Identification and classification of duplicated genes

Fabricio Almeida-Silva

2022-08-09

Contents

```r
library(here)
source(here("code", "utils.R"))
library(doubletrouble)
library(tidyverse)
library(syntenet)

set.seed(123)  # for reproducibility
dup_palette <- c("#1984c5", "#ffb400")
```

1 Overview

Here, we will describe the code to:

1. Identify duplicate gene pairs;
2. Classify duplicate gene pairs as **WGD-derived pairs** or **SSD-derived pairs**;
3. Calculate Ka, Ks, and Ka/Ks for each duplicate pair;
4. Separate duplicate pairs by Ks peaks;
5. Classify duplicated genes as **WGD-derived genes** or **SSD-derived genes**;

2 Identication and classification of duplicate gene pairs

Here, we will identify duplicate pairs and classify them with *doubletrouble*. We will use a binary classification scheme, so duplicate pairs will be classified as either **WGD-derived** or **SSD-derived**.

```r
#----Load sequence and annotation data----------------------------------------
load(here("data", "annotation.rda"))
proteomes <- get_proteomes()

#----Process input data-------------------------------------------------------
check_input(proteomes, annotation)
pdata <- process_input(proteomes, annotation)

#----Run DIAMOND only within species------------------------------------------
diamond_output <- here("products", "result_files", "blast_intra.rda")
if (!file.exists(diamond_output)) {
    blast_intra <- lapply(seq_along(pdata$seq), function(x) {

        l <- list(pdata$seq[[x]])
        species <- names(pdata$seq)[x]
        names(l) <- species

        blast <- run_diamond(seq = l, ... = "--sensitive")
        blast <- blast[paste0(species, "_", species)]
        return(blast)
    })
    blast_intra <- Reduce(c, blast_intra)
```

```
    # Save object to file
    save(blast_intra, file = diamond_output, compress = "xz")
} else {
    load(diamond_output)
}

#----Get anchor pairs------------------------------------------------------
anchors <- get_anchors_list(blast_list = blast_intra, annotation = pdata$annotation)
save(anchors, file = here("products", "result_files", "anchor_pairs_all_species.rda"),
    compress = "xz")

#----Classify duplicate pairs----------------------------------------------
class_duplicate_pairs <- classify_gene_pairs(blast_list = blast_intra,
    annotation = pdata$annotation, binary = TRUE)
save(class_duplicate_pairs, file = here("products", "result_files",
    "classified_dup_pairs.rda"), compress = "xz")
```

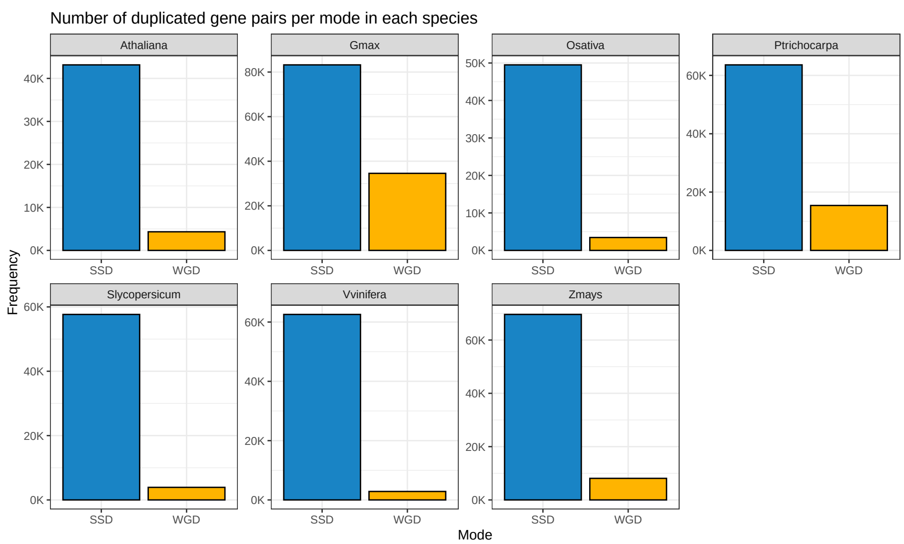
Now, let's explore some descriptive statistics per species.

```
library(tidyverse)
load(here("products", "result_files", "classified_dup_pairs.rda"))

# Count frequency of WGD and SSD-derived pairs per species
dup_stats <- lapply(seq_along(class_duplicate_pairs), function(x) {
    stats <- class_duplicate_pairs[[x]] %>%
        count(type) %>%
        mutate(Species = names(class_duplicate_pairs)[x])
    return(stats)
})
dup_stats <- Reduce(rbind, dup_stats)
names(dup_stats) <- c("Mode", "Frequency", "Species")

# Visualize results as a table
dup_stats %>%
    tidyr::pivot_wider(., names_from = Mode, values_from = Frequency) %>%
    knitr::kable()
```

