FQ3

2025-11-20

```
#install.packages("ISOweek")
library(ISOweek)
library(lme4)
```

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

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

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

```
library(readxl)
library(readr)
library(janitor)
```

```
##
## Attaching package: 'janitor'
```

```
## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

```
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 stringr 1.5.1
## v ggplot2 3.5.2     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)
#install.packages("tinytex")
#tinytex::tlmgr_update()
```

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`
```

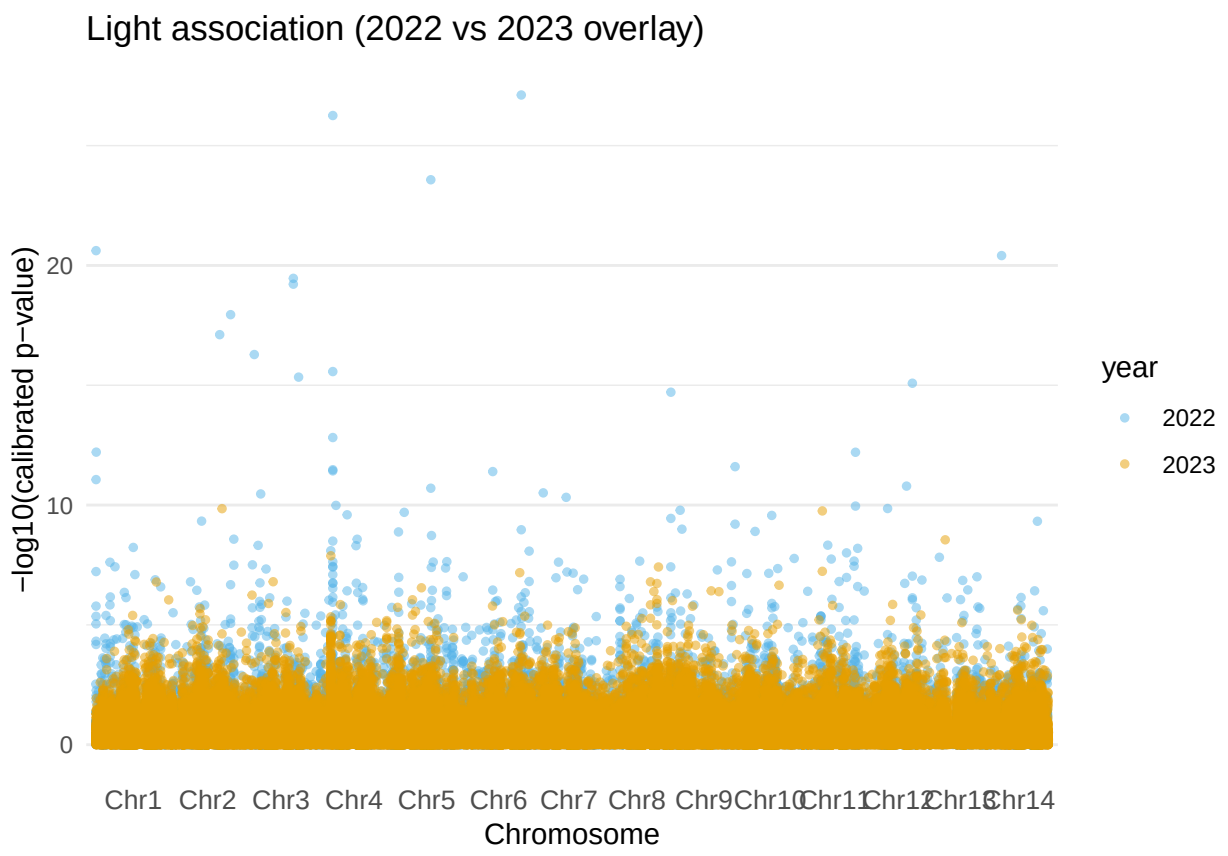
```
geno_23 <- read.delim("Data/merged_maf_common_WL2_2023.tsv", header = TRUE)

#process the data
Y_23 <- as.matrix(geno_23[,-1])
```

```r
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)
```

```r
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
```

