Coexpression network analysis of Platycaria transcriptomes

Fabricio Almeida-Silva

2022-07-08

Contents

1 Overview

Here, I will describe the code to:

- Infer gene coexpression networks from expression data
- Perform module-trait associations
- Perform module enrichment analyses
- Compare positively selected genes with network-based approaches

The species included in the expression data are:

- *Platycaria longipes*
- *Platycaria strobilacea*

```
library(here)
library(BioNERO)
library(SummarizedExperiment)
library(tidyverse)
set.seed(123)
```

2 Data preprocessing

Here, the expression data are stored in a `SummarizedExperiment` object. Let's load it and take a look at it.

```
# Load SummarizedExperiment object
load(here("se.rda"))

se

# Check sample metadata
colData(se)
```

For practical reasons, we will split expression data and sample metadata in different objects and add information on time points of the stress treatment. Here, we will infer a GCN for each species.

```
# Get expression data for each species
exp <- as.matrix(SummarizedExperiment::assay(se, "gene_TPM"))
exp_ps <- exp[, grep("PS", colnames(exp))]
exp_pl <- exp[, grep("PL", colnames(exp))]

# Get sample metadata
metadata <- as.data.frame(SummarizedExperiment::colData(se))[, c(1,2)]
names(metadata) <- c("Sample", "Tissue")
metadata <- metadata %>%
    mutate(
        Treatment = case_when(
            str_detect(Sample, "ONS") ~ "control",
            str_detect(Sample, "1d") ~ "1d",
            str_detect(Sample, "6h") ~ "6h",
            str_detect(Sample, "7d") ~ "7d",
            str_detect(Sample, "6h") ~ "6h",
            str_detect(Sample, "6h") ~ "6h",
            TRUE ~ Sample
        )
    ) %>%
    mutate(Tissue_Treatment = str_c(Tissue, Treatment, sep = "_"))

metadata_ps <- metadata[grep("PS", metadata$Sample), ]
metadata_pl <- metadata[grep("PL", metadata$Sample), ]
```

Now, let's preprocess the data with the function `exp_preprocess()`. Here, we will remove genes with median TPM <5. For every other processing step, we will use default parameters.

```
# Data preprocessing
exp_pl <- exp_preprocess(exp_pl, min_exp = 5, Zk_filtering = FALSE)
exp_ps <- exp_preprocess(exp_ps, min_exp = 5, Zk_filtering = FALSE)
```

2.1 Exploratory analysis

Now, let's do some exploratory analyses.

```
# Sample correlation heatmap
## Ps
plot_heatmap(
    exp_ps,
    col_metadata = metadata_ps[, c("Tissue", "Treatment")],
    type = "samplecor"
)

## Pl
plot_heatmap(
    exp_pl,
    col_metadata = metadata_pl[, c("Tissue", "Treatment")],
    type = "samplecor"
)

# Principal component analysis
## By tissue
### Ps
plot_PCA(
    exp_ps,
    metadata = metadata_ps[, "Tissue", drop = FALSE]
)

### Pl
plot_PCA(
    exp_pl,
    metadata = metadata_pl[, "Tissue", drop = FALSE]
)

## By treatment
```

```
### Ps
plot_PCA(
    exp_ps,
    metadata = metadata_ps[, "Treatment", drop = FALSE]
)

### Ps
plot_PCA(
    exp_pl,
    metadata = metadata_pl[, "Treatment", drop = FALSE]
)
```

3 Gene coexpression network inference

Before inferring the network, we need to find the best beta power based on the best SFT fit.

```
# Find beta power
## Ps GCN
sft_ps <- SFT_fit(exp_ps, net_type = "signed", cor_method = "pearson")
sft_ps$power # best: 17


## Pl GCN
sft_pl <- SFT_fit(exp_pl, net_type = "signed", cor_method = "pearson")
sft_pl$power # best: 19
```

