Lower panels in Figure 7B, Figure 8B, and Figure S8

Toshihiro Arae

General directory setting

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

Loading packages

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

Load common R scripts

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

Load script

```
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))
}
```

```r
source(fs::path(wd, "script_r", "MISC_PLOT_COV.R"))
```

```
Loading required package: GenomicRanges
```

```
Loading required package: stats4
```

```
Loading required package: BiocGenerics
```

```

Attaching package: 'BiocGenerics'
```

```
The following objects are masked from 'package:stats':

    IQR, mad, sd, var, xtabs
```

```
The following objects are masked from 'package:base':

    anyDuplicated, append, as.data.frame, basename, cbind, colnames,
    dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,
    grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,
    order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
    rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,
    union, unique, unsplit, which.max, which.min
```

```
Loading required package: S4Vectors
```

```

Attaching package: 'S4Vectors'
```

```
The following objects are masked from 'package:base':

    expand.grid, I, unname
```

```
Loading required package: IRanges
```

```
Loading required package: GenomeInfoDb
```

```

Attaching package: 'GenomicRanges'
```

```
The following object is masked from 'package:magrittr':

    subtract
```

```
Loading required package: AnnotationDbi
```

```
Loading required package: Biobase
```

```
Welcome to Bioconductor

    Vignettes contain introductory material; view with
    'browseVignettes()'. To cite Bioconductor, see
    'citation("Biobase")', and for packages 'citation("pkgname")'.
```

