PPI network reconstruction and analysis

Fabricio Almeida-Silva

2023-05-02

```
library(here)
library(tidyverse)
library(igraph)
library(BioNERO)
library(ggpubr)
library(ggstatsplot)
library(patchwork)

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

Overview

Here, we will describe the code to:

1. Obtain PPI networks for each species
2. Filter PPI networks
3. Explore network properties of PPI networks, such as degree, scale-free topology fit, and number interacting paralogs over time.

Obtaining PPI networks

Here, we will get PPI networks from STRING db (physical links only) and filter it to only contain edges with confidence level > 0.5 (500). STRING protein IDs will be converted to Ensembl gene IDs [1].

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

source(here("code", "utils.R"))

# Species and their NCBI IDs
id_map <- c(
    Gmax = 3847,
    Ptrichocarpa = 3694,
    Zmays = 4577,
    Osativa = 4530,
    Athaliana = 3702,
    Vvinifera = 29760,
```

[1] For *Vitis vinifera*, the mapping between STRING IDs and Ensembl gene IDs was obtained from BioMart on the Ensembl Plants release 53 web page and saved to *result_files/vvinifera_mapping_string_ensembl.tsv*. To obtain it, go to https://plants.ensembl.org/index.html, and click on `BioMart > Ensembl Plants Genes 54 > Vitis vinifera > Attributes > External References`, then tick `STRING ID`, click on `Results`, then `Go`.

```
    Slycopersicum = 4081
)

# Define function to create PPI network from STRINGdb for a species
stringdb2ppi <- function(species_id, threshold = 500,
                                id_map, duplicate_pairs, annotation,
                                duplicates_only = TRUE) {

    # Get PPI and alias URLs based on species ID
    url_ppi <- paste0(
        "https://stringdb-static.org/download/protein.physical.links.v11.5/",
        species_id, ".protein.physical.links.v11.5.txt.gz"
    )

    url_alias <- paste0(
        "https://stringdb-static.org/download/protein.aliases.v11.5/",
        species_id, ".protein.aliases.v11.5.txt.gz"
    )

    # Read PPI network and alias, and filter PPI based on threshold
    alias <- vroom::vroom(url_alias, delim = "\t", show_col_types = FALSE)
    ppi <- vroom::vroom(url_ppi, delim = " ", show_col_types = FALSE) %>%
        dplyr::filter(combined_score >= threshold)

    # Convert STRING IDs to Ensembl IDs
    species <- names(id_map)[id_map == species_id]
    fppi <- stringdb2ensembl(ppi, alias, species)
    names(fppi)[1:2] <- c("dup1", "dup2")

    # Keep only edges containing duplicated gene pairs
    pairs_ppi <- fppi
    if(duplicates_only) {
        pairs <- duplicate_pairs[[species]]
        pairs$pvalue <- NULL
        pairs_ppi <- merge(pairs, fppi, by = c("dup1", "dup2")) # 1 and 2
    }

    # Remove genes that are not present in annotation
    gene_list <- unique(annotation[[species]]$gene_id)
    pairs_ppi <- pairs_ppi[pairs_ppi$dup1 %in% gene_list, ]
    pairs_ppi <- pairs_ppi[pairs_ppi$dup2 %in% gene_list, ]

    message("Number of edges for ", species, ": ", nrow(pairs_ppi))
    return(pairs_ppi)
}

# Get PPI network, paralogous gene pairs only
ppi <- lapply(id_map, function(x) {
    paralogs_ppi <- stringdb2ppi(
        x,
        id_map = id_map,
        duplicate_pairs = duplicate_pairs,
        annotation = annotation
```

```
    )
    return(paralogs_ppi)
})

# Get PPI network, including non paralogs
ppi_full <- lapply(id_map, function(x) {
    paralogs_ppi <- stringdb2ppi(
        x,
        id_map = id_map,
        duplicate_pairs = duplicate_pairs,
        annotation = annotation,
        duplicates_only = FALSE
    )
    return(paralogs_ppi)
})


# Save PPI networks
save(
    ppi,
    file = here("products", "result_files", "ppi.rda"),
    compress = "xz"
)

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

Topological analysis of PPI networks

Here, we will explore the topology of the PPI networks to:

- check if PPI networks are scale-free
- compare the degree distribution of WGD- and SSD-derived gene pairs

First, let's count the number of interacting paralogs of each duplication mode for each species.

