# Quantile Normalization

Abigail Seeger

August 6, 2020

## Contents



The objective of quantile normalization is to normalize the data to account for technical differences.

Example: One lightbulb is better than another lightbulb used in a different micro-array experiment. So, every measurement might be brighter than every measurement from another experiment.

In this case, the technical differences are a result of machinery associated with the DEPI chamber. **This has nothing to do with biology!**

Here are the resources that I used to research quantile normalization:

https://www.youtube.com/watch?v=ecjN6Xpv6SE

How to implement quantile normalization in R:

https://davetang.org/muse/2014/07/07/quantile-normalisation-in-r/

**After looking into how this is done, I suspect I may run into a problem with experiments that have a different number of replicates per genotype!!**

### Read in the Data

Load necessary packages.

```
library(dplyr)
library(tidyverse)
library(reshape2)
library(ggthemes)
library(extrafont)
library(sm)
```

## Create Data Frame

```
### Read in the data
depi_jan <- read.table("C:/Users/Owner/Documents/Research/Shiu_Lab/Shiu_Lab_R/Data/Correct_January_Crea
    sep = ",", header = TRUE, stringsAsFactors = FALSE)
depi_dec_feb <- read.table("C:/Users/Owner/Documents/Research/Shiu_Lab/Shiu_Lab_R/Data/DEPI_analysis_Se
    sep = ",", header = FALSE)
depi_jan_growth <- read.table("C:/Users/Owner/Documents/Research/Shiu_Lab/Shiu_Lab_R/Data/Jan_Growth_Da
    sep = ",", header = TRUE, stringsAsFactors = FALSE)
### Add column names
names(depi_jan) <- c("measurement_ID", "plant_ID",
    "DEPI_ID", "time_point", "measured_value",
    "light_regimen", "measurement", "individual_plant_metadata",
    "genotype", "line", "subline", "full_subline_information",
    "experiment_number", "flat_number", "cell_number",
    "row_number", "column_number", "border",
    "treatment")
names(depi_dec_feb) <- c("individual_plant_metadata",
    "genotype", "line", "subline", "border",
    "flat_number", "measurement_ID", "plant_ID",
    "measurement", "time_point", "measured_value")
names(depi_jan_growth) <- c("individual_plant_metadata",
    "genotype", "line", "subline", "full_subline_information",
    "experiment_number", "flat_number", "cell_number",
    "row_number", "column_number", "border",
    "treatment", "measurement_ID", "plant_ID",
    "DEPI_ID", "time_point", "measured_value",
    "light_regimen", "measurement")

depi_jan <- rbind(depi_jan_growth, depi_jan)
```

**STOP! At this point, filter the correct Col0 sublines!**

I don't have the column that I need in the depi_dec_feb. But, I can add a column based on the information in the individual_plant_metadata file.

```
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_dec_feb <- merge(depi_dec_feb, indiv_plant_metadata,
    by = c("plant_ID"))

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

```
## [1] "Col1-3" "Col1-1" "Col2-3" "Col2-1" "Col1-2" "Col2-2" "Col2-4" "Col1-4"
```

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

```
## [1] "Col1_1" "Col1_3" "Col2_2" "Col2_4" "Col1_4" "Col2_1" "Col1_2" "Col2_3"
```

Now, I want to filter to only include Col1-X sublines!! This is different from selecting subline 1 from the subline column!!

Create subsets of the Feb and Dec data. Even though I will eventually re-combine these subsets to create on data frame, this will make it easier to shift the NPQ values so the minimum is 0, and address NA and negative phi2 values for each experiment.

Next, select only the individual plant metadata, genotype, flat number, measurement, time point, measured value, and border columns, and add a column with the month of the experiment.

```
### Select only the columns that we need
### and add a column with the month
dec_data <- depi_dec_feb %>% 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"))) %>%
    filter(substr(plant_ID, 1, 4) == "1217") %>%
    filter(border == FALSE) %>% select(individual_plant_metadata,
    genotype, flat_number, measurement, time_point,
    measured_value, border, subline, full_subline_information) %>%
    mutate(month = "Dec")

jan_data <- depi_jan %>% 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"))) %>%
    filter(border == FALSE) %>% select(individual_plant_metadata,
    genotype, flat_number, measurement, time_point,
    measured_value, border, subline, full_subline_information) %>%
    mutate(month = "Jan")

feb_data <- depi_dec_feb %>% 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"))) %>%
    filter(border == FALSE) %>% filter(substr(plant_ID,
    1, 4) == "0218") %>% select(individual_plant_metadata,
    genotype, flat_number, measurement, time_point,
    measured_value, border, subline, full_subline_information) %>%
    mutate(month = "Feb")
```

Remove the "X" in front of some time points.

```
dec_data$time_point <- as.numeric(gsub("X",
    "", dec_data$time_point))
feb_data$time_point <- as.numeric(gsub("X",
    "", feb_data$time_point))
```

Add a column with the day of the experiment.

```
dec_data <- add_day_col(dec_data)
jan_data <- add_day_col(jan_data)
feb_data <- add_day_col(feb_data)
```

Here, address negative and NA measured values.

```
### Shift all NPQ values so the minimum is
### 0
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))

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

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

### Remove the 2 NA values for phi2, and
### shift the phi2 measured values by the
### minimum values to ensure the minimum
### value is 0
dec_data <- na.omit(dec_data)

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))
```

Finally, merge these three experiments to create one data frame.

```
### Merge the two data frames, only using
### the columns they have in common
depi_all <- rbind(dec_data, jan_data, feb_data) %>%
    arrange(genotype, time_point, month)

### Filter the time points by the max time
### point of the shortest experiment -
### can't make comparisons on days that the
### three experiments don't share
depi_all <- as.data.frame(depi_all %>% filter(time_point <=
    min(c(max(jan_data$time_point), max(dec_data$time_point),
        max(feb_data$time_point)))))
```

```
depi_all$measurement[depi_all$measurement ==
    "size"] <- "leafarea"
depi_all$measurement[depi_all$measurement ==
    "growth"] <- "leafarea"
```

Now, just check to make sure I have the correct sublines of Col0.

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

```
## [1] "Col1-4" "Col1-2" "Col1-1" "Col1-3" "Col1_4" "Col1_2" "Col1_1" "Col1_3"
```

This is good!! Note that there are no Col1-2 sublines in the data, which explains why we don't see any here.

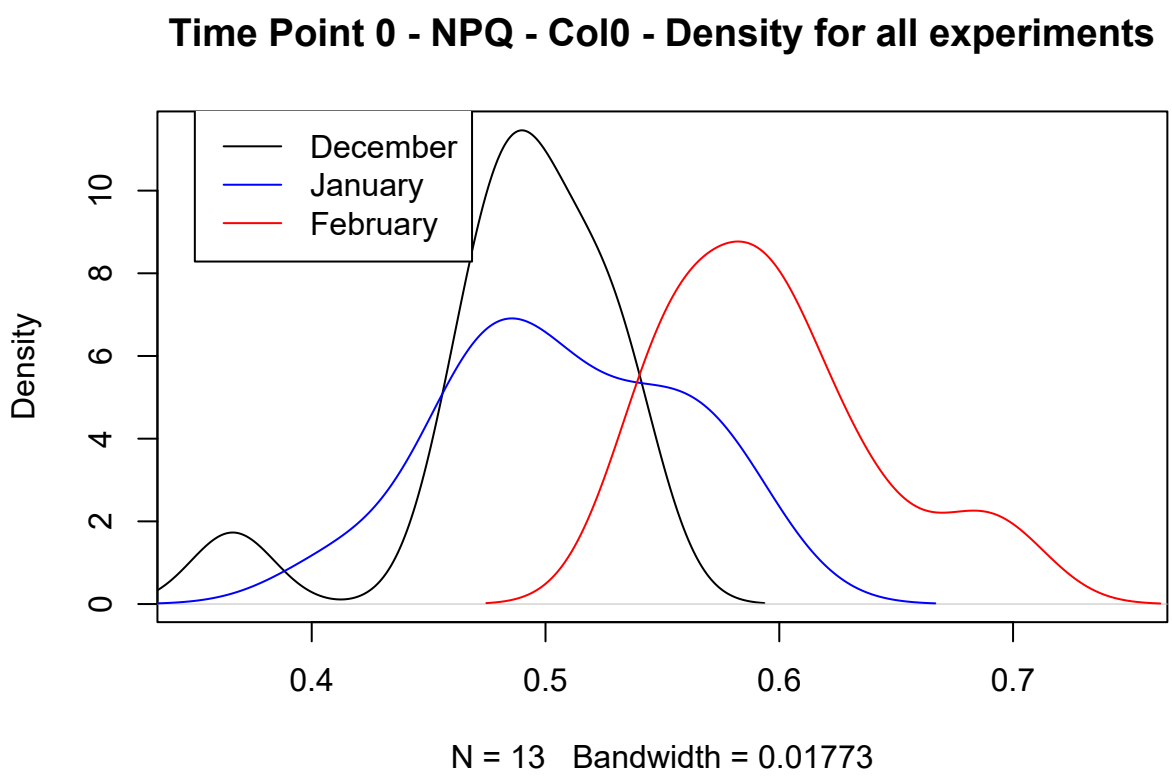
## Investigate Quantile Normalization

First, I want to look at one time point for one specific genotype. I will focus on Col0 at time point 0 for NPQ.

I will be quantile normalizing the three experiments.

```
dec_temp <- (depi_all %>% filter(genotype ==
    "Col0", measurement == "npq", time_point ==
    "0", month == "Dec") %>% arrange(measured_value))$measured_value
jan_temp <- (depi_all %>% filter(genotype ==
    "Col0", measurement == "npq", time_point ==
    "0", month == "Jan") %>% arrange(measured_value))$measured_value
feb_temp <- (depi_all %>% filter(genotype ==
    "Col0", measurement == "npq", time_point ==
    "0", month == "Feb") %>% arrange(measured_value))$measured_value
```

```
plot(density(dec_temp), xlim = c(0.35, 0.75),
    main = "Time Point 0 - NPQ - Col0 - Density for all experiments")
lines(density(jan_temp), col = "blue")
lines(density(feb_temp), col = "red")
legend(0.35, 12, legend = c("December", "January",
    "February"), col = c("black", "blue",
    "red"), lty = 1)
```




The goal of quantile normalization is to get all three of these density plots to look the same.

Clearly, I have some work to do - they have very different densities!

**But, how do I approach this when there are an unequal number of replicates between experiments?**

I couldn't find any obvious solutions online. Let's try this method with genotypes that have the same number of replicates in the three experiments.

