FQ4

2025-11-20

```
library(lme4)
```

```
## Loading required package: Matrix
```

```
library(tibble)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats   1.0.0     v readr     2.1.5
## v ggplot2   3.5.2     v stringr   1.5.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.4

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x tidyr::expand() masks Matrix::expand()
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x tidyr::pack()   masks Matrix::pack()
## x tidyr::unpack() masks Matrix::unpack()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(ggthemes)
library(ggrepel)
library(lfmm)
library(RSpectra)
#install.packages("fuzzyjoin")
library(fuzzyjoin)
load("pl_lt_t.lmer.RData")
load("growth_light_time.lmer.RData")
source("manhattan_plot.R")
```

```
##
## Attaching package: 'scales'
##
## The following object is masked from 'package:purrr':
##
##     discard
##
## The following object is masked from 'package:readr':
##
##     col_factor
```

```
#file_path <- file.choose()
#rmarkdown::render(file_path)
```

manhattan plot for 2022

```
pheno_22 <- ranef(pl_lt_t.lmer)$pop %>%
  as_tibble(rownames = "pop",.name_repair = "unique")%>%
  rename(blup_intercept = `(Intercept)...1`,
         blup_light = `mean_light_ly_day2`) %>%
  mutate(model = "Y2022")
```

```
## New names:
## * '(Intercept)' -> '(Intercept)...1'
## * '(Intercept)' -> '(Intercept)...3'
```

```
geno_22 <- read.delim("Data/merged_maf_common_UCD2022.tsv", header = TRUE)

Y <- as.matrix(geno_22[,-1])
Y <- t(Y)
X <- matrix(scale(pheno_22$blup_light), ncol = 1)
rownames(X) <- pheno_22$pop
colnames(X) <- "blup_light"
common <- intersect(rownames(X), rownames(Y))
Y <- Y[common,]
X <- X[common, , drop = FALSE]
mod.lfmm <- lfmm_ridge(Y = Y,
                       X = X,
                       K = 2)

 pv <- lfmm_test(Y = Y,
                 X = X,
                 lfmm = mod.lfmm,
                 calibrate = "gif")
pvalues <- pv$calibrated.pvalue

plot_data = tibble(loci = geno_22[,1], p_val = pv$pvalue[,1], calibrated.pvalue = pv$calibrated.pvalue[
separate_wider_delim(loci, delim = ":", names = c("chr", "bp")) |>
filter(str_detect(chr, "Chr")) |>
mutate(p.bonf = p.adjust(p_val, method = "bonferroni"),
bp = as.numeric(bp))
```

```
manhattan_22 = plot_manhattan(plot_data,
                        chr_col = "chr",
                        pos_col = "bp",
                        value_col = "calibrated.pvalue",
                        transform = "neglog10",
                        title = "Light association",
                        highlight_top_n = 20)
```

manhattan plot for WL2 2023

```
#import the X and Y
pheno_23 <- ranef(growth_light_time.lmer)$parent_pop %>%
  as_tibble(rownames = "parent_pop", .name_repair = "unique") %>%
  rename(blup_intercept = `(Intercept)...1`,