Species	SSD	WGD
Osativa	49498	3435
Zmays	69611	8095
Vvinifera	62571	2860
Gmax	83219	34570
Slycopersicum	57664	3906
Ptrichocarpa	63613	15400
Athaliana	43154	4329

```
# Visualize results as a barplot
dup_stats_plot <- ggplot(dup_stats, aes(x = Mode, y = Frequency,
    fill = Mode)) + geom_bar(stat = "identity", color = "black") +
    scale_y_continuous(labels = scales::label_number(suffix = "K",
```

```
        scale = 0.001)) + theme_bw() + facet_wrap(~Species, scales = "free",
    nrow = 2) + scale_fill_manual(values = dup_palette) + theme(legend.position = "none") +
    labs(title = "Number of duplicated gene pairs per mode in each species")

dup_stats_plot
```




```
# Save plot
ggsave(dup_stats_plot, file = here("products", "plots", "duplicates_stats.png"),
    dpi = 300, width = 10, height = 6)
```

3 Calculating Ka, Ks, and Ka/Ks for gene pairs

Here, we will use this pipeline, which is associated with the following paper:

> Qiao, X., Li, Q., Yin, H., Qi, K., Li, L., Wang, R., ... & Paterson, A. H. (2019). Gene duplication and evolution in recurring polyploidization–diploidization cycles in plants. Genome biology, 20(1), 1-23.

This pipeline requires the following external dependencies:

- BioPerl
- PAL2NAL
- MAFFT
- KaKs_Calculator

The Ka/Ks calculation pipeline requires only 2 files:

- FASTA file with CDS
- A tab-separated table with duplicate pairs

We will export our R objects as temporary files, run the Perl script using the temporary files as input, and then delete these files. We will only keep the output file.

```
# Export CDS files to a directory named CDS/
cds_dir <- file.path(tempdir(), "cds")
```

```
if (!dir.exists(cds_dir)) {
    dir.create(cds_dir, recursive = TRUE)
}
cds <- get_cds()
export_cds <- lapply(seq_along(cds), function(x) {

    # Get file name
    filename <- paste0(names(cds)[[x]], ".fasta")

    # Export FASTA file
    Biostrings::writeXStringSet(cds[[x]], filepath = file.path(cds_dir,
        filename))
    return(NULL)
})


# Export duplicates to a directory named pairs/
pairs_dir <- file.path(tempdir(), "pairs")
if (!dir.exists(pairs_dir)) {
    dir.create(pairs_dir, recursive = TRUE)
}
load(here("products", "result_files", "classified_dup_pairs.rda"))
export_pairs <- lapply(seq_along(class_duplicate_pairs), function(x) {

    pairs <- class_duplicate_pairs[[x]]
    # Make it match the format required by the pipeline
    pairs <- data.frame(Duplicate1 = gsub("^[a-zA-Z]{3}_", "",
        pairs$dup1), Location = "Chr", Duplicate2 = gsub("^[a-zA-Z]{3}_",
        "", pairs$dup2), Location = "Chr", Evalue = 0)

    # Define path to output file
    filename <- paste0(names(class_duplicate_pairs)[[x]], ".pairs")
    outfile <- file.path(pairs_dir, filename)
    readr::write_tsv(pairs, file = outfile)
    return(NULL)
})
```

Now, let's download the pipeline.

```
# Bash
# Download pipeline
wget https://raw.githubusercontent.com/qiao-xin/Scripts_for_GB/master/calculate_Ka_Ks_pipeline/calculate_Ka_K

wget https://raw.githubusercontent.com/qiao-xin/Scripts_for_GB/master/calculate_Ka_Ks_pipeline/fa_prepareDAT

wget https://raw.githubusercontent.com/qiao-xin/Scripts_for_GB/master/calculate_Ka_Ks_pipeline/parseFastaInto

# Grant execute permission to all Perl scripts
chmod a+x *.pl

# Create output directory for Ka/Ks values
```

```
mkdir output_kaks
```

