_functional_enrichment <- p_heatmap
save(
    p_heatmap_functional_enrichment,
    file = here("products", "plots", "p_heatmap_functional_enrichment.rda"),
    compress = "xz"
)
```

Session info

This document was created under the following conditions:

```
sessioninfo::session_info()
```

```
## - Session info ---------------------------------------------------------------
##  setting  value
##  version  R version 4.3.0 (2023-04-21)
##  os       Ubuntu 20.04.5 LTS
##  system   x86_64, linux-gnu
##  ui       X11
##  language (EN)
##  collate  en_US.UTF-8
##  ctype    en_US.UTF-8
##  tz       Europe/Brussels
##  date     2023-05-03
##  pandoc   2.19.2 @ /usr/lib/rstudio/resources/app/bin/quarto/bin/tools/ (via rmarkdown)
##
## - Packages -------------------------------------------------------------------
##  package              * version   date (UTC) lib source
##  abind                  1.4-5     2016-07-21 [1] CRAN (R 4.3.0)
##  annotate               1.78.0    2023-04-25 [1] Bioconductor
##  AnnotationDbi          1.62.0    2023-04-25 [1] Bioconductor
##  backports              1.4.1     2021-12-13 [1] CRAN (R 4.3.0)
##  base64enc              0.1-3     2015-07-28 [1] CRAN (R 4.3.0)
##  Biobase                2.60.0    2023-04-25 [1] Bioconductor
##  BiocGenerics           0.46.0    2023-04-25 [1] Bioconductor
##  BiocParallel           1.34.0    2023-04-25 [1] Bioconductor
##  BioNERO              * 1.8.0     2023-04-25 [1] Bioconductor
##  Biostrings             2.68.0    2023-04-25 [1] Bioconductor
##  bit                    4.0.5     2022-11-15 [1] CRAN (R 4.3.0)
##  bit64                  4.0.5     2020-08-30 [1] CRAN (R 4.3.0)
```

```
##  bitops                1.0-7    2021-04-24 [1] CRAN (R 4.3.0)
##  blob                  1.2.4    2023-03-17 [1] CRAN (R 4.3.0)
##  cachem                1.0.8    2023-05-01 [1] CRAN (R 4.3.0)
##  checkmate             2.2.0    2023-04-27 [1] CRAN (R 4.3.0)
##  circlize              0.4.15   2022-05-10 [1] CRAN (R 4.3.0)
##  cli                   3.6.1    2023-03-23 [1] CRAN (R 4.3.0)
##  clue                  0.3-64   2023-01-31 [1] CRAN (R 4.3.0)
##  cluster               2.1.4    2022-08-22 [4] CRAN (R 4.2.1)
##  coda                  0.19-4   2020-09-30 [1] CRAN (R 4.3.0)
##  codetools             0.2-19   2023-02-01 [4] CRAN (R 4.2.2)
##  colorspace            2.1-0    2023-01-23 [1] CRAN (R 4.3.0)
##  ComplexHeatmap      * 2.16.0   2023-04-25 [1] Bioconductor
##  crayon                1.5.2    2022-09-29 [1] CRAN (R 4.3.0)
##  data.table            1.14.8   2023-02-17 [1] CRAN (R 4.3.0)
##  DBI                   1.1.3    2022-06-18 [1] CRAN (R 4.3.0)
##  DelayedArray          0.26.1   2023-05-01 [1] Bioconductor
##  digest                0.6.31   2022-12-11 [1] CRAN (R 4.3.0)
##  doParallel            1.0.17   2022-02-07 [1] CRAN (R 4.3.0)
##  dplyr               * 1.1.2    2023-04-20 [1] CRAN (R 4.3.0)
##  dynamicTreeCut        1.63-1   2016-03-11 [1] CRAN (R 4.3.0)
##  edgeR                 3.42.0   2023-04-25 [1] Bioconductor
##  evaluate              0.20     2023-01-17 [1] CRAN (R 4.3.0)
##  fansi                 1.0.4    2023-01-22 [1] CRAN (R 4.3.0)
##  fastcluster           1.2.3    2021-05-24 [1] CRAN (R 4.3.0)
##  fastmap               1.1.1    2023-02-24 [1] CRAN (R 4.3.0)
##  forcats             * 1.0.0    2023-01-29 [1] CRAN (R 4.3.0)
##  foreach               1.5.2    2022-02-02 [1] CRAN (R 4.3.0)
##  foreign               0.8-82   2022-01-13 [4] CRAN (R 4.1.2)
##  Formula               1.2-5    2023-02-24 [1] CRAN (R 4.3.0)
##  genefilter            1.82.0   2023-04-25 [1] Bioconductor
##  generics              0.1.3    2022-07-05 [1] CRAN (R 4.3.0)
##  GENIE3                1.22.0   2023-04-25 [1] Bioconductor
##  GenomeInfoDb          1.36.0   2023-04-25 [1] Bioconductor
##  GenomeInfoDbData      1.2.10   2023-04-28 [1] Bioconductor
##  GenomicRanges         1.52.0   2023-04-25 [1] Bioconductor
##  GetoptLong            1.0.5    2020-12-15 [1] CRAN (R 4.3.0)
##  ggnetwork             0.5.12   2023-03-06 [1] CRAN (R 4.3.0)
##  ggnewscale            0.4.8    2022-10-06 [1] CRAN (R 4.3.0)
##  ggplot2             * 3.4.2    2023-04-03 [1] CRAN (R 4.3.0)
##  ggrepel               0.9.3    2023-02-03 [1] CRAN (R 4.3.0)
##  GlobalOptions         0.1.2    2020-06-10 [1] CRAN (R 4.3.0)
##  glue                  1.6.2    2022-02-24 [1] CRAN (R 4.3.0)
##  GO.db                 3.17.0   2023-05-02 [1] Bioconductor
##  gridExtra             2.3      2017-09-09 [1] CRAN (R 4.3.0)
##  gtable                0.3.3    2023-03-21 [1] CRAN (R 4.3.0)
##  here                * 1.0.1    2020-12-13 [1] CRAN (R 4.3.0)
##  highr                 0.10     2022-12-22 [1] CRAN (R 4.3.0)
##  Hmisc                 5.0-1    2023-03-08 [1] CRAN (R 4.3.0)
##  hms                   1.1.3    2023-03-21 [1] CRAN (R 4.3.0)
##  htmlTable             2.4.1    2022-07-07 [1] CRAN (R 4.3.0)
##  htmltools             0.5.5    2023-03-23 [1] CRAN (R 4.3.0)
##  htmlwidgets           1.6.2    2023-03-17 [1] CRAN (R 4.3.0)
##  httr                  1.4.5    2023-02-24 [1] CRAN (R 4.3.0)
##  igraph                1.4.2    2023-04-07 [1] CRAN (R 4.3.0)
```