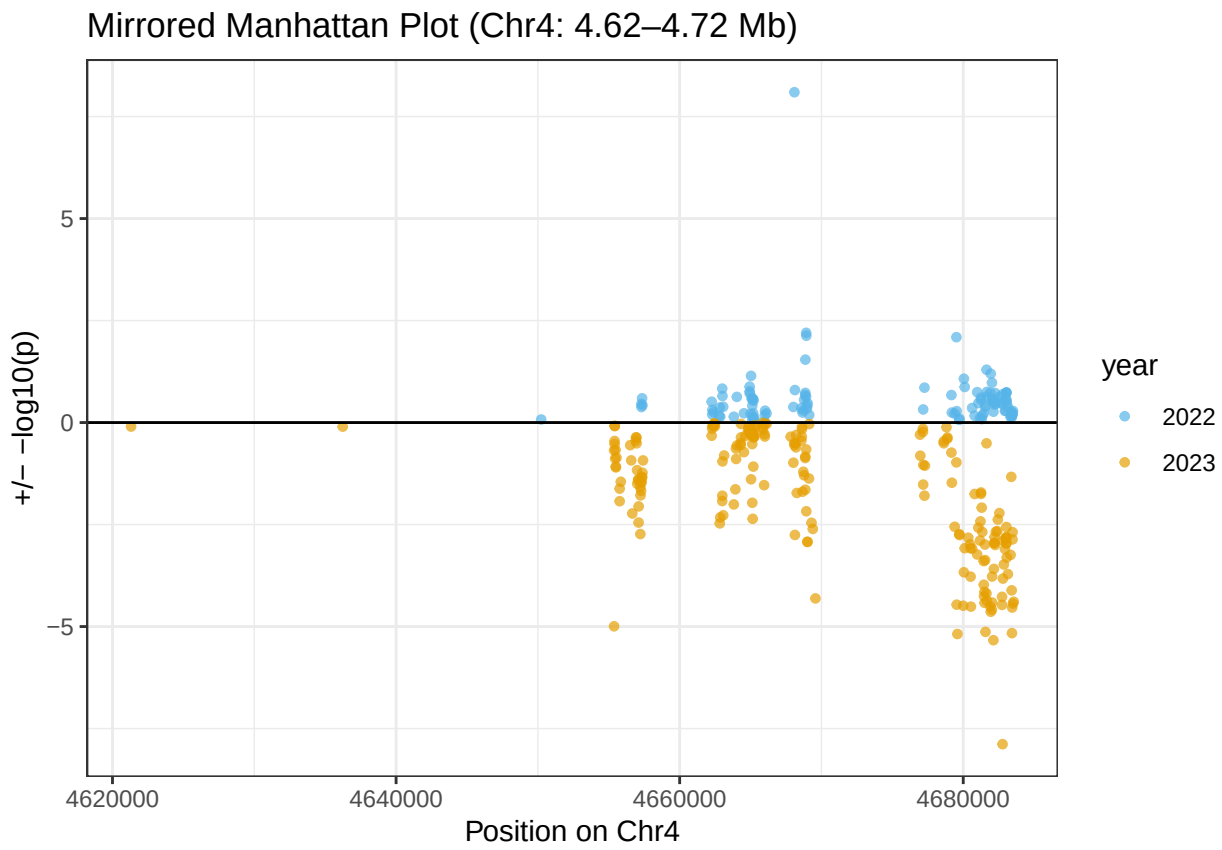
```
##  6 Sdiv_ptg00000~ Protein of~ Chr3       1            0 TRUE        FALSE
##  7 Sdiv_ptg00000~ Similar to~ Chr9       1            0 TRUE        FALSE
##  8 Sdiv_ptg00000~ Similar to~ Chr14      1            0 TRUE        FALSE
##  9 Sdiv_ptg00000~ Similar to~ Chr2       1            0 TRUE        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>
```

```
region_start <- 4618094
region_end   <- 4718094

# 2022 subset
chr4_22 <- prep22$plot_data %>%
  filter(chr == "Chr4",
         pos >= region_start,
         pos <= region_end) %>%
  mutate(logp_signed = plot_value)
```

```r
# 2023 subset
chr4_23 <- prep23$plot_data %>%
  filter(chr == "Chr4",
         pos >= region_start,
         pos <= region_end) %>%
  mutate(logp_signed = -plot_value)

# merge
mirror_df <- bind_rows(
  chr4_22 %>% mutate(year = "2022"),
  chr4_23 %>% mutate(year = "2023")
)

ggplot(mirror_df, aes(x = pos, y = logp_signed, color = year)) +
  geom_point(size = 1.2, alpha = 0.7) +
  geom_hline(yintercept = 0, color = "black") +
  labs(
    title = "Mirrored Manhattan Plot (Chr4: 4.62-4.72 Mb)",
    x = "Position on Chr4",
    y = "+/- -log10(p)"
  ) +
  scale_color_manual(values = c("2022" = "#56B4E9", "2023" = "#E69F00")) +
  theme_bw()
```