Now, run the following bash script:

```
#!/bin/bash
module load R/x86_64/4.1.3 gcc libpng perl KaKs_Calculator mafft

# Run the pipeline
# Usage: perl calculate_Ka_Ks_pipe.pl -d <cdsfile> -g <pairsfile> -o <outfile>
species=(Athaliana Gmax Osativa Ptrichocarpa Slycopersicum Vvinifera Zmays)

for s in "${species[@]}"
do
    perl calculate_Ka_Ks_pipe.pl -d cds/"$s".fasta -g pairs/"$s".pairs -o output_kaks/"$s"
    rm output_kaks/"$s".axt # Remove .axt file - too big and unnecessary
done
```

On the server, it can be executed with:

```
qsub -l h_vmem=50G calculate_ka_ks.sh
```

Now, let's read the output of `KaKs_Calculator` and save it as R objects.

```
# Read KaKs Calculator output and add duplication mode in
# another column
load(here("products", "result_files", "classified_dup_pairs.rda"))
species <- names(class_duplicate_pairs)

duplicate_pairs <- lapply(species, function(x) {

    message("Working on ", x)
    # Read KaKs Calculator output
    file <- here("output_kaks", x)
    kaks <- read.csv(file, header = FALSE, sep = "\t", skip = 2)
    names(kaks) <- c("dup1", "dup2", "ka", "ks", "kaks", "pvalue")

    # 1) Prepare object of duplicate pairs and modes for
    # merging
    duplicates <- class_duplicate_pairs[[x]]
    duplicates$dup1 <- gsub("^[a-zA-Z]{3}_", "", duplicates$dup1)
    duplicates$dup2 <- gsub("^[a-zA-Z]{3}_", "", duplicates$dup2)
    duplicates$pair <- paste0(duplicates$dup1, "_", duplicates$dup2)

    # 2) Prepare object of pairs and Ka/Ks values for
    # merging
    kaks$pair <- paste0(kaks$dup1, "_", kaks$dup2)

    # Merge objects
    duplicates_kaks <- merge(duplicates, kaks, by = "pair")
    dup_kaks_final <- duplicates_kaks[, c("dup1.x", "dup2.x",
        "type", "ka", "ks", "kaks", "pvalue")]
    names(dup_kaks_final)[1:2] <- c("dup1", "dup2")
```

```
    return(dup_kaks_final)
})
names(duplicate_pairs) <- species

save(duplicate_pairs, file = here("data", "duplicate_pairs.rda"),
    compress = "xz")
```

4 Classifying duplicate pairs by Ks peak

Here, we will split Ks pairs based on the Ks peak to which they belong (if there are more than 1). This is required to account for the impact of age in the number of connections in the network.

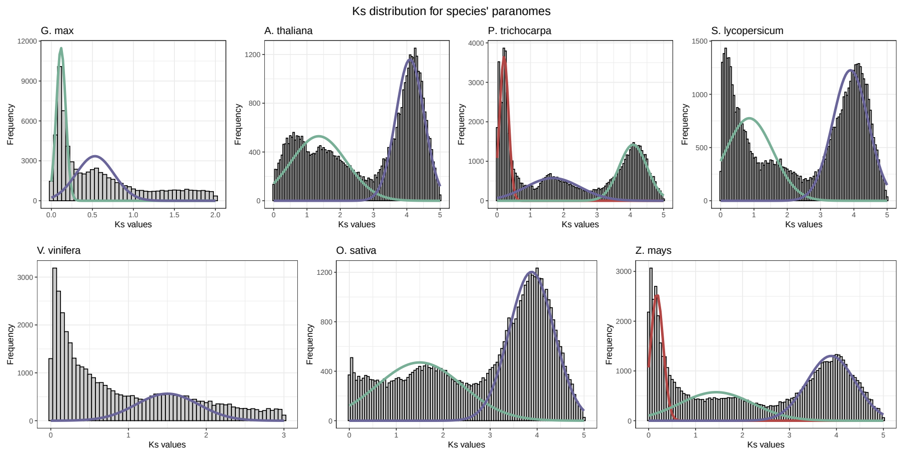
Here, we will use the following number of peaks for each species, based on previous works from Qiao et al., 2019. Genome Biology:

- *Glycine max*: 2 peaks
- *Oryza sativa:* 2 peaks
- *Zea mays*: 3 peaks
- *Solanum lycopersicum*: 2 peaks
- *Arabidopsis thaliana*: 2 peaks
- *Populus trichocarpa*: 2 or 3 peaks (if considering gamma-WGT)
- *Vitis vinifera*: 1 peak