         blup_light = `weekly_avg_SlrW2`) %>%
  mutate(model = "Y2023")
```

```
## New names:
## * `(Intercept)` -> `(Intercept)...1`
## * `(Intercept)` -> `(Intercept)...3`
```

```
geno_23 <- read.delim("Data/merged_maf_common_WL2_2023.tsv", header = TRUE)

#process the data
Y_23 <- as.matrix(geno_23[,-1])
Y_23 <- t(Y_23)
X_23 <- matrix(scale(pheno_23$blup_light), ncol = 1)
rownames(X_23) <- pheno_23$parent_pop
colnames(X_23) <- "blup_light"
common <- intersect(rownames(X_23), rownames(Y_23))
Y_23 <- Y_23[common,]
X_23 <- X_23[common, , drop = FALSE]

#fit the model
mod.lfmm_23 <- lfmm_ridge(Y = Y_23,
                        X = X_23,
                        K = 2)
pv_23 <- lfmm_test(Y = Y_23,
                 X = X_23,
                 lfmm = mod.lfmm_23,
                 calibrate = "gif")
pvalues_23 <- pv_23$calibrated.pvalue

#make the manhattan plot
plot_data_23 = tibble(loci = geno_23[,1], p_val = pv_23$pvalue[,1], calibrated.pvalue = pv_23$calibrate
separate_wider_delim(loci, delim = ":", names = c("chr", "bp")) |>
filter(str_detect(chr, "Chr")) |>
mutate(p.bonf = p.adjust(p_val, method = "bonferroni"),
bp = as.numeric(bp))


manhattan_23 = plot_manhattan(plot_data_23,
```

```
                          chr_col = "chr",
                          pos_col = "bp",
                          value_col = "calibrated.pvalue",
                          transform = "neglog10",
                          title = "Light association",
                          highlight_top_n = 20)

# Save plot
ggsave("light_interaction23.png", manhattan_23, width = 12, height = 6, dpi = 300)
```

```
prep_manhattan_data <- function(data,
                                chr_col = "chr",
                                pos_col = "bp",
                                value_col = "calibrated.pvalue",
                                transform = "neglog10") {
  plot_data <- data %>%
    rename(chr = !!sym(chr_col),
           pos = !!sym(pos_col),
           value = !!sym(value_col))

  if (transform == "log10") {
    plot_data <- plot_data %>% mutate(plot_value = log10(value))
  } else if (transform == "neglog10") {
    plot_data <- plot_data %>% mutate(plot_value = -log10(value))
  } else {
    plot_data <- plot_data %>% mutate(plot_value = value)
  }

  plot_data <- plot_data %>%
    filter(!is.na(plot_value), !is.infinite(plot_value)) %>%
    mutate(chr_num = as.numeric(gsub("\\D", "", chr))) %>%
    arrange(chr_num, pos)

  chr_lengths <- plot_data %>%
    group_by(chr, chr_num) %>%
    summarise(chr_len = max(pos), .groups = "drop") %>%
    arrange(chr_num) %>%
    mutate(tot = cumsum(as.numeric(chr_len)) - chr_len)

  plot_data <- plot_data %>%
    left_join(chr_lengths %>% select(chr, tot), by = "chr") %>%
    mutate(BPcum = pos + tot)

  axis_df <- plot_data %>%