```
# Load PPI network
load(here("products", "result_files", "ppi.rda"))

# Count
count_edges <- Reduce(rbind, lapply(seq_along(ppi), function(x) {
    species <- names(ppi)[x]
    net <- ppi[[x]]
    wgd_count <- nrow(net[net$type == "WGD", ])
    ssd_count <- nrow(net[net$type == "SSD", ])

    # Count frequency of edges between WGD- and SSD-derived pairs
    count <- data.frame(
        Species = species,
        WGD = wgd_count,
        SSD = ssd_count
```

```
    )
    count$Total <- count$WGD + count$SSD

    # Check if network is scale-free
    is_scale_free <- BioNERO::check_SFT(net[, c(1,2)])
    sft <- FALSE
    if(is_scale_free$KS.p >= 0.05) { sft <- TRUE }

    count$SFT <- sft

    return(count)
}))
count_edges
```

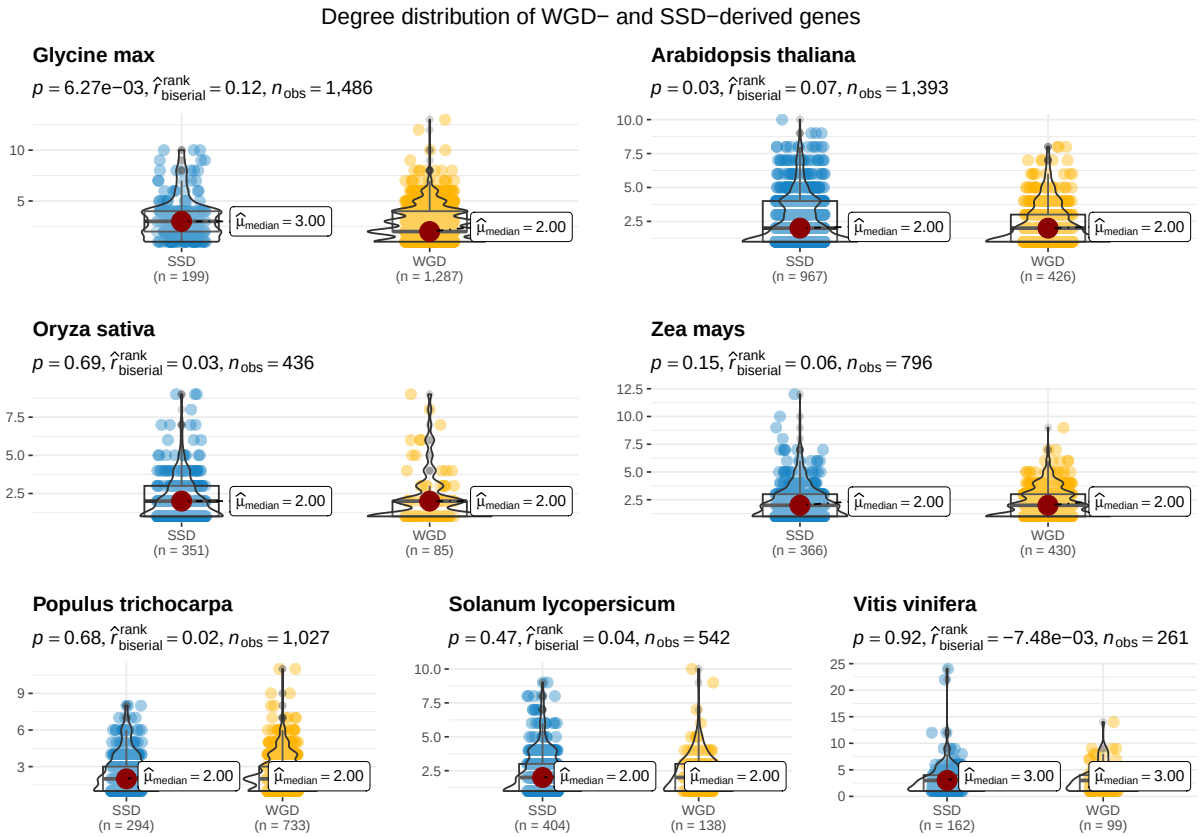
As we can see, all networks are scale-free.