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

```


library(magrittr)
library(rtracklayer)
library(GenomicRanges)
library(GenomicFeatures)

#' Extract GRanges object of gene(s) of interest
#' @param txdb TxDb-object
#' @param goi character vector of gene(s) of interest or NULL
#' @returns A GRanges
#' @export
#' @examples
#' AGI_GOI <- "AT1G01010"
#' txdb_arabi <- TxDb.Athaliana.BioMart.plantsmart51::TxDb.Athaliana.BioMart.plantsmart51
#' extract_gr_gene(txdb_arabi)
#' extract_gr_gene(txdb_arabi, AGI_GOI)
#'
extract_gr_gene <- function(txdb, goi = NULL) {
  if(is.null(goi)) {
    FILTER <- NULL
  } else {
    FILTER <- list("gene_id" = goi)
  }
  GenomicFeatures::genes(txdb, filter = FILTER)
}

#' Extract GRanges object of transcript(s) of gene(s) interest
#' @param txdb TxDb-object
#' @param goi character vector of gene(s) of interest or NULL
#' @returns A GRanges
#' @export
#' @examples
#' AGI_GOI <- "AT1G01010"
#' txdb_arabi <- TxDb.Athaliana.BioMart.plantsmart51::TxDb.Athaliana.BioMart.plantsmart51
#' extract_gr_transcript(txdb_arabi)
#' extract_gr_transcript(txdb_arabi, AGI_GOI)
#'
extract_gr_transcript <- function(txdb, goi = NULL) {
  if(is.null(goi)) {
    FILTER <- NULL
  } else {
    FILTER <- list("gene_id" = goi)
  }
  GenomicFeatures::transcripts(
    txdb,
    columns = c("gene_id", "tx_id", "tx_name"),
    filter = FILTER
  )
}

#' Extract GRanges object of exon(s) of gene(s) interest
#' @param txdb TxDb-object
```

```
#' @param goi character vector of gene(s) of interest or NULL
#' @returns A GRanges
#' @export
#' @examples
#' AGI_GOI <- "AT1G01010"
#' txdb_arabi <- TxDb.Athaliana.BioMart.plantsmart51::TxDb.Athaliana.BioMart.plantsmart51
#' extract_gr_exon(txdb_arabi)
#' extract_gr_exon(txdb_arabi, AGI_GOI)
#'
extract_gr_exon <- function(txdb, goi = NULL) {
  if(is.null(goi)) {
    FILTER <- NULL
  } else {
    FILTER <- list("gene_id" = goi)
  }
  GenomicFeatures::exons(
    txdb,
    columns = c("gene_id", "tx_id", "tx_name", "exon_id", "exon_name", "exon_rank"),
    filter = FILTER
  )
}

#' Extract GRanges object of CDS(s) of gene(s) interest
#' @param txdb TxDb-object
#' @param goi character vector of gene(s) of interest or NULL
#' @returns A GRanges
#' @export
#' @examples
#' AGI_GOI <- "AT1G01010"
#' txdb_arabi <- TxDb.Athaliana.BioMart.plantsmart51::TxDb.Athaliana.BioMart.plantsmart51
#' extract_gr_cds(txdb_arabi)
#' extract_gr_cds(txdb_arabi, AGI_GOI)
#'
extract_gr_cds <- function(txdb, goi = NULL) {
  if(is.null(goi)) {
    FILTER <- NULL
  } else {
    FILTER <- list("gene_id" = goi)
  }
  GenomicFeatures::cds(
    txdb,
    columns = c("gene_id", "tx_id", "tx_name", "exon_id", "exon_name", "exon_rank",
                "cds_id", "cds_name", "cds_phase"),
    filter = FILTER
  )
}

#' Extract GRanges object of 5'UTR(s) of gene(s) interest
#' @param txdb TxDb-object
#' @param goi character vector of gene(s) of interest or NULL
#' @returns A GRanges
#' @export
#' @examples
#' AGI_GOI <- "AT1G01010"
#' txdb_arabi <- TxDb.Athaliana.BioMart.plantsmart51::TxDb.Athaliana.BioMart.plantsmart51
#' extract_gr_futr(txdb_arabi)
#' extract_gr_futr(txdb_arabi, AGI_GOI)
#'
extract_gr_futr <- function(txdb, goi = NULL) {
  gr_futr <-
    GenomicFeatures::fiveUTRsByTranscript(txdb, use.names = TRUE) %>%
    unlist() %>%
```

```
    {plyranges::mutate(., tx_name = names(.))}
  if(is.null(goi)) {
    return(gr_futr)
  } else {
    plyranges::filter(gr_futr, grepl(paste(goi, collapse = "|"), tx_name))
  }
}

#' Extract GRanges object of 3'UTR(s) of gene(s) interest
#' @param txdb TxDb-object
#' @param goi character vector of gene(s) of interest or NULL
#' @returns A GRanges
#' @export
#' @examples
#' AGI_GOI <- "AT1G01010"
#' txdb_arabi <- TxDb.Athaliana.BioMart.plantsmart51::TxDb.Athaliana.BioMart.plantsmart51
#' extract_gr_tutr(txdb_arabi)
#' extract_gr_tutr(txdb_arabi, AGI_GOI)
#'
extract_gr_tutr <- function(txdb, goi = NULL) {
  gr_tutr <-
    GenomicFeatures::threeUTRsByTranscript(txdb, use.names = TRUE) %>%
    unlist() %>%
    {plyranges::mutate(., tx_name = names(.))}
  if(is.null(goi)) {
    return(gr_tutr)
  } else {
    plyranges::filter(gr_tutr, grepl(paste(goi, collapse = "|"), tx_name))
  }
}

#' Return an extended GRanges of the specified gene region
#' @param txdb TxDb-object
#' @param AGI character vector of gene of interest
#' @returns A GRanges
#' @export
#' @examples
#' AGI_GOI <- "AT1G01010"
#' txdb_arabi <- TxDb.Athaliana.BioMart.plantsmart51::TxDb.Athaliana.BioMart.plantsmart51
#' make_gr_goi_all(txdb_arabi, AGI_GOI)
#'
make_gr_goi_all <- function(txdb, AGI) {
  if(length(AGI) != 1L) stop("the length of AGI must be 1.")

  gr_gene_goi <- extract_gr_gene(txdb, AGI)
  gr_goi_all <- gr_gene_goi %>% plyranges::select(!gene_id)
  strand(gr_goi_all) <- "*"
  gr_goi_all %>% {plyranges::stretch(., extend = width(.) * 0.02)}
}

#' Extract GRanges objects overlapped specified region and return as a list
#' @param txdb TxDb-object
#' @param goi character vector of gene(s) of interest or NULL
#' @returns a list of GRanges
#' @export
#' @examples
#' txdb_arabi <- TxDb.Athaliana.BioMart.plantsmart51::TxDb.Athaliana.BioMart.plantsmart51
#' gr_all <- plyranges::as_granges(data.frame(seqnames = "1", start = 1, end = 10000))
#' extract_li_gr(txdb_arabi, gr_all)
#'
extract_li_gr <- function(txdb, gr) {
  purrr::map(
```

```
    list(
      "transcript" = extract_gr_transcript,
      "exon" = extract_gr_exon,
      "cds" = extract_gr_cds,
      "futr" = extract_gr_futr,
      "tutr" = extract_gr_tutr
    ),
    ~ .x(txdb)
  ) %>%
    purrr::map(plyranges::filter_by_overlaps, y = gr)
}

#' Transform a list of GRanges to a list of tibble
#' @param li_gr an output from `etract_li_gr()`
#' @export
#'
li_gr2li_tbl <- function(li_gr) {
  tbl_plot_start <-
    tibble::as_tibble(li_gr$transcript) %>%
    dplyr::mutate(tss = ifelse(strand == "+", start, end))
  tbl_plot_tail <-
    tibble::as_tibble(li_gr$transcript) %>%
    dplyr::mutate(tts = ifelse(strand == "+", end, start),
                  start = ifelse(strand == "+", tts-1, tts+1))