    group_by(chr, chr_num) %>%
    summarize(center = (max(BPcum) + min(BPcum)) / 2, .groups = "drop") %>%
    arrange(chr_num)

  list(plot_data = plot_data, axis_df = axis_df)
}

prep22 <- prep_manhattan_data(plot_data,
                              chr_col = "chr",
```

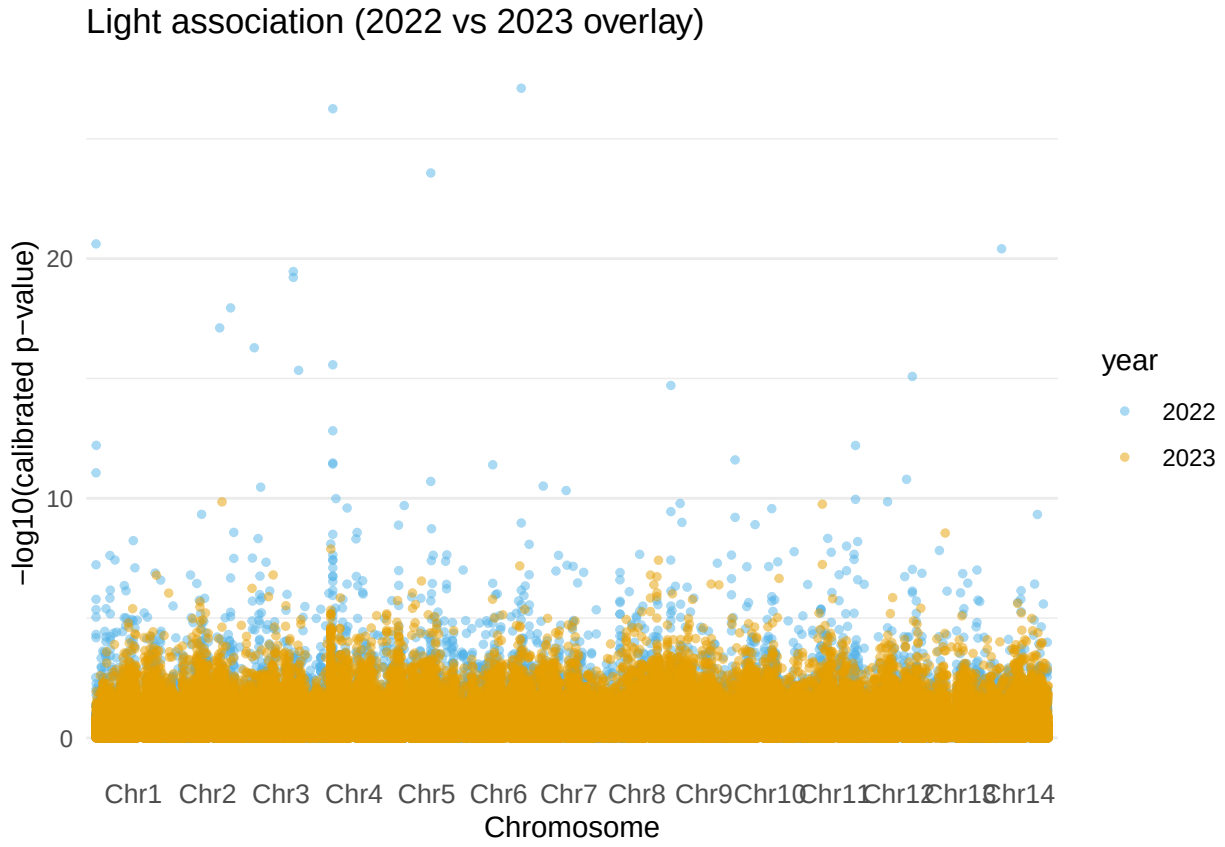
```
                              pos_col = "bp",
                              value_col = "calibrated.pvalue",
                              transform = "neglog10")

prep23 <- prep_manhattan_data(plot_data_23,
                              chr_col = "chr",
                              pos_col = "bp",
                              value_col = "calibrated.pvalue",
                              transform = "neglog10")

overlay_df <- bind_rows(
  prep22$plot_data %>% mutate(year = "2022"),
  prep23$plot_data %>% mutate(year = "2023")
)

overlay_plot <- ggplot(overlay_df,
                       aes(x = BPcum, y = plot_value)) +
  geom_point(aes(color = year),
             alpha = 0.5, size = 1) +
  scale_color_manual(values = c("2022" = "#56B4E9",
                                "2023" = "#E69F00")) +
  scale_x_continuous(breaks = prep22$axis_df$center,
                     labels = prep22$axis_df$chr,
                     expand = c(0.01, 0.01)) +
  labs(title = "Light association (2022 vs 2023 overlay)",
       x = "Chromosome",
       y = "-log10(calibrated p-value)") +
  theme_minimal() +
  theme(
    panel.grid.major.x = element_blank(),
    panel.grid.minor.x = element_blank(),
    axis.text.x = element_text(angle = 0, size = 10)
  )

print(overlay_plot)
```




```
top1_UCD = plot_data |>
slice_min(calibrated.pvalue, prop=0.001) |>
select(seqid = chr, bp, p_val, calibrated.pvalue, p.bonf)
write_tsv(top1_UCD, "top1_UCD_loc.tsv")

top1_WL2 = plot_data_23 |>
slice_min(calibrated.pvalue, prop=0.001) |>
select(seqid = chr, bp, p_val, calibrated.pvalue, p.bonf)
write_tsv(top1_WL2, "top1_WL2_loc.tsv")
```

```
w <- 1e5 / 2

reg22 <- top1_UCD %>%
  transmute(seqid, bp22=bp,start = bp - w, end = bp + w, p22 = calibrated.pvalue)

reg23 <- top1_WL2 %>%
  transmute(seqid, bp23=bp,start = bp - w, end = bp + w, p23 = calibrated.pvalue)

overlap_list <- list()

chroms <- intersect(reg22$seqid, reg23$seqid)

for (chr_i in chroms) {

  d1 <- reg22 %>% filter(seqid == chr_i)
```

```
  d2 <- reg23 %>% filter(seqid == chr_i)

  if (nrow(d1) > 0 & nrow(d2) > 0) {
    tmp <- fuzzyjoin::interval_inner_join(
      d1, d2,
      by = c("start", "end")
    ) %>%
      mutate(
        chr = chr_i,
        overlap_start = pmax(start.x, start.y),
        overlap_end   = pmin(end.x, end.y),
        width         = overlap_end - overlap_start
      ) %>%
      filter(width > 0)

    overlap_list[[chr_i]] <- tmp
  }
}

overlap <- bind_rows(overlap_list)
overlap_clean <- overlap %>%