Now, let's infer the GCNs.

```
# Ps
gcn_ps <- exp2gcn(exp_ps, net_type = "signed", SFTpower = 17,
                  cor_method = "pearson")
format(object.size(gcn_ps$adjacency_matrix), units = "Mb")
format(object.size(gcn_ps$moduleColors), units = "Mb")
format(object.size(gcn_ps$correlation_matrix), units = "Mb")
format(object.size(gcn_ps$dendro_plot_objects), units = "Mb")
format(object.size(gcn_ps), units = "Gb")

save(
    gcn_ps,
    file = here("gcn_ps.rda"),
    compress = "xz"
)


# Pl
gcn_pl <- exp2gcn(exp_pl, net_type = "signed", SFTpower = 19,
                  cor_method = "pearson")
save(
    gcn_pl,
    file = "gcn_pl.rda",
    compress = "xz"
)

genes_and_modules_pl <- gcn_pl$genes_and_modules
genes_and_modules_pl$Genes <- gsub("\\.[0-9]$", "", genes_and_modules_pl$Genes)
save(genes_and_modules_pl, file = "genes_and_modules_pl.rda", compress = "xz")


background_pl <- gsub("\\.[0-9]", "", rownames(exp_pl))
save(background_pl, file = "background_pl.rda", compress = "xz")
```

4 Coexpression network analysis

Now that we have the 2 networks, let's explore them.

4.1 Visual exploration of network inference

First, let's plot the dendrograms of module assignments for the 2 GCNs.

```
# Pl GCN
pdf(file = here("dendro_pl.pdf"), width = 10, height = 10)
plot_dendro_and_colors(gcn_pl)
dev.off()


# Ps GCN
pdf(file = here("dendro_ps.pdf"), width = 10, height = 10)
plot_dendro_and_colors(gcn_ps)
dev.off()
```

Now, let's plot eigengene networks.

```
# Pl
pdf(file = here("eigengene_network_pl.pdf"), width = 8, height = 8)
plot_eigengene_network(gcn_pl)
dev.off()

# Ps
pdf(file = here("eigengene_network_ps.pdf"), width = 8, height = 8)
plot_eigengene_network(gcn_ps)
dev.off()
```

Finally, let's plot the number of genes per module in each GCN.

```
# Pl
pl_ngenes <- plot_ngenes_per_module(gcn_pl)
ggsave(pl_ngenes, filename = here("ngenes_per_module_pl.pdf"),
       width = 12, height = 8)

# Ps
```

```
ps_ngenes <- plot_ngenes_per_module(gcn_ps)
ggsave(ps_ngenes, filename = here("ngenes_per_module_ps.pdf"),
       width = 14, height = 8)
```

4.2 Network comparison

First, let's find out which modules are preserved in both networks.

```
preservation_stats <- module_preservation(
    explist = list(pl = exp_pl, ps = exp_ps),
    ref_net = gcn_pl,
    test_net = gcn_ps
)

preservation_stats$
```

As we can see, 28 modules in the Pl GCN were preserved in the Ps network. They are: blue3, brown4, coral2, coral3, darkgreen, darkgrey, darkolivegreen, darkorange, darkseagreen4, darkslateblue, floralwhite, green, grey60, indianred3, lavenderblush2, lightblue4, lightyellow, maroon, mediumpurple4, mistyrose, navajowhite3, plum1, plum2, salmon, salmon1, salmon4, slateblue1, yellow3.

Now, let's see what these 28 modules represent in terms of percentage.

```
# Count number of modules for Pl GCN (with -1 to remove grey module)
nmodules_pl <- length(unique(gcn_pl$genes_and_modules$Modules)) - 1
28 / nmodules_pl
```

The Pl network has 37 modules, of which 28 (75.68%) were preserved in Ps. Let's save the information on preserved and non-preserved modules.

```
preserved_modules_pl <- c(
    "blue3", "brown4", "coral2", "coral3", "darkgreen", "darkgrey",
    "darkolivegreen", "darkorange", "darkseagreen4", "darkslateblue",
    "floralwhite", "green", "grey60", "indianred3", "lavenderblush2",
    "lightblue4", "lightyellow", "maroon", "mediumpurple4",
    "mistyrose", "navajowhite3", "plum1", "plum2", "salmon", "salmon1",
    "salmon4", "slateblue1", "yellow3"
)

non_preserved_modules_pl <- unique(gcn_pl$genes_and_modules$Modules)
non_preserved_modules_pl <- non_preserved_modules_pl[!non_preserved_modules_pl
                                                     %in% preserved_modules_pl]
non_preserved_modules_pl <- non_preserved_modules_pl[!non_preserved_modules_pl
                                                     %in% "grey"]
```

