Figure S2

Toshihiro Arae

General directory setting

```r
wd <- here::here()
shared <- fs::path(fs::path_dir(wd), "shared")
```

Loading packages

```r
library(magrittr)
library(ggplot2)
```

Load common R scripts

```r
#source(fs::path(wd, "script_r", "MISC.R"))
#source(fs::path(here::here(), "script_r", "MISC_PALETTE.R"))
```

Load script

```r
source(fs::path(wd, "script_r", "MISC_FIG.R"))
readLines(fs::path(wd, "script_r", "MISC_FIG.R")) %>% cat(sep = "\n")
```

```
library(magrittr)
library(ggplot2)

COL_PALETTE <-
  viridis::inferno(6, begin = .1, end = .9) %>%
  rev() %>%
  setNames(nm = c("ZT0", "ZT3", "ZT6", "ZT12", "ZT18", "ZT21"))

LABEL_PALETTE <-
  COL_PALETTE %>%
  prismatic::clr_darken(shift = .15) %>%
  setNames(names(COL_PALETTE))

label_number_si <-
  purrr::partial(scales::label_number, scale_cut = scales::cut_short_scale())

ggsave_single <- function(..., width = 86, height = 230, dpi = 300) {
  f <- purrr::partial(ggsave, width = width, height = height, dpi = dpi, units = "mm")
  f(...)
}

ggsave_double <- function(..., width = 178, height = 230, dpi = 300) {
  f <- purrr::partial(ggsave, width = width, height = height, dpi = dpi, units = "mm")
  f(...)
}

#' Utility functions for making secondary y-axis
#' @param y1 numeric vector
#' @param y2 numeric vector
#' @name util_2nd_axis
```

```r
#' @examples
#' make_scale_y1_to_y2(1:5, 6:10)(1:10)
#' make_scale_y2_to_y1(1:5, 6:10)(1:10)
#'
#' iris_ <- dplyr::select(iris, x = Sepal.Length, y1 = Petal.Length, y2 = Petal.Width)
#' gp1 <-
#'   iris_ %>%
#'   ggplot() +
#'   geom_point(aes(x, y1), color = "#CD3700") +
#'   geom_point(aes(x, y2), color = "#473C8B")
#'
#' to_y1 <- with(iris_, {make_scale_y2_to_y1(y1, y2)})
#' to_y2 <- with(iris_, {make_scale_y1_to_y2(y1, y2)})
#' gp2 <-
#'   iris_ %>%
#'   ggplot() +
#'   geom_point(aes(x, y1), color = "#CD3700") +
#'   geom_point(aes(x, y = to_y1(y2)), color = "#473C8B") +
#'   scale_y_continuous(sec.axis = sec_axis(trans = to_y2, name = "y2"))
#' patchwork::wrap_plots(gp1, gp2)
#'
NULL

#' Create transformation function of range(y1) to range(y2)
#' @rdname util_2nd_axis
#' @export
#'
make_scale_y1_to_y2 <- function(y1, y2) {
  function(n) {
    scales:::rescale.numeric(
      n,
      to = range(y2, na.rm = TRUE, finite = TRUE),
      from = range(y1, na.rm = TRUE, finite = TRUE)
    )
  }
}

#' Create transformation function of range(y2) to range(y1)
#' @rdname util_2nd_axis
#' @export
#'
make_scale_y2_to_y1 <- function(y1, y2) {
  function(n) {
    scales:::rescale.numeric(
      n,
      to = range(y1, na.rm = TRUE, finite = TRUE),
      from = range(y2, na.rm = TRUE, finite = TRUE)
    )
  }
}

#' Create transformation function of range(y2) to range(y1)
#' @rdname util_2nd_axis
#' @export
#'
make_scale_y2_to_y1_se <- function(y1, y2) {
  to <- range(y1, na.rm = TRUE, finite = TRUE)
  from <- range(y2, na.rm = TRUE, finite = TRUE)
  function(n) n / (diff(from) / diff(to))
}
```

Directory setting

```r
path_out <- function(...) fs::path(wd, "analysis", "fig", "figS02", ...)
fs::dir_create(path_out())
```

