Figure 7A and Figure S6

Toshihiro Arae

2024-03-27

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
#' @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

```
path_out_fig7a <- function(...) fs::path(wd, "analysis", "fig", "fig07A", ...)
fs::dir_create(path_out_fig7a())
path_out_figS6 <- function(...) fs::path(wd, "analysis", "fig", "figS06", ...)
fs::dir_create(path_out_figS6())
```

Define some functions

Data scaling

```
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, y2, y3))[,1]
}
```

Preparing data.frame

```
#' 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, ...) {
  tbl <-
    tbl %>%
    dplyr::mutate(rna_coef_zt0 = 0, .before = rna_coef_zt3) %>%
    dplyr::mutate(te_morf_coef_zt0 = 0, .before = te_morf_coef_zt3) %>%
    dplyr::mutate(te_uorf_coef_zt0 = 0, .before = te_uorf_coef_zt3)

  tbl %>%
    dplyr::arrange(...) %>%
    dplyr::mutate(AGI = forcats::fct_inorder(AGI)) %>%
    tidyr::pivot_longer(cols = !c(AGI, rna_Phase, morf_Phase, uorf_Phase, is_same_peak)) %>%
    dplyr::mutate(
      time =
        stringr::str_extract(name, "zt\\d+") %>%
        stringr::str_to_upper() %>%
        forcats::fct_inorder(),
      type =
        stringr::str_extract(name, "rna|morf|uorf") %>%
        {dplyr::case_when(
          . == "rna" ~ "RNA",
          . == "morf" ~ "TE mORF",
          . == "uorf" ~ "TE uORF"
        )} %>%
        forcats::fct_inorder()
    ) %>%
    dplyr::select(!name) %>%
    tidyr::pivot_wider(names_from = type, values_from = value) %>%
    tidyr::pivot_longer(cols = c("RNA", "TE mORF", "TE uORF"), names_to = "type") %>%
    dplyr::mutate(type = forcats::fct_relevel(type, c("RNA", "TE mORF", "TE uORF")))
}
```

```r
#' Scale the value column of tbl_plot by a given function
#' @param tbl tbl_plot
#' @param scaling_func a function to scale `value`
scale_tbl_plot <- function(tbl, scaling_func) {
  tbl %>%
    dplyr::group_by(AGI) %>%
    dplyr::arrange(type, time) %>%
    dplyr::mutate(scaled = scaling_func(value)) %>%
    dplyr::ungroup()
}

sort_tbl <- function(tbl, ...) {
  tbl %>%
    dplyr::with_groups(c(AGI, rna_Phase, morf_Phase, uorf_Phase),
                       tidyr::nest) %>%
    dplyr::arrange(...) %>%
    tidyr::unnest(cols = data)
}
```

Heatmap

```r
plot_heatmap <- function(tbl) {
  lab <- rev(levels(forcats::fct_inorder(tbl$AGI)))
  tbl_plot <-
    tbl %>%
    dplyr::filter(AGI %in% lab) %>%
    dplyr::mutate(AGI = forcats::fct_relevel(AGI, lab))

  tbl_plot %>%
    ggplot(aes(time, AGI)) +
    geom_raster(aes(fill = scaled)) +
    scale_x_discrete(expand = expansion(0)) +
    scale_y_discrete(expand = expansion(0)) +
    scale_fill_gradient2(low = scales::muted("blue"), high = scales::muted("red")) +
    facet_grid(cols = vars(type)) +
    theme_void(base_size = 10) +
    theme(
      axis.text.x = element_text(angle = 90, color = LABEL_PALETTE),
      strip.text = element_text(size = 10, vjust = 1.1),
      legend.title = element_text(angle = -90),
      plot.margin = margin(10/2, 10/2, 10/2, 10/2)
    ) +
    guides(fill = guide_colorbar(
      title = "Z-score of Log2FC",
      title.position = "right",
      barwidth = 2, barheight = 40, default.unit = "mm",
      label.position = "left", label.hjust = 1
    ))
}

theme_empty <- function() {
  list(
    theme_void(),
    theme(
      strip.background = element_blank(),
      strip.text = element_blank(),
      legend.position = "none"
    )
  )
}
```