4.3 In which module are the positively selected genes?

Here, we will look for modules where positively selected genes (in Pl) are in the Pl network.

```r
# Vector of IDs for positively selected genes in Pl
positive_selection <- c(
    "PstrChr01G000611", "PstrChr01G000895", "PstrChr04G000593",
    "PstrChr04G001560", "PstrChr05G000719", "PstrChr01G000755",
    "PstrChr14G001283"
)

gcn_pl$genes_and_modules$Genes <- gsub("\\.[0-9]$", "",
                                       gcn_pl$genes_and_modules$Genes)

mod_pos <- gcn_pl$genes_and_modules[gcn_pl$genes_and_modules$Genes %in%
                                        positive_selection, ]
mod_pos
```

Genes	Modules
PstrChr01G000611	mediumpurple4
PstrChr01G000755	darkgreen
PstrChr01G000895	darkolivegreen
PstrChr05G000719	indianred3

4.4 Module-trait associations

Here, we will perform module-trait associations for preserved modules in the Ps and Pl GCNs.

```r
# Perform module-trait associations
## Pl
pdf(file = here("Pl_module_trait_cor_tissue-treatment.pdf"),
    width = 8, height = 8)
MEtrait_pl1 <- module_trait_cor(
    exp = exp_pl,
    metadata = metadata_pl[, "Tissue_Treatment", drop = FALSE],
    MEs = gcn_pl$MEs,
    cor_method = "pearson"
)
dev.off()

pdf(file = here("Pl_module_trait_cor_tissue.pdf"),
    width = 5, height = 8)
MEtrait_pl2 <- module_trait_cor(
    exp = exp_pl,
    metadata = metadata_pl[, "Tissue", drop = FALSE],
    MEs = gcn_pl$MEs,
    cor_method = "pearson"
)
dev.off()

pdf(file = here("Pl_module_trait_cor_treatment.pdf"),
    width = 6, height = 8)
MEtrait_pl3 <- module_trait_cor(
    exp = exp_pl,
    metadata = metadata_pl[, "Treatment", drop = FALSE],
    MEs = gcn_pl$MEs,
    cor_method = "pearson"
)
dev.off()
```

4.5 Enrichment analysis

Now, let's perform a module enrichment analysis for the Pl GCN. First, let's get the annotation data.

```
# GO
annot_go <- readr::read_tsv(
    here("annotation", "Pstr.GO.2Cols"), col_names = c("Gene", "GO"),
    show_col_types = FALSE
) %>%
    separate_rows(GO, sep = ",") %>%
    mutate(GO = stringr::str_replace_all(
        GO,
        c(
            "Molecular Function" = "MF",
            "Biological Process" = "BP",
            "Cellular Component" = "CC"
        )
    )) %>%
    mutate(Gene = str_replace_all(Gene, "\\.[0-9]$", "")) %>%
    as.data.frame()

# InterPro
annot_interpro <- readr::read_tsv(
    here("annotation", "Pstr.IPRSCAN.3Cols"), col_names = c("Gene", "Interpro"),
    show_col_types = FALSE
) %>%
    select(Gene, Interpro) %>%
    mutate(Gene = str_replace_all(Gene, "\\.[0-9]$", "")) %>%
    as.data.frame()

# KEGG
annot_kegg <- readr::read_tsv(
    here("annotation", "Pstr.KEGG.2Cols"), col_names = c("Gene", "KEGG"),
    show_col_types = FALSE, skip = 1
) %>%
    separate_rows(KEGG, sep = ";") %>%
    mutate(KEGG = str_squish(KEGG)) %>%
    mutate(Gene = str_replace_all(Gene, "\\.[0-9]$", "")) %>%
    as.data.frame()

# PFAM
annot_pfam <- readr::read_tsv(
```

```r
    here("annotation", "Pstr.pfam.2Cols"), col_names = c("Gene", "PFAM"),
    show_col_types = FALSE, skip = 1
) %>%
    separate_rows(PFAM, sep = "/") %>%
    mutate(PFAM = str_squish(PFAM)) %>%
    mutate(Gene = str_replace_all(Gene, "\\.[0-9]$", "")) %>%
    as.data.frame()

annotation <- list(
  GO = annot_go,
  InterPro = annot_interpro,
  KEGG = annot_kegg,
  PFAM = annot_pfam
)

save(annotation, file = "annotation.rda", compress = "xz")
```