1.Create side-by-side plots of BLUP vs. allele frequency for the two closest significant SNPs from the UCD

and WL2 gardens

```
#find two closest significant SNPs around 4670000
target_pos <- 4670000
tol <- 30000

#UCD 2022
ucd_target <- plot_data %>%
  filter(chr == "Chr4",
         bp >= target_pos - tol,
         bp <= target_pos + tol) %>%
  arrange(calibrated.pvalue) %>%
  slice(1)

print(ucd_target)
```

```
## # A tibble: 1 x 5
##   chr        bp    p_val calibrated.pvalue p.bonf
##   <chr>   <dbl>    <dbl>             <dbl>  <dbl>
## 1 Chr4  4668094 0.000104     0.00000000801      1
```

```
#WL2 2023
wl2_target <- plot_data_23 %>%
  filter(chr == "Chr4",
         bp >= target_pos - tol,
         bp <= target_pos + tol) %>%
  arrange(calibrated.pvalue) %>%
  slice(1)

print(wl2_target)
```

```
## # A tibble: 1 x 5
##   chr        bp      p_val calibrated.pvalue p.bonf
##   <chr>   <dbl>      <dbl>             <dbl>  <dbl>
## 1 Chr4  4682775 0.00000488      0.0000000131  0.877
```

```
geno_table_22 = Y
colnames(geno_table_22)=geno_22[, 1]
geno_table_23 = Y_23
colnames(geno_table_23) = geno_23[, 1]

freq_UCD = as.data.frame(X) %>%
 rownames_to_column(var = "Population") |>
mutate(UCD_SNP = as.numeric(geno_table_22[, "Chr4:4668094"]))
freq_WL2 = as.data.frame(X_23) %>%
 rownames_to_column(var = "Population") |>
mutate(WL2_SNP = as.numeric(geno_table_23[, "Chr4:4682775"]))
freq_plot_both <- bind_rows(freq_UCD, freq_WL2)%>%
  pivot_longer(cols = UCD_SNP:WL2_SNP, names_to = "loci", values_to = "frequency")


freq_plot_both |>
```

```r
ggplot(aes(x = blup_light, y = frequency, label = Population)) +
geom_point(size = 3) +
geom_smooth(method = "lm", colour = "red") +
geom_text_repel(size = 6) +
facet_wrap(~loci, nrow = 1, scales = "free_y") +
theme_minimal()+
labs(x = "BLUP light", y = "Allele frequency")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

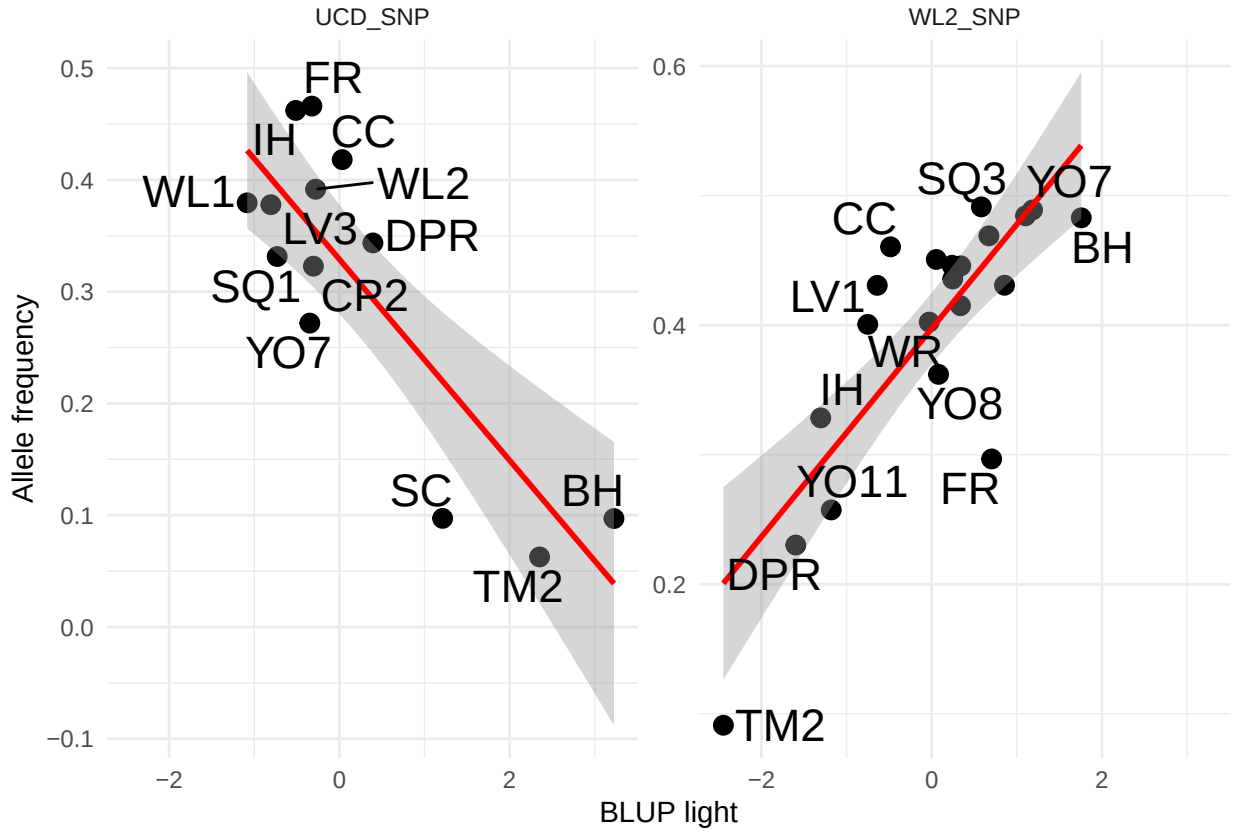
```
## Warning: Removed 35 rows containing non-finite outside the scale range
## (`stat_smooth()`).
```