  rename(
    chr      = chr,
    start22  = start.x,
    end22    = end.x,
    p22      = p22,
    start23  = start.y,
    end23    = end.y,
    p23      = p23
  )
head(overlap)
```

```
## # A tibble: 6 x 14
##   seqid.x    bp22 start.x   end.x      p22 seqid.y   bp23 start.y  end.y     p23
##   <chr>     <dbl>   <dbl>   <dbl>    <dbl> <chr>    <dbl>   <dbl>  <dbl>   <dbl>
## 1 Chr4    4668094 4618094 4718094  8.01e-9 Chr4    4.68e6 4632775 4.73e6 1.31e-8
## 2 Chr4    4668094 4618094 4718094  8.01e-9 Chr4    4.68e6 4632129 4.73e6 4.63e-6
## 3 Chr4    4668094 4618094 4718094  8.01e-9 Chr4    4.68e6 4629588 4.73e6 6.57e-6
## 4 Chr4    4668094 4618094 4718094  8.01e-9 Chr4    4.68e6 4633436 4.73e6 6.90e-6
## 5 Chr4    4668094 4618094 4718094  8.01e-9 Chr4    4.68e6 4631570 4.73e6 7.40e-6
## 6 Chr4    4668094 4618094 4718094  8.01e-9 Chr4    4.66e6 4605375 4.71e6 1.02e-5
## # i 4 more variables: chr <chr>, overlap_start <dbl>, overlap_end <dbl>,
## #   width <dbl>
```

```
df22 <- plot_data %>% mutate(logp = -log10(calibrated.pvalue))
sig22 <- df22 %>% slice_max(logp, prop = 0.01)
df23 <- plot_data_23 %>% mutate(logp = -log10(calibrated.pvalue))
sig23 <- df23 %>% slice_max(logp, prop = 0.01)

#combine SNPs within 50kb to make a peak region
merge_peaks <- function(df, max_gap = 50000){
  df %>%
    arrange(chr, bp) %>%
```

```
    group_by(chr) %>%
    mutate(gap = bp - lag(bp, default = first(bp)),
           new_region = gap > max_gap,
           region_id = cumsum(new_region)) %>%
    group_by(chr, region_id) %>%
    summarise(
      region_start = min(bp),
      region_end   = max(bp),
      peak_snp     = bp[which.max(logp)],
      peak_logp    = max(logp),
      .groups = "drop"
    )
}

peaks22 <- merge_peaks(sig22)
peaks23 <- merge_peaks(sig23)

peak_overlap <- interval_inner_join(
  peaks22 %>% rename(start = region_start, end = region_end),
  peaks23 %>% rename(start = region_start, end = region_end),
  by = c("start", "end")
) %>%
  mutate(
    overlap_start = pmax(start.x, start.y),
    overlap_end   = pmin(end.x, end.y),
    width = overlap_end - overlap_start
  ) %>%
  filter(width > 0)

peak_overlap
```

```
## # A tibble: 132 x 15
##    chr.x region_id.x  start.x    end.x peak_snp.x peak_logp.x chr.y region_id.y
##    <chr>       <int>    <dbl>    <dbl>      <dbl>       <dbl> <chr>       <int>
##  1 Chr1            6  3189599  3193702    3189599        4.45 Chr6            3
##  2 Chr1            7  3597659  3622098    3622098        5.39 Chr7           10
##  3 Chr1           19 10059687 10084759   10084759        6.36 Chr1           11
##  4 Chr1           19 10059687 10084759   10084759        6.36 Chr4           23
##  5 Chr1           21 10689169 10689203   10689203        6.14 Chr4           26
##  6 Chr1           25 12686280 12686428   