First, let's identify peaks in Ks distros.

```
load(here("data", "duplicate_pairs.rda"))

# Remove Ks values > 5 and remove NA
kslist <- lapply(duplicate_pairs, function(x) {
    y <- x[!is.na(x$ks) & x$ks <= 5, ]
    return(y)
})

# Find peaks and plot them Gmax
gmax_peaks <- find_ks_peaks(kslist$Gmax$ks[kslist$Gmax$ks <=
    1], npeaks = 2)
## number of iterations= 25

ksplot_gmax <- plot_ks_peaks(kslist$Gmax$ks[kslist$Gmax$ks <=
    2], gmax_peaks) + ggplot2::ggtitle("G. max")

## Slycopersicum
slycopersicum_peaks <- find_ks_peaks(kslist$Slycopersicum$ks,
    npeaks = 2)
## number of iterations= 25

ksplot_slycopersicum <- plot_ks_peaks(kslist$Slycopersicum$ks,
    slycopersicum_peaks) + ggplot2::ggtitle("S. lycopersicum")
```

```r
## Osativa
osativa_peaks <- find_ks_peaks(kslist$Osativa$ks, npeaks = 2)
## number of iterations= 33

ksplot_osativa <- plot_ks_peaks(kslist$Osativa$ks, osativa_peaks) +
    ggplot2::ggtitle("O. sativa")


## Zmays
zmays_peaks <- find_ks_peaks(kslist$Zmays$ks, npeaks = 3)
## number of iterations= 42

ksplot_zmays <- plot_ks_peaks(kslist$Zmays$ks, zmays_peaks) +
    ggplot2::ggtitle("Z. mays")


## Athaliana
athaliana_peaks <- find_ks_peaks(kslist$Athaliana$ks, npeaks = 2)
## number of iterations= 14

ksplot_athaliana <- plot_ks_peaks(kslist$Athaliana$ks, athaliana_peaks) +
    ggplot2::ggtitle("A. thaliana")

## Ptrichocarpa
ptrichocarpa_peaks <- find_ks_peaks(kslist$Ptrichocarpa$ks, npeaks = 3)
## number of iterations= 36

ksplot_ptrichocarpa <- plot_ks_peaks(kslist$Ptrichocarpa$ks,
    ptrichocarpa_peaks) + ggplot2::ggtitle("P. trichocarpa")

## Vvinifera - manually done because k = 1
vvinifera_peaks <- list(mean = 1.5, sd = 0.4, lambda = 0.28)

ksplot_vvinifera <- plot_ks_peaks(kslist$Vvinifera$ks[kslist$Vvinifera$ks <=
    3], vvinifera_peaks) + ggplot2::ggtitle("V. vinifera")


# Combining all plots into one
ks_plots1 <- patchwork::wrap_plots(ksplot_gmax, ksplot_athaliana,
    ksplot_ptrichocarpa, ksplot_slycopersicum, nrow = 1)
ks_plots2 <- patchwork::wrap_plots(ksplot_vvinifera, ksplot_osativa,
    ksplot_zmays, nrow = 1)

ks_plots <- ggpubr::ggarrange(ks_plots1, ks_plots2, nrow = 2)
ks_plots <- ggpubr::annotate_figure(ks_plots, top = ggpubr::text_grob("Ks distribution for species' paranomes
    size = 15))
ks_plots
```