  li <- purrr::map(li_gr, tibble::as_tibble)
  li[["start"]] <- tbl_plot_start
  li[["tail"]] <- tbl_plot_tail
  purrr::map(li, tidyr::unnest, cols = tidyselect::where(is.list))
}



#' Extract a transcript data and add transcript coordinates
#' @param li_tbl_plot an output from `li_gr2li_tbl()`
#' @param tx_name_goi a character vector to specify the transcript
#' @export
#'
make_li_tbl_plot_tx <- function(li_tbl_plot, tx_name_goi) {
  li_tbl_plot_tx <-
    li_tbl_plot %>%
    purrr::map(dplyr::filter, tx_name == tx_name_goi) %>%
    purrr::modify_if(
      .p = ~ any(names(.x) == "exon_rank"),
      .f = ~ dplyr::arrange(.x, exon_rank)
    ) %>%
    purrr::map(dplyr::mutate, tx_end = cumsum(width)) %>%
    purrr::map(dplyr::mutate, tx_start = tx_end - width)

  len_futr <- sum(li_tbl_plot_tx$futr$width)
  li_tbl_plot_tx$cds$tx_start <- li_tbl_plot_tx$cds$tx_start + len_futr
  li_tbl_plot_tx$cds$tx_end <- li_tbl_plot_tx$cds$tx_end + len_futr
  len_futr_cds <- max(li_tbl_plot_tx$cds$tx_end)
  li_tbl_plot_tx$tutr$tx_start <- li_tbl_plot_tx$tutr$tx_start + len_futr_cds
  li_tbl_plot_tx$tutr$tx_end <- li_tbl_plot_tx$tutr$tx_end + len_futr_cds

  li_tbl_plot_tx %>%
    purrr::map(dplyr::select, tx_start, tx_end, dplyr::everything())
}

#' Show the transcript feature length information
#' @param li_tbl_plot an output from `li_gr2li_tbl()`
#' @export
```

```
#'
show_transcript_info <- function(li_tbl_plot) {
  tx_names <- li_tbl_plot$transcript$tx_name
  f <- function(tbl, name) sum(dplyr::filter(tbl, tx_name %in% name)$width)
  len_exons <- purrr::map_int(tx_names, ~ f(li_tbl_plot$exon, .x))
  len_futrs <- purrr::map_int(tx_names, ~ f(li_tbl_plot$futr, .x))
  len_cdss <- purrr::map_int(tx_names, ~ f(li_tbl_plot$cds, .x))
  len_tutrs <- purrr::map_int(tx_names, ~ f(li_tbl_plot$tutr, .x))

  purrr::pmap_dfr(
    .l = list(tx_names, len_exons, len_futrs, len_cdss, len_tutrs),
    .f = ~ {
      tibble::tribble(
        ~ tx_name, ~ total, ~ "5'UTR", ~ CDS, ~ "3'UTR",
        ..1, ..2, ..3, ..4, ..5)
    }
  ) %>%
    dplyr::arrange(tx_name) %>%
    knitr::kable()
}

#' Plot gene structure with sequence
#' @param li_tbl_plot_tx an output from `make_li_tbl_plot_tx()`
#' @param zoom_x_min a numeric, default: NA
#' @param zoom_x_max a numeric, default: NA
#' @export
#'
plot_structure <- function(li_tbl_plot_tx, zoom_x_min = NA, zoom_x_max = NA) {
  TX_X_MIN <- li_tbl_plot_tx$transcript$start
  TX_X_MAX <- li_tbl_plot_tx$transcript$end
  shimashima <-
    annotate("rect",
             xmin = seq(TX_X_MIN, TX_X_MAX, by = 2) -.5,
             xmax = seq(TX_X_MIN, TX_X_MAX, by = 2) +.5,
             ymin = .5, ymax = 2.5,
             fill = "grey", alpha = .5)
  gp_check_transcript <-
    ggplot(mapping = aes(xmin = start, xmax = end, y = tx_name)) +
    annotate("rect", xmin = zoom_x_min, xmax = zoom_x_max,
             ymin = -Inf, ymax = Inf, fill = "grey") +
    geom_linerange(data = li_tbl_plot_tx$transcript, aes(group = tx_id)) +
    geom_linerange(data = li_tbl_plot_tx$cds, aes(group = exon_id), linewidth = 2) +
    geom_linerange(data = li_tbl_plot_tx$futr, aes(group = exon_id), linewidth = 1) +
    geom_linerange(data = li_tbl_plot_tx$tutr, aes(group = exon_id), linewidth = 1) +
    geom_point(data = li_tbl_plot_tx$start, aes(x = tss, group = tx_id)) +
    geom_segment(data = li_tbl_plot_tx$tail,
                 aes(x = start, xend = tts, yend = tx_name, group = tx_id),
                 arrow = arrow(type = "closed", length = unit(5, "points")))
    gp_check_transcript_mini <-
    ggplot(mapping = aes(xmin = start, xmax = end, y = 1)) +
    shimashima +
    geom_linerange(data = li_tbl_plot_tx$transcript, aes(group = tx_id)) +
   geom_linerange(data = li_tbl_plot_tx$cds, aes(group = exon_id), linewidth = 2, lineend = "butt")
+
   geom_linerange(data = li_tbl_plot_tx$futr, aes(group = exon_id), linewidth = 1, lineend = "butt")
+
   geom_linerange(data = li_tbl_plot_tx$tutr, aes(group = exon_id), linewidth = 1, lineend = "butt")
  gp_check_seq <-
    bsg_tair[[as.character(li_tbl_plot_tx$transcript$seqnames[1])]] %>%
    Biostrings::subseq(start = TX_X_MIN, end = TX_X_MAX) %>%
    as.character() %>%
    stringr::str_extract_all(".") %>%
```

```
    {tibble::tibble(coord = TX_X_MIN:TX_X_MAX, nucleotide = .[[1]])} %>%
    ggplot(aes(coord, 1, label = nucleotide, color = after_stat(label))) +
    shimashima +
    geom_text() +
    geom_text(aes(y = 2, label = bstringr::dstr_complement(nucleotide)))

  patchwork::wrap_plots(
    gp_check_transcript,
    gp_check_transcript_mini +
      coord_cartesian(xlim = c(zoom_x_min, zoom_x_max)),
    gp_check_seq +
      coord_cartesian(xlim = c(zoom_x_min, zoom_x_max), ylim = c(.5, 2.5)),
    ncol = 1, heights = c(3, 3, 3)) &
    theme(panel.background = element_blank())
}

plot_struct <- function(tbl_plot, linewidth = 1, color = "#191970", fill = "#191970") {
  if(nrow(tbl_plot) == 0) return(geom_linerange())
  geom_rect(
    data = tbl_plot,
    aes(
      xmin = tx_start - 0.5, xmax = tx_end + 0.5,
      ymin = ymin, ymax = ymax
    ),
    linewidth = linewidth, color = color, fill = fill)
}

plot_uorf <- function(tbl_plot, linewidth = 1, color = "#191970", fill = "#191970") {
  if(nrow(tbl_plot) == 0) return(geom_linerange())
  geom_rect(
    data = tbl_plot,
    aes(
      xmin = tx_start - 0.5, xmax = tx_end + 0.5,
      ymin = ymin, ymax = ymax,
      alpha = is_active
    ),
    linewidth = linewidth, color = color, fill = fill)
}

plot_uorf2 <- function(tbl_plot, linewidth = 1, color = "#191970", fill = "#191970") {
  if(nrow(tbl_plot) == 0) return(geom_linerange())
  geom_rect(
    data = tbl_plot,
    aes(
      xmin = tx_start - 0.5, xmax = tx_end + 0.5,
      ymin = ymin, ymax = ymax,
      alpha = is_active,
      linewidth = is_overlap
    ),
    # linewidth = linewidth, color = color, fill = fill)
    color = color, fill = fill, linetype = "dashed")
}

#' Plot gene structure with sequence
#' @param li_tbl_plot_tx an output from `make_li_tbl_plot_tx()`