12686280        4.95 Chr6           24
##  7 Chr1           25 12686280 12686428   12686280        4.95 Chr7           34
##  8 Chr1           27 13822575 13824709   13822575        7.10 Chr1           30
##  9 Chr1           39 20168900 20170476   20170381        4.11 Chr1           46
## 10 Chr1           39 20168900 20170476   20170381        4.11 Chr6           54
## # i 122 more rows
## # i 7 more variables: start.y <dbl>, end.y <dbl>, peak_snp.y <dbl>,
## #   peak_logp.y <dbl>, overlap_start <dbl>, overlap_end <dbl>, width <dbl>
```

upload the gene table and find the nearby genes

```
genes = read_tsv("Data/gene_description.tsv")
```

```
## Rows: 34643 Columns: 6
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (4): seqid, strand, gene_id, description
## dbl (2): start, end
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
window_size = 100000
```

```
top20_UCD = plot_data |>
slice_min(calibrated.pvalue, n = 20) |>
select(seqid = chr, bp, p_val, calibrated.pvalue, p.bonf)
write_tsv(top20_UCD, "top20_UCD_loc.tsv")

top20_WL2 = plot_data_23 |>
slice_min(calibrated.pvalue, n = 20) |>
select(seqid = chr, bp, p_val, calibrated.pvalue, p.bonf)
write_tsv(top20_WL2, "top20_WL2_loc.tsv")
```

```
genes_near_snps_UCD = top20_UCD |>
  left_join(genes, by = "seqid", relationship = "many-to-many") |>
  mutate(
    # Calculate distances
    dist_to_start = bp - start,
    dist_to_end = bp - end,

    # Check if SNP is within gene body
    in_gene = bp >= start & bp <= end,

    # Check if SNP is within window (upstream or downstream)
    in_window = (bp >= (start - window_size) & bp <= (end + window_size))
  ) |>
  filter(in_window) |>
  mutate(
    # Categorize position
    position = case_when(
      in_gene ~ "in_gene",
      bp < start ~ "upstream",
      bp > end ~ "downstream"
    ),
    distance = case_when(
      in_gene ~ 0,
      bp < start ~ start - bp,
      bp > end ~ bp - end
    )
  ) |>
  arrange(seqid, bp, distance)

#write_tsv(genes_near_snps_UCD, "UCD_2022_genes.tsv")
genes_near_snps_UCD
```

```
## # A tibble: 511 x 16
```

```
##    seqid     bp      p_val calibrated.pvalue p.bonf  start    end strand gene_id
##    <chr>  <dbl>      <dbl>             <dbl>  <dbl>  <dbl>  <dbl> <chr>  <chr>
##  1 Chr1  102763 0.00000136          2.41e-21  0.131  99758 103380 +      Sdiv_p~
##  2 Chr1  102763 0.00000136          2.41e-21  0.131 103721 107316 -      Sdiv_p~
##  3 Chr1  102763 0.00000136          2.41e-21  0.131  98238 100014 -      Sdiv_p~
##  4 Chr1  102763 0.00000136          2.41e-21  0.131  92330  97787 -      Sdiv_p~
##  5 Chr1  102763 0.00000136          2.41e-21  0.131 108145 110992 +      Sdiv_p~
##  6 Chr1  102763 0.00000136          2.41e-21  0.131 109567 110014 -      Sdiv_p~
##  7 Chr1  102763 0.00000136          2.41e-21  