```
##  impute                1.74.0    2023-04-25 [1] Bioconductor
##  intergraph            2.0-2     2016-12-05 [1] CRAN (R 4.3.0)
##  IRanges               2.34.0    2023-04-25 [1] Bioconductor
##  iterators             1.0.14    2022-02-05 [1] CRAN (R 4.3.0)
##  KEGGREST              1.40.0    2023-04-25 [1] Bioconductor
##  knitr                 1.42      2023-01-25 [1] CRAN (R 4.3.0)
##  lattice               0.20-45   2021-09-22 [4] CRAN (R 4.2.0)
##  lifecycle             1.0.3     2022-10-07 [1] CRAN (R 4.3.0)
##  limma                 3.56.0    2023-04-25 [1] Bioconductor
##  locfit                1.5-9.7   2023-01-02 [1] CRAN (R 4.3.0)
##  lubridate           * 1.9.2     2023-02-10 [1] CRAN (R 4.3.0)
##  magrene             * 1.2.0     2023-04-25 [1] Bioconductor
##  magrittr              2.0.3     2022-03-30 [1] CRAN (R 4.3.0)
##  Matrix                1.5-1     2022-09-13 [4] CRAN (R 4.2.1)
##  MatrixGenerics        1.12.0    2023-04-25 [1] Bioconductor
##  matrixStats           0.63.0    2022-11-18 [1] CRAN (R 4.3.0)
##  memoise               2.0.1     2021-11-26 [1] CRAN (R 4.3.0)
##  mgcv                  1.8-41    2022-10-21 [4] CRAN (R 4.2.1)
##  minet                 3.58.0    2023-04-25 [1] Bioconductor
##  munsell               0.5.0     2018-06-12 [1] CRAN (R 4.3.0)
##  NetRep                1.2.6     2023-01-06 [1] CRAN (R 4.3.0)
##  network               1.18.1    2023-01-24 [1] CRAN (R 4.3.0)
##  networkD3             0.4       2017-03-18 [1] CRAN (R 4.3.0)
##  nlme                  3.1-162   2023-01-31 [4] CRAN (R 4.2.2)
##  nnet                  7.3-18    2022-09-28 [4] CRAN (R 4.2.1)
##  patchwork           * 1.1.2     2022-08-19 [1] CRAN (R 4.3.0)
##  pillar                1.9.0     2023-03-22 [1] CRAN (R 4.3.0)
##  pkgconfig             2.0.3     2019-09-22 [1] CRAN (R 4.3.0)
##  plyr                  1.8.8     2022-11-11 [1] CRAN (R 4.3.0)
##  png                   0.1-8     2022-11-29 [1] CRAN (R 4.3.0)
##  preprocessCore        1.62.0    2023-04-25 [1] Bioconductor
##  purrr               * 1.0.1     2023-01-10 [1] CRAN (R 4.3.0)
##  R6                    2.5.1     2021-08-19 [1] CRAN (R 4.3.0)
##  RColorBrewer          1.1-3     2022-04-03 [1] CRAN (R 4.3.0)
##  Rcpp                  1.0.10    2023-01-22 [1] CRAN (R 4.3.0)
##  RCurl                 1.98-1.12 2023-03-27 [1] CRAN (R 4.3.0)
##  readr               * 2.1.4     2023-02-10 [1] CRAN (R 4.3.0)
##  reshape2              1.4.4     2020-04-09 [1] CRAN (R 4.3.0)
##  RhpcBLASctl           0.23-42   2023-02-11 [1] CRAN (R 4.3.0)
##  rjson                 0.2.21    2022-01-09 [1] CRAN (R 4.3.0)
##  rlang                 1.1.1     2023-04-28 [1] CRAN (R 4.3.0)
##  rmarkdown             2.21      2023-03-26 [1] CRAN (R 4.3.0)
##  rpart                 4.1.19    2022-10-21 [4] CRAN (R 4.2.1)
##  rprojroot             2.0.3     2022-04-02 [1] CRAN (R 4.3.0)
##  RSQLite               2.3.1     2023-04-03 [1] CRAN (R 4.3.0)
##  rstudioapi            0.14      2022-08-22 [1] CRAN (R 4.3.0)
##  S4Arrays              1.0.1     2023-05-01 [1] Bioconductor
##  S4Vectors             0.38.0    2023-04-25 [1] Bioconductor
##  scales                1.2.1     2022-08-20 [1] CRAN (R 4.3.0)
##  sessioninfo           1.2.2     2021-12-06 [1] CRAN (R 4.3.0)
##  shape                 1.4.6     2021-05-19 [1] CRAN (R 4.3.0)
##  statmod               1.5.0     2023-01-06 [1] CRAN (R 4.3.0)
##  statnet.common        4.8.0     2023-01-24 [1] CRAN (R 4.3.0)
##  stringi               1.7.12    2023-01-11 [1] CRAN (R 4.3.0)
```

