Fix P-Value Heat Maps

Abigail Seeger

January 21, 2021

Contents

The goal of this script is to correct the p-value heat maps - and all heat maps in general - so that the night isn't plotted in the heat map.

Per discussions with Melissa on January 12, I will first correct the p-values with the data that has not been quantile normalized. Then, I will use the same approach to correct the heat maps with the data that has been quantile normalized.

#All Data ##Load and clean data

To begin, load necessary packages:

```
library(dplyr)
library(tidyverse)
library(ggplot2)
### Lemon is used in ggplot2 -
### facet_rep_grid modification
library(lemon)
library(data.table)
library(ggthemes)
library(extrafont)
### Routliers is used for outliersmad to
### find outliers
library(Routliers)
library(stringi)
library(wesanderson)
library(viridis)
```

Next, load in the data.

```
depi_data <- read.table("C:/Users/Owner/Documents/Research/Shiu_Lab/Shiu_Lab_R/Data/DEPI_analysis_Seeger
    sep = ",", header = FALSE)
head(depi_data)
```

```
##                      V1      V2      V3 V4   V5 V6
## 1 0218_F_DEPI_SC_1_10_1 mpk1-16 SH1009b  3 TRUE  1
## 2 0218_F_DEPI_SC_1_10_1 mpk1-16 SH1009b  3 TRUE  1
## 3 0218_F_DEPI_SC_1_10_1 mpk1-16 SH1009b  3 TRUE  1
## 4 0218_F_DEPI_SC_1_10_1 mpk1-16 SH1009b  3 TRUE  1
## 5 0218_F_DEPI_SC_1_10_1 mpk1-16 SH1009b  3 TRUE  1
```

```
## 6 0218_F_DEPI_SC_1_10_1 mpk1-16 SH1009b  3 TRUE  1
##                                 V7                   V8   V9    V10    V11
## 1      0218_F_DEPI_SC_1_10_1_phi2_0 0218_F_DEPI_SC_1_10_1 phi2      0 0.6560
## 2      0218_F_DEPI_SC_1_10_1_phi2_1 0218_F_DEPI_SC_1_10_1 phi2      1 0.6703
## 3      0218_F_DEPI_SC_1_10_1_phi2_2 0218_F_DEPI_SC_1_10_1 phi2      2 0.6676
## 4      0218_F_DEPI_SC_1_10_1_phi2_3 0218_F_DEPI_SC_1_10_1 phi2      3 0.6595
## 5      0218_F_DEPI_SC_1_10_1_phi2_4 0218_F_DEPI_SC_1_10_1 phi2      4 0.6568
## 6 0218_F_DEPI_SC_1_10_1_phi2_5.0006 0218_F_DEPI_SC_1_10_1 phi2 5.0006 0.6580
```

We need to first add column names.

```
names(depi_data) <- c("individual_plant_metadata",
    "genotype", "line", "subline", "border",
    "flat_number", "measurement_ID", "plant_ID",
    "measurement", "time_point", "measured_value")
```

Add a column with the full subline information:

```
indiv_plant_metadata <- read.table("C:/Users/Owner/Documents/Research/Shiu_Lab/Shiu_Lab_R/Data/Individua
    sep = ",", header = FALSE, stringsAsFactors = FALSE)

indiv_plant_metadata <- indiv_plant_metadata %>%
    select(V1, V5) %>% rename(plant_ID = V1,
    full_subline_information = V5)

depi_data <- merge(depi_data, indiv_plant_metadata,
    by = c("plant_ID"))

head(depi_data)
```

```
##               plant_ID individual_plant_metadata genotype    line subline
## 1 0218_F_DEPI_SC_1_10_1     0218_F_DEPI_SC_1_10_1  mpk1-16 SH1009b       3
## 2 0218_F_DEPI_SC_1_10_1     0218_F_DEPI_SC_1_10_1  mpk1-16 SH1009b       3
## 3 0218_F_DEPI_SC_1_10_1     0218_F_DEPI_SC_1_10_1  mpk1-16 SH1009b       3
## 4 0218_F_DEPI_SC_1_10_1     0218_F_DEPI_SC_1_10_1  mpk1-16 SH1009b       3
## 5 0218_F_DEPI_SC_1_10_1     0218_F_DEPI_SC_1_10_1  mpk1-16 SH1009b       3
## 6 0218_F_DEPI_SC_1_10_1     0218_F_DEPI_SC_1_10_1  mpk1-16 SH1009b       3
##   border flat_number                    measurement_ID measurement time_point
## 1   TRUE           1      0218_F_DEPI_SC_1_10_1_phi2_0        phi2          0
## 2   TRUE           1      0218_F_DEPI_SC_1_10_1_phi2_1        phi2          1
## 3   TRUE           1      0218_F_DEPI_SC_1_10_1_phi2_2        phi2          2
## 4   TRUE           1      0218_F_DEPI_SC_1_10_1_phi2_3        phi2          3
## 5   TRUE           1      0218_F_DEPI_SC_1_10_1_phi2_4        phi2          4
## 6   TRUE           1 0218_F_DEPI_SC_1_10_1_phi2_5.0006        phi2     5.0006
##   measured_value full_subline_information
## 1         0.6560                SH1009b-3
## 2         0.6703                SH1009b-3
## 3         0.6676                SH1009b-3
## 4         0.6595                SH1009b-3
## 5         0.6568                SH1009b-3
## 6         0.6580                SH1009b-3
```