```
depi_all %>% group_by(month, genotype, measurement) %>%
    summarize(length(unique(individual_plant_metadata))) %>%
    select(-measurement) %>% distinct() %>%
    arrange(genotype)
```

```
## # A tibble: 113 x 3
## # Groups:   month, genotype [113]
##    month genotype `length(unique(individual_plant_metadata))`
##    <chr> <chr>                                          <int>
##  1 Dec   Col0                                              13
##  2 Feb   Col0                                              16
##  3 Jan   Col0                                              13
##  4 Dec   mpk1                                              13
##  5 Feb   mpk1                                              17
##  6 Jan   mpk1                                              13
##  7 Dec   mpk1-13                                            8
##  8 Feb   mpk1-13                                           15
```

```
##  9 Jan   mpk1-13                                                12
## 10 Dec   mpk1-14                                                 4
## # ... with 103 more rows
```

Okay - there are never genotypes that have the same amount of replicates per experiment, so that won't work.

While I wait to discuss with Shinhan and Melissa how to handle this roadblock, investigate other ways to normalize data.

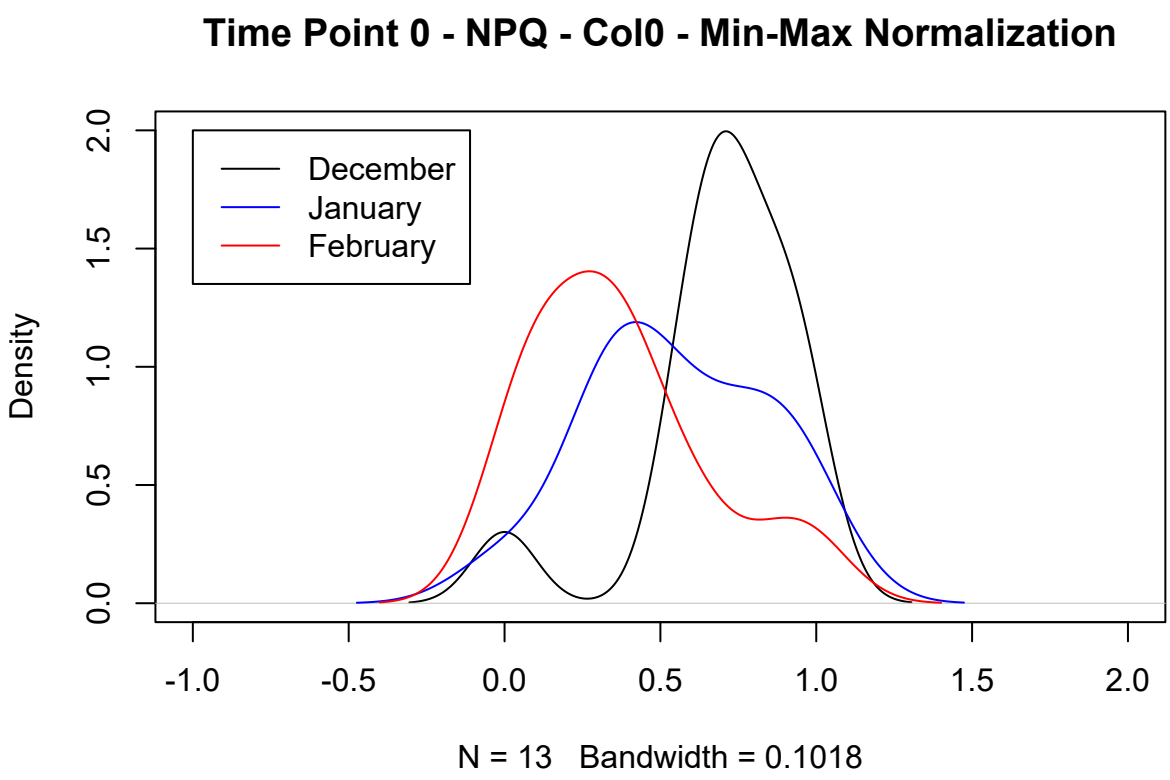
Another way to normalize data is to use feature scaling using min-max normalization.

While this won't make the distributions have the same "shape", it ensure that the data has the same min and max values relative to each other.

```
test_normalization <- depi_all %>% filter(genotype ==
    "Col0", measurement == "npq", time_point ==
    "0") %>% arrange(measured_value) %>%
    group_by(month) %>% mutate(min = min(measured_value),
    max = max(measured_value)) %>% mutate(scaled = (measured_value -
    min)/(max - min)) %>% select(month, scaled)

filter(test_normalization, month == "Jan")$scaled
```

```
##  [1] 0.0000000 0.3085415 0.3184195 0.3416618 0.3678094 0.4503196 0.5276002
##  [8] 0.5828007 0.6920395 0.8268449 0.8349797 0.9343405 1.0000000
```

```
plot(density(filter(test_normalization, month ==
    "Dec")$scaled), xlim = c(-1, 2), ylim = c(0,
    2), main = "Time Point 0 - NPQ - Col0 - Min-Max Normalization")
lines(density(filter(test_normalization,
    month == "Jan")$scaled), col = "blue")
lines(density(filter(test_normalization,
    month == "Feb")$scaled), col = "red")
legend(0.35, 16, legend = c("December", "January",
    "February"), col = c("black", "blue",
    "red"), lty = 1)
legend(-1, 2, legend = c("December", "January",
    "February"), col = c("black", "blue",
    "red"), lty = 1)
```




Here is a summary of each month:

```
### December
summary(filter(test_normalization, month ==
    "Dec")$scaled)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.6096  0.7095  0.7022  0.8628  1.0000
```

```
### January
summary(filter(test_normalization, month ==
    "Jan")$scaled)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.3417  0.5276  0.5527  0.8268  1.0000
```

```
### February
summary(filter(test_normalization, month ==
    "Feb")$scaled)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.1413  0.3510  0.3635  0.4874  1.0000
```

So, we have the range that we want, but the quantiles are still variant between experiments.

But, at least I have an idea of how to use dplyr to do min-max normalization!!

Here's a pipeline that will apply this to the entire dataset, if we chose to use this method:

```
test_normalization <- depi_all %>% group_by(genotype,
    measurement, time_point, month) %>% mutate(min = min(measured_value),
    max = max(measured_value)) %>% mutate(scaled = (measured_value -
    min)/(max - min))
```

Let's verify that this:

```
### Test 1
a <- filter(test_normalization, genotype ==
    "mpk1", time_point == "1", measurement ==
    "npq", month == "Dec")$measured_value

(a - min(a))/(max(a) - min(a))
```

```
##  [1] 0.4020537 0.3329384 0.2523697 0.1749605 0.6481043 1.0000000 0.3455766
##  [8] 0.5114534 0.7357820 0.3811216 0.0000000 0.2590837 0.3755924
```

```
verify_a <- filter(test_normalization, genotype ==
    "mpk1", time_point == "1", measurement ==
    "npq", month == "Dec")$scaled

verify_a
```

```
##  [1] 0.4020537 0.3329384 0.2523697 0.1749605 0.6481043 1.0000000 0.3455766
##  [8] 0.5114534 0.7357820 0.3811216 0.0000000 0.2590837 0.3755924
```

```
### Test 2
b <- filter(test_normalization, genotype ==
    "Col0", time_point == "0", measurement ==
    "phi2", month == "Feb")$measured_value

(b - min(b))/(max(b) - min(b))
```

```
##  [1] 0.2599338 0.2781457 0.4950331 0.3890728 0.3526490 0.3658940 0.7864238
##  [8] 0.8625828 0.2301325 1.0000000 0.1225166 0.0000000 0.4983444 0.6754967
## [15] 0.6738411 0.8129139
```

```
verify_b <- filter(test_normalization, genotype ==
    "Col0", time_point == "0", measurement ==
    "phi2", month == "Feb")$scaled