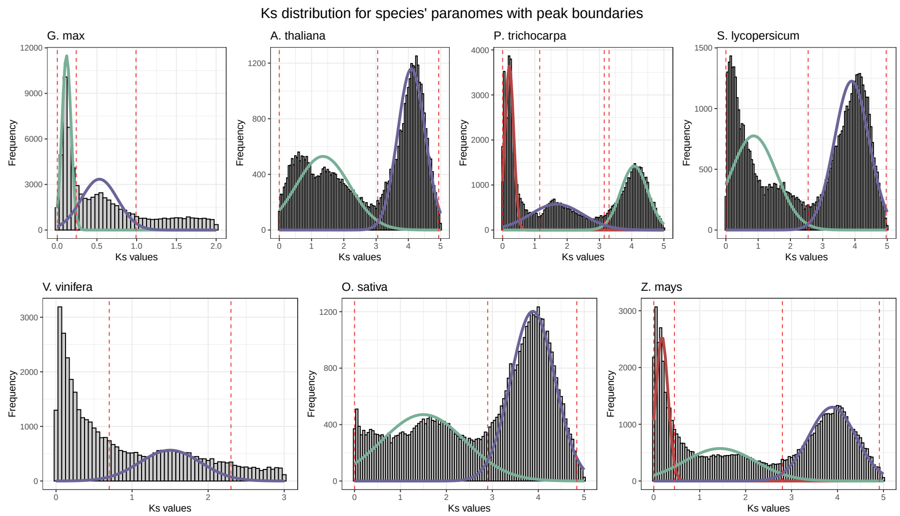

```
# Save peaks
ks_peaks <- list(Osativa = osativa_peaks, Gmax = gmax_peaks,
    Ptrichocarpa = ptrichocarpa_peaks, Athaliana = athaliana_peaks,
    Vvinifera = vvinifera_peaks, Zmays = zmays_peaks, Slycopersicum = slycopersicum_peaks)
save(ks_peaks, file = here("products", "result_files", "ks_peaks.rda"),
    compress = "xz")

# Save plots
ggplot2::ggsave(ks_plots, filename = here("products", "plots",
    "Ks_plots.png"), dpi = 300, width = 16, height = 8)
```

Now, let's split pairs by peak to which they belong.

```
# Osativa
osativa_spairs <- split_pairs_by_peak(
    kslist$Osativa[, c("dup1", "dup2", "ks", "type")], osativa_peaks
)

# Gmax
gmax_spairs <- split_pairs_by_peak(
    kslist$Gmax[kslist$Gmax$ks <= 2, c("dup1", "dup2", "ks", "type")],
    gmax_peaks
)

# Slycopersicum
slycopersicum_spairs <- split_pairs_by_peak(
    kslist$Slycopersicum[, c("dup1", "dup2", "ks", "type")],
    slycopersicum_peaks
)

# Zmays
zmays_spairs <- split_pairs_by_peak(
    kslist$Zmays[, c("dup1", "dup2", "ks", "type")], zmays_peaks
)

# Vvinifera - done manually, only 1 peak
vvinifera_spairs <- list(
```

```
        pairs = kslist$Vvinifera[, c("dup1", "dup2", "ks", "type")] %>%
            filter(ks >= vvinifera_peaks$mean - 2 * vvinifera_peaks$sd &
                       ks <= vvinifera_peaks$mean + 2 * vvinifera_peaks$sd) %>%
            mutate(peak = 1),
        plot = plot_ks_peaks(
            kslist$Vvinifera$ks[kslist$Vvinifera$ks <= 3], vvinifera_peaks
        ) +
            geom_vline(xintercept = vvinifera_peaks$mean - 2 * vvinifera_peaks$sd,
                       color = "brown2", linetype = "dashed") +
            geom_vline(xintercept = vvinifera_peaks$mean + 2 * vvinifera_peaks$sd,
                       color = "brown2", linetype = "dashed") +
            ggtitle("Ks distribution for V. vinifera paranome")

    )

    # Ptrichocarpa
    ptrichocarpa_spairs <- split_pairs_by_peak(
        kslist$Ptrichocarpa[, c("dup1", "dup2", "ks", "type")], ptrichocarpa_peaks
    )

    # Athaliana
    athaliana_spairs <- split_pairs_by_peak(
        kslist$Athaliana[, c("dup1", "dup2", "ks", "type")], athaliana_peaks
    )

    # Combine all plots into one
    ks_plots_with_boundaries1 <- patchwork::wrap_plots(
        gmax_spairs$plot + ggtitle("G. max"),
        athaliana_spairs$plot + ggtitle("A. thaliana"),
        ptrichocarpa_spairs$plot + ggtitle("P. trichocarpa"),
        slycopersicum_spairs$plot + ggtitle("S. lycopersicum"),
        nrow = 1
    )
    ks_plots_with_boundaries2 <- patchwork::wrap_plots(
        vvinifera_spairs$plot + ggtitle("V. vinifera"),
        osativa_spairs$plot + ggtitle("O. sativa"),
        zmays_spairs$plot + ggtitle("Z. mays"),
        nrow = 1
    )

    ks_plots_with_boundaries <- ggpubr::ggarrange(
        ks_plots_with_boundaries1,
        ks_plots_with_boundaries2,
        nrow = 2
    )
    ks_plots_with_boundaries <- ggpubr::annotate_figure(
        ks_plots_with_boundaries,
        top = ggpubr::text_grob(
            "Ks distribution for species' paranomes with peak boundaries",
            size = 16
        )
```

```
    )
    ks_plots_with_boundaries
```




```
    # Save plots
    ggsave(ks_plots_with_boundaries, file = here("products", "plots",
        "ks_plots_with_boundaries.png"), dpi = 300, width = 16, height = 8)

    # Save objects
    duplicate_pairs_with_boundaries <- list(Gmax = gmax_spairs$pairs,
        Athaliana = athaliana_spairs$pairs, Ptrichocarpa = ptrichocarpa_spairs$pairs,
        Slycopersicum = slycopersicum_spairs$pairs, Vvinifera = vvinifera_spairs$pairs,
        Osativa = osativa_spairs$pairs, Zmays = zmays_spairs$pairs)

    save(duplicate_pairs_with_boundaries, file = here("products",
        "result_files", "duplicate_pairs_with_boundaries.rda"), compress = "xz")
```