Upon investigation, some time points have an "X" in front of them. Remove the "X" in front of these time points.

```
### Proportion of time points that have an
### 'X':
sum(str_detect(depi_data$time_point, "X"))/nrow(depi_data)
```

```
## [1] 0.2832443
```

```
### Rows that have an 'X' in its time
### point:
depi_X <- depi_data[which(str_detect(depi_data$time_point,
    "X")), ]
head(depi_X)
```

```
##                   plant_ID individual_plant_metadata genotype   line subline
## 713408 1217_F_DEPI_SC_1_1_1      1217_F_DEPI_SC_1_1_1     mpk8 SH133P       4
## 713409 1217_F_DEPI_SC_1_1_1      1217_F_DEPI_SC_1_1_1     mpk8 SH133P       4
## 713410 1217_F_DEPI_SC_1_1_1      1217_F_DEPI_SC_1_1_1     mpk8 SH133P       4
## 713411 1217_F_DEPI_SC_1_1_1      1217_F_DEPI_SC_1_1_1     mpk8 SH133P       4
## 713412 1217_F_DEPI_SC_1_1_1      1217_F_DEPI_SC_1_1_1     mpk8 SH133P       4
## 713413 1217_F_DEPI_SC_1_1_1      1217_F_DEPI_SC_1_1_1     mpk8 SH133P       4
##        border flat_number                measurement_ID measurement time_point
## 713408   TRUE           1 1217_F_DEPI_SC_1_1_1_growth_0      growth         X0
## 713409   TRUE           1 1217_F_DEPI_SC_1_1_1_growth_1      growth         X1
## 713410   TRUE           1 1217_F_DEPI_SC_1_1_1_growth_2      growth         X2
## 713411   TRUE           1 1217_F_DEPI_SC_1_1_1_growth_3      growth         X3
## 713412   TRUE           1 1217_F_DEPI_SC_1_1_1_growth_4      growth         X4
## 713413   TRUE           1 1217_F_DEPI_SC_1_1_1_growth_5      growth         X5
##        measured_value full_subline_information
## 713408            224                 SH133P-4
## 713409            223                 SH133P-4
## 713410            227                 SH133P-4
## 713411            231                 SH133P-4
## 713412            241                 SH133P-4
## 713413            252                 SH133P-4
```

```
unique(depi_X$time_point)
```

```
##   [1] "X0"        "X1"        "X2"        "X3"        "X4"        "X5"
##   [7] "X6"        "X7"        "X8"        "X9"        "X10"       "X11"
##  [13] "X12"       "X13"       "X14"       "X15"       "X24"       "X24.5"
##  [19] "X25"       "X25.5"     "X26"       "X26.5"     "X27"       "X27.5"
##  [25] "X28"       "X28.5"     "X29"       "X29.5"     "X30"       "X30.5"
##  [31] "X31"       "X31.5"     "X32"       "X32.4997"  "X33"       "X33.5"
##  [37] "X34"       "X34.5"     "X35"       "X35.5"     "X36.0003"  "X36.5"
##  [43] "X37"       "X37.5"     "X38"       "X38.5"     "X38.9997"  "X39.5"
##  [49] "X48"       "X48.1667"  "X48.5"     "X48.6664"  "X49"       "X49.1667"
##  [55] "X49.5"     "X49.6667"  "X50"       "X50.1664"  "X50.5"     "X50.6667"
##  [61] "X51"       "X51.1667"  "X51.5"     "X51.6667"  "X52"       "X52.1667"
##  [67] "X52.5"     "X52.6667"  "X53"       "X53.1667"  "X53.5"     "X53.6667"
##  [73] "X54"       "X54.1667"  "X54.5"     "X54.6667"  "X55"       "X55.1667"
##  [79] "X55.5"     "X55.6667"  "X56"       "X56.1667"  "X56.5"     "X56.6667"
##  [85] "X57"       "X57.1667"  "X57.5"     "X57.6667"  "X58"       "X58.1664"
##  [91] "X58.5"     "X58.6667"  "X59"       "X59.1667"  "X59.5"     "X59.6667"
```

```
##  [97] "X60"       "X60.1667"  "X60.5"     "X60.6667"  "X61"       "X61.1667"
## [103] "X61.5"     "X61.6667"  "X62"       "X62.1667"  "X62.5"     "X62.6667"
## [109] "X63"       "X63.1667"  "X63.5"     "X63.6667"  "X72"       "X72.9997"
## [115] "X74"       "X75.0003"  "X76"       "X77"       "X78.0003"  "X79"
## [121] "X80"       "X81.0003"  "X82"       "X83"       "X84"       "X85"
## [127] "X86"       "X87"       "X96"       "X96.1667"  "X96.5"     "X96.6669"
## [133] "X97"       "X97.1667"  "X97.5"     "X97.6667"  "X98.0003"  "X98.1669"
## [139] "X98.5"     "X98.6667"  "X99.0003"  "X99.1669"  "X99.5"     "X99.6667"
## [145] "X100"      "X100.1667" "X100.5"    "X100.6669" "X101"      "X101.1669"
## [151] "X101.5"    "X101.6667" "X102.0003" "X102.1669" "X102.5"    "X102.6667"
## [157] "X103.0003" "X103.1667" "X103.5"    "X103.6667" "X104"      "X104.1667"
## [163] "X104.5"    "X104.6669" "X105.0003" "X105.1667" "X105.5"    "X105.6667"
## [169] "X106"      "X106.1667" "X106.5"    "X106.6667" "X107"      "X107.1667"
## [175] "X107.5"    "X107.6667" "X108"      "X108.1667" "X108.5003" "X108.6669"
## [181] "X109.0003" "X109.1667" "X109.4994" "X109.6667" "X110"      "X110.1667"
## [187] "X110.5"    "X110.6667" "X111"      "X111.1669" "X111.5"    "X111.6669"
## [193] "X120"      "X121.0003" "X122.0003" "X123"      "X124"      "X125"
## [199] "X126"      "X127"      "X128"      "X129"      "X130.0003" "X131"
## [205] "X132"      "X133.0003" "X134.0003" "X135"      "X144"      "X145"
## [211] "X146"      "X147"      "X148"      "X149"      "X150"      "X151"
## [217] "X152"      "X153.0003" "X154"      "X155"      "X156"      "X157"
## [223] "X158"      "X159"
```

```
unique(depi_X$genotype)
```

```
##  [1] "mpk8"     "mpk14-20" "mpk14-17" "mpk13"    "mpk3-16"  "mpk9-18"
##  [7] "mpk8-17"  "mpk17"    "mpk6-20"  "b1b3"     "mpk3"     "mpk13-20"
## [13] "ftsz-dbl" "Col0"     "mpk14-16" "mpk9"     "mpk16"    "mpk18-20"
## [19] "mpk6-8"   "ftsz2-2"  "mpk1-16"  "mpk14"    "mpk6"     "mpk1-3"
## [25] "mpk18"    "ftsz2-1"  "mpk5-6"   "b3"       "mpk5-16"  "b1"
## [31] "mpk14-18" "mpk1-18"  "mpk6-18"  "mpk6-9"   "mpk1-17"  "mpk1-14"
## [37] "mpk17-20" "mpk20"    "mpk1"     "mpk1-13"  "mpk8-20"  "mpk5"
## [43] "mpk9-16"
```

```
unique(depi_X$measurement)
```

```
## [1] "growth" "phi2"
```

```
nrow(unique(filter(depi_X, substr(plant_ID,
    1, 4) == "1217")))
```

```
## [1] 351232
```

```
nrow(unique(filter(depi_X, substr(plant_ID,
    1, 4) == "0218")))
```

```
## [1] 0
```

```
### Remove these X values
depi_data$time_point <- as.numeric(gsub("X",
    "", depi_data$time_point))
```

It looks like the X time points area only in December. They are for a range of time points and genotypes. The X only appears in front of time points that are measuring growth and phi2, not npq.