Define some functions

Data scaling

```r
z_score <- function(x) scale(x)[,1]
centering <- function(x) scale(x, scale = FALSE)[,1]
do_nothing <- function(x) x

centering_then_z_score <- function(x) {
  y1 <- scale(x[1:6], scale = FALSE)[,1]
  y2 <- scale(x[7:12], scale = FALSE)[,1]
  y3 <- scale(x[13:18], scale = FALSE)[,1]
  # scale(c(y1))[,1]
  # scale(c(y1, y2))[,1]
  scale(c(y1, y2, y3))[,1]
}

# centering_then_z_score(c(1:6, (7:12)*2, (13:18)*.5)) %>% plot()
```

Preparing data.frame

```r
#' Convert input filtered tibble to tibble for plotting
#' @param tbl a input tibble
#' @param ... will be passed to `dplyr::arrange()`
convert_to_tbl_plot <- function(tbl, ...) {
  SAMPLES <-
    colnames(tbl) %>%
    stringr::str_extract("^(rna|morf|te_morf)_", group = 1) %>%
    unique() %>%
    {.[!is.na(.)]}

  if(any(SAMPLES %in% "rna"))
    tbl <- tbl %>% dplyr::mutate(rna_coef_zt0 = 0, .before = rna_coef_zt3)
  if(any(SAMPLES %in% "morf"))
    tbl <- tbl %>% dplyr::mutate(morf_coef_zt0 = 0, .before = morf_coef_zt3)
  if(any(SAMPLES %in% "te_morf"))
    tbl <- tbl %>% dplyr::mutate(te_morf_coef_zt0 = 0, .before = te_morf_coef_zt3)

  tbl %>%
    dplyr::arrange(...) %>%
    dplyr::mutate(AGI = forcats::fct_inorder(AGI)) %>%
    tidyr::pivot_longer(cols = !c(AGI)) %>%
    dplyr::mutate(
      time =
        stringr::str_extract(name, "zt\\d+") %>%
        stringr::str_to_upper() %>%
        forcats::fct_inorder(),
      type =
        stringr::str_extract(name, "^(rna|morf|te_morf)_", group = 1) %>%
        {dplyr::case_when(
          . == "rna" ~ "RNA",
          . == "morf" ~ "RPF",
          . == "te_morf" ~ "TE mORF"
        )} %>%
        forcats::fct_inorder()
    ) %>%
    dplyr::select(!name) %>%
    tidyr::pivot_wider(names_from = type, values_from = value) %>%
    tidyr::pivot_longer(cols =
                          c(rna = "RNA", morf = "RPF", te_morf = "TE mORF")[SAMPLES] %>%
                          set_names(NULL),
                        names_to = "type") %>%
    dplyr::mutate(type = forcats::fct_relevel(type, c("RNA", "RPF", "TE mORF")))
}

#' Scale the value column of tbl_plot by a given function
#' Note: This function perform scaling for each groups (AGI and **type**)
#' @param tbl tbl_plot
#' @param scaling_func a function to scale `value`
scale_tbl_plot <- function(tbl, scaling_func) {
  tbl %>%
    dplyr::group_by(AGI, type) %>%
    dplyr::mutate(scaled = scaling_func(value)) %>%
    dplyr::ungroup()
}
```

Heatmap

```r
plot_heatmap <- function(tbl) {
  tbl %>%
    ggplot(aes(time, AGI)) +
    geom_tile(data = \(x) dplyr::filter(x, type != "TE mORF"), aes(fill = scaled)) +
    scale_x_discrete(expand = expansion(0)) +
    scale_y_discrete(expand = expansion(.5)) +
    scale_fill_gradient2(
      low = scales::muted("blue"), high = scales::muted("red"),
      limits = c(-5, 4),
      guide = guide_colorbar(
        title = "Log2FC against to Ave.\n(for RNA and RPF)",
        title.position = "right",
        barwidth = 2, barheight = 40, default.unit = "mm",
        label.position = "left", label.hjust = 1
      )) +
    facet_grid(rows = vars(category), cols = vars(type), scales = "free_y") +
    theme_minimal(base_size = 10) +
    theme(
      axis.text.x = element_text(angle = 90),
      strip.text = element_text(size = 10, vjust = 1.1),
      strip.text.y.right = element_text(size = 10, vjust = .5, angle = 0),
      legend.title = element_text(angle = -90),
      plot.margin = margin(10/2, 0, 0, 0)
    ) +
    ggnewscale::new_scale_fill() +
    geom_tile(data = \(x) dplyr::rename(dplyr::filter(x, type == "TE mORF"), scaled2 = scaled),
              aes(fill = scaled2)) +
    scale_fill_gradient2(
      low = scales::muted("blue"), high = scales::muted("red"),
      limits = c(-2, 2),
      guide = guide_colorbar(
        title = "Log2FC against to Ave.\n(for TE mORF)",
        title.position = "right",
        barwidth = 2, barheight = 40, default.unit = "mm",
        label.position = "left", label.hjust = 1
      )) +
    NULL
}
```