Finally, let's get duplicated genes from duplicate pairs. That is, we will get a list of unique genes assigned to each mode of duplication.

```
    # Classify duplicated genes
    load(here("products", "result_files", "duplicate_pairs_with_boundaries.rda"))

    duplicated_genes <- lapply(duplicate_pairs_with_boundaries, function(x) {

        # Create a list for each peak
        pairs_list <- x
        pairs_list$ks <- NULL
        names(pairs_list) <- c("dup1", "dup2", "type", "peak")
        pairs_list <- split(pairs_list, pairs_list$peak)

        # Classify duplicated genes for each peak
        dup_genes <- Reduce(rbind, lapply(pairs_list, function(y) {
            peak_id <- unique(y$peak)
            dups <- y
```

```
            dups$peak <- NULL
            dups <- list(dups)
            class_dups <- classify_genes(dups)[[1]]
            class_dups$peak <- peak_id
            return(class_dups)
        }))

        ref <- c("WGD", "SSD")
        dups <- dup_genes[order(match(dup_genes$type, ref)), ]
        dups <- dup_genes[!duplicated(dup_genes$gene), ]
        return(dups)
    })

str(duplicated_genes)
```

```
# Save object
save(duplicated_genes, file = here("data", "duplicated_genes.rda"),
    compress = "xz")
```