#' @param tbl_uorf an output from `make_tbl_uorf()`
#' @param zoom_x_min a numeric, default: NA
#' @param zoom_x_max a numeric, default: NA
#' @export
#'
plot_structure_w_uorf <- function(li_tbl_plot_tx, tbl_uorf, zoom_x_min = NA, zoom_x_max = NA) {
  TX_X_MIN <- min(li_tbl_plot_tx$exon$tx_start) + 1
```

```
  TX_X_MAX <- max(li_tbl_plot_tx$exon$tx_end)
  zoom_x_min <- ifelse(is.na(zoom_x_min), TX_X_MIN, zoom_x_min)
  zoom_x_max <- ifelse(is.na(zoom_x_max), TX_X_MAX, zoom_x_max)
  shimashima <-
    annotate("rect",
             xmin = seq(TX_X_MIN, TX_X_MAX, by = 2) -.5,
             xmax = seq(TX_X_MIN, TX_X_MAX, by = 2) +.5,
             ymin = -Inf, ymax = Inf,
             fill = "grey", alpha = .5)

  tbl_uorf_plot <-
    tbl_uorf %>%
    dplyr::with_groups(c(ID, is_active), dplyr::summarise,
                       tx_start = min(tx_start)+1, tx_end = max(tx_end)) %>%
    dplyr::mutate(frame = tx_start %% 3)

    summarise <- function(tbl) dplyr::summarise(tbl, tx_start = min(tx_start) + 1, tx_end =
max(tx_end), .by = tx_name)
  mutate_ymin_ymax <- function(tbl, y, offset) dplyr::mutate(summarise(tbl), ymin = y - offset, ymax
= y + offset)
  mutate_y_uorf <- function(tbl, offset) dplyr::mutate(tbl, ymin = frame - offset + 3, ymax = frame
+ offset + 3)

  gp_check_transcript <-
    ggplot() +
    annotate("rect", xmin = zoom_x_min, xmax = zoom_x_max,
             ymin = -Inf, ymax = Inf, fill = "grey") +
    plot_struct(mutate_ymin_ymax(li_tbl_plot_tx$cds, 1, 1), fill = NA) +
    plot_struct(mutate_ymin_ymax(li_tbl_plot_tx$futr, 1, .3), color = NA) +
    plot_struct(mutate_ymin_ymax(li_tbl_plot_tx$tutr, 1, .3), color = NA) +
    plot_uorf(mutate_y_uorf(tbl_uorf_plot, .45), color = NA) +
    scale_alpha_manual(values = c("TRUE" = .8, "FALSE" = .4))

  gp_check_transcript_mini <-
    ggplot() +
    shimashima +
    plot_struct(mutate_ymin_ymax(li_tbl_plot_tx$cds, 1, 1), fill = NA) +
    plot_struct(mutate_ymin_ymax(li_tbl_plot_tx$futr, 1, .3), color = NA) +
    plot_struct(mutate_ymin_ymax(li_tbl_plot_tx$tutr, 1, .3), color = NA) +
    plot_uorf(mutate_y_uorf(tbl_uorf_plot, .45), color = NA) +
    scale_alpha_manual(values = c("TRUE" = .8, "FALSE" = .4))

  seqname <- as.character(li_tbl_plot_tx$transcript$seqnames[1])
  exon_seq <-
    bsg_tair[[seqname]] %>%
    as.character() %>%
    bstringr::as_dstr(n = seqname) %>%
    bstringr::bstr_sub_all(from = li_tbl_plot_tx$exon$start,
                           to = li_tbl_plot_tx$exon$end) %>%
    .[[1]]

  if(li_tbl_plot_tx$transcript$strand == "-")
    exon_seq <- bstringr::dstr_rev_comp(exon_seq)

  gp_check_seq <-
    exon_seq %>%
    paste(collapse = "") %>%
    stringr::str_extract_all(".") %>%
    {tibble::tibble(coord = TX_X_MIN:TX_X_MAX, nucleotide = .[[1]])} %>%
    ggplot(aes(coord, 1, label = nucleotide, color = after_stat(label))) +
    shimashima +
    geom_text() +
```

```
    theme(legend.position = "none")

  patchwork::wrap_plots(
    gp_check_transcript,
    gp_check_transcript_mini +
      coord_cartesian(xlim = c(zoom_x_min, zoom_x_max)),
    gp_check_seq +
      coord_cartesian(xlim = c(zoom_x_min, zoom_x_max), ylim = c(.5, 1.5)),
    ncol = 1, heights = c(3, 3, 3)) &
    theme(panel.background = element_blank())
}


theme_gene_structure <- function(gr_all) {
  list(
    labs(x = as.character(seqnames(gr_all)), y = "transcripts"),
    theme_minimal(),
    theme(
      line = element_blank(),
      axis.ticks.x = element_line(),
      plot.margin = unit(c(0,0,0,0), "pt")
    ),
    scale_x_continuous(
      limits = c(start(gr_all), end(gr_all)),
      expand = expansion(mult = c(0, 0)),
      breaks = scales::breaks_pretty(n = 6),
      labels = scales::label_number(
        big.mark = ",",
        suffix = "bp",
        scale_cut = c(0, " k" = 1000)
      )
    )
  )
}


#' @param bigwig_files path to the bigwig files
#' @param gr_region
read_bigwig_region_directed <- function(bigwig_files, gr_region, gr_feature) {
  strand <- ifelse(as.character(gr_feature@strand) == "+", "plus", "minus")
  bw_selection <-
    gr_goi_all %>%
    rtracklayer::BigWigSelection()

  path_bigwig_files %>%
    {.[stringr::str_detect(., strand)]} %>%
    purrr::map(rtracklayer::import, selection = bw_selection)
}

#' Read coverage on the specified region from bigwig files
#' @param bigwig_files path to the bigwig files
#' @param gr_region
read_bigwig_region <- function(bigwig_files, gr_region) {
  bw_selection <-
    gr_region %>%
    rtracklayer::BigWigSelection()

  bigwig_files %>%
    purrr::map(rtracklayer::import, selection = bw_selection)
}

#' Convert GenomicRanges to tibble
```

```
#' @param gr GRanges object
gr2tbl <- function(gr) {
  if(!inherits(gr, "GRanges")) stop("gr is not a GRanges object.")
  if(length(gr) == 0) return(NULL)
  tibble::as_tibble(gr) %>%
    dplyr::rowwise() %>%
    dplyr::mutate(coord = list(start:end)) %>%
    tidyr::unnest(cols = c(coord)) %>%
    dplyr::select(seqnames, coord, strand, score)
}

# "https://raw.githubusercontent.com/adibender/pammtools/master/R/ggplot-extensions.R" %>%
#   readr::read_lines() %>%
#   readr::write_lines(fs::path(wd, "script_r", "geom_stepribbon.R"))
source(fs::path(wd, "script_r", "geom_stepribbon.R"))

mutate_rel_to_maximum_roi <- function(tbl, start, end) {
  maximum <-
    tbl %>%
    dplyr::filter(start <= coord, coord <= end) %>%
    dplyr::pull(score) %>%
    max()
  tbl %>%
    dplyr::mutate(score2 = score / maximum)
}

theme_coverage <- function() {
  list(
    theme_minimal(),
    theme(
      axis.ticks.x = element_blank(),
      axis.text.x = element_blank(),
      axis.text.y = element_blank(),
      line = element_blank(),
      legend.position = "",
      panel.background = element_blank(),
      plot.margin = unit(c(0, 0, -5, 0), "pt")
    ),
    scale_x_continuous(
      limits = c(start(gr_goi_all), end(gr_goi_all)),
      expand = expansion(mult = c(0, 0))
    ),
    scale_y_continuous(limits = c(0, 1), expand = expansion(mult = c(.1, .1))),
    scale_fill_manual(values = LABEL_PALETTE)
  )
}
```