There are some genotypes that are not MPK mutants in this data set. Create a subset of only MPK mutants and wildtype Col0 to use in subsequent analysis. Also, only use subline 1 of Col0, because prior investigations have shown that other sublines may behave differently.

```
depi_subset <- depi_data %>% filter(!genotype %in%
    c("b1", "b3", "b1b3", "ftsz2-1", "ftsz2-2",
        "ftsz-dbl", "Col0") | (genotype ==
    "Col0" & full_subline_information %in%
    c("Col1-1", "Col1-3", "Col1-4", "Col1-2")))

unique(filter(depi_subset, genotype == "Col0")$full_subline_information)
```

```
## [1] "Col1-3" "Col1-1" "Col1-2" "Col1-4"
```

Next, the December collection period has "growth" as a measurement, but the February collection period has "size". These are the same measurement - leaf area. Change both growth and size to leafarea.

```
### The December collection period has
### 'growth', while the February collection
### period has 'size' Both of these
### measurements are recording leaf area,
### so change both to leafarea
levels(depi_subset$measurement)[levels(depi_subset$measurement) ==
    "size"] <- "leafarea"
levels(depi_subset$measurement)[levels(depi_subset$measurement) ==
    "growth"] <- "leafarea"
```

Create dec_data and feb_data from the depi_subset. This will allow us to analyze each collection period seperately.

```
### Create two data sets based on
### collection period and remove border
### plants
feb_data <- filter(depi_subset, substr(plant_ID,
    1, 4) == "0218", border == FALSE)
dec_data <- filter(depi_subset, substr(plant_ID,
    1, 4) == "1217", border == FALSE)
```

Finally, lets look at the genotype we have:

```
unique(feb_data$genotype)
```

```
##  [1] "mpk17"    "mpk20"    "mpk6-18"  "mpk16"    "mpk3"     "Col0"
##  [7] "mpk1-16"  "mpk8-20"  "mpk1-13"  "mpk1"     "mpk17-20" "mpk13-20"
## [13] "mpk14"    "mpk6-8"   "mpk6-9"   "mpk14-17" "mpk5-16"  "mpk9-18"
## [19] "mpk6-20"  "mpk14-16" "mpk8"     "mpk18-20" "mpk1-14"  "mpk9"
## [25] "mpk6"     "mpk5-6"   "mpk14-18" "mpk5-17"  "mpk18"    "mpk8-17"
## [31] "mpk1-3"   "mpk1-18"  "mpk3-16"  "mpk1-17"  "mpk9-16"  "mpk13"
## [37] "mpk5"     "mpk14-20"
```

```
unique(dec_data$genotype)
```

```
##  [1] "mpk6-20"  "mpk3"     "mpk13-20" "Col0"     "mpk14-16" "mpk8"
##  [7] "mpk9"     "mpk18-20" "mpk6-8"   "mpk8-17"  "mpk1-16"  "mpk14"
## [13] "mpk6"     "mpk1-3"   "mpk18"    "mpk17"    "mpk16"    "mpk5-6"
## [19] "mpk14-18" "mpk14-20" "mpk1-18"  "mpk13"    "mpk6-18"  "mpk1-14"
## [25] "mpk17-20" "mpk20"    "mpk1"     "mpk3-16"  "mpk6-9"   "mpk1-17"
## [31] "mpk5"     "mpk1-13"  "mpk8-20"  "mpk14-17" "mpk9-16"  "mpk5-16"
## [37] "mpk9-18"
```

It looks like they have the same genotypes, except the February experiment has mpk5-17 while the December experiment does not.

##Create functions

These functions will be applied to the December and February data seperately.

###add_day_col

The add_day_col function adds a column with the day of the experiment to the data frame. This function first looks for breaks in the data - data was only collected when the lights were on, so a break of greater than five hours indicates a change of day. Then, a number is assigned to indicate the day of the experiment.

```
add_day_col <- function(data_frame) {
    unique_time <- sort(unique(data_frame$time_point))
    diff <- c()
    for (i in 1:length(unique_time)) {
        if (i == 1) {
            diff[1] <- 0
        } else {
            diff[i] <- unique_time[i] - unique_time[i -
                1]
        }
    }
    breaks <- c(0)
    for (i in 1:length(diff)) {
        if (diff[i] > 5)
            breaks <- append(breaks, unique_time[i])
    }
    out <- data.frame()
    for (i in 1:length(breaks)) {
        if (i == length(breaks)) {
            indiv <- data_frame %>% filter(time_point >=
                breaks[i]) %>% mutate(day = i)
        } else {
            indiv <- data_frame %>% filter(time_point >=
                breaks[i] & time_point <
                breaks[i + 1]) %>% mutate(day = i)
        }

        out <- rbind(as.data.frame(indiv),
            out)
    }
    return(out)
}
```

###remove_outliers

The remove_outliers function replaces measured values that are outliers with NA. Because we will analyze the data using nonparametric analysis, this function will not be applied to the data, because outliers have little influence on the median value.

First, group by genotype, measurement type, and time point. Once we've focused in on this, conduct the outliers_mad test using a conservative threshold value of 3.5. If the outliers_mad function finds an outlier, use the position of the outlier to replace the measured value with NA.

```
remove_outliers <- function(df) {
    out <- df %>% group_by(genotype, measurement,
        time_point) %>% mutate(measured_value = replace(measured_value,
        outliers_mad(measured_value, b = 1.4826,
            threshold = 3.5, na.rm = TRUE)$outliers_pos,
        NA)) %>% arrange(genotype, measurement,
        time_point)
    return(out)
}
```

###p_value

The p_value function finds the p-value for the comparison of each genotype to wildtype in order to answer the question - does the phenotype (either leaf area, npq, or phi2) significantly differ from wild type? The test is repeated for each time point for each genotype.

More information on the specifics used in the Wilcox test can be found here and here.

http://courses.atlas.illinois.edu/spring2016/STAT/STAT200/RProgramming/NonParametricStats.html and https://data.library.virginia.edu/the-wilcoxon-rank-sum-test/

The function is heavily commented, explaining each step in the pipeline.

```
p_value <- function(data_frame) {
    ### Initialize an empty data frame
    out = data.frame()
    ### For each genotype:
    for (i in unique(filter(data_frame, genotype !=
        "Col0")$genotype)) {
        ### We don't want to make comparisons of WT
        ### to itself - this could impact FDR
        ### correction
        indiv_data <- data_frame %>% ### Focus on each time point and
        ### measurement
        group_by(time_point, measurement) %>%
            ### Create a column with the number of WT
        ### individual plants and the number of
        ### plants for each genotype Use this later
        ### to calculate effect size
        mutate(n_genotype = length(measured_value[genotype ==
            i]), n_wt = length(measured_value[genotype ==
            "Col0"])) %>% ### Create a column of p-values using a
        ### nonparametric Wilcox test

        ### Default set to exact = TRUE, because
        ### our sample sizes are too small to use a
        ### normal approximation But, when there
```

```
        ### are ties in the values (i.e. one value
        ### appears twice in the ranking process),
        ### wilcox.test returns to the normal
        ### approximation and spits out a warning
        ### message This may be a problem - include
        ### correct = FALSE to stop this from
        ### happening


        ### Paired = FALSE, because the Col0 plants
        ### are independet from each genotype
        ### Correct = FALSE turns off the
        ### continuity correction
        mutate(p = (wilcox.test(measured_value[genotype ==
            i], measured_value[genotype ==
            "Col0"], correct = FALSE, paired = FALSE))$p.value) %>%

        ### Add a column with each genotype
        mutate(genotype = i) %>% # mutate(number = unique(filter(feb_data,
        # genotype == i)$number))%>%
        # mutate(number_2 =
        # unique(filter(feb_data, genotype ==
        # i)$number_2))%>%
        select(time_point, genotype, measurement,
            day, p, n_wt, n_genotype)


        ### Add individual information to the main
        ### data frame
        out <- rbind(as.data.frame(indiv_data),
            out)
    }
    return(out)
}
```

###corrected_p_value

The corrected_p_value function corrects the p-values using an FDR correction. First, the data is grouped by measurement and time point. Then, the p-value are corrected by the number of genotypes. This function also computes effect size.