```
##  stringr              * 1.5.0     2022-12-02 [1] CRAN (R 4.3.0)
##  SummarizedExperiment   1.30.1    2023-05-01 [1] Bioconductor
##  survival               3.5-3     2023-02-12 [4] CRAN (R 4.2.2)
##  sva                    3.48.0    2023-04-25 [1] Bioconductor
##  tibble               * 3.2.1     2023-03-20 [1] CRAN (R 4.3.0)
##  tidyr                * 1.3.0     2023-01-24 [1] CRAN (R 4.3.0)
##  tidyselect             1.2.0     2022-10-10 [1] CRAN (R 4.3.0)
##  tidyverse            * 2.0.0     2023-02-22 [1] CRAN (R 4.3.0)
##  timechange             0.2.0     2023-01-11 [1] CRAN (R 4.3.0)
##  tzdb                   0.3.0     2022-03-28 [1] CRAN (R 4.3.0)
##  utf8                   1.2.3     2023-01-31 [1] CRAN (R 4.3.0)
##  vctrs                  0.6.2     2023-04-19 [1] CRAN (R 4.3.0)
##  vroom                  1.6.3     2023-04-28 [1] CRAN (R 4.3.0)
##  WGCNA                  1.72-1    2023-01-18 [1] CRAN (R 4.3.0)
##  withr                  2.5.0     2022-03-03 [1] CRAN (R 4.3.0)
##  xfun                   0.39      2023-04-20 [1] CRAN (R 4.3.0)
##  XML                    3.99-0.14 2023-03-19 [1] CRAN (R 4.3.0)
##  xtable                 1.8-4     2019-04-21 [1] CRAN (R 4.3.0)
##  XVector                0.40.0    2023-04-25 [1] Bioconductor
##  yaml                   2.3.7     2023-01-23 [1] CRAN (R 4.3.0)
##  zlibbioc               1.46.0    2023-04-25 [1] Bioconductor
##
##  [1] /home/faalm/R/x86_64-pc-linux-gnu-library/4.3
##  [2] /usr/local/lib/R/site-library
##  [3] /usr/lib/R/site-library
##  [4] /usr/lib/R/library
##
## ------------------------------------------------------------------------------
```