verify_b
```

```
##  [1] 0.2599338 0.2781457 0.4950331 0.3890728 0.3526490 0.3658940 0.7864238
##  [8] 0.8625828 0.2301325 1.0000000 0.1225166 0.0000000 0.4983444 0.6754967
## [15] 0.6738411 0.8129139
```

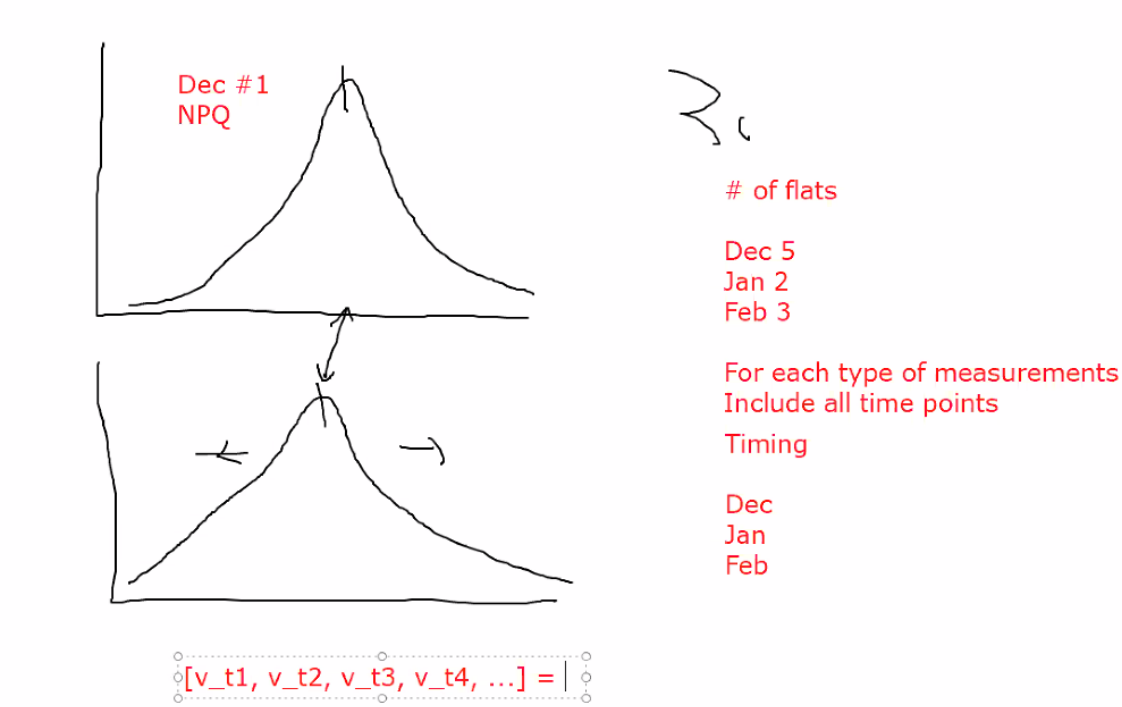



Figure 1: A schematic of how Shinhan proposed I tackle this.

## Notes from Friday, 7/31 Meeting

Now, I change approach after meeting with Shinhan and Melissa.

General approach:

1. Normalization done at the per flat level, for each type of measurement. But, include all time points and all genotypes.

OR

2. Normalization done on the level of experiment.

Then, always do:

3. Normalize against the WT on that flat. Once we compare before and after the quantile normalization - variance should be smaller after quantile. (Divide each measured value by the median measured value of Col0 on the same flat.)

**The goal is to use the approach that minimizes the variance.**

When doing this, group all of the plants on the flat together, regardless of their genotype.

There will be two different levels of normalization - first level, normalize between three sets of data - December, January, and February. Then, normalize at the unit of a flat.

To start, try normalize.quantiles.

I should have a variance for each genotype. And, then I can draw the distributions.

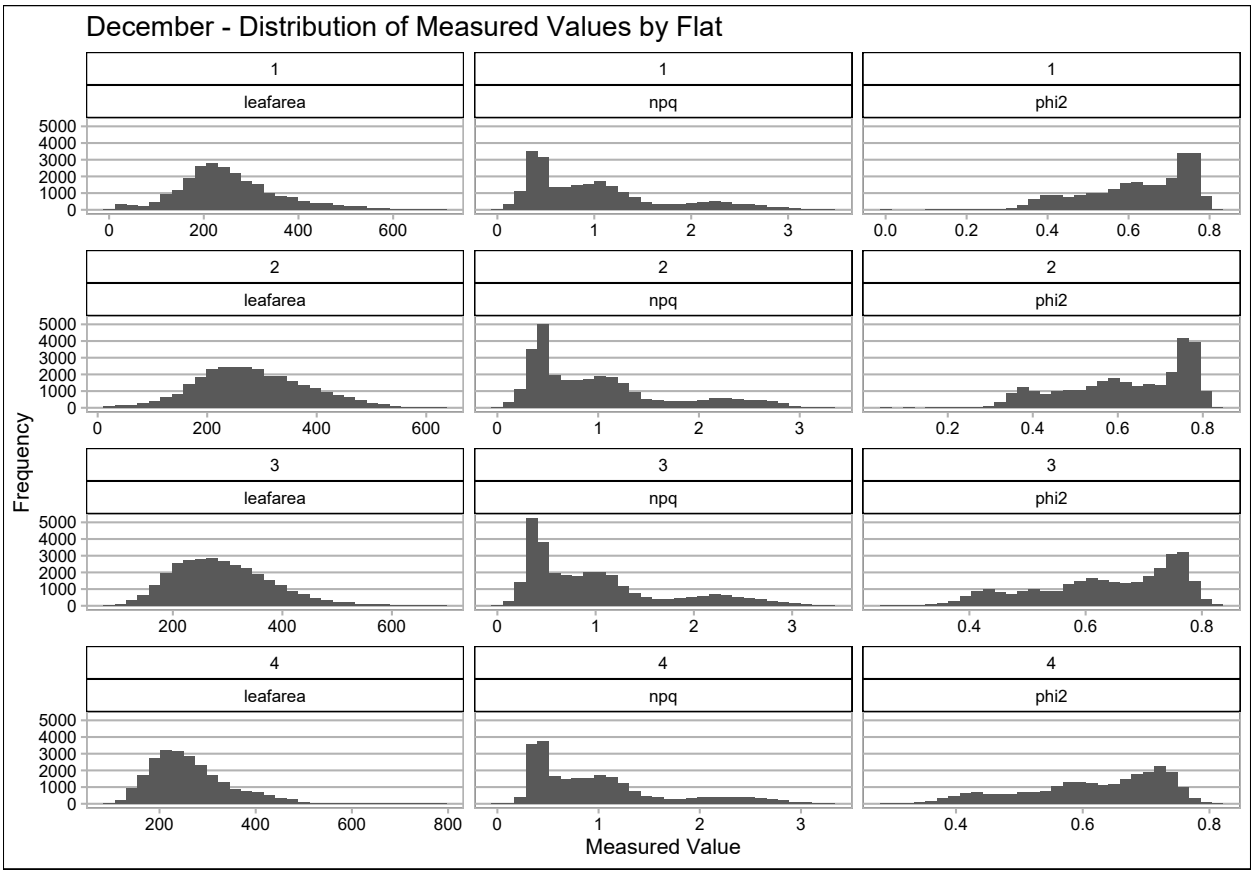
ANOTHER CRAZY IDEA:

Essentially, the trait value = g + g x flat + g x season + flat x season + g x f x s

This linear regression is used to estimate genotype effect. We will do this for each genotype - anticipate intercept to be 0. We will build a different model for each time point for each trait.

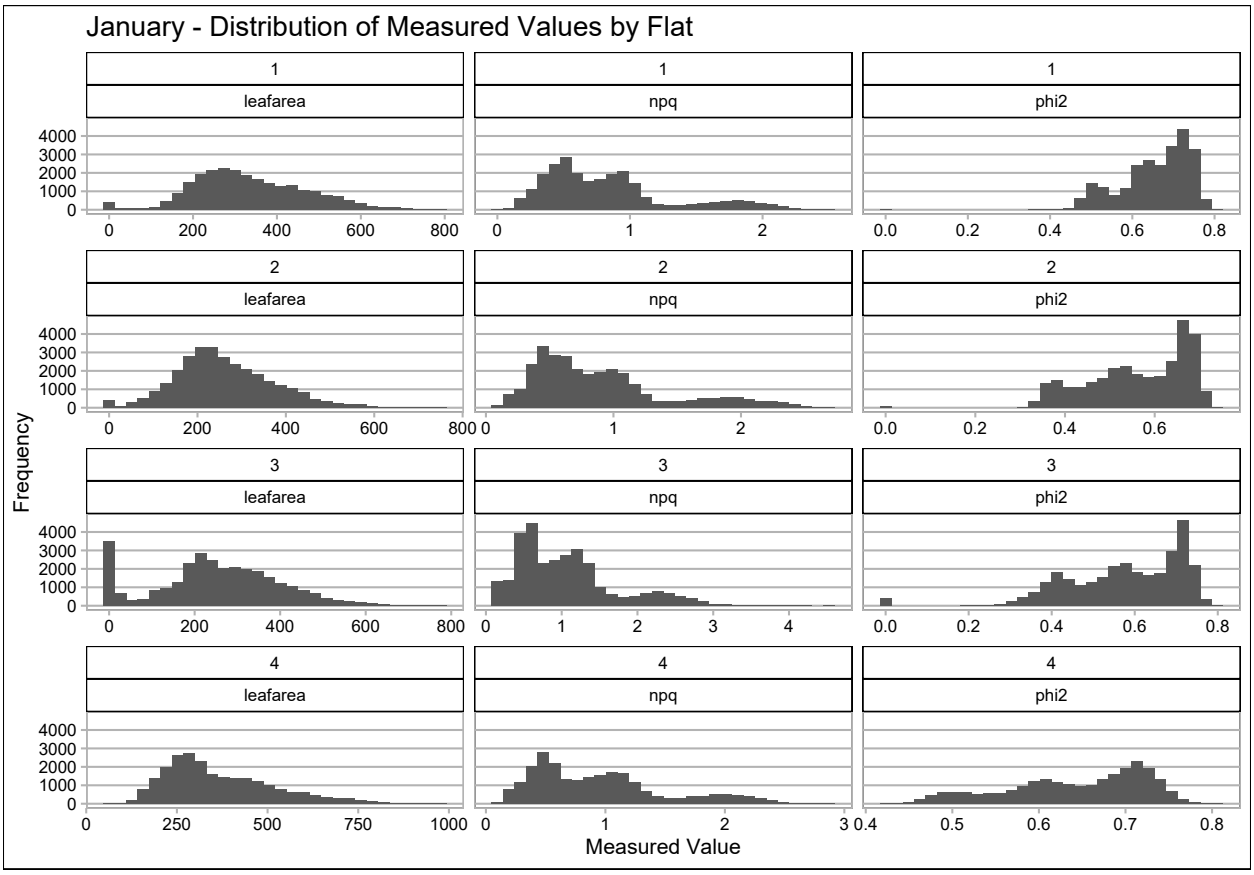
Note that this model may have a problem - flat is nested in season.