Directory setting

```
path_out_fig7Blp <- function(...) fs::path(wd, "analysis", "fig", "fig07Blp", ...)
path_out_fig8Blp <- function(...) fs::path(wd, "analysis", "fig", "fig08Blp", ...)
path_out_figS8lp <- function(...) fs::path(wd, "analysis", "fig", "figS08lp", ...)
fs::dir_create(path_out_fig7Blp())
fs::dir_create(path_out_fig8Blp())
fs::dir_create(path_out_figS8lp())
```

Load input data

RPF coverage

P-site の bigwiggle ファイルを読み込む。

```
bw_files <-
  fs::path(wd, "data_preproc", "bam_coverage", "bam_psite") %>%
  fs::dir_ls(regexp = ".bw$")
```

Genome annotation

Araport11 のアノテーション（gff ファイル）を読み込む。

```
txdb <-
  fs::path(wd, "misc", "gff_gtf",
           "Araport11_GFF3_genes_transposons.201606_mod.gff") %>%
  GenomicFeatures::makeTxDbFromGFF()
```

```
Import genomic features from the file as a GRanges object ... OK
Prepare the 'metadata' data frame ... OK
Make the TxDb object ...
```

```
Warning in .get_cds_IDX(mcols0$type, mcols0$phase): The "phase" metadata column contains non-NA
values for features of type
  exon. This information was ignored.
```

```
OK
```

uORF annotation

```
tbl_uorf_all <-
  dplyr::left_join(
    fs::path(wd, "data_modified", "gff_gtf", "araport11_uorf_ribotricer.gff3") %>%
      rtracklayer::import.gff3() %>%
      tibble::as_tibble() %>%
      tidyr::unnest_longer(ID) %>%
      purrr::modify_if(is.character, stringr::str_remove_all, "\""),
    fs::path(wd, "data_modified", "gff_gtf", "araport11_active_uorf_ribotricer.gff3") %>%
      rtracklayer::import.gff3() %>%
      tibble::as_tibble() %>%
      tidyr::unnest_longer(ID) %>%
      purrr::modify_if(is.character, stringr::str_remove_all, "\"") %>%
      dplyr::select(ID) %>%
      dplyr::distinct() %>%
      dplyr::mutate(is_active = TRUE),
    by = "ID"
  ) %>%
  dplyr::mutate(is_active = ifelse(is.na(is_active), FALSE, TRUE))
```

Define misc funcitons

```
#' Extract uORF data on a specific transcript and add transcript coordinates
#' @param li_tbl_plot_tx an output from `make_li_tbl_plot_tx()`
#' @param tx_name_goi a character vector to specify the transcript
#' @export
make_tbl_uorf <- function(li_tbl_plot_tx, tx_name_goi) {
  tbl_empty <- tibble::tibble(
    tx_start = NA, tx_end = NA, ID = NA_character_, width = NA, exon_id = NA, is_active = NA
  )
  if(!any(grepl(paste0(tx_name_goi, "_"), tbl_uorf_all$ID))) {
    return(tbl_empty)
  }
```

```
  li_tbl_uorf <-
    tbl_uorf_all %>%
    dplyr::filter(grepl(paste0(tx_name_goi, "_"), ID)) %>%
    dplyr::with_groups(ID, dplyr::mutate, exon_id = paste0(ID, "_", dplyr::n()))

  if(li_tbl_uorf$strand[1] == "+") {
    li_tbl_uorf <- dplyr::arrange(li_tbl_uorf, start)
  } else {
    li_tbl_uorf <- dplyr::arrange(li_tbl_uorf, desc(start))
  }

  li_tbl_uorf <-
    li_tbl_uorf %>%
    split(.$ID) %>%
    purrr::map(dplyr::mutate, tx_end = cumsum(width)) %>%
    purrr::map(dplyr::mutate, tx_start = tx_end - width) %>%
    purrr::map(dplyr::select, tx_start, tx_end, dplyr::everything())

  if(nrow(li_tbl_plot_tx$futr) == 0) {
    return(tbl_empty)
  }

  if(li_tbl_plot_tx$transcript$strand == "-") {
    exon_start <- li_tbl_plot_tx$exon$start
    exon_end <- li_tbl_plot_tx$exon$end
    exon_tx_start <- li_tbl_plot_tx$exon$tx_start
    exon_tx_end <- li_tbl_plot_tx$exon$tx_end
    exon_width <- li_tbl_plot_tx$exon$width
    for(i in seq_along(li_tbl_uorf)) {
      temp_tbl_uorf <- li_tbl_uorf[[i]]
      temp_pos <-
        which(exon_start <= temp_tbl_uorf$end[1] & temp_tbl_uorf$end[1] <= exon_end)
      uorf_tx_start <- exon_tx_start[temp_pos] + (exon_end[temp_pos] - temp_tbl_uorf$end[1])
      li_tbl_uorf[[i]] <-
        dplyr::mutate(
          temp_tbl_uorf,
          tx_start = uorf_tx_start + tx_start,
          tx_end = uorf_tx_start + cumsum(width)
        )
    }
  } else {
    futr_start <- li_tbl_plot_tx$futr$start
    futr_width <- li_tbl_plot_tx$futr$width
    for(i in seq_along(li_tbl_uorf)) {
      temp_tbl_uorf <- li_tbl_uorf[[i]]
      temp_start <-
        futr_start[temp_tbl_uorf$start[1] >= futr_start] %>%
        {.[length(.)]}
      temp_width <-
        futr_width[temp_tbl_uorf$start[1] >= futr_start] %>%
        {.[0:(length(.) - 1)]} %>%
        sum()

      li_tbl_uorf[[i]] <-
        dplyr::mutate(
          temp_tbl_uorf,
          tx_start = tx_start + temp_width + (temp_tbl_uorf$start[1] - temp_start),
          tx_end = tx_end + temp_width + (temp_tbl_uorf$start[1] - temp_start)
        )
    }
  }
  tbl_uorf <- dplyr::bind_rows(li_tbl_uorf)
```

```
  if(nrow(tbl_uorf) != 0) {
    tbl_uorf <- dplyr::mutate(tbl_uorf, tx_name= ID)
  }
  return(tbl_uorf)
}

make_tbl_y <- function(tbl, ymin = 1, ymax = 1, add_offset = FALSE) {
  tx_num <- length(unique(tbl$tx_name)) + add_offset
  y_vec <- seq(from = ymin, to = ymax, length.out = tx_num)
  if(add_offset) {
    y_vec <- y_vec[-1]
  }
  tbl %>%
    split(.$tx_name) %>%
    purrr::map2(y_vec, ~ dplyr::mutate(.x, y = .y)) %>%
    dplyr::bind_rows()
}

make_tbl_plot_coverage_tx <- function(li_tbl_plot_tx) {
  tbl <-
    read_bigwig_region(
      bw_files,
      plyranges::as_granges(li_tbl_plot_tx$exon)
    ) %>%
    purrr::map(gr2tbl) %>%
    purrr::map_if(
      .p = ~ !is.null(.x),
      .f = ~ .x,
      .else = ~ tibble::tibble(coord = NA, seqnames = NA, strand = NA, score = NA)
    ) %>%
    purrr::imap(~ dplyr::mutate(.x, fname = fs::path_file(.y))) %>%
    dplyr::bind_rows() %>%
    dplyr::mutate(ID = stringr::str_extract(fname, "zt\\d+_[12]")) %>%
    dplyr::mutate(
      sample_info_short =
        stringr::str_extract(ID, "^zt\\d+") %>%
        toupper() %>%
        forcats::fct_relevel(names(COL_PALETTE)),
      rep = stringr::str_sub(ID, -1)
    )

  if(li_tbl_plot_tx$transcript$strand == "+") {
    tbl <- dplyr::arrange(tbl, coord)
  } else {
    tbl <- dplyr::arrange(tbl, desc(coord))
  }
  tbl %>%
    dplyr::with_groups(fname, dplyr::mutate, tx_coord = dplyr::row_number())
}

theme_fontsize <- function() {
  list(
    theme(text = element_text(color = "black", size = 16)
    )
  )
}

theme_strip <- function() {
  list(
    theme(strip.text.y = element_text(angle = 0), strip.background.y = element_blank())
  )
}
```

```
# from https://stackoverflow.com/questions/65045019/ggplot2-both-axis-labels-inside-plot-area
move_axes_inside <- function(p) {
  b <- ggplot_build(p)
  x_breaks <- b$layout$panel_scales_x[[1]]$break_info()
  y_breaks <- b$layout$panel_scales_y[[1]]$break_info()
  x_range <- b$layout$panel_params[[1]]$x.range
  y_range <- b$layout$panel_params[[1]]$y.range
  y_breaks$major <- diff(y_breaks$range)/diff(y_range) * y_breaks$major +
    (y_breaks$range[1] - y_range[1])/diff(y_range)
  x_breaks$major <- diff(x_breaks$range)/diff(x_range) * x_breaks$major +
    (x_breaks$range[1] - x_range[1])/diff(x_range)
  y <- grid::yaxisGrob(at = y_breaks$major, label = y_breaks$labels, main = FALSE)
  x <- grid::xaxisGrob(at = x_breaks$major, label = x_breaks$labels, main = FALSE)
  p +
    annotation_custom(y, xmin = x_range[1], xmax = x_range[1]) +
    theme(axis.text.y = element_blank(),
          axis.ticks = element_blank())
}
```