5 Session info

This document was created under the following conditions:

```
sessioninfo::session_info()
## - Session info ---------------------------------------------------------------
##  setting  value
##  version  R version 4.2.1 (2022-06-23)
##  os       Ubuntu 20.04.4 LTS
##  system   x86_64, linux-gnu
##  ui       X11
##  language (EN)
##  collate  en_US.UTF-8
##  ctype    en_US.UTF-8
##  tz       Europe/Brussels
##  date     2022-08-09
##  pandoc   2.18 @ /usr/lib/rstudio/bin/quarto/bin/tools/ (via rmarkdown)
##
## - Packages -------------------------------------------------------------------
##  package          * version  date (UTC) lib source
##  abind              1.4-5    2016-07-21 [1] CRAN (R 4.2.0)
##  assertthat         0.2.1    2019-03-21 [1] CRAN (R 4.2.0)
##  backports          1.4.1    2021-12-13 [1] CRAN (R 4.2.0)
##  BiocGenerics       0.42.0   2022-04-26 [1] Bioconductor
##  BiocManager        1.30.18  2022-05-18 [1] CRAN (R 4.2.0)
##  BiocStyle        * 2.25.0   2022-06-15 [1] Github (Bioconductor/BiocStyle@7150c28)
##  Biostrings         2.64.0   2022-04-26 [1] Bioconductor
##  bitops             1.0-7    2021-04-24 [1] CRAN (R 4.2.0)
##  bookdown           0.27     2022-06-14 [1] CRAN (R 4.2.0)
##  broom              0.8.0    2022-04-13 [1] CRAN (R 4.2.0)
```

```
##  car               3.1-0    2022-06-15 [1] CRAN (R 4.2.0)
##  carData           3.0-5    2022-01-06 [1] CRAN (R 4.2.0)
##  cellranger        1.1.0    2016-07-27 [1] CRAN (R 4.2.0)
##  cli               3.3.0    2022-04-25 [1] CRAN (R 4.2.0)
##  cluster           2.1.3    2022-03-28 [1] CRAN (R 4.2.0)
##  coda              0.19-4   2020-09-30 [1] CRAN (R 4.2.0)
##  colorspace        2.0-3    2022-02-21 [1] CRAN (R 4.2.0)
##  cowplot           1.1.1    2020-12-30 [1] CRAN (R 4.2.0)
##  crayon            1.5.1    2022-03-26 [1] CRAN (R 4.2.0)
##  DBI               1.1.3    2022-06-18 [1] CRAN (R 4.2.0)
##  dbplyr            2.2.1    2022-06-27 [1] CRAN (R 4.2.1)
##  digest            0.6.29   2021-12-01 [1] CRAN (R 4.2.0)
##  doubletrouble   * 0.99.0   2022-07-06 [1] Bioconductor
##  dplyr           * 1.0.9    2022-04-28 [1] CRAN (R 4.2.0)
##  ellipsis          0.3.2    2021-04-29 [1] CRAN (R 4.2.0)
##  evaluate          0.15     2022-02-18 [1] CRAN (R 4.2.0)
##  fansi             1.0.3    2022-03-24 [1] CRAN (R 4.2.0)
##  farver            2.1.0    2021-02-28 [1] CRAN (R 4.2.0)
##  fastmap           1.1.0    2021-01-25 [1] CRAN (R 4.2.0)
##  forcats         * 0.5.1    2021-01-27 [1] CRAN (R 4.2.0)
##  formatR           1.12     2022-03-31 [1] CRAN (R 4.2.0)
##  fs                1.5.2    2021-12-08 [1] CRAN (R 4.2.0)
##  generics          0.1.2    2022-01-31 [1] CRAN (R 4.2.0)
##  GenomeInfoDb      1.32.2   2022-05-15 [1] Bioconductor
##  GenomeInfoDbData  1.2.8    2022-05-06 [1] Bioconductor
##  GenomicRanges     1.48.0   2022-04-26 [1] Bioconductor
##  ggnetwork         0.5.10   2021-07-06 [1] CRAN (R 4.2.0)
##  ggplot2         * 3.3.6    2022-05-03 [1] CRAN (R 4.2.0)
##  ggpubr            0.4.0    2020-06-27 [1] CRAN (R 4.2.0)
##  ggsignif          0.6.3    2021-09-09 [1] CRAN (R 4.2.0)
##  glue              1.6.2    2022-02-24 [1] CRAN (R 4.2.0)
##  gridExtra         2.3      2017-09-09 [1] CRAN (R 4.2.0)
##  gtable            0.3.0    2019-03-25 [1] CRAN (R 4.2.0)
##  haven             2.5.0    2022-04-15 [1] CRAN (R 4.2.0)
##  here            * 1.0.1    2020-12-13 [1] CRAN (R 4.2.0)
##  hms               1.1.1    2021-09-26 [1] CRAN (R 4.2.0)
##  htmltools         0.5.2    2021-08-25 [1] CRAN (R 4.2.0)
##  htmlwidgets       1.5.4    2021-09-08 [1] CRAN (R 4.2.0)
##  httr              1.4.3    2022-05-04 [1] CRAN (R 4.2.0)
##  igraph            1.3.2    2022-06-13 [1] CRAN (R 4.2.0)
##  intergraph        2.0-2    2016-12-05 [1] CRAN (R 4.2.0)
##  IRanges           