Load input data

```
inf <- fs::path(wd, "analysis", "list_summary", "summary_all.csv")

# 19375 genes
# which were tested their mORF TE are changed or not,
# and their successfully finished eJTK analysis.
AGI_19375 <-
  readr::read_csv(inf, show_col_types = FALSE) %>%
  dplyr::filter(!grepl("AT[CM]G", AGI)) %>%
  dplyr::filter(!is.na(te_morf_padj), !is.na(rna_BF_BH), !is.na(morf_BF_BH)) %>%
  .$AGI
str(AGI_19375)
```

```
 chr [1:19375] "AT1G01010" "AT1G01020" "AT1G01030" "AT1G01040" "AT1G01050" ...
```

```
# 4739 genes
# which were satisfy the above conditions,
# and also perform P-site read count for uORFs.
AGI_4739 <-
readr::read_csv(inf, show_col_types = FALSE) %>%
  dplyr::filter(AGI %in% AGI_19375) %>%
  dplyr::filter(!is.na(uorf_baseMean)) %>%
  .$AGI
str(AGI_4739)
```

```
 chr [1:4739] "AT1G01060" "AT1G01210" "AT1G01230" "AT1G01240" "AT1G01260" ...
```

Prepare lists of genes

```
tbl_padj_morf <-
  readr::read_csv(fs::path(wd, "analysis", "deseq2_te", "deg_all.csv"),
                  show_col_types = FALSE) %>%
  dplyr::select(AGI, padj_morf = padj)

tbl_padj_uorf <-
  readr::read_csv(fs::path(wd, "analysis", "deseq2_te_uorf", "deg_all.csv"),
                  show_col_types = FALSE) %>%
  dplyr::select(AGI, padj_uorf = padj)

tbl_phase <-
  fs::path(wd, "analysis", "list_summary", "summary_all.csv") %>%
  readr::read_csv(show_col_types = FALSE) %>%
  dplyr::select(AGI, morf_Phase, uorf_Phase)

tbl_for_filter <-
  readr::read_csv(fs::path(wd, "analysis", "deseq2_te_morf_uorf", "deg_all.csv"),
                  show_col_types = FALSE) %>%
  dplyr::select(AGI, padj) %>%
  dplyr::left_join(tbl_padj_morf, by = "AGI") %>%
  dplyr::left_join(tbl_padj_uorf, by = "AGI") %>%
  dplyr::left_join(tbl_phase, by = "AGI") %>%
  dplyr::filter(AGI %in% AGI_4739) %>%
  dplyr::filter(!is.na(padj) & !is.na(padj_uorf))
tbl_for_filter
```

```
# A tibble: 4,641 × 6
   AGI          padj padj_morf     padj_uorf morf_Phase uorf_Phase
   <chr>      <dbl>     <dbl>         <dbl>      <dbl>      <dbl>
 1 AT1G01060 0.0103  2.03e- 4 0.0254              0          0
 2 AT1G01210 0.406   7.07e- 2 0.103              21         18
 3 AT1G01230 0.310   1.24e- 2 0.00655             6          3
 4 AT1G01240 0.647   1.09e-17 0.00000000756       0          6
 5 AT1G01260 0.618   8.97e- 2 0.813               0          9
 6 AT1G01370 0.691   8.69e- 1 0.690               3          6
 7 AT1G01430 0.931   1.59e- 2 0.00454             0         15
 8 AT1G01440 0.817   5.35e- 7 0.00123             0          0
 9 AT1G01490 0.0514  6.63e- 8 0.215               0         18
10 AT1G01540 0.334   3.87e-29 0.000156           18         21
# ℹ 4,631 more rows
```

```
# purrr::map(tbl_for_filter, ~ table(is.na(.x)))
# tbl_for_filter %>% dplyr::filter(is.na(morf_Phase))
```

For Fig. 7a

```
# (mORF TE) < 0.05 & (uORF TE) < 0.05 & (diff of mORF and uORF) < 0.05
# 255 genes
AGI_SIGUORF_SIGMORF_SIGUMORF <-
  tbl_for_filter %>%
  dplyr::filter(padj_uorf < 0.05) %>%
  dplyr::filter(padj_morf < 0.05) %>%
  dplyr::filter(padj < 0.05) %>%
  dplyr::pull(AGI)
str(AGI_SIGUORF_SIGMORF_SIGUMORF)
```

```
 chr [1:255] "AT1G01060" "AT1G03930" "AT1G06040" "AT1G07030" "AT1G07520" ...
```

For Fig. S6a

```
# (uORF TE) >= 0.05
# 3029 genes
AGI_NOSIGUORF <-
  tbl_for_filter %>%
  dplyr::filter(padj_uorf >= 0.05) %>%
  dplyr::pull(AGI)
str(AGI_NOSIGUORF)
```

```
 chr [1:3029] "AT1G01210" "AT1G01260" "AT1G01370" "AT1G01490" "AT1G01630" ...
```

For Fig. S6b

```
# (mORF TE) < 0.05 & (uORF TE) < 0.05 & (diff of mORF and uORF) >= 0.05
# 998 genes
AGI_SIGUORF_SIGMORF_NOSIGUMORF <-
  tbl_for_filter %>%
  dplyr::filter(padj_uorf < 0.05) %>%
  dplyr::filter(padj_morf < 0.05) %>%
  dplyr::filter(padj >= 0.05) %>%
  dplyr::pull(AGI)
str(AGI_SIGUORF_SIGMORF_NOSIGUMORF)
```

```
 chr [1:998] "AT1G01230" "AT1G01240" "AT1G01430" "AT1G01440" "AT1G01540" ...
```