https://stats.stackexchange.com/questions/133077/effect-size-to-wilcoxon-signed-rank-test

Once again, the function is heavily commented, explaining each step of the pipeline.

```
corrected_p_value <- function(data_frame){
  out <- data_frame%>%
  ###For some reason, I have multiple copies of each row
  distinct()%>%
  ###Group by time point and measurement - we are correcting by the number of genotypes
  group_by(time_point, measurement)%>%
  mutate(p_adj = p.adjust(p, method = "fdr"))%>%

  ###Only report an effect size if the p-value is significant; otherwise, NA
  mutate(effect = ifelse(p < 0.05, (abs(qnorm(p_adj))/sqrt(n_wt+n_genotype)), NA))%>%
```

```
  mutate(effect_size = case_when(
    ###Make sure these are the right cut offs for magnitude of effect size
    (effect <0.1)~"small",
    (effect>0.1 & effect < 0.5)~"medium",
    (effect>0.5)~"large"))
  ###Gather the data to make it easier to plot according to whether p was adjusted
  out <- gather(out, type, p, p, p_adj)%>%arrange(genotype, time_point)
  return(out)
}
```

###add_number

The add_number function adds two columns - number and number_2. number is used to sort the heat maps that include all genotypes, while number_2 is used to sort the heat maps that are a trio of double and single mutants.

The code is not intuitive - it is commented throughout to explain each step.

```
add_number <- function(data_frame) {
    ### First, if the genotype is Col0 (only
    ### genotype with length 4), assign 0 as
    ### number Else, assign number as genotype
    ### with 'mpk' removed Example: mpk1 will
    ### be 1, mpk1-17 will be 1-17
    data_frame <- data_frame %>% mutate(number = ifelse(genotype !=
        "Col0", (stri_sub(genotype, 4, length(genotype))),
        0))
    ### Next, for all double mutants, replace
    ### '-' with '0' Example: 1-17 becomes 1017
    data_frame$number <- as.numeric(gsub("-",
        "0", data_frame$number))
    ### Almost there! There's a problem with
    ### two single digit double mutants We need
    ### a four digit number to sort correctly
    ### Example: mpk1-3 -> 1-3 -> 103, but we
    ### need it to be 1003 to sort correctly
    data_frame$number[data_frame$number ==
        "103"] <- "1003"
    data_frame$number[data_frame$number ==
        "506"] <- "5006"
    data_frame$number[data_frame$number ==
        "608"] <- "6008"
    data_frame$number[data_frame$number ==
        "609"] <- "6009"
    ### Convert number to a numberic in order
    ### to sort
    data_frame$number <- as.numeric(data_frame$number)
    data_frame <- data_frame %>% arrange(number)
    data_frame <- data_frame %>% mutate(number_2 = number)
    data_frame$number_2[nchar(data_frame$number_2) ==
        4] <- 0
    data_frame$number_2[nchar(data_frame$number_2) ==
        5] <- 0
    return(data_frame)
}
```

###cell_370_data

The cell_370_data function prepares the data to use in visualizations mirrored after page 370 of the Cell paper Dynamic Environmental Photosynthetic Imaging Reveals Emergent Phenotypes.

This function groups the data by genotype, and for each time point and each measurement finds the median measured value of the plants. Because of how variable the leaf area measurements are between time points, only the beginning and end of each day will be plotted.

```
cell_370_data <- function(data_frame) {
    npq_phi2 <- data_frame %>% filter(measurement %in%
        c("npq", "phi2")) %>% group_by(genotype,
        time_point, measurement) %>% summarize(med = median(measured_value))
    ### For each day, we want the minimum and
    ### maximum time point for leaf area

    ### Previously included the midpoint -
    ### leave code in case we want to use in
    ### the future, just commented out

    ### If there are an odd number of time
    ### points, use the median time point If
    ### there are an even number of time
    ### points, instead of finding the average
    ### of the two center values, choose the
    ### larger value start_mid_end <-
    ### unique((data_frame%>%group_by(day)%>%filter(time_point
    ### %in% c(min(time_point),
    ### max(time_point),
    ### ifelse(length(time_point %% 2 == 0),
    ### median(time_point[-1]),
    ### median(time_point)))))$time_point)

    start_end <- unique((data_frame %>% group_by(day) %>%
        filter(time_point %in% c(min(time_point),
            max(time_point))))$time_point)

    leaf_area <- data_frame %>% filter(measurement ==
        "leafarea") %>% filter(time_point %in%
        start_end) %>% group_by(genotype,
        time_point, measurement) %>% summarize(med = median(measured_value))

    out <- rbind(npq_phi2, leaf_area) %>%
        group_by(genotype, time_point, measurement)

    return(as.data.frame(out))
}
```

###cell_371_data

The cell_371_data_function prepares the data to use to create visualizations mirrored after page 371 in the Cell paper Dynamic Environmental Photosynthetic Imaging Reveals Emergent Phenotypes.

This function:

- groups the data by time point and measurement and genotype
- divides the measured value of each genotype by the median wild type value
- calculates the log 2 value for the median of these values

Similar to above, we are only interested in the start and end time points for the leaf area measurement.

```
cell_371_data <- function(data_frame) {

    npq_phi2 <- data_frame %>% filter(measurement %in%
        c("npq", "phi2")) %>% group_by(time_point,
        measurement) %>% mutate_each(funs(./median(.[genotype ==
        "Col0"])), measured_value) %>% group_by(time_point,
        measurement, genotype) %>% mutate(log2_fold = log2(median(measured_value)))

    # start_mid_end <-
    # unique((data_frame%>%group_by(day)%>%filter(time_point
    # %in% c(min(time_point),
    # max(time_point),
    # ifelse(length(time_point %% 2 == 0),
    # median(time_point[-1]),
    # median(time_point))))$time_point)
    start_end <- unique((data_frame %>% group_by(day) %>%
        filter(time_point %in% c(min(time_point),
            max(time_point))))$time_point)

    leaf_area <- data_frame %>% filter(measurement ==
        "leafarea") %>% filter(time_point %in%
        start_end) %>% group_by(time_point,
        measurement) %>% mutate_each(funs(./median(.[genotype ==
        "Col0"])), measured_value) %>% group_by(time_point,
        measurement, genotype) %>% mutate(log2_fold = log2(median(measured_value)))

    out <- rbind(npq_phi2, leaf_area) %>%
        group_by(genotype, time_point, measurement)

    return(as.data.frame(out))
}
```

###Genotype combinations

Finally, define a list of all combinations of single and double mutants. We will use this multiple times in the following analysis to loop through each combination to create plots.

```
genotype_combinations <- list(c("mpk1-17",
    "mpk1", "mpk17"), c("mpk1-16", "mpk1",
    "mpk16"), c("mpk6-9", "mpk6", "mpk9"),
    c("mpk17-20", "mpk17", "mpk20"), c("mpk14-17",
        "mpk14", "mpk17"), c("mpk8-17", "mpk8",
        "mpk17"), c("mpk8-20", "mpk8", "mpk20"),
    c("mpk6-18", "mpk6", "mpk18"), c("mpk1-13",
        "mpk1", "mpk13"), c("mpk17-20", "mpk17",
        "mpk20"), c("mpk13-20", "mpk13",
        "mpk20"), c("mpk6-8", "mpk6", "mpk8"),
    c("mpk9-18", "mpk9", "mpk18"), c("mpk6-20",
```

```
        "mpk6", "mpk20"), c("mpk14-16", "mpk14",
        "mpk16"), c("mpk18-20", "mpk18",
        "mpk20"), c("mpk5-6", "mpk5", "mpk6"),
    c("mpk14-18", "mpk14", "mpk18"), c("mpk5-6",
        "mpk5", "mpk6"), c("mpk14-18", "mpk14",
        "mpk18"), c("mpk5-17", "mpk5", "mpk17"),
    c("mpk1-3", "mpk1", "mpk3"), c("mpk1-17",
        "mpk1", "mpk17"), c("mpk3-16", "mpk3",
        "mpk16"), c("mpk9-16", "mpk9", "mpk16"),
    c("mpk14-20", "mpk14", "mpk20"))
```

#December Analysis ##Further Cleaning Here, we repeat the analysis used in February for December.