Load input data

```r
inf <- fs::path(wd, "analysis", "list_summary", "summary_all.csv")
tbl_input <-
  readr::read_csv(inf, show_col_types = FALSE) %>%
  dplyr::filter(!grepl("AT[CM]G", AGI)) %>%
  dplyr::filter(!is.na(te_morf_padj)) %>%
  dplyr::arrange(rna_Phase, morf_Phase)

MAP_AGI2CATEGORY <-
  c(
    tbl_input %>%
      dplyr::filter(
        te_morf_padj < 0.05,
        rna_BF_BH < 0.05, (rna_Max_Amp/rna_Max) > 0.5,
        morf_BF_BH >= 0.05, (morf_Max_Amp/morf_Max) <= 0.5) %>%
      dplyr::pull(AGI) %>%
      {setNames(rep("rnaonly", length(.)), .)},
    tbl_input %>%
      dplyr::filter(
        te_morf_padj < 0.05,
        morf_BF_BH < 0.05, (morf_Max_Amp/morf_Max) > 0.5,
        rna_BF_BH >= 0.05, (rna_Max_Amp/rna_Max) <= 0.5) %>%
      dplyr::pull(AGI) %>%
      {setNames(rep("riboonly", length(.)), .)},
    tbl_input %>%
      dplyr::filter(
        te_morf_padj > 0.05,
        morf_BF_BH < 0.05, (morf_Max_Amp/morf_Max) > 0.5,
        rna_BF_BH < 0.05, (rna_Max_Amp/rna_Max) > 0.5) %>%
      dplyr::pull(AGI) %>%
      {setNames(rep("synchro", length(.)), .)},
    tbl_input %>%
      dplyr::filter(
        te_morf_padj < 0.05,
        morf_BF_BH < 0.05, (morf_Max_Amp/morf_Max) > 0.5,
        rna_BF_BH < 0.05, (rna_Max_Amp/rna_Max) > 0.5) %>%
      dplyr::filter(rna_Phase != morf_Phase) %>%
      dplyr::pull(AGI) %>%
      {setNames(rep("asynchro_phase", length(.)), .)},
    tbl_input %>%
      dplyr::filter(
        te_morf_padj < 0.05,
        morf_BF_BH < 0.05, (morf_Max_Amp/morf_Max) > 0.5,
        rna_BF_BH < 0.05, (rna_Max_Amp/rna_Max) > 0.5) %>%
      dplyr::filter(rna_Phase == morf_Phase) %>%
      dplyr::pull(AGI) %>%
      {setNames(rep("asynchro_amplitude", length(.)), .)}
  )
str(MAP_AGI2CATEGORY)
```

```
 Named chr [1:5272] "rnaonly" "rnaonly" "rnaonly" "rnaonly" "rnaonly" ...
 - attr(*, "names")= chr [1:5272] "AT1G30690" "AT1G79160" "AT2G21300" "AT2G21830" ...
```

```r
li_go_photo <-
  c(
    "GO:0009522", # photosystem I
    "GO:0009523", # photosystem II
    "GO:0009765"  # photosynthesis, light harvesting
  )

tbl_go2 <-
```

```
  AnnotationDbi::as.list.Bimap(org.At.tair.db::org.At.tairGO2ALLTAIRS) %>%
  {.[li_go_photo]} %>%
  purrr::imap(~ {
    tibble::tibble(AGI = .x, GOID = .y, term = clusterProfiler::go2term(.y)$Term, ONTO =
clusterProfiler::go2ont(.y)$Ontology)
  }) %>%
  dplyr::bind_rows()
```