```
plot_data_dec <- depi_all %>% filter(month ==
    "Dec")
plot_data_jan <- depi_all %>% filter(month ==
    "Jan")
plot_data_feb <- depi_all %>% filter(month ==
    "Feb")

ggplot(data = plot_data_dec, aes(x = measured_value)) +
    geom_histogram() + facet_wrap(flat_number ~
    measurement, scales = "free_x", ncol = 3) +
    labs(x = "Measured Value", y = "Frequency",
        title = "December - Distribution of Measured Values by Flat") +
    theme_calc(base_size = 8)
```

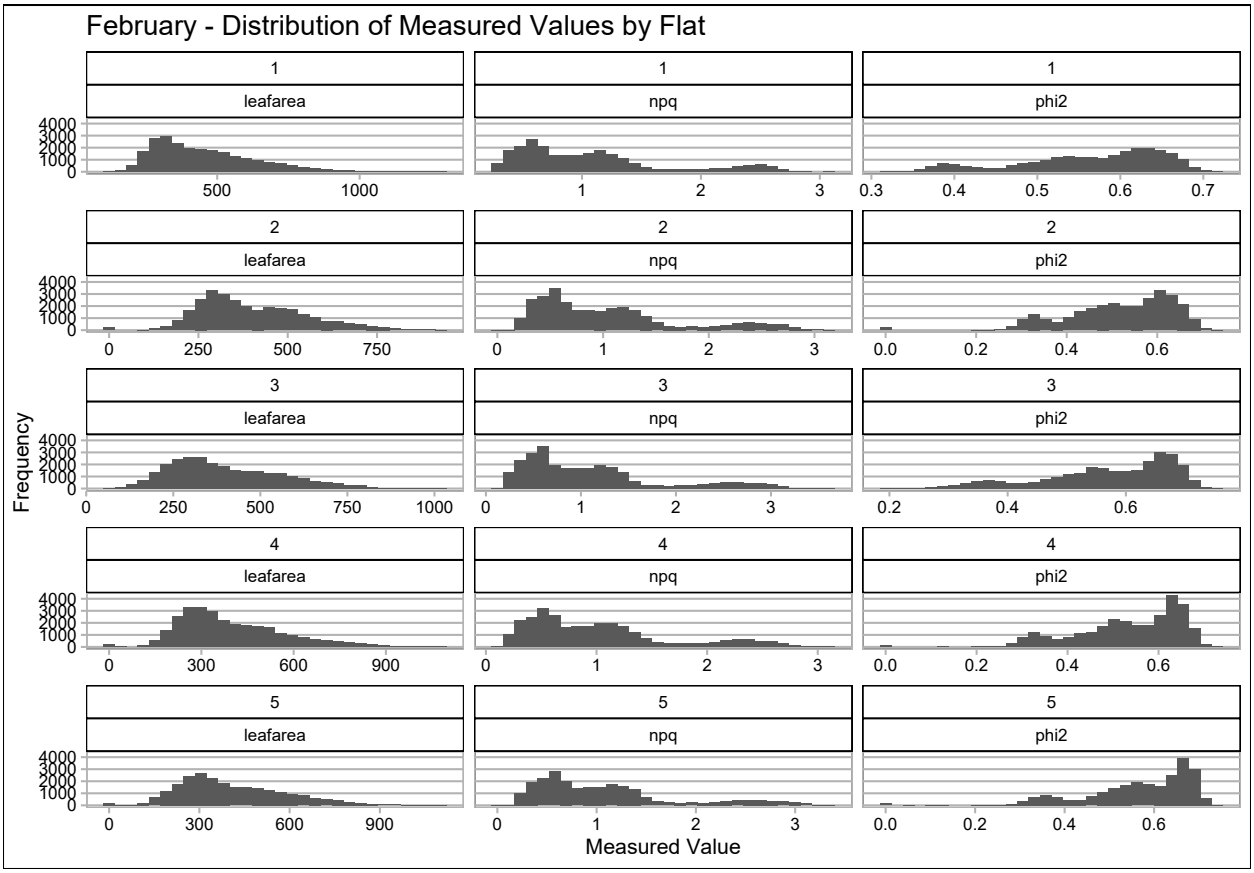



```
ggplot(data = plot_data_jan, aes(x = measured_value)) +
    geom_histogram() + facet_wrap(flat_number ~
    measurement, scales = "free_x", ncol = 3) +
    labs(x = "Measured Value", y = "Frequency",
```

```
        title = "January - Distribution of Measured Values by Flat") +
    theme_calc(base_size = 8)
```




```
ggplot(data = plot_data_feb, aes(x = measured_value)) +
    geom_histogram() + facet_wrap(flat_number ~
    measurement, scales = "free_x", ncol = 3) +
    labs(x = "Measured Value", y = "Frequency",
        title = "February - Distribution of Measured Values by Flat") +
    theme_calc(base_size = 8)
```




So, the goal is to get each column to have the same distribution. In other words: I want the same distribution for each flat and measurement in each experiment.

## normalize.quantiles()

Here, load the necessary package and have a short example to see what's actually happening with normalize.quantiles.

```
library(preprocessCore)
# the function expects a matrix create a
# matrix using the same example
mat <- matrix(c(5, 2, 3, 4, 4, 1, 4, 2, 3,
    4, 6, 8), ncol = 3)
mat
```

```
##      [,1] [,2] [,3]
## [1,]    5    4    3
## [2,]    2    1    4
## [3,]    3    4    6
## [4,]    4    2    8
```

```
normalize.quantiles(mat)
```

```
##          [,1]     [,2]     [,3]
```

```
## [1,] 5.666667 5.166667 2.000000
## [2,] 2.000000 2.000000 3.000000
## [3,] 3.000000 5.166667 4.666667
## [4,] 4.666667 3.000000 5.666667
```

```
summary(mat)
```

```
##        V1              V2              V3
##  Min.   :2.00   Min.   :1.00   Min.   :3.00
##  1st Qu.:2.75   1st Qu.:1.75   1st Qu.:3.75
##  Median :3.50   Median :3.00   Median :5.00
##  Mean   :3.50   Mean   :2.75   Mean   :5.25
##  3rd Qu.:4.25   3rd Qu.:4.00   3rd Qu.:6.50
##  Max.   :5.00   Max.   :4.00   Max.   :8.00
```

```
summary(normalize.quantiles(mat))
```

```
##        V1              V2              V3
##  Min.   :2.000   Min.   :2.000   Min.   :2.000
##  1st Qu.:2.750   1st Qu.:2.750   1st Qu.:2.750
##  Median :3.833   Median :4.083   Median :3.833
##  Mean   :3.833   Mean   :3.833   Mean   :3.833
##  3rd Qu.:4.917   3rd Qu.:5.167   3rd Qu.:4.917
##  Max.   :5.667   Max.   :5.167   Max.   :5.667
```

```
# Function to add vector as column
addToDF <- function(df, v) {
    nRow <- nrow(df)
    lngth <- length(v)
    if (nRow > lngth) {
        length(v) <- nRow
    } else if (nRow < lngth) {
        df[(nRow + 1):lngth, ] <- NA
    }
    cbind(df, v)
}
```

So, a summary of the normalize.quantile data returns identical summary statistics for each column.

Now, apply this function to each experiment to normalize between flats. Use December and NPQ as an example.

```
temp_flat1 <- filter(depi_all, month == "Dec",
    measurement == "npq", flat_number ==
        "1")$measured_value
temp_flat2 <- filter(depi_all, month == "Dec",
    measurement == "npq", flat_number ==
        "2")$measured_value
temp_flat3 <- filter(depi_all, month == "Dec",
    measurement == "npq", flat_number ==
        "3")$measured_value
temp_flat4 <- filter(depi_all, month == "Dec",
```

```
    measurement == "npq", flat_number ==
        "4")$measured_value

count <- 1:max(c(length(temp_flat1), length(temp_flat2),
    length(temp_flat3), length(temp_flat4)))

test <- as.matrix(cbind(temp_flat1, temp_flat2,
    temp_flat3, temp_flat4, count))

test_normalize <- normalize.quantiles(test[,
    1:4])

summary(test)
```

```
##    temp_flat1       temp_flat2       temp_flat3       temp_flat4
##  Min.   :0.0070   Min.   :0.0149   Min.   :0.0000   Min.   :0.0313
##  1st Qu.:0.4377   1st Qu.:0.4372   1st Qu.:0.4219   1st Qu.:0.4468
##  Median :0.8420   Median :0.7976   Median :0.7909   Median :0.8216
##  Mean   :1.0103   Mean   :0.9894   Mean   :0.9936   Mean   :1.0004
##  3rd Qu.:1.3053   3rd Qu.:1.2657   3rd Qu.:1.2674   3rd Qu.:1.2598
##  Max.   :3.4314   Max.   :3.3075   Max.   :3.3774   Max.   :3.3102
##      count
##  Min.   :    1
##  1st Qu.: 7561
##  Median :15120
##  Mean   :15120
##  3rd Qu.:22680
##  Max.   :30240
```

```
summary(test_normalize)
```

```
##        V1               V2               V3               V4
##  Min.   :0.0151   Min.   :0.0133   Min.   :0.0133   Min.   :0.0151
##  1st Qu.:0.4359   1st Qu.:0.4359   1st Qu.:0.4359   1st Qu.:0.4359
##  Median :0.8131   Median :0.8130   Median :0.8130   Median :0.8130
##  Mean   :0.9984   Mean   :0.9984   Mean   :0.9984   Mean   :0.9984
##  3rd Qu.:1.2745   3rd Qu.:1.2746   3rd Qu.:1.2747   3rd Qu.:1.2746
##  Max.   :3.3506   Max.   :3.3566   Max.   :3.3566   Max.   :3.3566
```

Okay - this does what I want it to! The summary for each column after normalization are essentially identical. But, I need to make sure that I don't mix up what measurement or experiment the data is from - the matrix just has measured values, and no other identifying information.

Here is a loop that accomplishes this for all measurements and experiments:

```
### Create a loop for each measurement and
### experiment
for (i in c("npq", "phi2", "leafarea")) {
    for (j in c("Dec", "Jan", "Feb")) {
        ### Filter to select each specific month
        ### and measured value
        temp_vector <- filter(depi_all, month ==
```

```
            j, measurement == i)
        ### Initialize an empty data frame
        temp_df <- data.frame()
        ### Loop through each flat
        for (k in 1:length(unique(temp_vector$flat_number))) {
            ### Create a temporary data frame - each
            ### column is the measured values for each
            ### flat
            temp <- filter(temp_vector, flat_number ==
                k)$measured_value
            temp_df <- addToDF(temp_df, temp)
        }

        ### Normalize across the flats
        temp_normalize <- as.data.frame(normalize.quantiles(as.matrix(temp_df))) %>%
            ### Add columns with the measurement and
        ### experiment to be certain there hasn't
        ### been any mix-ups
        mutate(measurement = i, experiment = j)
        ### Create a name to give the normalized
        ### data based on the measurement and
        ### experiment
        temp_name <- paste(tolower(j), "_",
            i, "_normalize", sep = "")
        ### Rename the columns
        if (ncol(temp_normalize) == 6) {
            temp_normalize <- temp_normalize %>%
                rename(flat_1 = V1, flat_2 = V2,
                  flat_3 = V3, flat_4 = V4)
        } else {
            temp_normalize <- temp_normalize %>%
                rename(flat_1 = V1, flat_2 = V2,
                  flat_3 = V3, flat_4 = V4,
                  flat_5 = V5)
        }
        ### Assign the name to the data frame
        assign(temp_name, temp_normalize)
    }
}
```

## Summary - Normalized by Flat

Here are the summary statistics for each month and measured value. Note that the summary statistics for each column - each column represents a flat - are essentially identical.