Here is a summary of the measured values.

```
summary(dec_data$measured_value)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.     NA's
##  -0.1628   0.4779   0.6865  91.8606 208.0000 781.0000        2
```

```
dec_data %>% group_by(measurement) %>% summarize(min = min(measured_value))
```

```
## # A tibble: 3 x 2
##   measurement    min
##   <chr>        <dbl>
## 1 growth       5
## 2 npq         -0.163
## 3 phi2        NA
```

It looks like we've run into a few problems. First, npq values are negative, yet we know they should all be positive. And, for some reason we have NA values in phi2.

To address these problems, start by shifting all of the npq values by the minimum value.

```
dec_data$measured_value[dec_data$measurement ==
    "npq"] <- (dec_data$measured_value[dec_data$measurement ==
    "npq"]) + abs(min((filter(dec_data, measurement ==
    "npq"))$measured_value))
```

Next, address the problems with phi2.

```
### Here is the percentage of negative phi2
### values:
nrow(filter(dec_data, measurement == "phi2",
    measured_value < 0))/nrow(dec_data) *
    100
```

```
## [1] 0.0006265664
```

```
### Here are the rows with NA for a
### measured value:
dec_data[which(is.na(dec_data$measured_value)),
    ]
```

```
##                    plant_ID individual_plant_metadata genotype   line subline
## 11268 1217_F_DEPI_SC_1_12_4     1217_F_DEPI_SC_1_12_4    mpk18 SH139P       3
## 11273 1217_F_DEPI_SC_1_12_4     1217_F_DEPI_SC_1_12_4    mpk18 SH139P       3
##       border flat_number                      measurement_ID measurement
## 11268  FALSE           1 1217_F_DEPI_SC_1_12_4_phi2_52.6667        phi2
## 11273  FALSE           1      1217_F_DEPI_SC_1_12_4_phi2_54        phi2
##       time_point measured_value full_subline_information
## 11268    52.6667             NA                 SH139P-3
## 11273    54.0000             NA                 SH139P-3
```

```
### Because only two rows have negative
### phi2 values, remove these rows
dec_data <- na.omit(dec_data)
### Now, shift all phi2 values by the
### minimum measured_value
dec_data$measured_value[dec_data$measurement ==
    "phi2"] <- (dec_data$measured_value[dec_data$measurement ==
    "phi2"]) + abs(min((filter(dec_data,
    measurement == "phi2"))$measured_value))
```

Before moving forward, reexamine the summary of measured values.

```
summary(dec_data$measured_value)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##   0.0000   0.5924   0.8132  91.9499 208.0000 781.0000
```

```
dec_data %>% group_by(measurement) %>% summarize(min = min(measured_value))
```

```
## # A tibble: 3 x 2
##   measurement   min
##   <chr>       <dbl>
## 1 growth          5
## 2 npq             0
## 3 phi2            0
```

Everything looks good!! We can continue.

Next, add a column with the day to the data, and columns with number and number_2 that we will use later to sort visualizations.

```
dec_data <- add_day_col(dec_data)
```

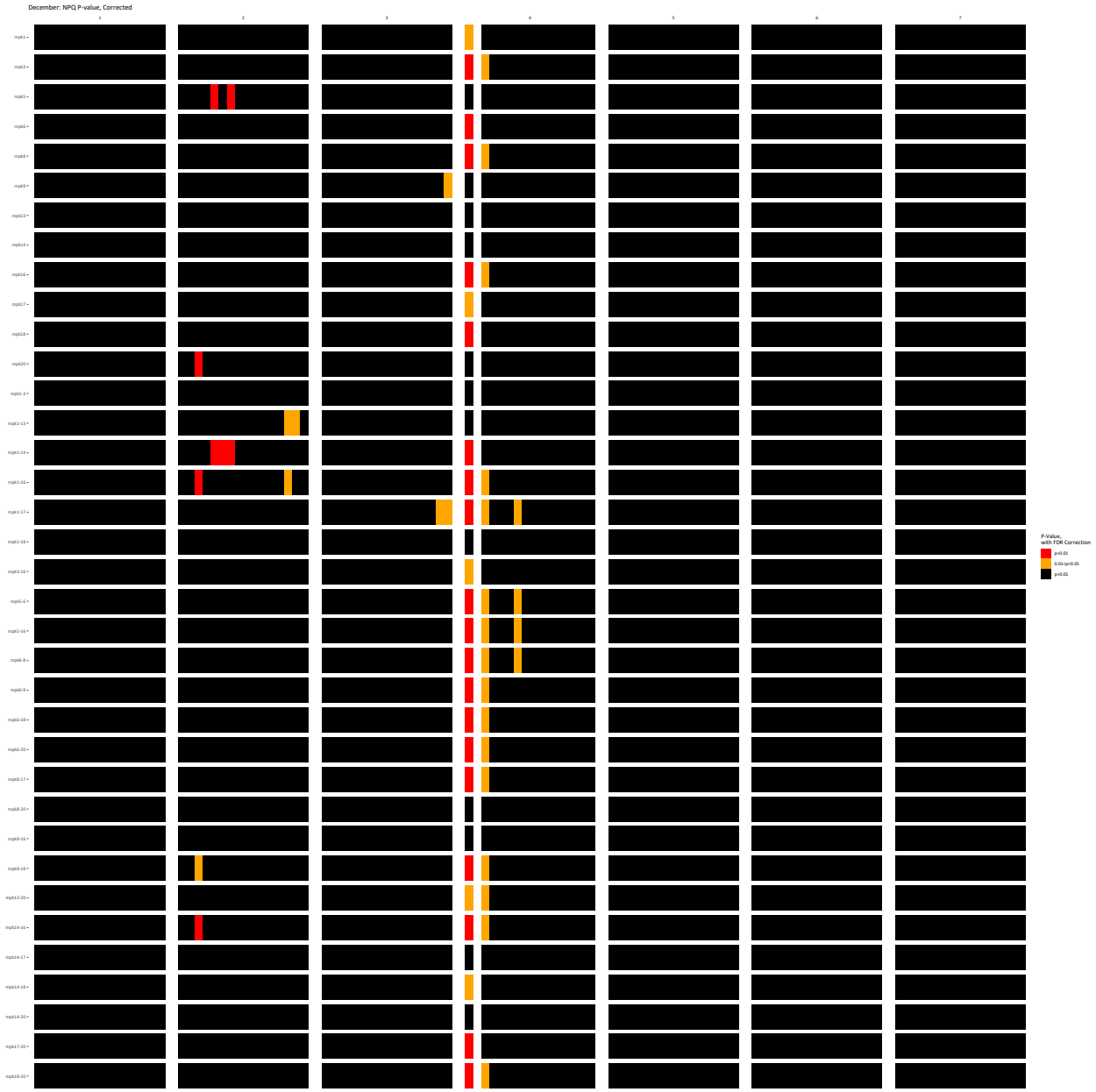
```
dec_data_p <- p_value(dec_data)
dec_data_p_corrected <- corrected_p_value(dec_data_p)
dec_data_plot <- add_number(dec_data_p_corrected)
```

```
### Create seperate data frames for each
### measurement to use in heat maps; only
### use adjusted p-values
dec_npq_adj <- filter(dec_data_plot, measurement ==
    "npq", type == "p_adj") %>% ### Create a new column with the p-value
### 'bins'
mutate(bin = case_when((p < 0.01) ~ "p<0.01",
    (p > 0.01 & p < 0.05) ~ "0.01<p<0.05",
    (p > 0.05) ~ "p>0.05"))
dec_phi2_adj <- filter(dec_data_plot, measurement ==
    "phi2", type == "p_adj") %>% mutate(bin = case_when((p >
    0.01 & p < 0.05) ~ "0.01<p<0.05", (p >
    0.05) ~ "p>0.05"))
```

```
### In order to use these bins as the fill
### in a heat map, convert to a factor
dec_npq_adj$bin <- as.factor(dec_npq_adj$bin)
dec_phi2_adj$bin <- as.factor(dec_phi2_adj$bin)
### We want p<0.1 to be first in the
### legend, so refactor with p<0.01 as the
### first term
dec_npq_adj$bin <- relevel(dec_npq_adj$bin,
    "p<0.01")
dec_phi2_adj$bin <- relevel(dec_phi2_adj$bin,
    "0.01<p<0.05")
### Reorder by number so heat map has WT
### first, then single, then double mutants
dec_npq_adj$genotype <- reorder(dec_npq_adj$genotype,
    dec_npq_adj$number)
dec_phi2_adj$genotype <- reorder(dec_phi2_adj$genotype,
    dec_phi2_adj$number)
```