```
# Save table
readr::write_tsv(
    count_edges,
    file = here("products", "tables", "ppi_network_summary_stats.tsv")
)
```

Now, let's compare if SSD- and WGD-derived pairs differ in terms of degree.

```
load(here("data", "duplicated_genes.rda"))
load(here("products", "result_files", "ppi.rda"))

# Create data frame of degree distribution for each species by duplication mode
degree_distros <- Reduce(rbind, lapply(seq_along(ppi), function(x) {
    species <- names(ppi)[x]
    net <- ppi[[x]]
    dups <- duplicated_genes[[species]]

    # Get degree distribution as a data frame
    degree <- graph_from_data_frame(net[, 1:2]) %>% degree()
    degree_df <- data.frame(
        gene = names(degree),
        degree = degree
    ) %>% dplyr::inner_join(., dups, by = "gene") %>%
        dplyr::select(gene, degree, type) %>%
        dplyr::mutate(species = species)

    return(degree_df)
}))

# Visualize degree distributions as a violin plot
plot_violin <- function(data) {
    p <- ggbetweenstats(
        data = data, x = type, y = degree,
        type = "nonparametric", pairwise.display = "all",
        p.adjust.method = "BH"
    ) +
        ggplot2::scale_color_manual(values = dup_palette) +
        labs(x = "", y = "") +
        theme(plot.subtitle = element_text(size = 8.5))
```

```
    subtitle <- extract_subtitle(p)
    subtitle[c(2, 5, 6)] <- NULL
    p <- p + labs(subtitle = subtitle)

    return(p)
}
sdegree_distros <- split(degree_distros, degree_distros$species)

plot_degree_gma <- plot_violin(sdegree_distros$Gmax) +
    labs(title = "Glycine max") +
    theme(plot.subtitle = element_text(size = 11))

plot_degree_ath <- plot_violin(sdegree_distros$Athaliana) +
    labs(title = "Arabidopsis thaliana") +
    theme(plot.subtitle = element_text(size = 11))

plot_degree_ptr <- plot_violin(sdegree_distros$Ptrichocarpa) +
    labs(title = "Populus trichocarpa") +
    theme(plot.subtitle = element_text(size = 11))

plot_degree_sly <- plot_violin(sdegree_distros$Slycopersicum) +
    labs(title = "Solanum lycopersicum") +
    theme(plot.subtitle = element_text(size = 11))

plot_degree_vvi <- plot_violin(sdegree_distros$Vvinifera) +
    labs(title = "Vitis vinifera") +
    theme(plot.subtitle = element_text(size = 11))

plot_degree_osa <- plot_violin(sdegree_distros$Osativa) +
    labs(title = "Oryza sativa") +
    theme(plot.subtitle = element_text(size = 11))

plot_degree_zma <- plot_violin(sdegree_distros$Zmays) +
    labs(title = "Zea mays") +
    theme(plot.subtitle = element_text(size = 11))

# Combine plots
p_deg_final <- wrap_plots(
    wrap_plots(plot_degree_gma, plot_degree_ath, nrow = 1),
    wrap_plots(plot_degree_osa, plot_degree_zma, nrow = 1),
    wrap_plots(plot_degree_ptr, plot_degree_sly, plot_degree_vvi, nrow = 1),
    nrow = 3
) +
    plot_annotation(
        "Degree distribution of WGD- and SSD-derived genes",
        theme = theme(plot.title = element_text(hjust = 0.5))
    )

p_deg_final
```



SSD- and WGD-derived genes do not diverge in degree. This is good, because differences in degree distributions could lead to a greater number of motifs that is solely due to the greater number of connections.

Note: *A. thaliana* seems to be the only exception, with $P < 0.03$ for the Mann-Whitney test, which suggests that there is a difference in degree for WGD- and SSD-derived genes. However, the effect size is very small (r rank biserial = 0.07), so the difference doesn't have any impact.