0.131 111553 114116 +      Sdiv_p~
##  8 Chr1  102763 0.00000136          2.41e-21  0.131  85739  92723 +      Sdiv_p~
##  9 Chr1  102763 0.00000136          2.41e-21  0.131 115206 121670 -      Sdiv_p~
## 10 Chr1  102763 0.00000136          2.41e-21  0.131 118937 119409 +      Sdiv_p~
## # i 501 more rows
## # i 7 more variables: description <chr>, dist_to_start <dbl>,
## #   dist_to_end <dbl>, in_gene <lgl>, in_window <lgl>, position <chr>,
## #   distance <dbl>
```

```
genes_near_snps_WL2 = top20_WL2 |>
  left_join(genes, by = "seqid", relationship = "many-to-many") |>
  mutate(
    # Calculate distances
    dist_to_start = bp - start,
    dist_to_end = bp - end,

    # Check if SNP is within gene body
    in_gene = bp >= start & bp <= end,

    # Check if SNP is within window (upstream or downstream)
    in_window = (bp >= (start - window_size) & bp <= (end + window_size))
  ) |>
  filter(in_window) |>
  mutate(
    # Categorize position
    position = case_when(
      in_gene ~ "in_gene",
      bp < start ~ "upstream",
      bp > end ~ "downstream"
    ),
    distance = case_when(
      in_gene ~ 0,
      bp < start ~ start - bp,
      bp > end ~ bp - end
    )
  ) |>
  arrange(seqid, bp, distance)

#write_tsv(genes_near_snps_WL2, "WL2_2023_genes.tsv")
genes_near_snps_WL2
```

```
## # A tibble: 278 x 16
##    seqid       bp    p_val calibrated.pvalue p.bonf  start    end strand gene_id
##    <chr>    <dbl>    <dbl>             <dbl>  <dbl>  <dbl>  <dbl> <chr>  <chr>
##  1 Chr1  21374778  1.39e-5       0.000000163      1 2.14e7 2.14e7 +      Sdiv_p~
##  2 Chr1  21374778  1.39e-5       0.000000163      1 2.14e7 2.14e7 +      Sdiv_p~
```

```
##  3 Chr1  21374778  1.39e-5       0.000000163      1 2.14e7 2.14e7 +      Sdiv_p~
##  4 Chr1  21374778  1.39e-5       0.000000163      1 2.14e7 2.14e7 -      Sdiv_p~
##  5 Chr1  21374778  1.39e-5       0.000000163      1 2.14e7 2.14e7 -      Sdiv_p~
##  6 Chr1  21374778  1.39e-5       0.000000163      1 2.14e7 2.14e7 -      Sdiv_p~
##  7 Chr1  21374778  1.39e-5       0.000000163      1 2.14e7 2.14e7 +      Sdiv_p~
##  8 Chr1  21374778  1.39e-5       0.000000163      1 2.14e7 2.14e7 -      Sdiv_p~
##  9 Chr1  21374778  1.39e-5       0.000000163      1 2.14e7 2.14e7 +      Sdiv_p~
## 10 Chr1  21374778  1.39e-5       0.000000163      1 2.13e7 2.14e7 +      Sdiv_p~
## # i 268 more rows
## # i 7 more variables: description <chr>, dist_to_start <dbl>,
## #   dist_to_end <dbl>, in_gene <lgl>, in_window <lgl>, position <chr>,
## #   distance <dbl>
```

```
prioritized_genes <- genes_near_snps_UCD %>%
  group_by(gene_id, description, seqid) %>%
  summarise(
    n_snps = n(),
    min_distance = min(distance),
    any_in_gene = any(position == "in_gene"),
    any_promoter = any(distance <= 1000 & position=="upstream"),