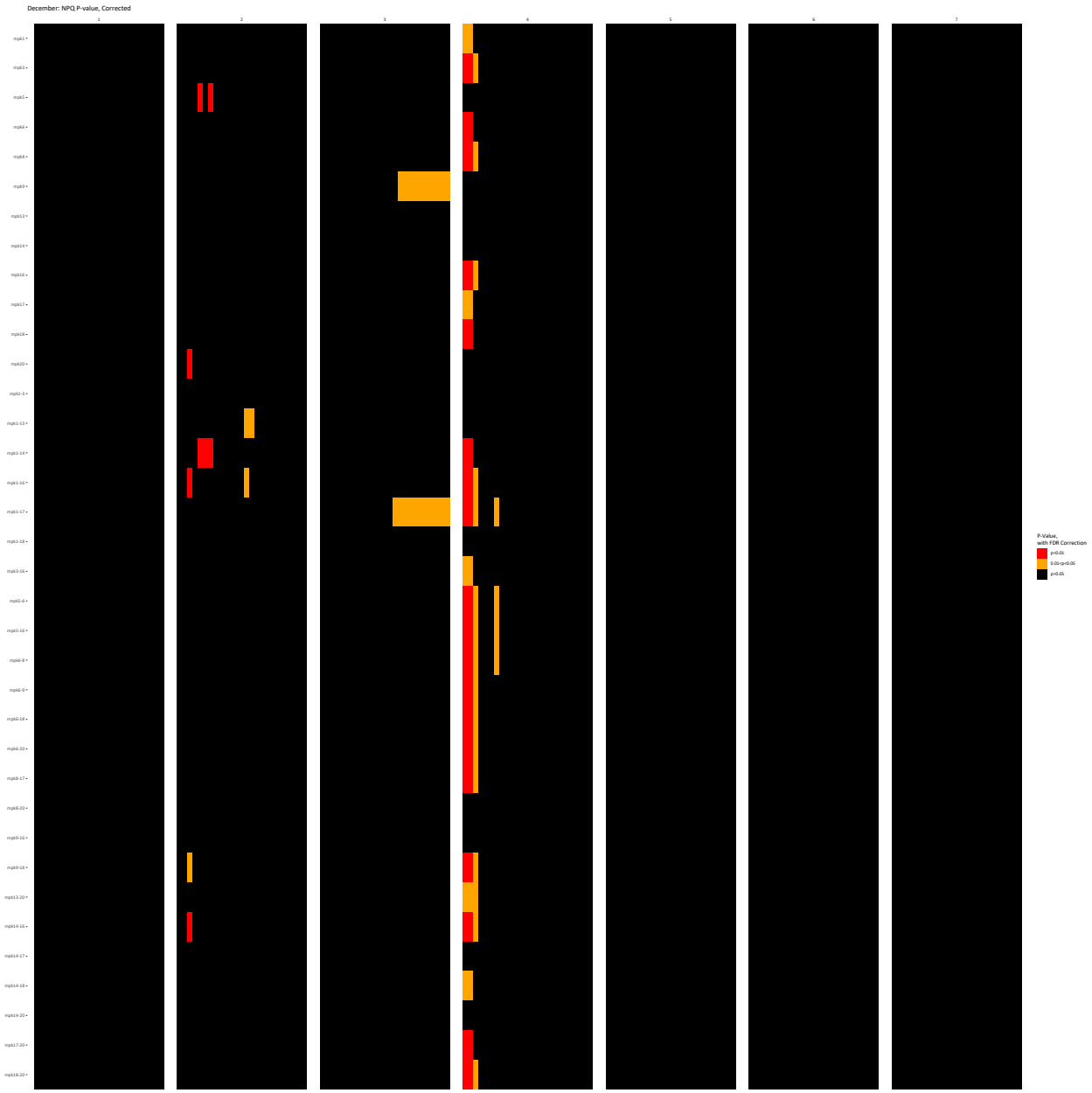
This is the INCORRECT heat map:

```
##### ----- Dec p-value heat map - NPQ ------
ggplot(data = dec_npq_adj, aes(x = time_point,
    y = genotype, fill = bin)) + labs(fill = "P-Value, \nwith FDR Correction",
    x = "Hours", y = NULL, title = "December: NPQ P-value, Corrected") +
    geom_tile(width = 1, height = 1) + facet_grid(genotype ~
    day, scales = "free", switch = "y") +
    # scale_x_continuous(breaks =
# round(c(0,15,24,39.5,48,63.7,72,87,96,112,
# 120,135,144,159,168,183,192,207,216,231,240,255,264,279),0))+
theme_tufte(base_family = "Calibri", base_size = 10) +
    theme(strip.background.y = element_blank(),
        strip.text.y = element_blank(), axis.title.x = element_blank(),
        axis.text.x = element_blank(), axis.ticks.x = element_blank(),
        panel.spacing = unit(0, "lines")) +
    scale_fill_manual(values = c("red", "orange",
        "black"))
```




```
# ggsave('dec_npq_corrected.png', scale =
# 2, path = 'C:/Users/Joan
# Seeger/Documents/Shiu Lab -
# R/Current/Visualizations')

##### ----- Dec p-value heat map - NPQ ------
ggplot(data = dec_npq_adj, aes(x = time_point,
    y = genotype, fill = bin)) + labs(fill = "P-Value, \nwith FDR Correction",
    x = "Hours", y = NULL, title = "December: NPQ P-value, Corrected") +
    geom_tile(width = 10, height = 20) +
    facet_grid(genotype ~ day, scales = "free",
        switch = "y") + # scale_x_continuous(breaks =
# round(c(0,15,24,39.5,48,63.7,72,87,96,112,
# 120,135,144,159,168,183,192,207,216,231,240,255,264,279),0))+
```

```
theme_tufte(base_family = "Calibri", base_size = 10) +
    theme(strip.background.y = element_blank(),
        strip.text.y = element_blank(), axis.title.x = element_blank(),
        axis.text.x = element_blank(), axis.ticks.x = element_blank(),
        panel.spacing = unit(0, "lines")) +
    scale_fill_manual(values = c("red", "orange",
        "black"))
```




```
# ggsave('dec_npq_corrected.png', scale =
# 2, path = 'C:/Users/Joan
# Seeger/Documents/Shiu Lab -
# R/Current/Visualizations')
```