```
# Saving figure and degree distros object
## Degree distros
save(
    degree_distros,
    file = here("products", "result_files", "degree_distros.rda"),
    compress = "xz"
)

## Plot
p_degree_distros_ppi_network <- p_deg_final
save(
    p_degree_distros_ppi_network,
    file = here::here("products", "plots", "p_degree_distros_ppi_network.rda"),
    compress = "xz"
)
```

Exploring hypotheses

Now, we will use data to answer some questions we have.

Is there any association between duplication type and interaction tendency?

To answer this question, we will perform a Fisher's exact test to test for association between the two variables.

```
# Load required data
load(here("data", "duplicated_genes.rda"))
load(here("products", "result_files", "ppi_full.rda"))
load(here("data", "annotation.rda"))

# Define function to perform Fisher's exact tests for each species
fisher_dup_interaction <- function(duplicated_genes, ppi, annotation) {

    species_list <- names(duplicated_genes)
    test <- Reduce(rbind, lapply(species_list, function(x) {

        # Define background: all duplicated genes
        bg <- annotation[[x]]$gene_id

        # Define genes to test and duplication mode 'annotation'
        genes_that_interact <- unique(c(ppi[[x]]$dup1, ppi[[x]]$dup2))
        duplicate_annotation <- duplicated_genes[[x]][, c("gene", "type")]

        # Perform test
        enrichment <- BioNERO::enrichment_analysis(
            genes_that_interact,
            bg,
            annotation = duplicate_annotation,
            column = "type"
        )
        if(!is.null(enrichment)) {
            enrichment <- enrichment[, c("TermID", "padj")]
            enrichment$species <- x
        }
        return(enrichment)
    }))
    return(test)
}

# Perform test
dup_ppi_association <- fisher_dup_interaction(
    duplicated_genes, ppi_full, annotation
)
dup_ppi_association
```

```
##    TermID          padj       species
## 2     WGD  0.000000e+00          Gmax
## 21    WGD  3.523975e-50     Athaliana
## 22    WGD  0.000000e+00  Ptrichocarpa
## 23    WGD  8.848942e-86 Slycopersicum
## 24    WGD  2.847985e-63     Vvinifera
## 25    WGD  7.618464e-29       Osativa
## 1     SSD  4.001073e-05       Osativa
## 26    WGD 9.936232e-261         Zmays
```

As we can see, the PPI network of all species are enriched in WGD-derived genes. That means that WGD-derived genes in these species tend to interact more than the expected by chance in a scenario where the

null hypothesis is true.

```
# Save the results to a table
write_tsv(
    dup_ppi_association,
    file = here(
        "products", "tables",
        "fisher_test_association_between_duplication_type_and_ppi.tsv"
    )
)
```

Are duplicated genes that interact enriched in any process/domain?

To answer this question, we will perform an enrichment analysis for GO-BP terms and InterPro domains using the package *BioNERO*. As background, we will use all duplicated genes. We will perform enrichment analyses separately for WGD- and SSD-derived duplicates.

```
# Load required data
load(here("products", "result_files", "ppi.rda"))
load(here("data", "duplicated_genes.rda"))
load(here("data", "functional_annotation.rda"))

# Define function to perform enrichment analysis for each species
dup_ppi_sea <- function(duplicated_genes, ppi, mode = "WGD") {

    species_list <- names(duplicated_genes)
    enrich <- Reduce(rbind, lapply(species_list, function(x) {

        # Define background: all duplicated genes
        bg <- duplicated_genes[[x]]$gene

        # Define genes to test and duplication mode 'annotation'
        ppi_dup <- ppi[[x]][ppi[[x]]$type == mode, ]
        ppi_dup <- unique(c(ppi_dup$dup1, ppi_dup$dup2))

        annot_gobp <- functional_annotation[[x]]$GOBP
        annot_gobp <- annot_gobp[annot_gobp$gene %in% bg, ]
        annot_interpro <- functional_annotation[[x]]$InterPro
        annot_interpro <- annot_interpro[annot_interpro$gene %in% bg, ]