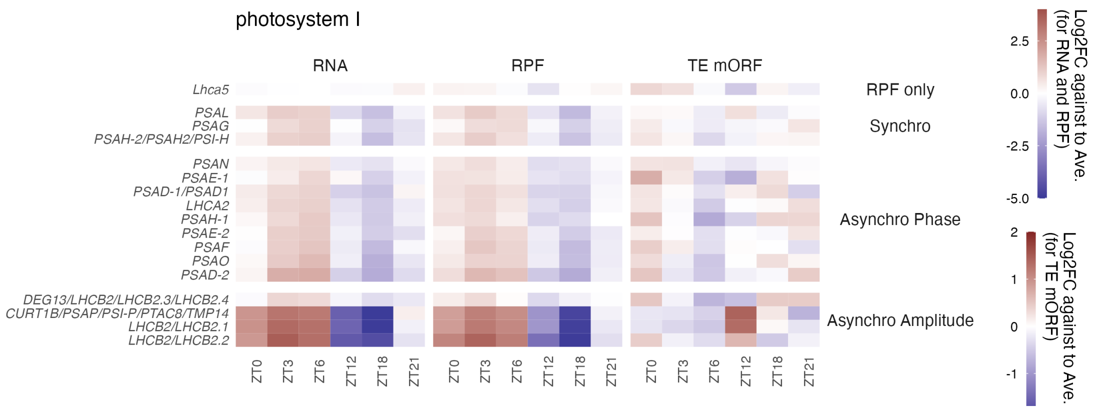
```
li_tbl_go <- tbl_go2 %>% split(.$GOID)

calc_mm <- function(x) sum(grid::convertUnit(x, "mm"))

AGI2SYMBOL <- function(AGI) {
  AnnotationDbi::select(
    x = org.At.tair.db::org.At.tair.db,
    keys = AGI,
    columns = "SYMBOL",
    keytype = "TAIR"
  ) %>%
    tibble::as_tibble() %>%
    dplyr::with_groups(TAIR, dplyr::summarise, symbol = paste(SYMBOL, collapse = "/")) %>%
    # return()
    dplyr::mutate(symbol = ifelse(symbol == "NA", TAIR, symbol)) %>%
    dplyr::pull(symbol)
}
```