For Fig. S6c

```r
# (mORF TE) >= 0.05 & (uORF TE) < 0.05
# 359 genes
AGI_SIGUORF_NOSIGMORF <-
  tbl_for_filter %>%
  dplyr::filter(padj_uorf < 0.05) %>%
  dplyr::filter(padj_morf >= 0.05) %>%
  dplyr::pull(AGI)
str(AGI_SIGUORF_NOSIGMORF)
```

```
 chr [1:359] "AT1G04430" "AT1G04440" "AT1G07630" "AT1G09800" "AT1G11200" ...
```

Plotting

```r
f <- function(AGI_GOI, ...) {
  inf <- fs::path(wd, "analysis", "list_summary", "summary_all.csv")
  tbl_plot <-
    readr::read_csv(inf, show_col_types = FALSE) %>%
    dplyr::filter(AGI %in% AGI_GOI) %>%
    dplyr::arrange(rna_Phase, morf_Phase, uorf_Phase) %>%
    dplyr::mutate(AGI = forcats::fct_inorder(AGI)) %>%
    dplyr::mutate(is_same_peak = morf_Phase == uorf_Phase) %>%
    dplyr::select(AGI, rna_Phase, morf_Phase, uorf_Phase, is_same_peak,
                  dplyr::matches("rna_coef_z"),
                  dplyr::matches("^te_morf_coef_z"),
                  dplyr::starts_with("te_uorf_coef_z")) %>%
    convert_to_tbl_plot(AGI) %>%
    scale_tbl_plot(scaling_func = centering_then_z_score) %>%
    sort_tbl(...)
  gp_hm <- plot_heatmap(tbl_plot)
  gt <- ggplotGrob(gp_hm)
  gt$heights[8] <- unit(length(AGI_GOI)*.1, units = "mm")
  gt$widths[c(5, 7, 9)] <- rep(unit(30, units = "mm"), 3)
  gt
}

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

Figure 7a

```r
gt <- f(AGI_GOI = AGI_SIGUORF_SIGMORF_SIGUMORF, rna_Phase, morf_Phase, uorf_Phase)
```

```
Warning: Vectorized input to `element_text()` is not officially supported.
ℹ Results may be unexpected or may change in future versions of ggplot2.
```

```r
label <- "fig7a"
ggsave(path_out_fig7a(glue::glue("{label}.png")), gt,
       width = calc_mm(gt$widths), height = calc_mm(gt$heights), units = "mm")
ggsave(path_out_fig7a(glue::glue("{label}.svg")), gt,
       width = calc_mm(gt$widths), height = calc_mm(gt$heights), units = "mm")
gt_trim <- gt[8, 5:9]
ggsave(path_out_fig7a(glue::glue("{label}_trim.png")), gt_trim,
       width = calc_mm(gt_trim$widths), height = calc_mm(gt_trim$heights), units = "mm")
```

```r
knitr::include_graphics(path_out_fig7a("fig7a.png"))
```

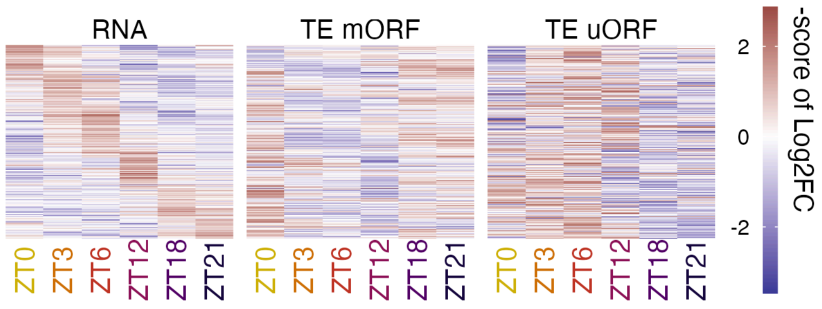
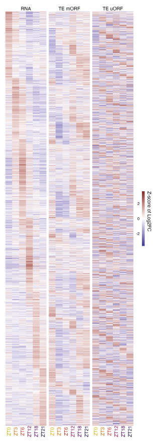


Figure S6

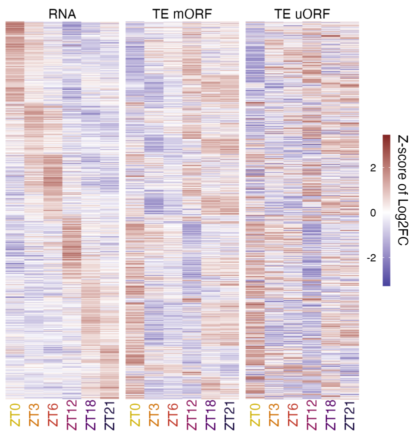
```r
li_AGIs <- list(
  "figS06a" = AGI_NOSIGUORF,
  "figS06b" = AGI_SIGUORF_SIGMORF_NOSIGUMORF,
  "figS06c" = AGI_SIGUORF_NOSIGMORF
)