Now, we can perform the enrichment analyses.

```r
# Perform module enrichment analyses
load("background_pl.rda")
bg <- background_pl

# Start SEA
enrichment_go <- module_enrichment(gcn_pl, bg, annot_go, column = "GO")
enrichment_interpro <- module_enrichment(gcn_pl, bg, annotation$InterPro, column = "Interpro")
enrichment_kegg <- module_enrichment(gcn_pl, bg, annotation$KEGG, column = "KEGG")
enrichment_pfam <- module_enrichment(gcn_pl, bg, annotation$PFAM, column = "PFAM")

# Save enrichment results
enrichment_results <- list(
  GO = enrichment_go,
  InterPro = enrichment_interpro,
  KEGG = enrichment_kegg,
  PFAM = enrichment_pfam
)

lapply(seq_along(enrichment_results), function(x) {
  n <- names(enrichment_results)[x]
  readr::write_tsv(enrichment_results[[x]], file = paste0(n, "enrichment.tsv"))
})

save(enrichment_results, file = "enrichment_results.rda", compress = "xz")
```

Now, let's add information on modules that were preserved in both sets and modules that were not.

```r
load(here("enrichment_results.rda"))
load(here("genes_and_modules_pl.rda"))

all_modules <- unique(genes_and_modules_pl$Modules)
```

```
# Preserved modules
preserved_modules <- data.frame(
  Module = c(
    "blue3", "brown4", "coral2", "coral3", "darkgreen", "darkgrey",
    "darkolivegreen", "darkorange", "darkseagreen4", "darkslateblue",
    "floralwhite", "green", "grey60", "indianred3", "lavenderblush2",
    "lightblue4", "lightyellow", "maroon", "mediumpurple4",
    "mistyrose", "navajowhite3", "plum1", "plum2", "salmon", "salmon1",
    "salmon4", "slateblue1", "yellow3"
  ),
  Preserved = TRUE
)

# Non-preserved modules
nonpreserved_modules <- data.frame(
  Module = all_modules[!all_modules %in% preserved_modules$Module],
  Preserved = FALSE
)

# Data frame of preservation info
preservation_status <- rbind(preserved_modules, nonpreserved_modules)
readr::write_tsv(
  preservation_status,
  file = here("tables", "module_preservation_status.tsv")
)

# Add preservation info to enrichment results
final_enrichment_results <- Reduce(rbind, enrichment_results) %>%
  inner_join(., preservation_status)

readr::write_tsv(
  final_enrichment_results,
  file = here("tables", "enrichment_results_with_module_preservation.tsv")
)
```

5 Network-based comparison of positively selected genes

Here, we will calculate pairwise Pearson correlation coefficients between genes using both expression data sets (Pl and Ps). Then, we will compare positively selected genes in terms of:

- degree
- percentage of shared neighbors
- function of shared neighbors

To start, let's preprocess both sets prior to calculating PCC.

```r
# Load expression data
load(here("se.rda"))

# Create vector of positively selected genes in Pl
pos_sel <- c(
    "PstrChr01G000611", "PstrChr01G000895", "PstrChr04G000593",
    "PstrChr04G001560", "PstrChr05G000719", "PstrChr01G000755",
    "PstrChr14G001283"
)

# Remove genes with median <=1, correct for false-positives, and
# apply quantile-normalization
exp <- as.matrix(SummarizedExperiment::assay(se, "gene_TPM"))

## Pl
exp_pl <- exp[, grep("PL", colnames(exp))]
exp_pl <- exp_preprocess(exp_pl, min_exp = 1, Zk_filtering = FALSE)
rownames(exp_pl) <- gsub("\\.[0-9]$", "", rownames(exp_pl))

## Ps
exp_ps <- exp[, grep("PS", colnames(exp))]
exp_ps <- exp_preprocess(exp_ps, min_exp = 1, Zk_filtering = FALSE)
rownames(exp_ps) <- gsub("\\.[0-9]$", "", rownames(exp_ps))

# Check if all positively selected genes are included in both sets
pos_sel %in% rownames(exp_pl)
pos_sel %in% rownames(exp_ps)
```