Plotting

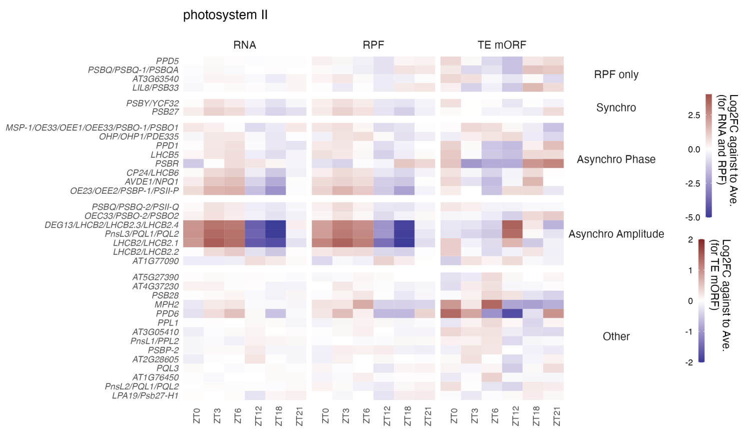
```
for(i in seq_along(li_tbl_go)) {
  temp_tbl_go <- li_tbl_go[[i]]
  temp_id <- names(li_tbl_go)[i]
  temp_label <- paste0(temp_tbl_go$term[1])

  tbl <-
    tbl_input %>%
    dplyr::filter(AGI %in% temp_tbl_go$AGI) %>%
    dplyr::mutate(AGI = forcats::fct_inorder(AGI)) %>%
    dplyr::select(AGI,
                  dplyr::starts_with("rna_coef_z"),
                  dplyr::starts_with("morf_coef_z"),
                  dplyr::starts_with("te_morf_coef_z"))

  tbl_plot <- convert_to_tbl_plot(tbl, AGI) %>% scale_tbl_plot(scaling_func = centering)
  tbl_plot <-
    tbl_plot %>%
    dplyr::arrange(type, time) %>%
    dplyr::with_groups(AGI, tidyr::nest) %>%
    dplyr::mutate(
      maximum = purrr::map_dbl(data, ~ max(unlist(dplyr::pull(.x, scaled)[1:6]))),
      zt_index = purrr::map_dbl(data, ~ which.max(unlist(dplyr::pull(.x, scaled)[1:6])))
    ) %>%
    dplyr::arrange(desc(zt_index), desc(maximum)) %>%
    dplyr::mutate(AGI = forcats::fct_inorder(AGI)) %>%
    dplyr::select(AGI, data) %>%
    tidyr::unnest(data) %>%
```

```r
    dplyr::mutate(category = MAP_AGI2CATEGORY[as.character(AGI)]) %>%
    dplyr::mutate(category = dplyr::case_when(
      category == "asynchro_phase" ~ "Asynchro Phase",
      category == "asynchro_amplitude" ~ "Asynchro Amplitude",
      category == "riboonly" ~ "RPF only",
      category == "rnaonly" ~ "RNA only",
      category == "synchro" ~ "Synchro",
      .default = "Other"
    )) %>%
    dplyr::mutate(category = forcats::fct_relevel(category, c(
      "RNA only", "RPF only", "Synchro",
      "Asynchro Phase", "Asynchro Amplitude", "Other"
    )))

  gp_hm <-
    tbl_plot %>%
    plot_heatmap() +
    labs(title = temp_label, x = "", y = "") +
    scale_y_discrete(label = AGI2SYMBOL) +
    theme(
      axis.text.y = element_text(face = "italic", size = 8, hjust = 1),
      panel.grid = element_blank(),
      strip.clip = "off"
    )
  N <-
    split(tbl_plot, tbl_plot$category) %>%
    purrr::map_int(~ length(unique(.x$AGI)))
  gt <- ggplotGrob(gp_hm)
  gt$widths[5] <- ggplot2::unit(40, units = "mm")
  gt$widths[7] <- ggplot2::unit(40, units = "mm")
  gt$widths[9] <- ggplot2::unit(40, units = "mm")
  gt$heights[4] <- ggplot2::unit(3, units = "mm")
  gt$heights[8] <- ggplot2::unit(N[1]*3, units = "mm")
  if(length(N) > 1) gt$heights[10] <- ggplot2::unit(N[2]*3, units = "mm")
  if(length(N) > 2) gt$heights[12] <- ggplot2::unit(N[3]*3, units = "mm")
  if(length(N) > 3) gt$heights[14] <- ggplot2::unit(N[4]*3, units = "mm")
  if(length(N) > 4) gt$heights[16] <- ggplot2::unit(N[5]*3, units = "mm")
  if(length(N) > 5) gt$heights[18] <- ggplot2::unit(N[6]*3, units = "mm")
  label <- paste0(i, "_", stringr::str_remove(temp_id, ':'))
  ggsave_ <- purrr::partial(ggsave, width = calc_mm(gt$widths),
                            height = calc_mm(gt$heights), units = "mm")
  ggsave_(gt, filename = path_out(glue::glue("{label}.png")))
  # ggsave_(gt, filename = path_out(glue::glue("{label}.svg")))
  # ggsave_(gt, filename = path_out(glue::glue("{label}.pdf")), device = cairo_pdf)
}
```