```
summary(dec_leafarea_normalize)
```

```
##      flat_1           flat_2           flat_3           flat_4
##  Min.   : 49.25   Min.   : 49.25   Min.   : 49.25   Min.   : 49.25
##  1st Qu.:207.75   1st Qu.:207.75   1st Qu.:207.75   1st Qu.:207.75
##  Median :261.75   Median :260.75   Median :260.75   Median :261.75
##  Mean   :272.55   Mean   :272.55   Mean   :272.56   Mean   :272.56
```

```
##  3rd Qu.:327.50   3rd Qu.:326.75   3rd Qu.:326.75   3rd Qu.:326.75
##  Max.   :700.00   Max.   :700.00   Max.   :700.00   Max.   :700.00
##  NA's   :6944     NA's   :1344                      NA's   :6272
##  measurement         experiment
##  Length:30240       Length:30240
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
##
```

```
summary(jan_leafarea_normalize)
```

```
##      flat_1           flat_2          flat_3           flat_4
##  Min.   : 24.25   Min.   : 27.0   Min.   : 34.25   Min.   : 19.25
##  1st Qu.:213.00   1st Qu.:213.0   1st Qu.:213.00   1st Qu.:213.00
##  Median :286.00   Median :287.0   Median :287.00   Median :287.00
##  Mean   :304.96   Mean   :305.0   Mean   :305.02   Mean   :304.95
##  3rd Qu.:393.25   3rd Qu.:393.2   3rd Qu.:393.25   3rd Qu.:393.25
##  Max.   :827.50   Max.   :827.5   Max.   :827.50   Max.   :825.37
##  NA's   :5600                                      NA's   :6048
##  measurement         experiment
##  Length:30240       Length:30240
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
##
```

```
summary(feb_leafarea_normalize)
```

```
##      flat_1           flat_2            flat_3           flat_4
##  Min.   :  39.4   Min.   :  47.45   Min.   :  39.58   Min.   :  48.0
##  1st Qu.: 285.8   1st Qu.: 286.22   1st Qu.: 286.00   1st Qu.: 286.0
##  Median : 377.5   Median : 377.20   Median : 377.80   Median : 376.8
##  Mean   : 412.6   Mean   : 412.58   Mean   : 412.56   Mean   : 412.6
##  3rd Qu.: 519.6   3rd Qu.: 519.60   3rd Qu.: 519.60   3rd Qu.: 519.8
##  Max.   :1085.2   Max.   :1085.20   Max.   :1085.20   Max.   :1085.2
##  NA's   :4704                       NA's   :1344
##      flat_5        measurement         experiment
##  Min.   :  47.47   Length:29120       Length:29120
##  1st Qu.: 286.00   Class :character   Class :character
##  Median : 377.01   Mode  :character   Mode  :character
##  Mean   : 412.57
##  3rd Qu.: 519.60
##  Max.   :1085.20
##  NA's   :4928
```

```
summary(dec_phi2_normalize)
```

```
##      flat_1          flat_2          flat_3          flat_4
##  Min.   :0.153   Min.   :0.1532   Min.   :0.1532   Min.   :0.153
##  1st Qu.:0.539   1st Qu.:0.5386   1st Qu.:0.5386   1st Qu.:0.539
##  Median :0.649   Median :0.6487   Median :0.6488   Median :0.649
##  Mean   :0.626   Mean   :0.6256   Mean   :0.6256   Mean   :0.626
##  3rd Qu.:0.736   3rd Qu.:0.7356   3rd Qu.:0.7356   3rd Qu.:0.736
##  Max.   :0.827   Max.   :0.8269   Max.   :0.8269   Max.   :0.827
##  NA's   :6946    NA's   :1344                      NA's   :6272
##  measurement         experiment
##  Length:30240       Length:30240
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
##
```

```
summary(jan_phi2_normalize)
```

```
##      flat_1          flat_2          flat_3          flat_4
##  Min.   :0.110   Min.   :0.2131   Min.   :0.3123   Min.   :0.106
##  1st Qu.:0.541   1st Qu.:0.5409   1st Qu.:0.5409   1st Qu.:0.541
##  Median :0.628   Median :0.6277   Median :0.6277   Median :0.628
##  Mean   :0.610   Mean   :0.6099   Mean   :0.6102   Mean   :0.610
##  3rd Qu.:0.700   3rd Qu.:0.6999   3rd Qu.:0.7000   3rd Qu.:0.700
##  Max.   :0.788   Max.   :0.7879   Max.   :0.7879   Max.   :0.788
##  NA's   :5600                                      NA's   :6048
##  measurement         experiment
##  Length:30240       Length:30240
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
##
```

```
summary(feb_phi2_normalize)
```

```
##      flat_1          flat_2          flat_3          flat_4
##  Min.   :0.102   Min.   :0.1240   Min.   :0.1017   Min.   :0.1214
##  1st Qu.:0.487   1st Qu.:0.4873   1st Qu.:0.4872   1st Qu.:0.4873
##  Median :0.572   Median :0.5725   Median :0.5725   Median :0.5724
##  Mean   :0.550   Mean   :0.5501   Mean   :0.5502   Mean   :0.5502
##  3rd Qu.:0.639   3rd Qu.:0.6390   3rd Qu.:0.6390   3rd Qu.:0.6391
##  Max.   :0.735   Max.   :0.7348   Max.   :0.7348   Max.   :0.7348
##  NA's   :4704                     NA's   :1344
##      flat_5      measurement         experiment
##  Min.   :0.123   Length:29120       Length:29120
##  1st Qu.:0.487   Class :character   Class :character
##  Median :0.572   Mode  :character   Mode  :character
##  Mean   :0.550
##  3rd Qu.:0.639
##  Max.   :0.735
##  NA's   :4928
```

```
summary(dec_npq_normalize)
```

```
##      flat_1          flat_2           flat_3          flat_4
##  Min.   :0.013   Min.   :0.0133   Min.   :0.0133   Min.   :0.013
##  1st Qu.:0.435   1st Qu.:0.4350   1st Qu.:0.4351   1st Qu.:0.435
##  Median :0.810   Median :0.8102   Median :0.8101   Median :0.810
##  Mean   :0.996   Mean   :0.9963   Mean   :0.9963   Mean   :0.996
##  3rd Qu.:1.272   3rd Qu.:1.2717   3rd Qu.:1.2718   3rd Qu.:1.272
##  Max.   :3.357   Max.   :3.3566   Max.   :3.3566   Max.   :3.357
##  NA's   :6944    NA's   :1344                      NA's   :6272
##  measurement         experiment
##  Length:30240       Length:30240
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
##
```

```
summary(jan_npq_normalize)
```

```
##      flat_1          flat_2            flat_3            flat_4
##  Min.   :0.055   Min.   :0.05505   Min.   :0.05505   Min.   :0.055
##  1st Qu.:0.503   1st Qu.:0.50324   1st Qu.:0.50321   1st Qu.:0.503
##  Median :0.814   Median :0.81437   Median :0.81442   Median :0.814
##  Mean   :0.935   Mean   :0.93485   Mean   :0.93482   Mean   :0.935
##  3rd Qu.:1.166   3rd Qu.:1.16631   3rd Qu.:1.16624   3rd Qu.:1.166
##  Max.   :3.121   Max.   :3.12062   Max.   :3.12062   Max.   :3.121
##  NA's   :5600                                        NA's   :6048
##  measurement         experiment
##  Length:30240       Length:30240
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
##
```

```
summary(feb_npq_normalize)
```

```
##      flat_1          flat_2           flat_3          flat_4
##  Min.   :0.119   Min.   :0.1194   Min.   :0.1194   Min.   :0.1194
##  1st Qu.:0.544   1st Qu.:0.5444   1st Qu.:0.5444   1st Qu.:0.5444
##  Median :0.895   Median :0.8949   Median :0.8949   Median :0.8949
##  Mean   :1.063   Mean   :1.0629   Mean   :1.0629   Mean   :1.0629
##  3rd Qu.:1.325   3rd Qu.:1.3252   3rd Qu.:1.3253   3rd Qu.:1.3252
##  Max.   :3.273   Max.   :3.2725   Max.   :3.2725   Max.   :3.2725
##  NA's   :4704                     NA's   :1344
##      flat_5      measurement         experiment
##  Min.   :0.119   Length:29120       Length:29120
##  1st Qu.:0.544   Class :character   Class :character
```

```
##  Median :0.895   Mode  :character   Mode  :character
##  Mean   :1.063
##  3rd Qu.:1.325
##  Max.   :3.273
##  NA's   :4928
```

**I see a potential issue! Check to make sure that these NA values make sense.**

I expect that the flat with the most measured values will have 0 NA values. I expect that flats with less measured values will fill in the gaps with NA.

Verify that this is the case:

```
depi_all %>% group_by(month, measurement,
    flat_number) %>% summarize(num_measured_values = length(measured_value)) %>%
    ungroup() %>% group_by(month, measurement) %>%
    mutate(expected_NA = num_measured_values -
        min(num_measured_values))
```

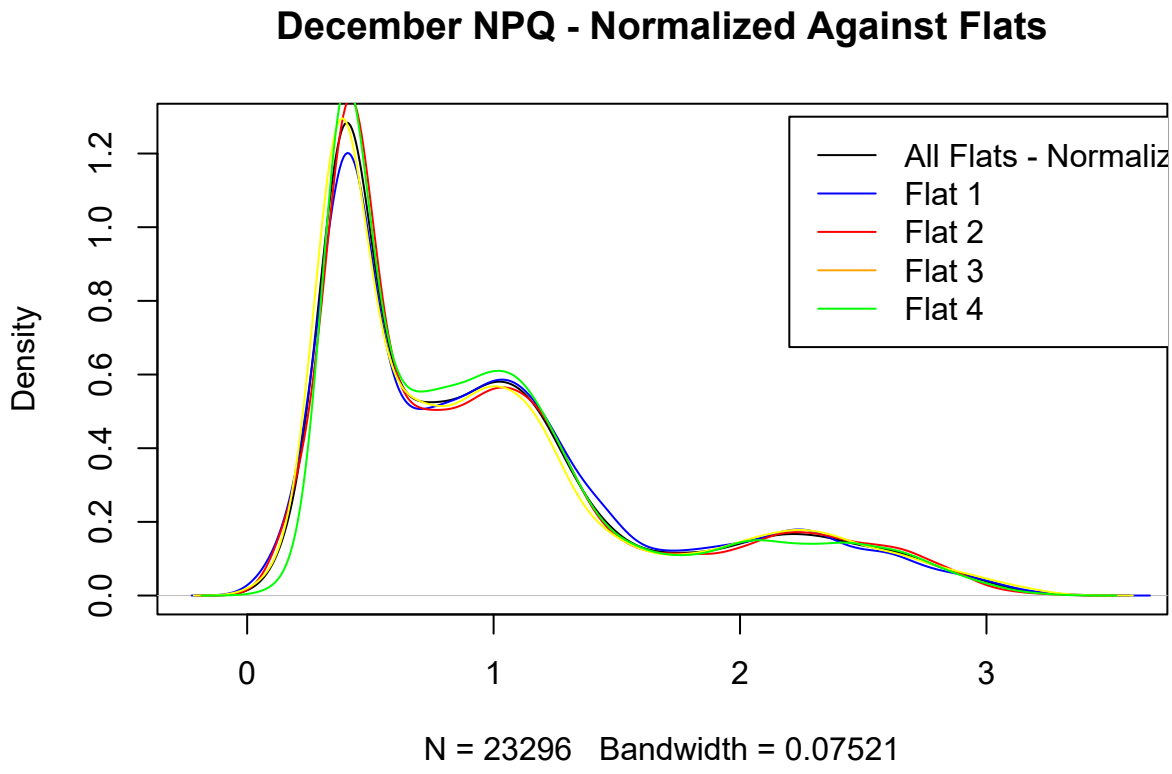
```
## # A tibble: 39 x 5
## # Groups:   month, measurement [9]
##    month measurement flat_number num_measured_values expected_NA
##    <chr> <chr>             <int>                <int>       <int>
##  1 Dec   leafarea              1                23296           0
##  2 Dec   leafarea              2                28896        5600
##  3 Dec   leafarea              3                30240        6944
##  4 Dec   leafarea              4                23968         672
##  5 Dec   npq                   1                23296           0
##  6 Dec   npq                   2                28896        5600
##  7 Dec   npq                   3                30240        6944
##  8 Dec   npq                   4                23968         672
##  9 Dec   phi2                  1                23294           0
## 10 Dec   phi2                  2                28896        5602
## # ... with 29 more rows
```

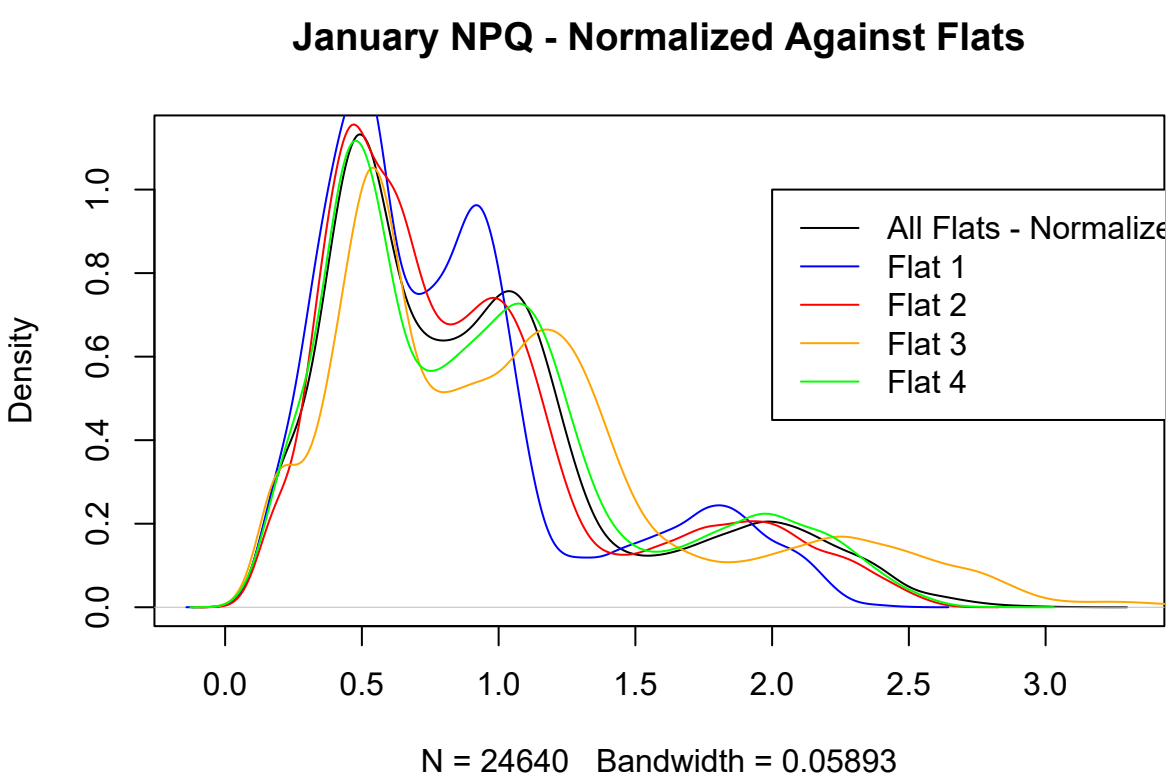