```
## Warning: The following aesthetics were dropped during statistical transformation: label.
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a `group` aesthetic or to convert a numerical
##   variable into a factor?
## The following aesthetics were dropped during statistical transformation: label.
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a `group` aesthetic or to convert a numerical
##   variable into a factor?
```

```
## Warning: Removed 35 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

```
## Warning: Removed 35 rows containing missing values or values outside the scale range
## (`geom_text_repel()`).
```

```
## Warning: ggrepel: 10 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```



2. Plot the change in light intensity over time for both years.

```
#weekly solar radiation for UCD garden in 2022-2023 year
ucd_met <- read_csv("Data/UCD_met_data.csv")%>%clean_names()
```

```
## New names:
## Rows: 188 Columns: 33
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (17): Stn Name, CIMIS Region, Date, Jul, qc...9, qc...11, qc...13, qc..... dbl
## (15): Stn Id, ETo (in), Precip (in), Sol Rad (Ly/day), Avg Vap Pres (mBa... lgl
## (1): qc...7
## 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.
## * `qc` -> `qc...7`
## * `qc` -> `qc...9`
## * `qc` -> `qc...11`
## * `qc` -> `qc...13`
## * `qc` -> `qc...15`
## * `qc` -> `qc...17`
## * `qc` -> `qc...19`
## * `qc` -> `qc...21`
## * `qc` -> `qc...23`
## * `qc` -> `qc...25`
## * `qc` -> `qc...27`
```

```
## * `qc` -> `qc...29`
## * `qc` -> `qc...31`
## * `qc` -> `qc...33`
```

```
ucd_daily <- ucd_met %>%
  mutate(
    date = mdy(date),
    year = year(date),
    week = isoweek(date),
    light=sol_rad_ly_day,
    garden="UCD"
  ) %>%
  select(date, year, week, light,garden)
```

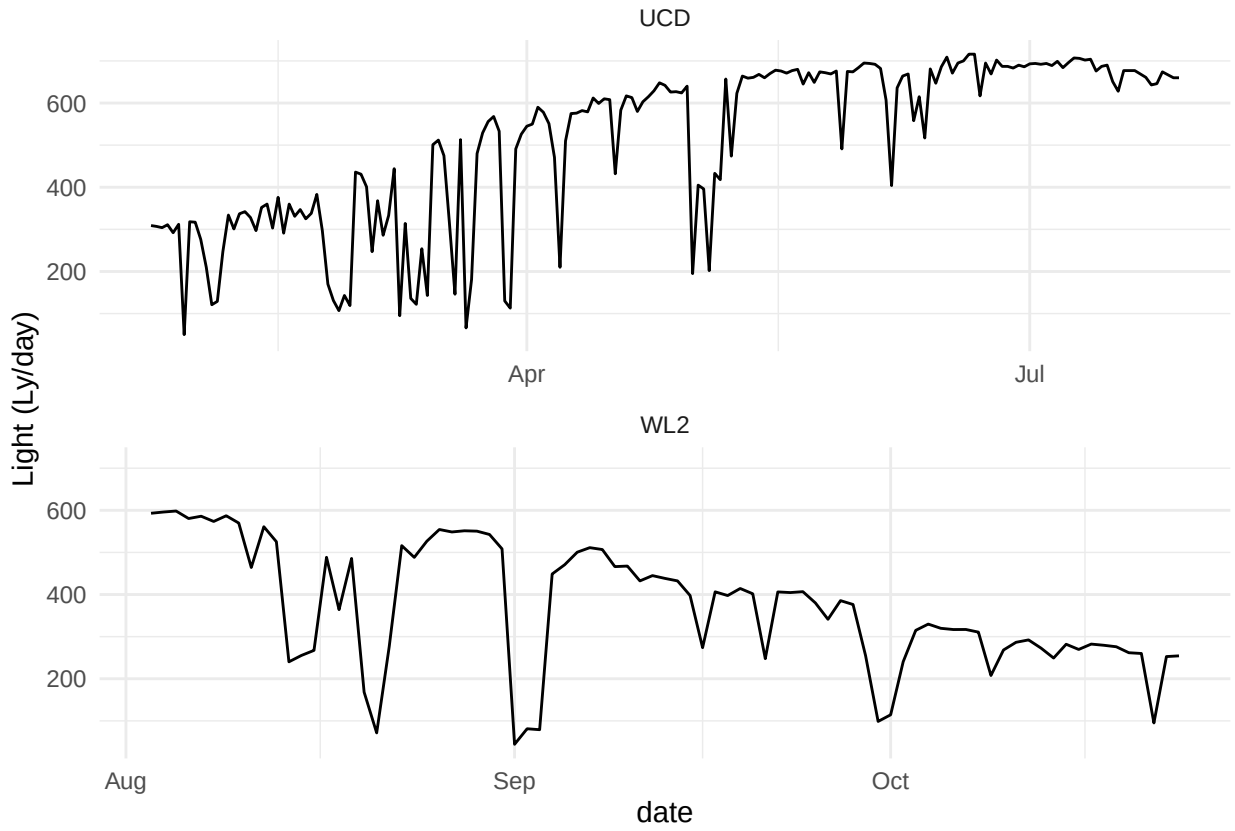
```
#weekly solar radiation for WL2 garden in 2023 year
wl2_met <- read_csv("Data/IntBioHalfHourTable_clean.txt")%>%clean_names()
```