for(i in seq_along(li_AGIs)) {
  label <- names(li_AGIs)[i]
  gt <- f(li_AGIs[[i]], rna_Phase, morf_Phase, uorf_Phase)
  ggsave(path_out_figS6(glue::glue("{label}.png")), gt,
         width = calc_mm(gt$widths), height = calc_mm(gt$heights), units = "mm")
  ggsave(path_out_figS6(glue::glue("{label}.svg")), gt,
         width = calc_mm(gt$widths), height = calc_mm(gt$heights), units = "mm")
  gt_trim <- gt[8, 5:9]
  ggsave(path_out_figS6(glue::glue("{label}_trim.png")), gt_trim,
         width = calc_mm(gt_trim$widths), height = calc_mm(gt_trim$heights), units = "mm")
}
```

```
Warning: Vectorized input to `element_text()` is not officially supported.
i Results may be unexpected or may change in future versions of ggplot2.
Vectorized input to `element_text()` is not officially supported.
i Results may be unexpected or may change in future versions of ggplot2.
Vectorized input to `element_text()` is not officially supported.
i Results may be unexpected or may change in future versions of ggplot2.
```

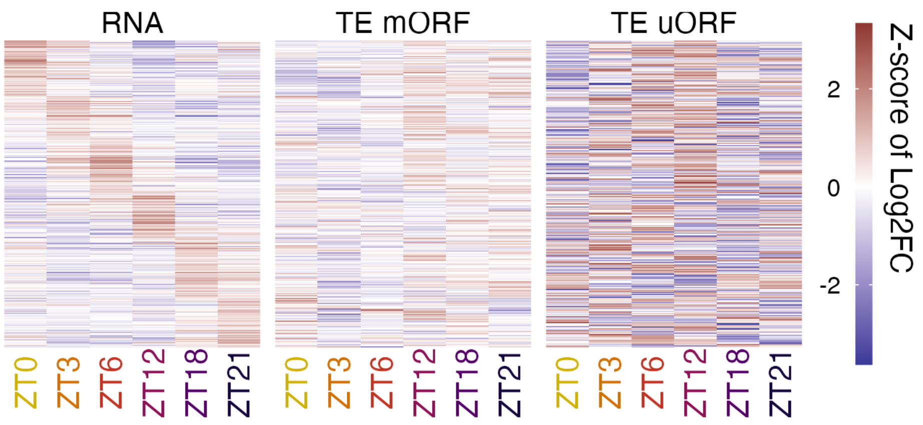
figS06a.png



figS06b.png



figS06c.png



Sessioninfo

```
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] tidyselect_1.2.0    xfun_0.40           purrr_1.0.1
 [4] colorspace_2.0-3    vctrs_0.6.1         generics_0.1.3
 [7] htmltools_0.5.3     viridisLite_0.4.1   yaml_2.3.6
[10] utf8_1.2.2          rlang_1.1.0         pillar_1.9.0
[13] glue_1.6.2          withr_2.5.0         bit64_4.0.5
[16] lifecycle_1.0.3     stringr_1.5.0       munsell_0.5.0
[19] gtable_0.3.1        ragg_1.2.5          evaluate_0.20
[22] labeling_0.4.2      knitr_1.42          forcats_1.0.0
[25] tzdb_0.3.0          fastmap_1.1.0       parallel_4.2.1
```

```
[28] fansi_1.0.3         Rcpp_1.0.11         readr_2.1.4
[31] renv_1.0.3          scales_1.2.1        BiocManager_1.30.18
[34] magick_2.7.3        vroom_1.6.0         jsonlite_1.8.4
[37] systemfonts_1.0.4   farver_2.1.1        fs_1.5.2
[40] bit_4.0.5           textshaping_0.3.6   gridExtra_2.3
[43] png_0.1-7           hms_1.1.3           digest_0.6.31
[46] stringi_1.7.12      dplyr_1.1.1         grid_4.2.1
[49] rprojroot_2.0.3     here_1.0.1          cli_3.6.0
[52] tools_4.2.1         tibble_3.2.1        crayon_1.5.2
[55] tidyr_1.3.0         pkgconfig_2.0.3     svglite_2.1.0
[58] rmarkdown_2.24      rstudioapi_0.14     viridis_0.6.2
[61] R6_2.5.1            prismatic_1.1.1     compiler_4.2.1
```