## Density - Comparing Normalized Data to Each Flat

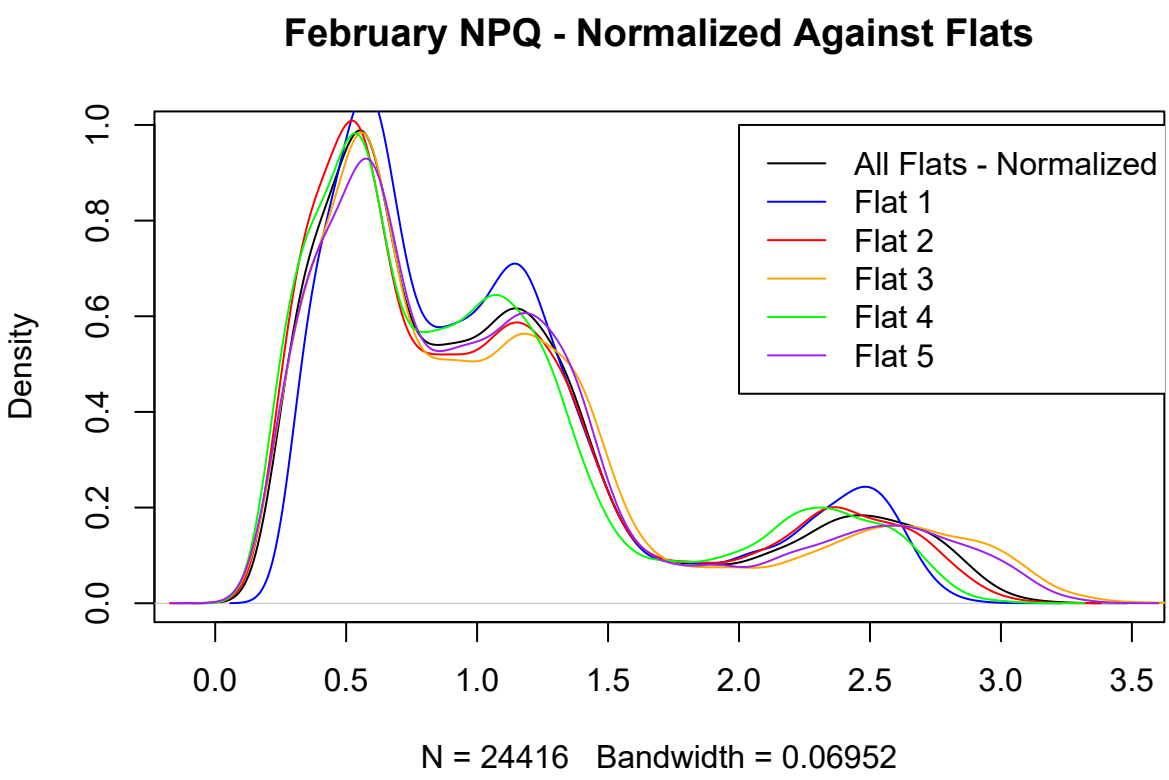
I only show the code to create the first plot. This is the same code used to create the other plots, with just the month and measurement changed.

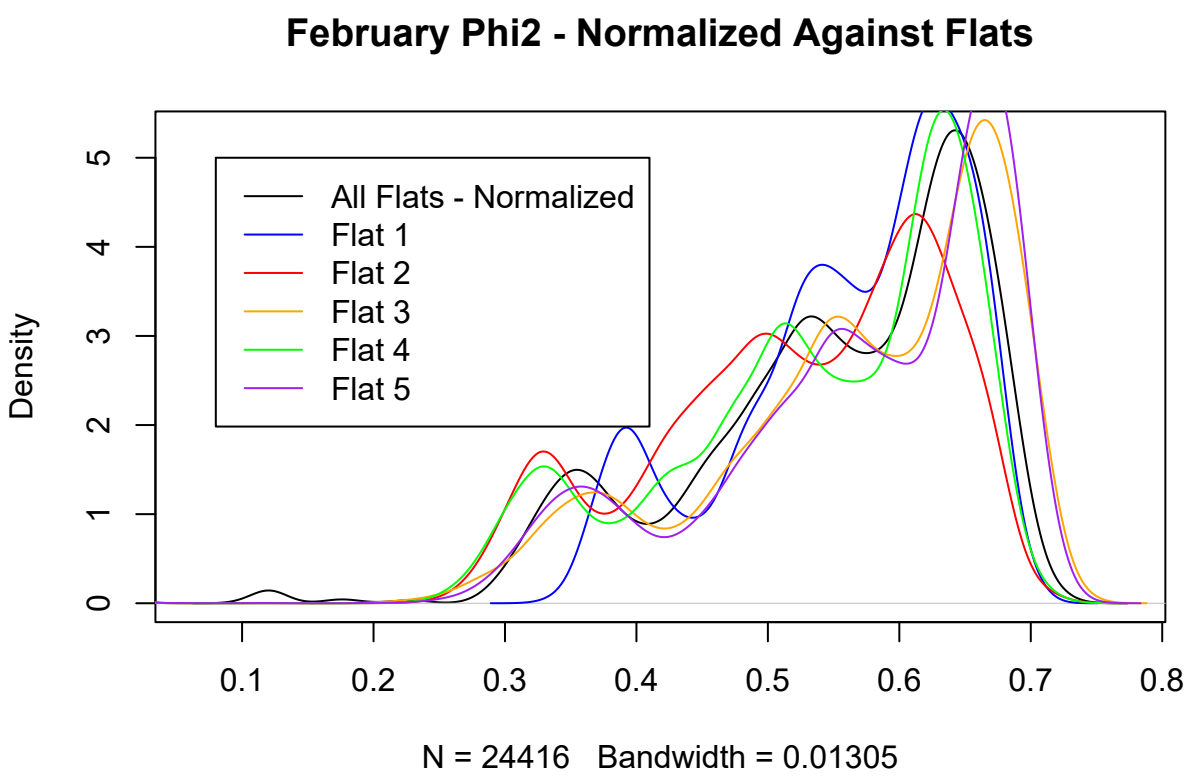
```
plot(density(dec_npq_normalize$flat_1, na.rm = TRUE),
    main = "December NPQ - Normalized Against Flats")
lines(density(filter(depi_all, month == "Dec",
    measurement == "npq", flat_number ==
        "1")$measured_value), col = "blue")
lines(density(filter(depi_all, month == "Dec",
    measurement == "npq", flat_number ==
        "2")$measured_value), col = "red")
lines(density(filter(depi_all, month == "Dec",
    measurement == "npq", flat_number ==
        "3")$measured_value), col = "yellow")
lines(density(filter(depi_all, month == "Dec",
    measurement == "npq", flat_number ==
```

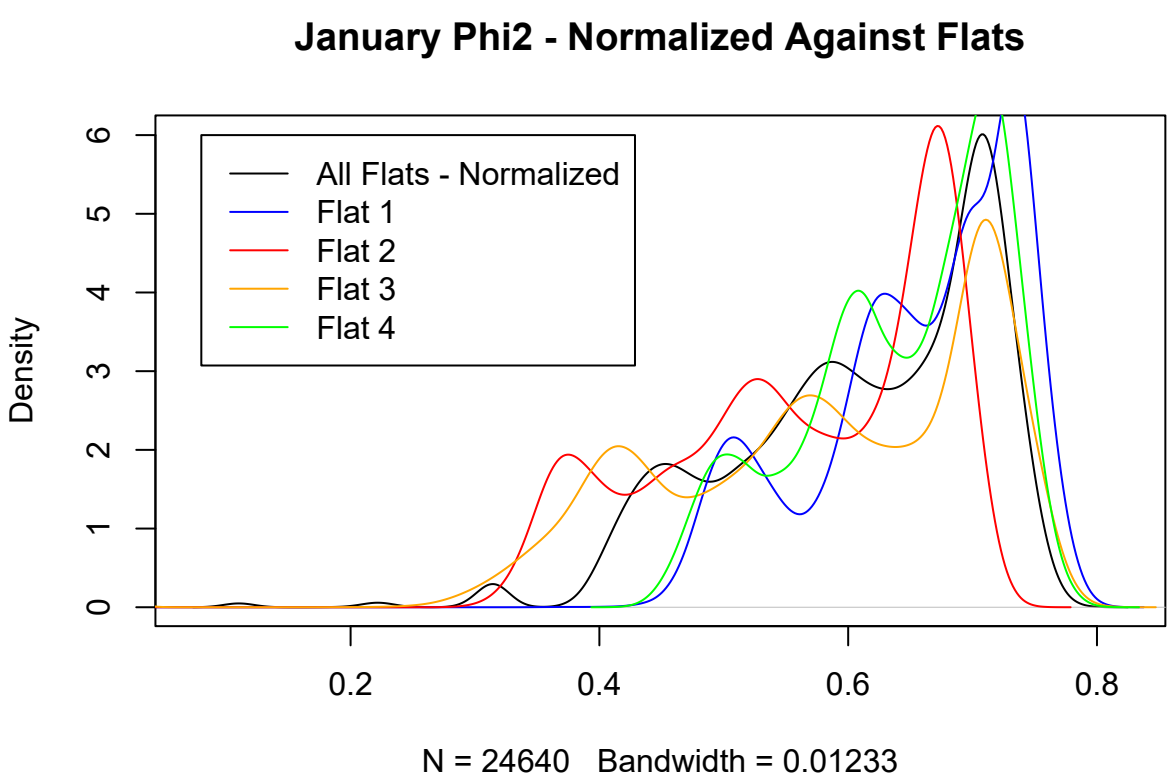
```
        "4")$measured_value), col = "green")
legend(2.2, 1.3, legend = c("All Flats - Normalized",
    "Flat 1", "Flat 2", "Flat 3", "Flat 4"),
    col = c("black", "blue", "red", "orange",
        "green"), lty = 1)
```

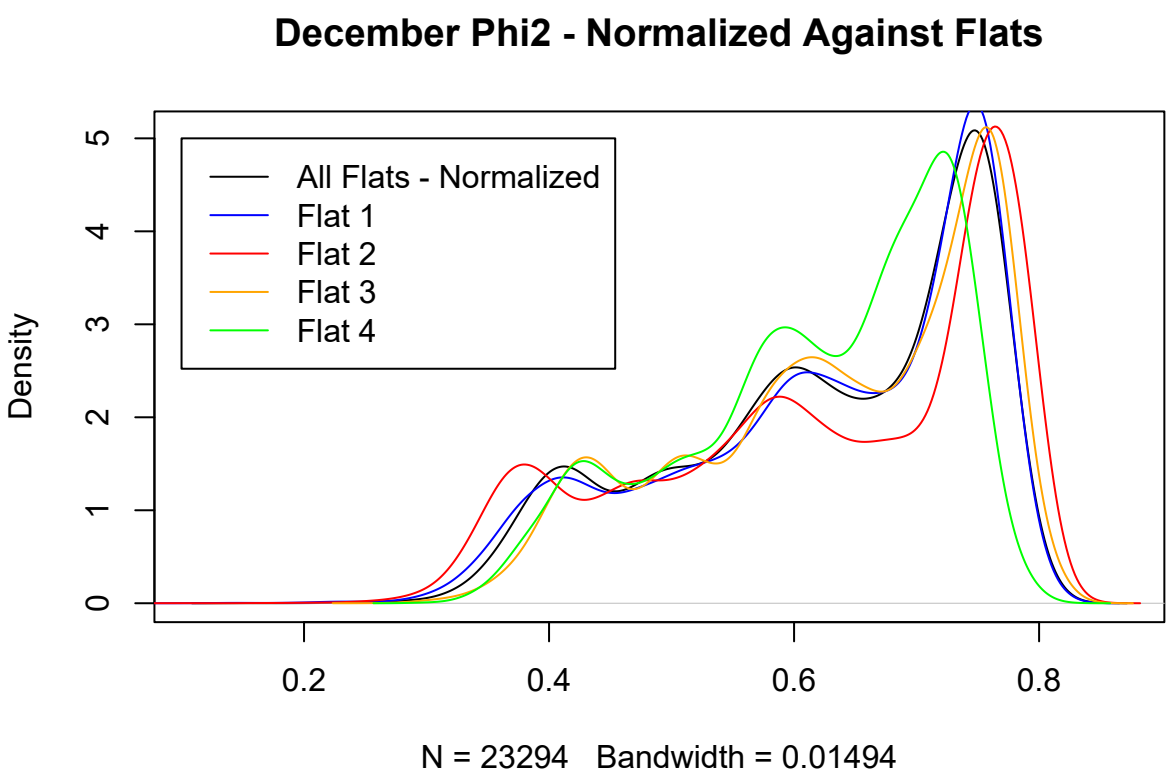


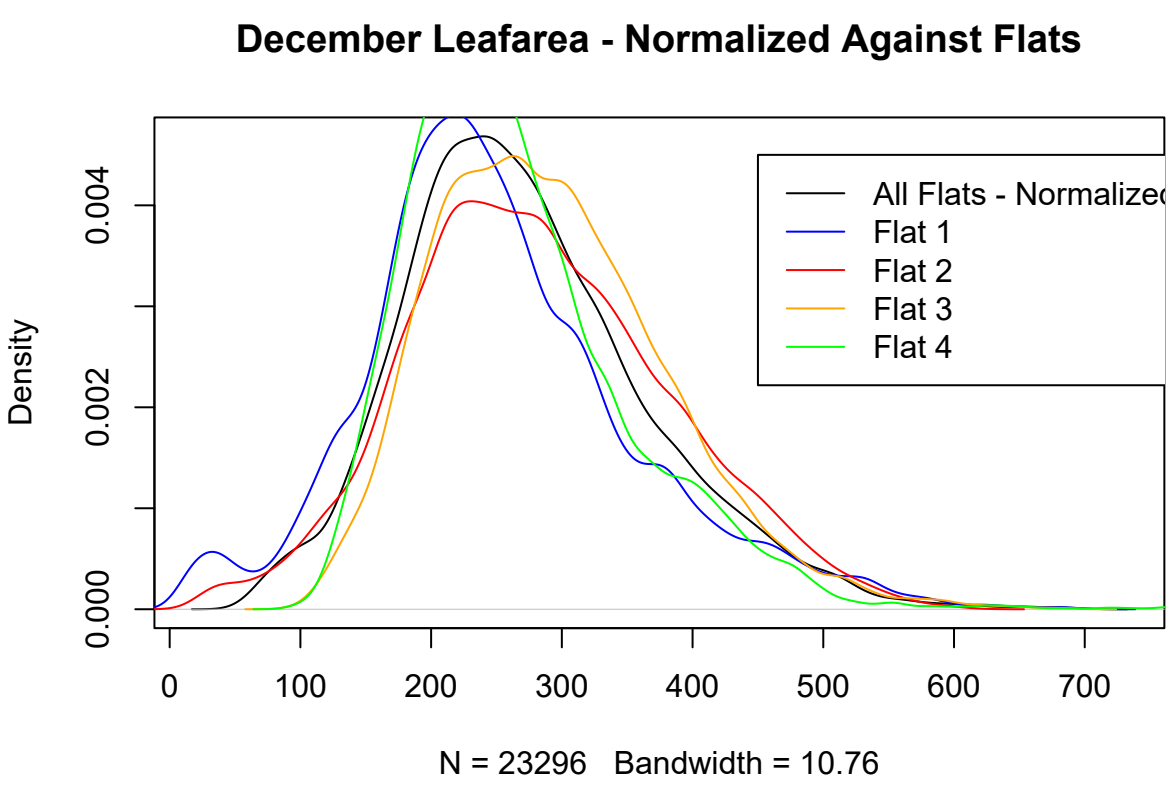

January NPQ - Normalized Against Flats
Density