```
## Rows: 4063 Columns: 139
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl  (138): RECORD, BattV_Max, PTemp_C_Max, SlrW_Avg, SlrW_Max, SlrW_Min, Sl...
## dttm   (1): TIMESTAMP
##
## 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.
```

```
wl2_daily <- wl2_met %>%
  mutate(
  timestamp = as.POSIXct(timestamp),
  date = as.Date(timestamp),
  week= isoweek(timestamp),
  year = 2023
  ) %>%
  group_by(year,week,date) %>%
  summarise(
    n_records   = n(),
    total_J_m2  = sum(slr_w_avg, na.rm = TRUE) * 1800,
    light    = total_J_m2 / 41840,
    .groups = "drop"
  )%>%
  filter(n_records>=40)%>%
  mutate(garden="WL2")


wl2_weekly <- wl2_daily %>%
  group_by(year, week) %>%
  summarise(weekly_avg_SlrW = mean(light, na.rm = TRUE), .groups = "drop")%>%
  mutate(iso_week  = paste0(year, "-W", sprintf("%02d", week)),
    week = ISOweek2date(paste0(iso_week, "-1")) - days(1))


#combine UCD and WL2
```

```
daily_light <- bind_rows(ucd_daily, wl2_daily)%>%
  select(-n_records, -total_J_m2)
```

```
daily_light %>%
  ggplot(aes(x = date, y = light)) +
  geom_line() +
  facet_wrap(~ garden, nrow = 2, scales = "free_x") +
  labs(x = "date", y = "Light (Ly/day)") +
  theme_minimal()
```



3.Generate a side-by-side plot of slope vs. population, incorporating the time factors.

UCD

```
plant_2223 <- read.csv("C:/Users/Tobyz/Desktop/Toby  /Maloof Lab/S.tort-light-growth/Data/UCD_2022_23_si
  clean_names() %>%
  mutate(survey_date = as.Date(survey_date, format = "%m/%d/%Y"))
#PID
pl_gr <- plant_2223 %>%
  unite("PID", pop:rep, sep = "_", remove = FALSE) %>%
  mutate(survey_date = as.Date(survey_date))
#filter out plants with negative growth < -5
pl_gr_cleaned <- pl_gr

repeat {
  pl_gr_cleaned <- pl_gr_cleaned %>%
```

```
    arrange(PID, survey_date) %>%
    group_by(PID) %>%
    mutate(growth = height - lag(height)) %>%
    filter(is.na(growth) | growth >= -5) %>%
    select(-growth) %>%
    ungroup()

  check <- pl_gr_cleaned %>%
    arrange(PID, survey_date) %>%
    group_by(PID) %>%
    mutate(growth = height - lag(height)) %>%
    filter(growth < -5)

  if (nrow(check) == 0) break
}
#define daily growth rate
pl_gr_daily <- pl_gr_cleaned %>%
  arrange(PID, survey_date) %>%
  group_by(PID) %>%
  mutate(
    prev_height = lag(height),
    prev_date = lag(survey_date),
    days_elapsed = as.numeric(survey_date - prev_date),
    daily_growth = (height - prev_height) / days_elapsed
  ) %>%
  ungroup()

pl_lt <- pl_gr_daily %>%
  rowwise() %>%
  mutate(
    mean_light_ly_day = mean(
      ucd_daily$light[
        !is.na(ucd_daily$date) &
        ucd_daily$date > prev_date &
        ucd_daily$date <= survey_date
      ],
      na.rm = TRUE
    ),
    n_days_light = sum(
      !is.na(ucd_daily$date) &
      ucd_daily$date > prev_date &
      ucd_daily$date <= survey_date,
      na.rm = TRUE
    )
  ) %>%
  ungroup()