This is the CORRECT heat map:

First, create a new data frame for these plots. This is because the time points for day 4 skips a time point, results in a column of all white:

```
unique(filter(dec_data, dec_data$day == 4)$time_point)
```

```
##  [1] 72.0000 74.0000 75.0000 76.0000 77.0000 78.0000 79.0000 80.0000 81.0000
## [10] 82.0000 83.0000 84.0000 85.0000 86.0000 87.0000 72.9997 75.0003 78.0003
## [19] 81.0003
```

So, for time point 73, I need to include a column of gray, to show that this data is missing, and isn't denoting a change in day.

To do this, I will introduce dummy data for this time point for each genotype, and specify this to be coded as gray.

I will let this dummy value be "No Data". This is convinient, because it will show in the legend.

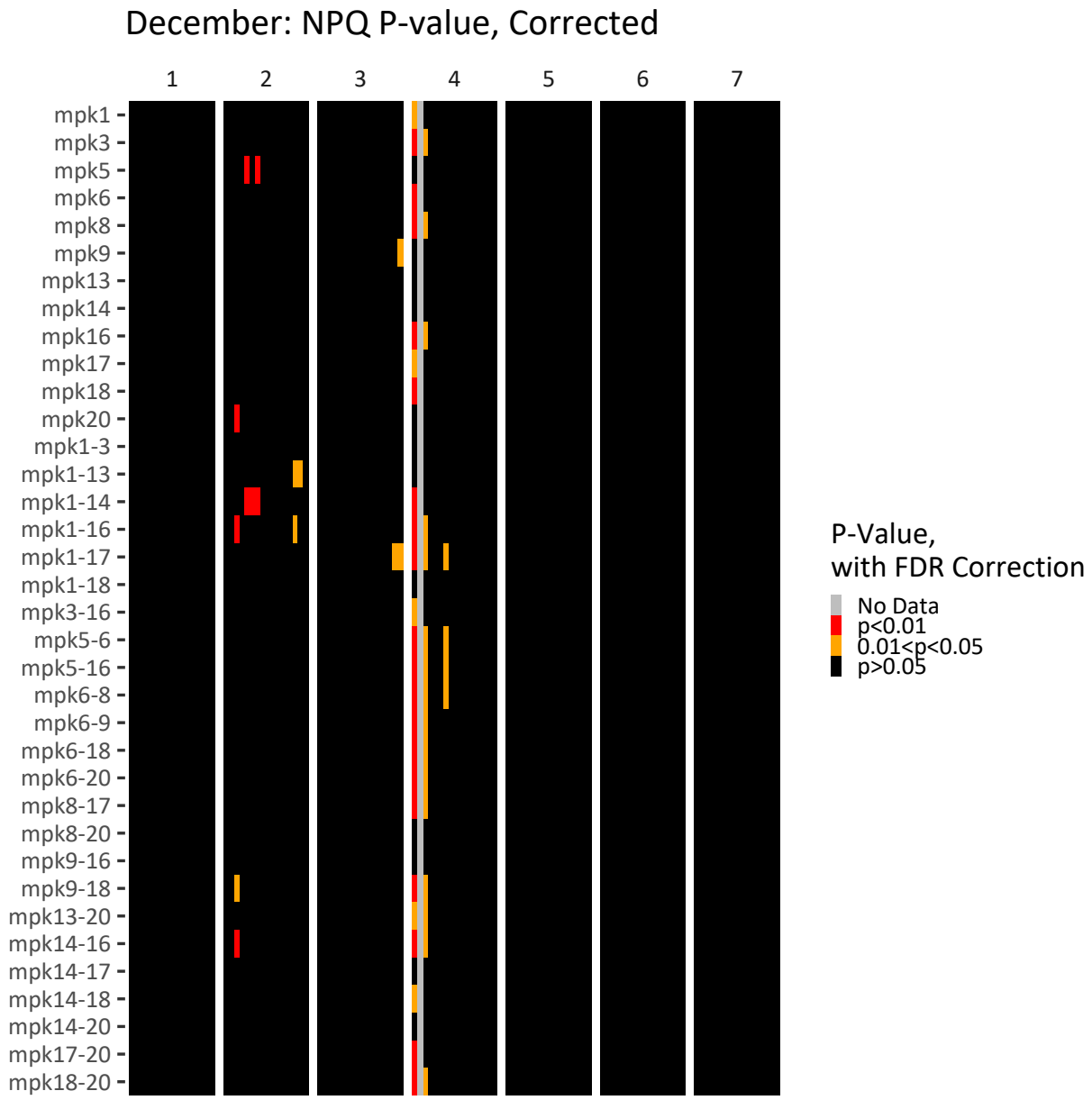
```
final_plot_data <- dec_npq_adj

dummy_data_frame <- data.frame(time_point = rep(73, 36),
                               genotype = unique(dec_npq_adj$genotype),
                               measurement = rep("npq", 36),
                               day = rep(4, 36),
                               n_wt = rep(NA, 36),
                               n_genotype = rep(NA, 36),
                               effect = rep(NA, 36),
                               effect_size = rep(NA, 36),
                               type = rep("p_adj", 36),
                               p = rep(2, 36),
                               number = (dec_npq_adj%>%filter(time_point == 1))$number,
                               number_2 = rep(NA, 36),
                               bin = rep("No Data", 36)

                               )

final_plot_data <- rbind(final_plot_data, dummy_data_frame)
```

```
final_plot_data$bin <- as.factor(final_plot_data$bin)

final_plot_data$bin <- relevel(final_plot_data$bin,
    "p<0.01", first = TRUE)
final_plot_data$bin <- relevel(final_plot_data$bin,
    "No Data", first = FALSE)
```

```
plot_data <- dec_data %>% filter(genotype %in%
    c("mpk1-17", "mpk1", "mpk17", "Col0"),
    measurement == "npq") %>% group_by(genotype,
    time_point) %>% summarize(med = median(measured_value))

plot <- ggplot(data = plot_data, aes(x = time_point,
    y = med)) + geom_line(aes(color = genotype),
    size = 1) + # facet_rep_grid(~ day, scales = 'free' ,
# switch = 'y', repeat.tick.labels =
```

```
# FALSE)+
labs(x = "Hours", y = NULL) + theme_tufte(base_family = "Calibri",
    base_size = 20) + theme(strip.background.x = element_blank(),
    axis.title.x = element_blank(), axis.text.x = element_blank(),
    axis.ticks.x = element_blank(), panel.border = element_rect(color = "black",
        fill = NA, size = 1), axis.line = element_line(),
    panel.spacing = unit(1, "lines")) + # scale_x_continuous(breaks
# round(c(0,15,24,39.5,48,63.7,72,87,96,112,120,135,144,
# 159,168,183,192,207,216,231,240,255,264,279),0))+
scale_color_viridis_d(begin = 0, end = 1,
    option = "viridis", aesthetics = c("colour",
        "fill"))
```