Function to create a coverage plot

```
plot_version5 <- function(li_tbl_plot_tx, tbl_uorf, tbl_plot_rna_coverage_tx,
                          score_modifier, title_y = "Count") {
  TX_X_MIN <- min(li_tbl_plot_tx$exon$tx_start) + 1
  TX_X_MAX <- max(li_tbl_plot_tx$exon$tx_end)

  tbl_uorf_plot <-
    tbl_uorf %>%
    dplyr::with_groups(c(ID, is_active), dplyr::summarise,
                       tx_start = min(tx_start)+1, tx_end = max(tx_end)) %>%
    dplyr::mutate(frame = tx_start %% 3)

  tbl_plot_rna_coverage_tx <-
    tbl_plot_rna_coverage_tx %>%
    dplyr::with_groups(fname, dplyr::mutate, score = score_modifier(score)) %>%
    dplyr::group_by(seqnames, coord, sample_info_short, tx_coord) %>%
    dplyr::summarise(score = mean(score)) %>%
    dplyr::ungroup() %>%
    dplyr::mutate(score = ifelse(is.nan(score), 0, score))

  ALPHA_UORF_BOTTOM <- c("TRUE" = .8, "FALSE" = .4)
  ALPHA_UORF_MIDDLE <- ALPHA_UORF_BOTTOM / 3

  num_max <-
    tbl_plot_rna_coverage_tx %>%
    {max(.$score)}
   summarise <- function(tbl) dplyr::summarise(tbl, tx_start = min(tx_start) + 1, tx_end =
max(tx_end), .by = tx_name)
  mutate_ymin_ymax <- function(tbl, y, offset) dplyr::mutate(summarise(tbl), ymin = y - offset, ymax
= y + offset)
  mutate_y_uorf <- function(tbl, offset) dplyr::mutate(tbl, ymin = frame - offset + 3, ymax = frame
+ offset + 3)

  gp_tx <-
    ggplot() +
    plot_struct(mutate_ymin_ymax(li_tbl_plot_tx$cds, 1, 1), fill = NA) +
    plot_struct(mutate_ymin_ymax(li_tbl_plot_tx$futr, 1, .3), color = NA) +
    plot_struct(mutate_ymin_ymax(li_tbl_plot_tx$tutr, 1, .3), color = NA) +
    plot_uorf(mutate_y_uorf(tbl_uorf_plot, .45), color = NA) +
    scale_y_continuous(limits = c(0, 7)) +
```

```
  scale_alpha_manual(values = ALPHA_UORF_BOTTOM)

gp_cov <-
  ggplot() +
  geom_stepribbon(
    data = tbl_plot_rna_coverage_tx,
    aes(x = tx_coord, ymin = 0, ymax = score, group = sample_info_short,
        fill = I(alpha(LABEL_PALETTE[as.character(sample_info_short)], 1)))
  ) +
  facet_grid(rows = vars(sample_info_short)) +
  scale_alpha_manual(values = ALPHA_UORF_MIDDLE) +
  scale_x_continuous(
    expand = expansion(mult = 0),
    labels = scales::label_number(big.mark = ",", suffix = "")) +
  scale_y_continuous(
    breaks = c(num_max),
    expand = expansion(mult = c(0, .1)),
    labels = scales::label_number(accuracy = .001)) +
  theme_strip() +
  theme_fontsize() +
  theme(
    panel.background = element_rect(fill = "grey95"),
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    legend.position = "none"
  )

theme_scale_x_tx <- function() {
  default_breaks <- scales::breaks_extended()
  modified_breaks <- function(x) {
    b <- default_breaks(x)
    c(1, b[b != 0])
  }
  list(
    scale_x_continuous(
      expand = expansion(mult = 0),
      labels = scales::label_number(big.mark = ",", suffix = ""),
      breaks = modified_breaks
    )
  )
}

patchwork::wrap_plots(
    move_axes_inside(gp_cov) +
    labs(y = title_y) +
    theme(axis.text.x = element_blank(), title = element_blank(),
          plot.margin = unit(c(0,0,0,0), "npc"),
          axis.ticks.y = element_blank()),
  gp_tx +
    labs(x = TITLE_X, y = "") +
    theme_fontsize() +
    theme_scale_x_tx() +
    theme(
      plot.margin = unit(c(0,0,0,0), "npc"),
      axis.ticks.y = element_blank(),
      axis.text.y = element_blank(),
      panel.background = element_blank(),
      panel.grid.major = element_blank(),
      panel.grid.minor = element_blank(),
      legend.position = "none"
    ),
  heights = c(6, 1)
```

```
  )
}
```