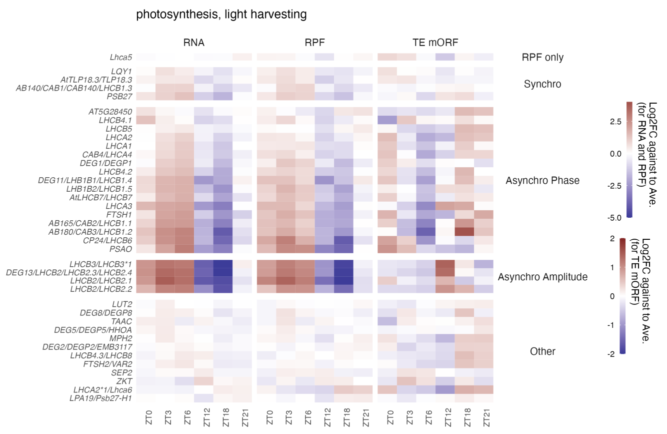
1_GO0009522.png



2_GO0009523.png



3_GO0009765.png



Sessioninfo

```r
sessionInfo()
```

```
R version 4.2.1 (2022-06-23)
Platform: aarch64-apple-darwin20 (64-bit)
Running under: macOS Ventura 13.1

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/lib/libRlapack.dylib

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

attached base packages:
[1] stats     graphics  grDevices datasets  utils     methods   base

other attached packages:
[1] ggplot2_3.4.2  magrittr_2.0.3

loaded via a namespace (and not attached):
  [1] ggnewscale_0.4.8       fgsea_1.22.0           colorspace_2.0-3
  [4] ggtree_3.4.4           rprojroot_2.0.3        qvalue_2.28.0
  [7] XVector_0.36.0         fs_1.5.2               aplot_0.1.8
 [10] rstudioapi_0.14        farver_2.1.1           org.At.tair.db_3.15.1
 [13] graphlayouts_1.0.1     ggrepel_0.9.4          bit64_4.0.5
 [16] AnnotationDbi_1.58.0   fansi_1.0.3            scatterpie_0.1.8
 [19] codetools_0.2-18       splines_4.2.1          cachem_1.0.6
 [22] GOSemSim_2.22.0        knitr_1.42             polyclip_1.10-4
 [25] jsonlite_1.8.4         GO.db_3.15.0           png_0.1-7
 [28] ggforce_0.4.1          BiocManager_1.30.18    readr_2.1.4
 [31] compiler_4.2.1         httr_1.4.5             Matrix_1.6-4
 [34] fastmap_1.1.0          lazyeval_0.2.2         cli_3.6.0
 [37] tweenr_2.0.2           htmltools_0.5.3        tools_4.2.1
 [40] igraph_1.3.5           gtable_0.3.1           glue_1.6.2
 [43] GenomeInfoDbData_1.2.8 reshape2_1.4.4         DO.db_2.9
 [46] dplyr_1.1.1            fastmatch_1.1-3        Rcpp_1.0.11
 [49] enrichplot_1.16.2      Biobase_2.56.0         vctrs_0.6.1
 [52] Biostrings_2.64.1      ape_5.6-2              nlme_3.1-157
 [55] ggraph_2.1.0           xfun_0.40              stringr_1.5.0
 [58] lifecycle_1.0.3        clusterProfiler_4.4.4  renv_1.0.3
 [61] DOSE_3.22.1            zlibbioc_1.42.0        MASS_7.3-57
 [64] scales_1.2.1           tidygraph_1.2.2        vroom_1.6.0
 [67] ragg_1.2.5             hms_1.1.3              parallel_4.2.1
 [70] RColorBrewer_1.1-3     prismatic_1.1.1        yaml_2.3.6
 [73] memoise_2.0.1          gridExtra_2.3          downloader_0.4
 [76] ggfun_0.1.1            yulab.utils_0.0.5      stringi_1.7.12
 [79] RSQLite_2.2.18         S4Vectors_0.34.0       tidytree_0.4.1
 [82] BiocGenerics_0.42.0    BiocParallel_1.30.4    GenomeInfoDb_1.32.4
 [85] systemfonts_1.0.4      rlang_1.1.0            pkgconfig_2.0.3
 [88] bitops_1.0-7           evaluate_0.20          lattice_0.20-45
 [91] purrr_1.0.1            labeling_0.4.2         treeio_1.20.2
 [94] patchwork_1.1.2        shadowtext_0.1.2       bit_4.0.5
 [97] tidyselect_1.2.0       here_1.0.1             plyr_1.8.7
[100] R6_2.5.1               magick_2.7.3           IRanges_2.30.1
[103] generics_0.1.3         DBI_1.1.3              pillar_1.9.0
[106] withr_2.5.0            KEGGREST_1.36.3        RCurl_1.98-1.9
[109] tibble_3.2.1           crayon_1.5.2           utf8_1.2.2
[112] tzdb_0.3.0             rmarkdown_2.24         viridis_0.6.2
[115] grid_4.2.1             data.table_1.14.4      blob_1.2.3
[118] forcats_1.0.0          digest_0.6.31          tidyr_1.3.0
[121] textshaping_0.3.6      gridGraphics_0.5-1     stats4_4.2.1
[124] munsell_0.5.0          viridisLite_0.4.1      ggplotify_0.1.0
```