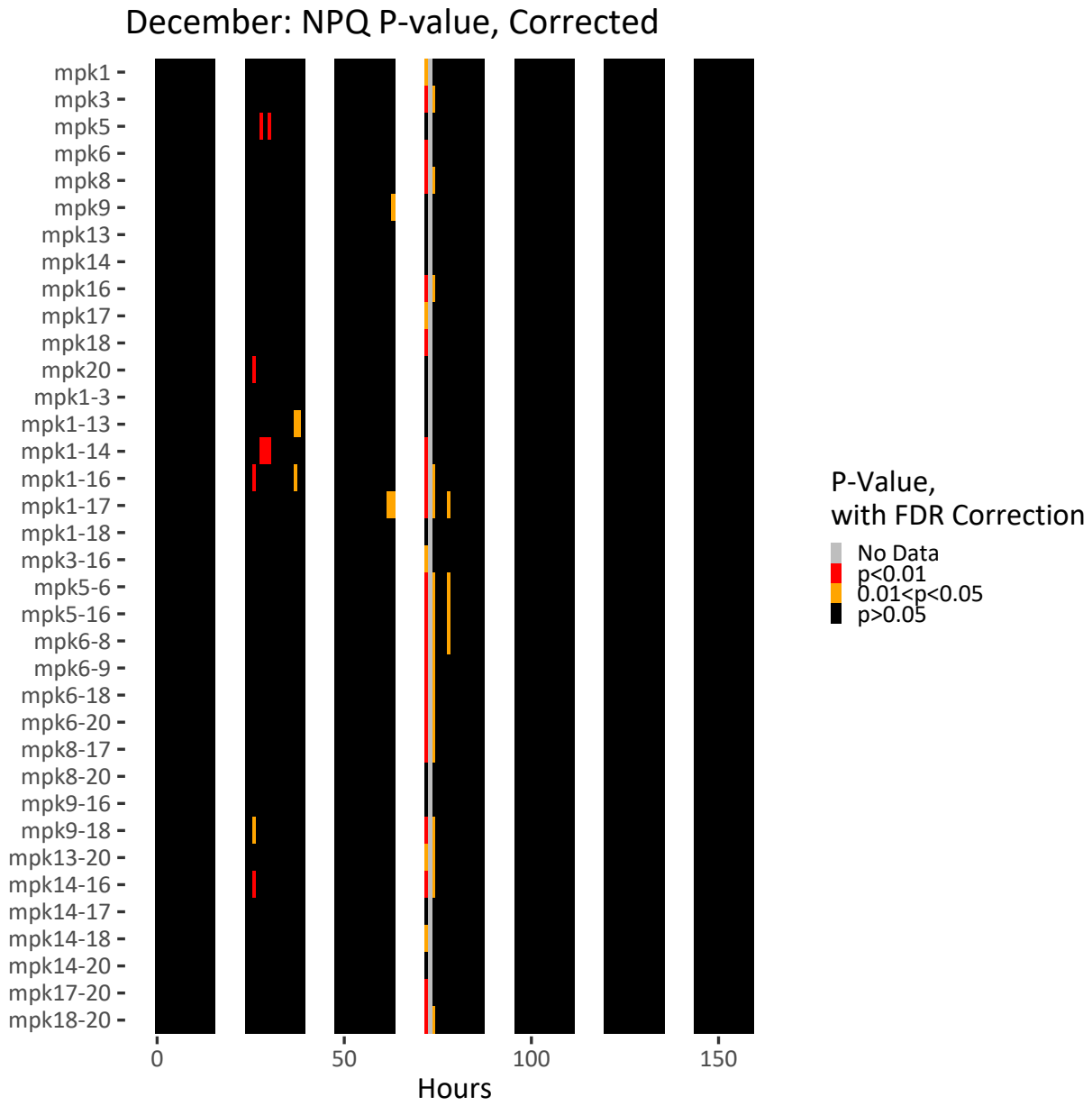
```
##### ----- Dec p-value heat map - NPQ ------
ggplot(data = final_plot_data, aes(x = time_point,
    y = genotype, fill = bin)) + labs(fill = "P-Value, \nwith FDR Correction",
    x = "Hours", y = NULL, title = "December: NPQ P-value, Corrected") +
    geom_tile(height = 2, width = 1.1) +
    facet_grid(genotype ~ day, scales = "free",
        switch = "y") + theme_tufte(base_family = "Calibri",
    base_size = 50) + theme(strip.background.y = element_blank(),
    strip.text.y = element_blank(), axis.title.x = element_blank(),
    axis.text.x = element_blank(), axis.ticks.x = element_blank(),
    panel.spacing = unit(0, "lines")) + scale_fill_manual(values = c("gray",
    "red", "orange", "black"))
```




```
# ggsave('Dec_NPQ_Correct_No_QN.png',
# scale = 2, path =
# 'C:/Users/Owner/Documents/Research/Shiu_Lab/Shiu_Lab_R/Updated_Plots',
# limitsize = FALSE)
```

```
final_plot_data_2 <- final_plot_data
final_plot_data_2$genotype <- reorder(final_plot_data_2$genotype,
    final_plot_data_2$number * -1)

ggplot(data = final_plot_data_2, aes(x = time_point,
    y = genotype, fill = bin)) + geom_tile(height = 1,
    width = 1.1) + scale_fill_manual(values = c("gray",
    "red", "orange", "black")) + theme_tufte(base_family = "Calibri",
    base_size = 50) + labs(fill = "P-Value, \nwith FDR Correction",
```

```
    x = "Hours", y = NULL, title = "December: NPQ P-value, Corrected")
```




```
# ggsave('Dec_NPQ_Correct_No_QN_V2.png',
# scale = 2, path =
# 'C:/Users/Owner/Documents/Research/Shiu_Lab/Shiu_Lab_R/Updated_Plots',
# limitsize = FALSE)
```