Figure 7B and Figure 8B

```
tbl_goi <-
  tibble::tribble(
    ~AGI,        ~gene_name,
    "AT5G57360", "ZTL",
    "AT5G14060", "AK2",
    "AT3G02020", "AK3",
  )

for(i in seq_along(tbl_goi$AGI)) {
  if(i == 1) {
    path_out_temp <- path_out_fig7Blp
  } else {
    path_out_temp <- path_out_fig8Blp
  }

  temp_AGI_GOI <- tbl_goi$AGI[i]
  temp_name <- tbl_goi$gene_name[i]
  temp_label <- stringr::str_glue("{temp_AGI_GOI}_{temp_name}")

  gr_goi_all <- make_gr_goi_all(txdb, temp_AGI_GOI)
  li_gr <- extract_li_gr(txdb, gr_goi_all)
  li_tbl_plot <- li_gr2li_tbl(li_gr)
  tx <-
    li_tbl_plot$transcript$tx_name %>%
    {.[grep(temp_AGI_GOI, .)]}
  for(AGI_GOI_TX in tx) {
    li_tbl_plot_tx <- make_li_tbl_plot_tx(li_tbl_plot, AGI_GOI_TX)
    tbl_uorf <- make_tbl_uorf(li_tbl_plot_tx, AGI_GOI_TX)
    tbl_plot_rna_coverage_tx <-
      make_tbl_plot_coverage_tx(li_tbl_plot_tx) %>%
      return()

    if(nrow(li_tbl_plot_tx$futr) != 0) {
      TITLE_X <- paste0(AGI_GOI_TX, " mRNA (nt)")
     plot_version5(li_tbl_plot_tx, tbl_uorf, tbl_plot_rna_coverage_tx, score_modifier = function(x)
x / sum(x)) &
        coord_cartesian(xlim = c(1, max(li_tbl_plot_tx$futr$tx_end) + 100))
      path_out_temp(stringr::str_glue("{temp_label}/{AGI_GOI_TX}.png")) %>%
        ggsave(width = 178, height = 200, dpi = 300, units = "mm")
      path_out_temp(stringr::str_glue("{temp_label}/{AGI_GOI_TX}.pdf")) %>%
        ggsave(width = 178, height = 200, units = "mm")
      rm(TITLE_X)
    }
    rm(li_tbl_plot_tx, tbl_uorf, tbl_plot_rna_coverage_tx)
  }
  rm(temp_AGI_GOI, gr_goi_all, li_tbl_plot)
}
```

```
`summarise()` has grouped output by 'seqnames', 'coord', 'sample_info_short'.
You can override using the `.groups` argument.
`summarise()` has grouped output by 'seqnames', 'coord', 'sample_info_short'.
You can override using the `.groups` argument.
```

```
Warning: Removed 1 rows containing missing values (`geom_rect()`).
Removed 1 rows containing missing values (`geom_rect()`).
```

```
`summarise()` has grouped output by 'seqnames', 'coord', 'sample_info_short'.
You can override using the `.groups` argument.
`summarise()` has grouped output by 'seqnames', 'coord', 'sample_info_short'.
You can override using the `.groups` argument.
`summarise()` has grouped output by 'seqnames', 'coord', 'sample_info_short'.
You can override using the `.groups` argument.
`summarise()` has grouped output by 'seqnames', 'coord', 'sample_info_short'.
You can override using the `.groups` argument.
```

Figure 7B upper panel

AT5G57360.1.png

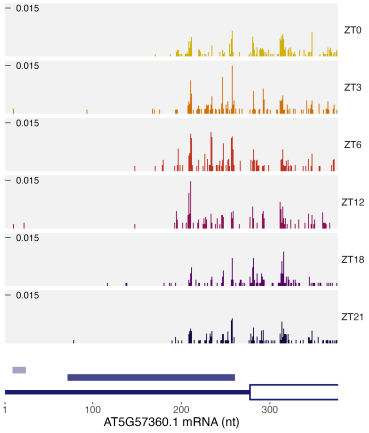
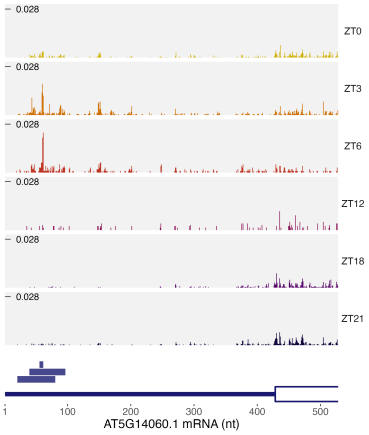