#Standardization
pl_lt$mean_light_ly_day2 <- scale(pl_lt$mean_light_ly_day, center = TRUE, scale = TRUE)
mean_lt <- mean(pl_lt$mean_light_ly_day, na.rm = TRUE)
sd_lt   <- sd(pl_lt$mean_light_ly_day, na.rm = TRUE)

#Change Data type
```

```
pl_lt <- pl_lt %>%
  mutate(
    pop = factor(pop),
    PID        = factor(PID),
    block      = factor(block)
  )
pl_lt <- pl_lt %>%
arrange(survey_date) %>%
mutate(
week = as.integer((as.numeric(survey_date - min(survey_date)) %/% 7) + 1)
)
pl_lt$week_f <- as.factor(pl_lt$week)
pl_lt_t.lmer <- lmer(
daily_growth ~ mean_light_ly_day2 + (1 + mean_light_ly_day2 | pop) + (week|pop),
data = pl_lt, REML = TRUE
)
```

```
## boundary (singular) fit: see help('isSingular')
```

```
summary(pl_lt_t.lmer)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: daily_growth ~ mean_light_ly_day2 + (1 + mean_light_ly_day2 |
##     pop) + (week | pop)
##    Data: pl_lt
##
## REML criterion at convergence: 1643.8
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -4.1993 -0.2980 -0.0511  0.1352  8.5497
##
## Random effects:
##  Groups   Name               Variance  Std.Dev. Corr
##  pop      (Intercept)        2.677e-02 0.16362
##           mean_light_ly_day2 6.161e-03 0.07849  1.00
##  pop.1    (Intercept)        3.565e-04 0.01888
##           week               4.761e-05 0.00690  1.00
##  Residual                    1.127e-01 0.33569
## Number of obs: 2415, groups:  pop, 23
##
## Fixed effects:
##                    Estimate Std. Error t value
## (Intercept)         0.13735    0.04729   2.904
## mean_light_ly_day2  0.07728    0.02339   3.303
##
## Correlation of Fixed Effects:
##             (Intr)
## mn_lght_l_2 0.956
## optimizer (nloptwrap) convergence code: 0 (OK)
## boundary (singular) fit: see help('isSingular')
```

```r
# 1. Fixed effect slope
b_fix <- fixef(pl_lt_t.lmer)["mean_light_ly_day2"]

# 2. Random slope deviations (only from the 'pop' block)
rand_eff <- ranef(pl_lt_t.lmer)$pop
rand_slope <- rand_eff[, "mean_light_ly_day2"]

# 3. Get variance of the random slope (from the pop block only)
vc <- as.data.frame(VarCorr(pl_lt_t.lmer))

rand_slope_var <- vc %>%
  filter(grp == "pop", var1 == "mean_light_ly_day2", is.na(var2)) %>%
  pull(vcov)

rand_slope_sd <- sqrt(rand_slope_var)

# 4. Construct slopes & CI
pop_slopes <- tibble(
  pop        = rownames(rand_eff),
  slope      = b_fix + rand_slope,
  slope_SD   = rand_slope_sd,
  lower      = slope - 1.96 * slope_SD,
  upper      = slope + 1.96 * slope_SD
)

pop_slopes
```

```
## # A tibble: 23 x 5
##    pop    slope slope_SD   lower upper
##    <chr>  <dbl>    <dbl>   <dbl> <dbl>
##  1 BH    0.259    0.0785  0.105  0.413
##  2 CC    0.0791   0.0785 -0.0747 0.233
##  3 CP2   0.0601   0.0785 -0.0937 0.214
##  4 DPR   0.0995   0.0785 -0.0544 0.253
##  5 FR    0.0591   0.0785 -0.0948 0.213
##  6 IH    0.0484   0.0785 -0.105  0.202
##  7 LV1   0.0735   0.0785 -0.0803 0.227
##  8 LV3   0.0320   0.0785 -0.122  0.186
##  9 LVTR  0.0690   0.0785 -0.0849 0.223
## 10 LVTR1 0.0270   0.0785 -0.127  0.181
## # i 13 more rows
```

```r
# 5. Plot
UCD_slope <- ggplot(pop_slopes,
       aes(x = reorder(pop, slope), y = slope)) +
  geom_hline(yintercept = 0, linetype = 2) +
  geom_pointrange(aes(ymin = lower, ymax = upper)) +
  coord_flip() +
  labs(title = "UCD: Population-specific slopes (95% CI)",
       x = "Population",
       y = "Slope (cm/day per Ly/day)") +
  theme_bw()
```

WL2