        # Perform SEA
        ## GO
        enrichment_go <- BioNERO::enrichment_analysis(
            genes = ppi_dup,
            background_genes = bg,
            annotation = annot_gobp,
            column = "GO"
        )
        if(!is.null(enrichment_go)) {
            enrichment_go <- enrichment_go[, c("TermID", "padj")]
            enrichment_go$class <- "GOBP"
        }
        ## InterPro
        enrichment_interpro <- BioNERO::enrichment_analysis(
            genes = ppi_dup,
```

```
            background_genes = bg,
            annotation = annot_interpro,
            column = "interpro"
        )
        if(!is.null(enrichment_interpro)) {
            enrichment_interpro <- enrichment_interpro[, c("TermID", "padj")]
            enrichment_interpro$class <- "interpro"
        }

        enrichment <- rbind(enrichment_go, enrichment_interpro)
        if(!is.null(enrichment)) {
            enrichment <- enrichment[, c("TermID", "padj")]
            enrichment$species <- x
        }
        return(enrichment)
    }))
    return(enrich)
}

# Perform enrichment analysis
wgd_ppi_enrichment <- dup_ppi_sea(
    duplicated_genes, ppi, mode = "WGD"
)

ssd_ppi_enrichment <- dup_ppi_sea(
    duplicated_genes, ppi, mode = "SSD"
)

# Save enrichment analysis for duplicated genes that interact
readr::write_tsv(
    wgd_ppi_enrichment,
    file = here::here(
        "products", "tables",
        "functional_enrichment_wgd_genes_that_interact.tsv"
    )
)
readr::write_tsv(
    ssd_ppi_enrichment,
    file = here::here(
        "products", "tables",
        "functional_enrichment_ssd_genes_that_interact.tsv"
    )
)
```

A deeper inspection shows that WGD-derived genes that interact at the protein level are enriched in process associated to:

- *Glycine max*: lipid metabolism, photosynthesis, heme biosynthesis, signal transduction, glucose oxidation, proteolysis, amino acid oxidation, cutin biosynthesis, cell wall biogenesis, redox homeostasis, nucleic acid metabolism, mRNA processing, translation, transcriptional regulation.
- *A. thaliana*: cell cycle, photosynthesis, signal transduction, glucose oxidation, mRNA processing, vesicle trafficking, proteolysis, amino acid oxidation, redox homeostasis, translation, lipid metabolism, transcriptional regulation.

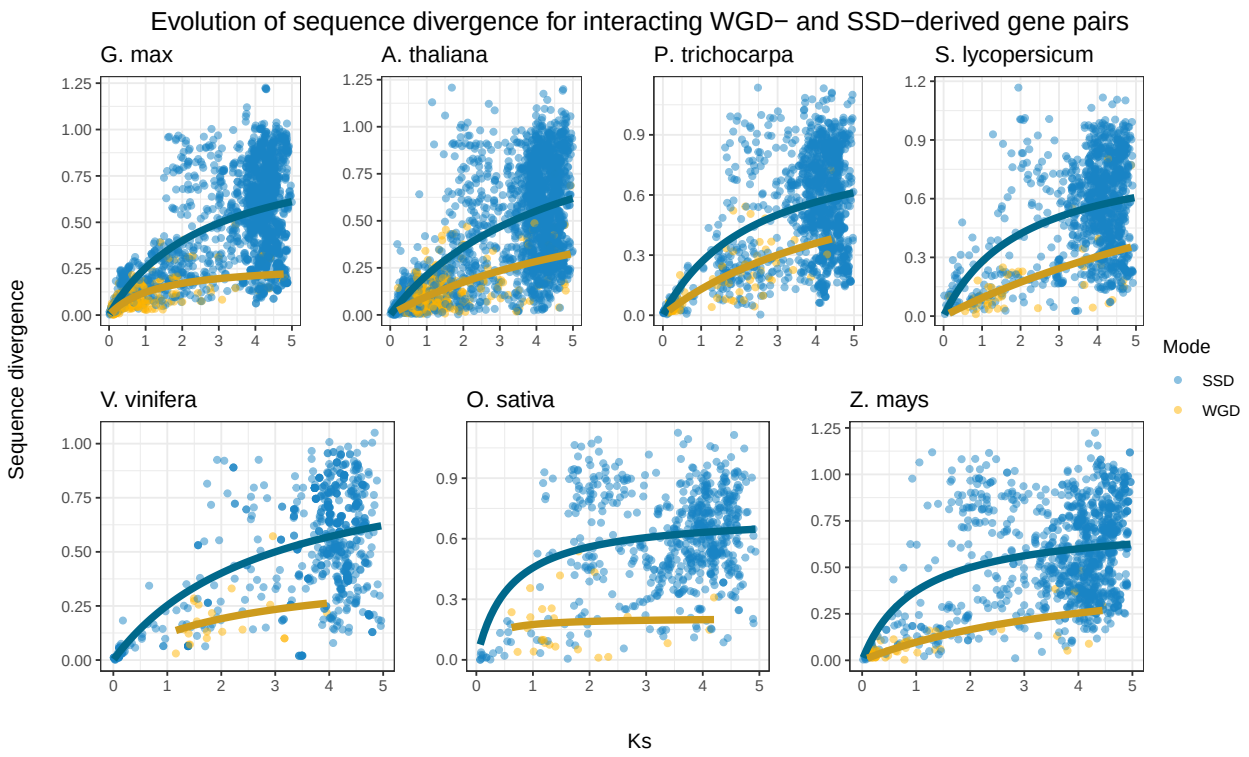
- *P. trichocarpa*: lipid metabolism, glucose oxidation, signal transduction, amino acid oxidation, photosynthesis, nitrate assimilation, cell wall biogenesis, redox homeostasis, translation, transcriptional regulation.
- *S. lycopersicum*: photosynthesis, glucose oxidation, lipid metabolism, signal transduction
- *V. vinifera*: signal transduction, lipid metabolism, glucose oxidation, redox homeostasis
- *O. sativa*: cell cycle, signal transduction, translation.
- *Z. mays*: photosynthesis, glucose oxidation, cell wall biogenesis, translation, signal transduction, redox homeostasis, transcriptional regulation.

Are WGD-derived genes more constrained to evolve divergent functions?

To answer this question, we will explore sequence divergence over time for WGD- and SSD-derived genes. Ka will be used as a proxy for sequence divergence, and Ks will be used to represent time. We will fit Michaelis-Menten curves to the scatterplot.