2.30.0   2022-04-26 [1] Bioconductor
##  jsonlite          1.8.0    2022-02-22 [1] CRAN (R 4.2.0)
##  kernlab           0.9-31   2022-06-09 [1] CRAN (R 4.2.0)
##  knitr             1.39     2022-04-26 [1] CRAN (R 4.2.0)
##  labeling          0.4.2    2020-10-20 [1] CRAN (R 4.2.0)
##  lattice           0.20-45  2021-09-22 [1] CRAN (R 4.2.0)
##  lifecycle         1.0.1    2021-09-24 [1] CRAN (R 4.2.0)
##  lubridate         1.8.0    2021-10-07 [1] CRAN (R 4.2.0)
##  magrittr          2.0.3    2022-03-30 [1] CRAN (R 4.2.0)
##  MASS              7.3-57   2022-04-22 [1] CRAN (R 4.2.0)
```

```
## Matrix            1.4-1    2022-03-23 [1] CRAN (R 4.2.0)
## mgcv              1.8-40   2022-03-29 [1] CRAN (R 4.2.0)
## mixtools          1.2.0    2020-02-07 [1] CRAN (R 4.2.0)
## modelr            0.1.8    2020-05-19 [1] CRAN (R 4.2.0)
## munsell           0.5.0    2018-06-12 [1] CRAN (R 4.2.0)
## network           1.17.2   2022-05-21 [1] CRAN (R 4.2.0)
## networkD3         0.4      2017-03-18 [1] CRAN (R 4.2.0)
## nlme              3.1-158  2022-06-15 [1] CRAN (R 4.2.0)
## patchwork         1.1.1    2020-12-17 [1] CRAN (R 4.2.0)
## permute           0.9-7    2022-01-27 [1] CRAN (R 4.2.0)
## pheatmap          1.0.12   2019-01-04 [1] CRAN (R 4.2.0)
## pillar            1.7.0    2022-02-01 [1] CRAN (R 4.2.0)
## pkgconfig         2.0.3    2019-09-22 [1] CRAN (R 4.2.0)
## purrr           * 0.3.4    2020-04-17 [1] CRAN (R 4.2.0)
## R6                2.5.1    2021-08-19 [1] CRAN (R 4.2.0)
## RColorBrewer      1.1-3    2022-04-03 [1] CRAN (R 4.2.0)
## Rcpp              1.0.8.3  2022-03-17 [1] CRAN (R 4.2.0)
## RCurl             1.98-1.7 2022-06-09 [1] CRAN (R 4.2.0)
## readr           * 2.1.2    2022-01-30 [1] CRAN (R 4.2.0)
## readxl            1.4.0    2022-03-28 [1] CRAN (R 4.2.0)
## reprex            2.0.1    2021-08-05 [1] CRAN (R 4.2.0)
## rlang             1.0.3    2022-06-27 [1] CRAN (R 4.2.1)
## rmarkdown         2.14     2022-04-25 [1] CRAN (R 4.2.0)
## rprojroot         2.0.3    2022-04-02 [1] CRAN (R 4.2.0)
## rstatix           0.7.0    2021-02-13 [1] CRAN (R 4.2.0)
## rstudioapi        0.13     2020-11-12 [1] CRAN (R 4.2.0)
## rvest             1.0.2    2021-10-16 [1] CRAN (R 4.2.0)
## S4Vectors         0.34.0   2022-04-26 [1] Bioconductor
## scales            1.2.0    2022-04-13 [1] CRAN (R 4.2.0)
## segmented         1.6-0    2022-05-31 [1] CRAN (R 4.2.0)
## sessioninfo       1.2.2    2021-12-06 [1] CRAN (R 4.2.0)
## statnet.common    4.6.0    2022-05-02 [1] CRAN (R 4.2.0)
## stringi           1.7.6    2021-11-29 [1] CRAN (R 4.2.0)
## stringr         * 1.4.0    2019-02-10 [1] CRAN (R 4.2.0)
## survival          3.3-1    2022-03-03 [1] CRAN (R 4.2.0)
## syntenet        * 0.99.1   2022-06-19 [1] Bioconductor
## tibble          * 3.1.7    2022-05-03 [1] CRAN (R 4.2.0)
## tidyr           * 1.2.0    2022-02-01 [1] CRAN (R 4.2.0)
## tidyselect        1.1.2    2022-02-21 [1] CRAN (R 4.2.0)
## tidyverse       * 1.3.1    2021-04-15 [1] CRAN (R 4.2.0)
## tzdb              0.3.0    2022-03-28 [1] CRAN (R 4.2.0)
## utf8              1.2.2    2021-07-24 [1] CRAN (R 4.2.0)
## vctrs             0.4.1    2022-04-13 [1] CRAN (R 4.2.0)
## vegan             2.6-2    2022-04-17 [1] CRAN (R 4.2.0)
## withr             2.5.0    2022-03-03 [1] CRAN (R 4.2.0)
## xfun              0.31     2022-05-10 [1] CRAN (R 4.2.0)
## xml2              1.3.3    2021-11-30 [1] CRAN (R 4.2.0)
## XVector           0.36.0   2022-04-26 [1] Bioconductor
## yaml              2.3.5    2022-02-21 [1] CRAN (R 4.2.0)
## zlibbioc          1.42.0   2022-04-26 [1] Bioconductor
##
```

```
## [1] /home/faalm/R/x86_64-pc-linux-gnu-library/4.2
## [2] /usr/local/lib/R/site-library
## [3] /usr/lib/R/site-library
## [4] /usr/lib/R/library
##
## ------------------------------------------------------------------------------
```