```r
plant <- read.csv("Data/WL2-2023_Size_Combined.csv") %>%
  clean_names() %>%
  mutate(survey_date = as.Date(survey_date, format = "%m/%d/%Y"))
#PID
plant_growth <- plant %>%
  unite("PID", genotype:rep, sep = "_", remove = FALSE) %>%
  mutate(survey_date = as.Date(survey_date))
#filter out plants with negative growth < -5
plant_growth_cleaned <- plant_growth

repeat {
  plant_growth_cleaned <- plant_growth_cleaned %>%
    arrange(PID, survey_date) %>%
    group_by(PID) %>%
    mutate(growth = height_cm - lag(height_cm)) %>%
    filter(is.na(growth) | growth >= -5) %>%
    select(-growth) %>%
    ungroup()

  check <- plant_growth_cleaned %>%
    arrange(PID, survey_date) %>%
    group_by(PID) %>%
    mutate(growth = height_cm - lag(height_cm)) %>%
    filter(growth < -5)

  if (nrow(check) == 0) break
}
#define daily growth rate
plant_growth_daily <- plant_growth_cleaned %>%
  arrange(PID, survey_date) %>%
  group_by(PID) %>%
  mutate(
    prev_height = lag(height_cm),
    prev_date = lag(survey_date),
    days_elapsed = as.numeric(survey_date - prev_date),
    daily_growth = (height_cm - prev_height) / days_elapsed
  ) %>%
  ungroup()
#Align plant growth data to week
plant_weekly <- plant_growth_daily %>%
  filter(!is.na(daily_growth), days_elapsed > 0) %>%
  mutate(week = floor_date(survey_date, "week"))

#Adds `weekly_avg_SlrW` to plant data
plant_with_light <- plant_weekly %>%
  left_join(wl2_weekly, by = "week")

#Standardization
plant_with_light$weekly_avg_SlrW2 <- scale(plant_with_light$weekly_avg_SlrW, center = TRUE, scale = TRU
mean_light <- mean(plant_with_light$weekly_avg_SlrW, na.rm = TRUE)
sd_light   <- sd(plant_with_light$weekly_avg_SlrW, na.rm = TRUE)

#Change Data type
```

```
plant_with_light <- plant_with_light %>%
  mutate(
    parent_pop = factor(parent_pop),
    PID        = factor(PID),
    block      = factor(block)
  )
```

```
plant_with_light <- plant_with_light %>%
arrange(survey_date) %>%
mutate(
week = as.integer((as.numeric(survey_date - min(survey_date)) %/% 7) + 1)
)
plant_with_light$week_f <- as.factor(plant_with_light$week)
growth_light_time.lmer <- lmer(
daily_growth ~ weekly_avg_SlrW2 + (1 + weekly_avg_SlrW2 | parent_pop) + (week|parent_pop),
data = plant_with_light, REML = TRUE
)
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge with max|grad| = 0.0115239 (tol = 0.002, component 1)
```

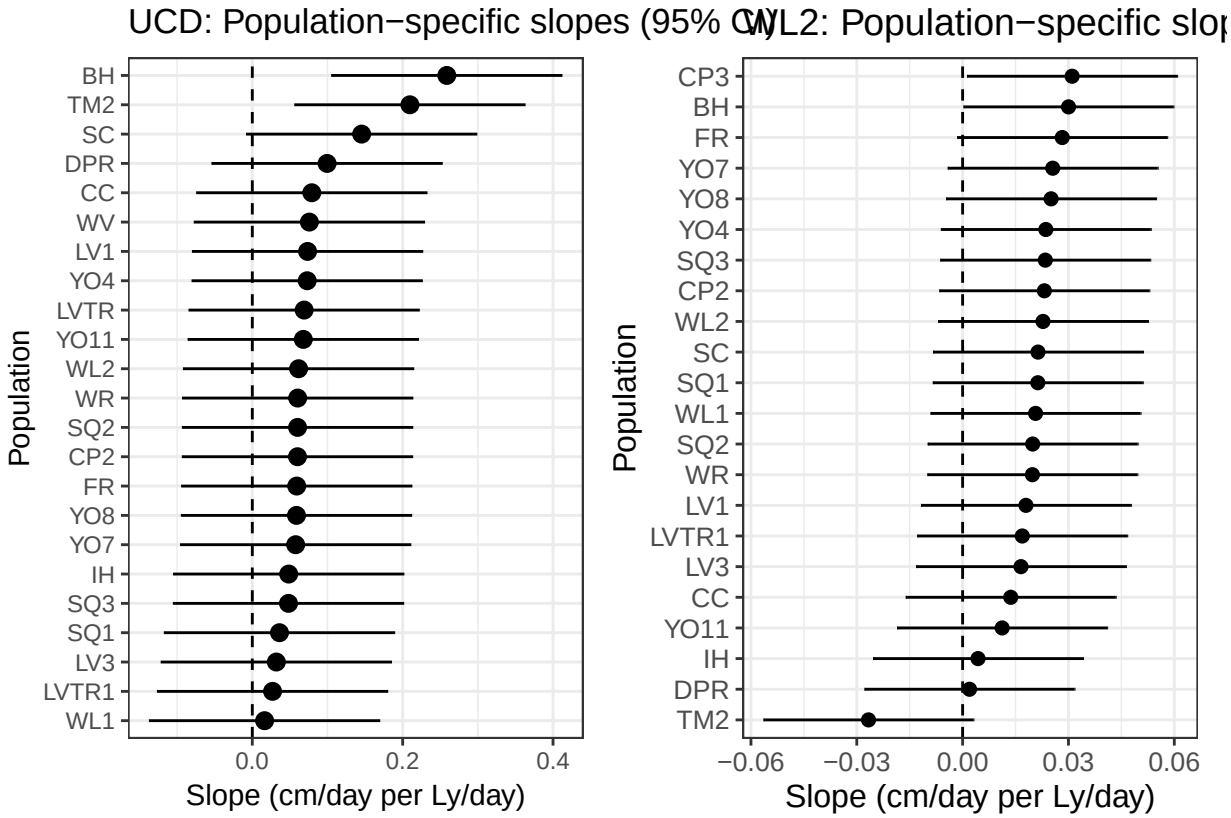
```
summary(growth_light_time.lmer)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula:
## daily_growth ~ weekly_avg_SlrW2 + (1 + weekly_avg_SlrW2 | parent_pop) +
##     (week | parent_pop)
##    Data: plant_with_light
##
## REML criterion at convergence: -6476.5
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -6.1031 -0.4732 -0.0083  0.4445  8.4529
##
## Random effects:
##  Groups       Name             Variance  Std.Dev. Corr
##  parent_pop   (Intercept)      1.448e-03 0.03805
##               weekly_avg_SlrW2 2.304e-04 0.01518  -0.56
##  parent_pop.1 (Intercept)      2.714e-03 0.05210
##               week             4.957e-05 0.00704  -0.99
##  Residual                      1.897e-02 0.13773
## Number of obs: 5870, groups:  parent_pop, 22
##
## Fixed effects:
##                  Estimate Std. Error t value
## (Intercept)      0.016011   0.008514   1.880
## weekly_avg_SlrW2 0.017809   0.004178   4.263
##
## Correlation of Fixed Effects:
##             (Intr)
## wkly_vg_SW2 -0.400
```