    .groups = "drop"
  ) %>%
  mutate(
    priority_score =
      3 * any_in_gene +
      2 * any_promoter +
      1 * (min_distance <= 3000) +
      n_snps
  ) %>%
  arrange(desc(priority_score), min_distance)

prioritized_genes
```

```
## # A tibble: 358 x 8
##    gene_id        description seqid n_snps min_distance any_in_gene any_promoter
##    <chr>          <chr>       <chr>  <int>        <dbl> <lgl>       <lgl>
##  1 Sdiv_ptg00001~ Similar to~ Chr4       5            0 TRUE        FALSE
##  2 Sdiv_ptg00001~ Similar to~ Chr1       2            0 TRUE        FALSE
##  3 Sdiv_ptg00001~ Similar to~ Chr1       2            0 TRUE        FALSE
##  4 Sdiv_ptg00001~ Similar to~ Chr4       5         1652 FALSE       FALSE
##  5 Sdiv_ptg00000~ Similar to~ Chr11      1            0 TRUE        FALSE
##  6 Sdiv_ptg00000~ Protein of~ Chr3       1            0 TRUE        FALSE
##  7 Sdiv_ptg00000~ Similar to~ Chr9       1            0 TRUE        FALSE
##  8 Sdiv_ptg00000~ Similar to~ Chr14      1            0 TRUE        FALSE
##  9 Sdiv_ptg00000~ Similar to~ Chr2       1            0 TRUE        FALSE
## 10 Sdiv_ptg00001~ Protein of~ Chr1       2          698 FALSE       TRUE
## # i 348 more rows
## # i 1 more variable: priority_score <dbl>
```

```
prioritized_genes <- genes_near_snps_WL2 %>%
  group_by(gene_id, description, seqid) %>%
  summarise(
    n_snps = n(),
```

```
    min_distance = min(distance),
    any_in_gene = any(position == "in_gene"),
    any_promoter = any(distance <= 1000 & position=="upstream"),
    .groups = "drop"
  ) %>%
  mutate(
    priority_score =
      3 * any_in_gene +
      2 * any_promoter +
      1 * (min_distance <= 3000) +
      n_snps
  ) %>%
  arrange(desc(priority_score), min_distance)

prioritized_genes
```

```
## # A tibble: 262 x 8
##    gene_id        description seqid n_snps min_distance any_in_gene any_promoter
##    <chr>          <chr>       <chr>  <int>        <dbl> <lgl>       <lgl>
##  1 Sdiv_ptg00000~ Similar to~ Chr10      1            0 TRUE        FALSE
##  2 Sdiv_ptg00000~ Similar to~ Chr3       1            0 TRUE        FALSE
##  3 Sdiv_ptg00001~ Similar to~ Chr4       1            0 TRUE        FALSE
##  4 Sdiv_ptg00001~ Similar to~ Chr1       1            0 TRUE        FALSE
##  5 Sdiv_ptg00001~ Protein of~ Chr1       1            0 TRUE        FALSE
##  6 Sdiv_ptg00000~ Protein of~ Chr3       1          694 FALSE       TRUE
##  7 Sdiv_ptg00000~ Similar to~ Chr8       2         2864 FALSE       FALSE
##  8 Sdiv_ptg00000~ Similar to~ Chr9       1         1075 FALSE       FALSE
##  9 Sdiv_ptg00001~ Similar to~ Chr4       1         1614 FALSE       FALSE
## 10 Sdiv_ptg00000~ Protein of~ Chr8       1         1749 FALSE       FALSE
## # i 252 more rows
## # i 1 more variable: priority_score <dbl>
```