The gene PstrChr14G001283 is not present in any of the sets, which means its median expression is lower than 1. As this is too low, we will ignore this gene.

```r
# Calculate pairwise gene-gene Pearson correlation coefficients
pl_cor <- WGCNA::cor(t(exp_pl))
ps_cor <- WGCNA::cor(t(exp_ps))
```

Now, let's calculate the degree of the positively selected genes in each network.

```r
library(tidyverse)

# Calculate degree
degree_pl <- data.frame(
  Gene = rownames(pl_cor),
  Degree = matrixStats::rowSums2(pl_cor)
)

degree_ps <- data.frame(
  Gene = rownames(ps_cor),
  Degree = matrixStats::rowSums2(ps_cor)
)

# Find hubs
```

```r
hubs_pl <- degree_pl %>%
  arrange(-Degree) %>%
  slice_head(n = nrow(degree_pl) * 0.1)

hubs_ps <- degree_ps %>%
  arrange(-Degree) %>%
  slice_head(n = nrow(degree_ps) * 0.1)


# Are positively selected genes hubs?
pos_sel %in% hubs_pl$Gene
pos_sel %in% hubs_ps$Gene
```

The gene PstrChr01G000755 is a hub in the Ps network. Let's compare the degree of all positively selected genes.

```r
degree_comparison <- degree_pl %>%
  filter(Gene %in% pos_sel) %>%
  dplyr::rename(Degree_Pl = Degree) %>%
  inner_join(., degree_ps) %>%
  rename(Degree_Ps = Degree)

head(degree_comparison)
```

Now, let's perform a differential coexpression analysis with the R package `diffcoexp` to check if the positively selected genes are differentially coexpressed.

```r
library(diffcoexp)
allowWGCNAThreads()

# Keep only genes that appear in both expression sets
genes_pl <- rownames(exp_pl)
gene_intersect <- genes_pl[genes_pl %in% rownames(exp_ps)]

exp_pl_int <- exp_pl[gene_intersect, ]
exp_ps_int <- exp_ps[gene_intersect, ]

dim(exp_pl_int)
dim(exp_ps_int)

# Perform differential coexpression analysis
diffcoexp_res <- diffcoexp(
  exprs.1 = exp_pl_int,
  exprs.2 = exp_ps_int,
  r.method = "pearson"
)

dcg <- diffcoexp_res$DCGs
save(dcg,
     file = here("result_files", "dcg.rda"),
     compress = "xz")
```