Figure 8B upper panels

AT5G14060.1.png



AT3G02020.1.png

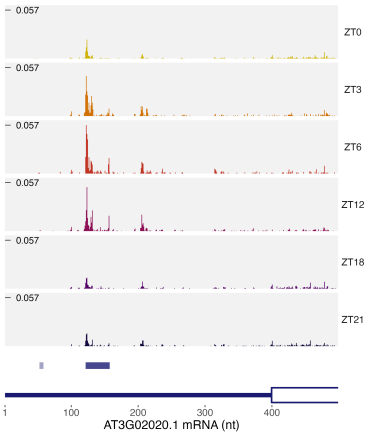


Figure S8

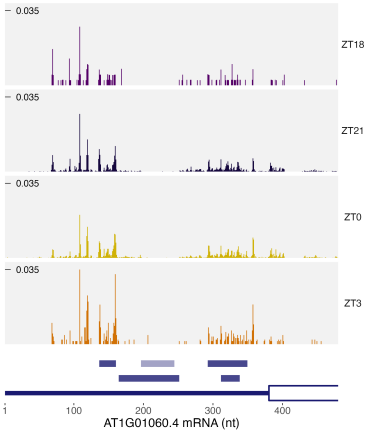
```
tbl_goi <-
  tibble::tribble(
    ~AGI,        ~gene_name,
    "AT1G01060", "LHY",
  )

for(i in seq_along(tbl_goi$AGI)) {
  temp_AGI_GOI <- tbl_goi$AGI[i]
  temp_name <- tbl_goi$gene_name[i]
  temp_label <- stringr::str_glue("{temp_AGI_GOI}_{temp_name}")
  fs::dir_create(path_out_figS8lp(temp_label))

  gr_goi_all <- make_gr_goi_all(txdb, temp_AGI_GOI)
  li_gr <- extract_li_gr(txdb, gr_goi_all)
  li_tbl_plot <- li_gr2li_tbl(li_gr)
  tx <-
    li_tbl_plot$transcript$tx_name %>%
    {.[grep(temp_AGI_GOI, .)]}
  for(AGI_GOI_TX in tx) {
    li_tbl_plot_tx <- make_li_tbl_plot_tx(li_tbl_plot, AGI_GOI_TX)
    tbl_uorf <- make_tbl_uorf(li_tbl_plot_tx, AGI_GOI_TX)
    tbl_plot_rna_coverage_tx <-
      make_tbl_plot_coverage_tx(li_tbl_plot_tx) %>%
      dplyr::filter(!(sample_info_short %in% c("ZT6", "ZT12"))) %>%
      return()

    tbl_plot_rna_coverage_tx <-
      tbl_plot_rna_coverage_tx %>%
      dplyr::mutate(
        sample_info_short =
          forcats::fct_relevel(sample_info_short, paste0("ZT", c(18,21,0,3,6,12)))
      )

    if(nrow(li_tbl_plot_tx$futr) != 0) {
      TITLE_X <- paste0(AGI_GOI_TX, " mRNA (nt)")
     plot_version5(li_tbl_plot_tx, tbl_uorf, tbl_plot_rna_coverage_tx, score_modifier = function(x)
x / sum(x)) &
        coord_cartesian(xlim = c(1, max(li_tbl_plot_tx$futr$tx_end) + 100))
      path_out_figS8lp(stringr::str_glue("{temp_label}/{AGI_GOI_TX}.png")) %>%
        ggsave(width = 178, height = 200, dpi = 300, units = "mm")
      path_out_figS8lp(stringr::str_glue("{temp_label}/{AGI_GOI_TX}.pdf")) %>%
        ggsave(width = 178, height = 200, units = "mm")
      rm(TITLE_X)
    }
    rm(li_tbl_plot_tx, tbl_uorf, tbl_plot_rna_coverage_tx)
  }
  rm(temp_AGI_GOI, gr_goi_all, li_tbl_plot)
}
```

```
`summarise()` has grouped output by 'seqnames', 'coord', 'sample_info_short'.
You can override using the `.groups` argument.
`summarise()` has grouped output by 'seqnames', 'coord', 'sample_info_short'.
You can override using the `.groups` argument.
`summarise()` has grouped output by 'seqnames', 'coord', 'sample_info_short'.
You can override using the `.groups` argument.
`summarise()` has grouped output by 'seqnames', 'coord', 'sample_info_short'.
You can override using the `.groups` argument.
`summarise()` has grouped output by 'seqnames', 'coord', 'sample_info_short'.
You can override using the `.groups` argument.
`summarise()` has grouped output by 'seqnames', 'coord', 'sample_info_short'.
```

```
You can override using the `.groups` argument.
`summarise()` has grouped output by 'seqnames', 'coord', 'sample_info_short'.
You can override using the `.groups` argument.
`summarise()` has grouped output by 'seqnames', 'coord', 'sample_info_short'.
You can override using the `.groups` argument.
```

Figure S8 upper panel

AT1G01060.4.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] stats4    stats     graphics  grDevices datasets  utils     methods
[8] base

other attached packages:
 [1] GenomicFeatures_1.48.4 AnnotationDbi_1.58.0   Biobase_2.56.0
 [4] rtracklayer_1.56.1     GenomicRanges_1.48.0   GenomeInfoDb_1.32.4
```

```
 [7] IRanges_2.30.1         S4Vectors_0.34.0       BiocGenerics_0.42.0
[10] ggplot2_3.4.2          magrittr_2.0.3

loaded via a namespace (and not attached):
 [1] bitops_1.0-7                matrixStats_0.62.0
 [3] fs_1.5.2                    bit64_4.0.5
 [5] filelock_1.0.2              progress_1.2.2
 [7] httr_1.4.5                  rprojroot_2.0.3
 [9] tools_4.2.1                 utf8_1.2.2
[11] R6_2.5.1                    DBI_1.1.3
[13] colorspace_2.0-3            withr_2.5.0
[15] tidyselect_1.2.0            gridExtra_2.3
[17] prettyunits_1.1.1           bit_4.0.5
[19] curl_4.3.3                  compiler_4.2.1
[21] textshaping_0.3.6           cli_3.6.0
[23] xml2_1.3.3                  DelayedArray_0.22.0
[25] labeling_0.4.2              prismatic_1.1.1
[27] scales_1.2.1                rappdirs_0.3.3
[29] systemfonts_1.0.4           stringr_1.5.0
[31] digest_0.6.31               Rsamtools_2.12.0
[33] rmarkdown_2.24              XVector_0.36.0
[35] pkgconfig_2.0.3             htmltools_0.5.3
[37] MatrixGenerics_1.8.1        dbplyr_2.3.2
[39] fastmap_1.1.0               rlang_1.1.0
[41] rstudioapi_0.14             RSQLite_2.2.18
[43] BiocIO_1.6.0                farver_2.1.1
[45] generics_0.1.3              jsonlite_1.8.4
[47] BiocParallel_1.30.4         dplyr_1.1.1
[49] RCurl_1.98-1.9              GenomeInfoDbData_1.2.8
[51] patchwork_1.1.2             Matrix_1.6-4
[53] Rcpp_1.0.11                 munsell_0.5.0
[55] fansi_1.0.3                 viridis_0.6.2
[57] lifecycle_1.0.3             stringi_1.7.12
[59] yaml_2.3.6                  SummarizedExperiment_1.26.1
[61] zlibbioc_1.42.0             BiocFileCache_2.4.0
[63] grid_4.2.1                  blob_1.2.3
[65] parallel_4.2.1              forcats_1.0.0
[67] crayon_1.5.2                lattice_0.20-45
[69] Biostrings_2.64.1           hms_1.1.3
[71] KEGGREST_1.36.3             magick_2.7.3
[73] knitr_1.42                  pillar_1.9.0
[75] rjson_0.2.21                codetools_0.2-18
[77] biomaRt_2.52.0              XML_3.99-0.11
[79] glue_1.6.2                  evaluate_0.20
[81] renv_1.0.3                  BiocManager_1.30.18
[83] png_0.1-7                   vctrs_0.6.1
[85] gtable_0.3.1                purrr_1.0.1
[87] tidyr_1.3.0                 cachem_1.0.6
[89] xfun_0.40                   restfulr_0.0.15
[91] ragg_1.2.5                  viridisLite_0.4.1
[93] tibble_3.2.1                GenomicAlignments_1.32.1
[95] plyranges_1.16.0            memoise_2.0.1
[97] here_1.0.1
```