```
## optimizer (nloptwrap) convergence code: 0 (OK)
## Model failed to converge with max|grad| = 0.0115239 (tol = 0.002, component 1)
```

```
b_fix <- fixef(growth_light_time.lmer)["weekly_avg_SlrW2"]

re <- ranef(growth_light_time.lmer)$parent_pop
rand_slope <- re[, "weekly_avg_SlrW2", drop = TRUE]

vc_mat  <- VarCorr(growth_light_time.lmer)$parent_pop
sl_col  <- which(colnames(vc_mat) == "weekly_avg_SlrW2")
rand_slope_sd <- attr(vc_mat, "stddev")[sl_col]

pop_slope <- tibble(
  Population = rownames(re),
  slope      = as.numeric(b_fix + rand_slope),       # cm/day per W/m^2
  slope_SD   = rand_slope_sd,
  lower      = slope - 1.96 * slope_SD,
  upper      = slope + 1.96 * slope_SD
)
sd_light <- sd(plant_with_light[["weekly_avg_SlrW2"]], na.rm = TRUE)

pop_slope_std <- pop_slope %>%
  mutate(
    slope_per_Wm2 = slope / sd_light,
    lower_per_Wm2 = lower / sd_light,
    upper_per_Wm2 = upper / sd_light
  ) %>%
  arrange(slope_per_Wm2)

WL2_slope <- ggplot(pop_slope,
       aes(x = slope,
           y = reorder(Population, slope))) +
  geom_point(size = 2) +
  geom_errorbarh(aes(xmin = lower, xmax = upper), height = 0) +
  geom_vline(xintercept = 0, linetype = "dashed") +
  labs(
    title = "WL2: Population-specific slopes with 95% CI",
    x = "Slope (cm/day per Ly/day)",
    y = "Population"
  ) +
  theme_bw() +
  theme(
    text = element_text(size = 12)
  )
```

```
library(patchwork)
UCD_slope+WL2_slope
```


UCD: Population−specific slopes (95% CI)
WL2: Population−specific slop
Population
BH
TM2
SC
DPR
CC
WV
LV1
YO4
LVTR
YO11
WL2
WR
SQ2
CP2
FR
YO8
YO7
IH
SQ3
SQ1
LV3
LVTR1
WL1
0.0
0.2
0.4
Slope (cm/day per Ly/day)
Population
CP3
BH
FR
YO7
YO8
YO4
SQ3
CP2
WL2
SC
SQ1
WL1
SQ2
WR
LV1
LVTR1
LV3
CC
YO11
IH
DPR
TM2
−0.06
−0.03
0.00
0.03
0.06
Slope (cm/day per Ly/day)