```
load(here("products", "result_files", "ppi.rda"))

# Define function to plot scatterplot with Michaelis-Menten curve
scatter_mm <- function(ppi) {

    # Remove Ks values > 5
    ppi <- ppi %>%
        filter(ks <= 5)

    # Plot scatterplot with Michaelis-Menten curves
    p <- ggplot(ppi %>% dplyr::rename(Mode = type), aes(x = ks, y = ka)) +
        geom_point(aes(color = Mode), alpha = 0.5) +
        theme_bw() +
        labs(x = "", y = "") +
        scale_color_manual(values = dup_palette) +
        geom_smooth(
            method = "nls", formula = y ~ Vmax * x / (Km + x),
            start = list(Vmax = 50, Km = 2),
            se = FALSE,
            colour = "deepskyblue4", size = 2,
            data = filter(ppi, type == "SSD")
        ) +
        geom_smooth(
            method = "nls", formula = y ~ Vmax * x / (Km + x),
            start = list(Vmax = 50, Km = 2),
            se = FALSE,
            colour = "goldenrod3", size = 2,
            data = filter(ppi, type == "WGD")
        )

    return(p)
}

plot_mm <- lapply(seq_along(ppi), function(x) {

    # Create character scalar of plot title
    species <- names(ppi)[x]
```

```
    title <- paste0(
        substr(species, 1, 1), ". ",
        substr(species, 2, nchar(species))
    )

    # Plot scatterplot with Michaelis-Menten curve + species name in title
    p <- scatter_mm(ppi[[species]]) +
        labs(title = title) +
        theme(legend.position = "none")
    return(p)
})
names(plot_mm) <- names(ppi)