The genes PstrChr01G000611, PstrChr04G001560, PstrChr05G000719, and PstrChr01G000755 were differentially coexpressed. Now, let's see a network plot of each gene and its coexpression partners in each network.

```r
# Get edge list for positively selected genes
## Removing gene that is absent in both sets
pos_sel <- c(
    "PstrChr01G000611", "PstrChr01G000895", "PstrChr04G000593",
    "PstrChr04G001560", "PstrChr05G000719", "PstrChr01G000755"
)
r <- 0.6

# Pl
cormat_pl <- pl_cor[pos_sel, ]
cormat_pl[cormat_pl < r] <- NA
fnet_pl <- list(
  params = list(net_type = "signed", power = 17),
  correlation_matrix = cormat_pl
)

# Ps
cormat_ps <- ps_cor[pos_sel, ]
cormat_ps[cormat_ps < r] <- NA
fnet_ps <- list(
  params = list(net_type = "signed", power = 19),
  correlation_matrix = cormat_ps
)

edges_pos <- lapply(pos_sel, function(x) {

  # Get neighbors
  neighbors_pl <- get_neighbors(x, fnet_pl, cor_threshold = r)[[1]]
  neighbors_ps <- get_neighbors(x, fnet_ps, cor_threshold = r)[[1]]

  # Get intersection information
  both <- intersect(neighbors_pl, neighbors_ps)
  ps_only <- neighbors_ps[!neighbors_ps %in% both]
  pl_only <- neighbors_pl[!neighbors_pl %in% both]

  color_by <- data.frame(
    Gene = union(neighbors_ps, neighbors_pl)
  ) %>%
    mutate(Set = case_when(
      Gene %in% both ~ "Both",
      Gene %in% ps_only ~ "Ps",
      Gene %in% pl_only ~ "Pl"
    ))
  color_by <- rbind(
    color_by,
    data.frame(Gene = x, Set = "Selected")
  )
```

```
    # Create edgelist
    edges <- data.frame(
      node1 = x,
      node2 = color_by$Gene
    )

    result <- list(edges = edges, color_by = color_by)
    return(result)
  })
  names(edges_pos) <- pos_sel

  # Plot networks
  library(ggnetwork)
  library(ggpubr)
  plots <- lapply(edges_pos, function(x) {

    selected <- x$color_by$Gene[x$color_by$Set == "Selected"]
    x$color_by$Selected <- ifelse(x$color_by$Gene == selected, TRUE, FALSE)
    graph <- igraph::graph_from_data_frame(
      d = x$edges, vertices = x$color_by, directed = FALSE
    )
    n <- ggnetwork(graph, arrow.gap = 0)
    p <- ggplot(n, aes_(x = ~x, y = ~y, xend = ~xend, yend = ~yend)) +
      geom_edges(color = "grey75", alpha = 0.5, show.legend = FALSE) +
      geom_nodes(aes_(color = ~Set), show.legend = TRUE) +
      geom_nodelabel_repel(
        aes_(label = ~name), color = "azure4",
        box.padding = ggnetwork::unit(1, "lines"),
        data = function(x) {
          x[x$Selected, ]
        }, show.legend = FALSE
      ) +
      scale_color_manual(values = c(
        "Both" = "#374E55FF", "Pl" = "#DF8F44FF",
        "Ps" = "#00A1D5FF", "Selected" = "#B24745FF"
      )) +
      ggnetwork::theme_blank()
    p
    return(p)
  })

  network_plots <- ggpubr::ggarrange(
    plots[[1]], plots[[2]], plots[[3]],
    plots[[4]], plots[[5]], plots[[6]],
    ncol = 3, nrow = 2, common.legend = TRUE,
    labels = "AUTO"
  )

  network_plots_final <- ggpubr::annotate_figure(
    network_plots,
    top = text_grob("Positively selected genes and their coexpression partners")
```

```
)

ggsave(
  network_plots_final,
  file = here("plots", "networks_positively_selected_genes.png"),
  dpi = 300, width = 12, height = 7
)

ggsave(
  network_plots_final,
  file = here("plots", "networks_positively_selected_genes.pdf"),
  width = 12, height = 7
)
```

Session information

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     2022-07-08
##  pandoc   2.17.1.1 @ /usr/lib/rstudio/bin/quarto/bin/ (via rmarkdown)
##
## - Packages -------------------------------------------------------------------
##  package     * version date (UTC) lib source
##  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)
##  bookdown      0.27    2022-06-14 [1] CRAN (R 4.2.0)
##  cli           3.3.0   2022-04-25 [1] CRAN (R 4.2.0)
##  digest        0.6.29  2021-12-01 [1] CRAN (R 4.2.0)
##  evaluate      0.15    2022-02-18 [1] CRAN (R 4.2.0)
##  fastmap       1.1.0   2021-01-25 [1] CRAN (R 4.2.0)
##  htmltools     0.5.2   2021-08-25 [1] CRAN (R 4.2.0)
##  knitr         1.39    2022-04-26 [1] CRAN (R 4.2.0)
##  magrittr      2.0.3   2022-03-30 [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)
##  rstudioapi    0.13    2020-11-12 [1] CRAN (R 4.2.0)
##  sessioninfo   1.2.2   2021-12-06 [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)
##  xfun          0.31    2022-05-10 [1] CRAN (R 4.2.0)
##  yaml          2.3.5   2022-02-21 [1] CRAN (R 4.2.0)
##
##  [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
##
## ------------------------------------------------------------------------------
```