# Combine plots
## Get legend
legend <- ggpubr::get_legend(scatter_mm(ppi$Gmax))

# Get upper and lower plots
p_mm_upper <- ggpubr::ggarrange(
    plot_mm$Gmax, plot_mm$Athaliana, plot_mm$Ptrichocarpa,
    plot_mm$Slycopersicum, nrow = 1
)
p_mm_lower <- ggpubr::ggarrange(
    plot_mm$Vvinifera, plot_mm$Osativa, plot_mm$Zmays, nrow = 1
)

## Combine upper and lower plots and add common legend
p_mm_final <- ggarrange(
    p_mm_upper, p_mm_lower,
    nrow = 2,
    common.legend = TRUE,
    legend = "right", legend.grob = legend
)

p_mm_final <- annotate_figure(
    p_mm_final,
    top = text_grob(
        "Evolution of sequence divergence for interacting WGD- and SSD-derived gene pairs",
        size = 15
    ),
    left = text_grob("Sequence divergence", rot = 90),
    bottom = "Ks"
)

p_mm_final
```



The figure shows that interacting WGD-derived gene pairs are more constrained to diverge in sequence than interacting SSD-derived gene pairs. This finding is in line with previous reports that demonstrate that WGD-derived genes encode proteins associated with intricate systems, such as components of signal transduction networks, transcriptional regulation, multi-subunit protein complexes.

```
# Save plot
p_sequence_divergence_duplicates <- p_mm_final
save(
    p_sequence_divergence_duplicates,
    file = here(
        "products", "plots",
        "p_sequence_divergence_duplicates.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     2023-05-02
##  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)
##  bayestestR             0.13.1    2023-04-07 [1] CRAN (R 4.3.0)
##  Biobase                2.60.0    2023-04-25 [1] Bioconductor
##  BiocGenerics           0.46.0    2023-04-25 [1] Bioconductor
##  BiocManager            1.30.20   2023-02-24 [1] CRAN (R 4.3.0)
##  BiocParallel           1.34.0    2023-04-25 [1] Bioconductor
##  BiocStyle              2.28.0    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)
##  broom                  1.0.4     2023-03-11 [1] CRAN (R 4.3.0)
##  cachem                 1.0.8     2023-05-01 [1] CRAN (R 4.3.0)
##  car                    3.1-2     2023-03-30 [1] CRAN (R 4.3.0)
##  carData                3.0-5     2022-01-06 [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
##  correlation            0.8.4     2023-04-06 [1] CRAN (R 4.3.0)
##  cowplot                1.1.1     2020-12-30 [1] CRAN (R 4.3.0)
##  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)
##  datawizard             0.7.1     2023-04-03 [1] CRAN (R 4.3.0)
##  DBI                    1.1.3     2022-06-18 [1] CRAN (R 4.3.0)
##  DelayedArray           0.25.0    2022-11-01 [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
##  effectsize             0.8.3     2023-01-28 [1] CRAN (R 4.3.0)
##  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)
##  farver                 2.1.1     2022-07-06 [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)
##  ggpubr              * 0.6.0    2023-02-10 [1] CRAN (R 4.3.0)
##  ggrepel               0.9.3    2023-02-03 [1] CRAN (R 4.3.0)
##  ggsignif              0.6.4    2022-10-13 [1] CRAN (R 4.3.0)
##  ggstatsplot         * 0.11.1   2023-04-14 [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
##  insight               0.19.1   2023-03-18 [1] CRAN (R 4.3.0)
##  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)
##  labeling              0.4.2    2020-10-20 [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)
##  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)
##  paletteer              1.5.0     2022-10-19 [1] CRAN (R 4.3.0)
##  parameters             0.21.0    2023-04-19 [1] CRAN (R 4.3.0)
##  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)
##  rematch2               2.1.2     2020-05-01 [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)
##  rstatix                0.7.2     2023-02-01 [1] CRAN (R 4.3.0)
##  rstudioapi             0.14      2022-08-22 [1] CRAN (R 4.3.0)
##  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)
##  statsExpressions       1.5.0     2023-02-19 [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   1.30.0    2023-04-25 [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)
##  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)
##  zeallot               0.1.0     2018-01-28 [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
##
## ------------------------------------------------------------------------------
```