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
All Flats - Normalize
Flat 1
Flat 2
Flat 3
Flat 4
N = 24640 Bandwidth = 0.05893

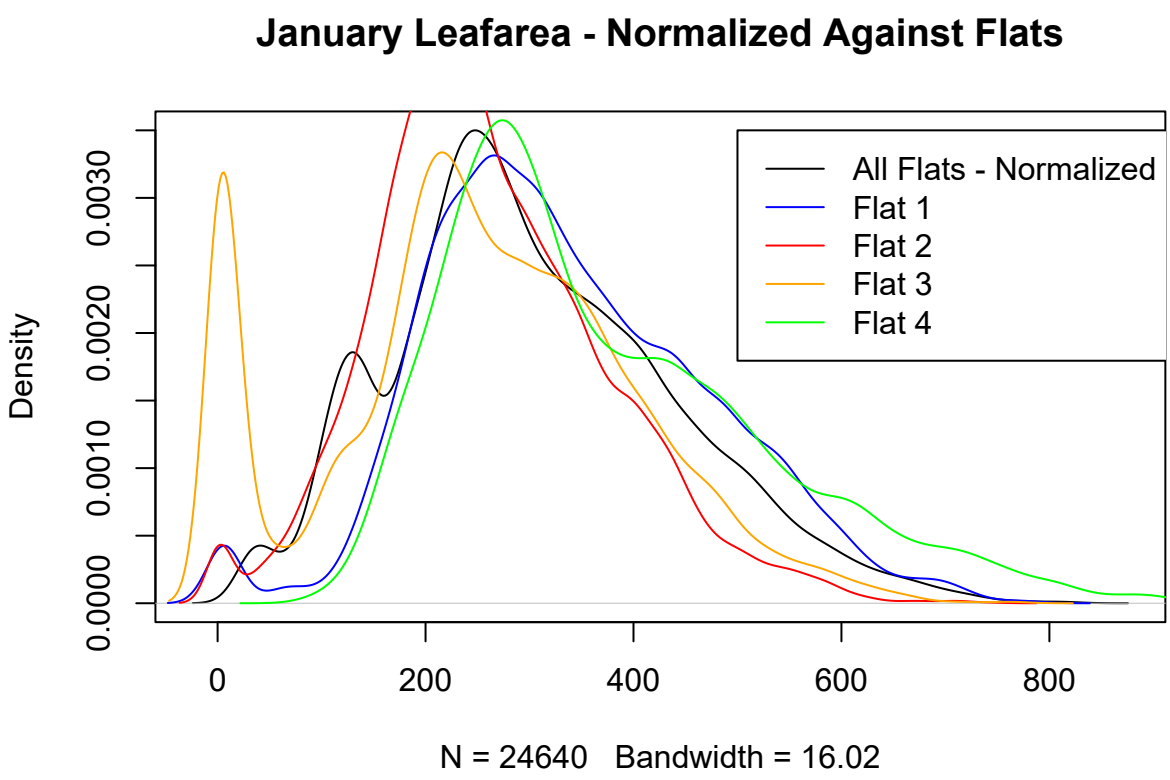

February NPQ - Normalized Against Flats
Density
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
All Flats - Normalized
Flat 1
Flat 2
Flat 3
Flat 4
Flat 5
N = 24416 Bandwidth = 0.06952

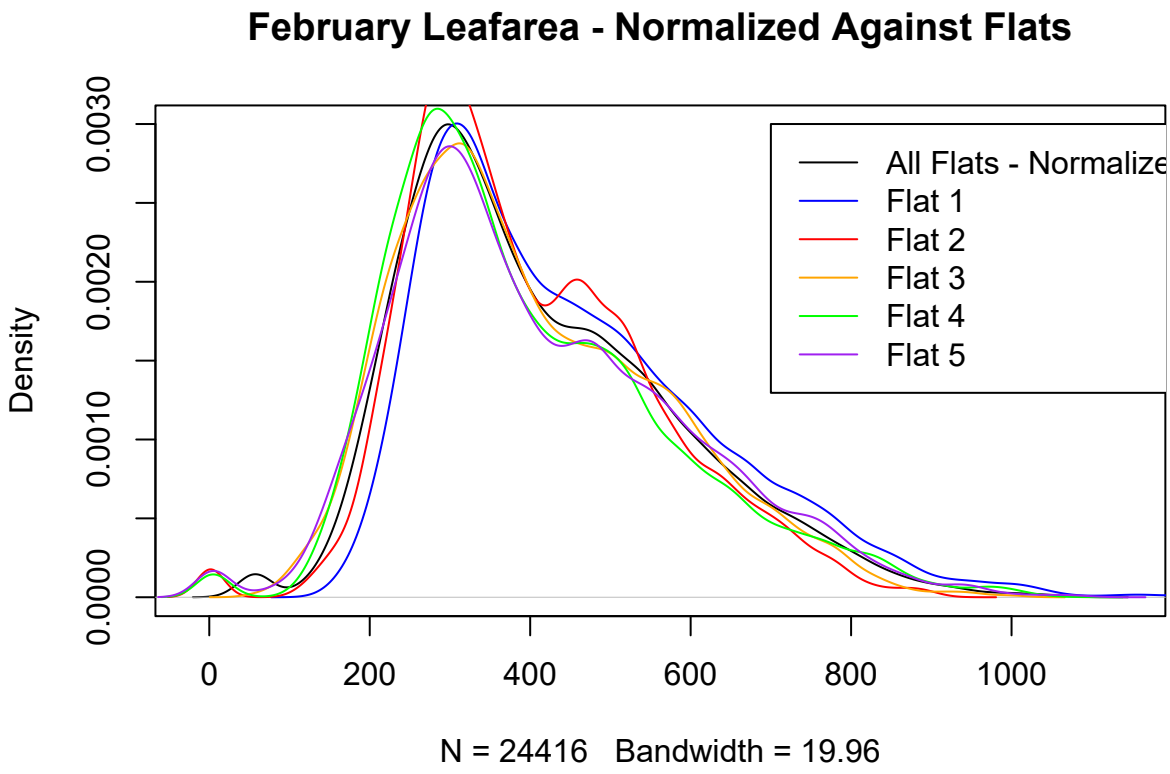

February Phi2 - Normalized Against Flats
All Flats - Normalized
Flat 1
Flat 2
Flat 3
Flat 4
Flat 5
Density
0
1
2
3
4
5
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
N = 24416 Bandwidth = 0.01305


January Phi2 - Normalized Against Flats
All Flats - Normalized
Flat 1
Flat 2
Flat 3
Flat 4
Density
0
1
2
3
4
5
6
0.2
0.4
0.6
0.8
N = 24640 Bandwidth = 0.01233


December Phi2 - Normalized Against Flats
All Flats - Normalized
Flat 1
Flat 2
Flat 3
Flat 4
Density
0
1
2
3
4
5
0.2
0.4
0.6
0.8
N = 23294 Bandwidth = 0.01494


December Leafarea - Normalized Against Flats
Density
0.000
0.002
0.004
0
100
200
300
400
500
600
700
All Flats - Normalized
Flat 1
Flat 2
Flat 3
Flat 4
N = 23296 Bandwidth = 10.76


January Leafarea - Normalized Against Flats
Density
0.0000
0.0010
0.0020
0.0030
0
200
400
600
800
All Flats - Normalized
Flat 1
Flat 2
Flat 3
Flat 4
N = 24640 Bandwidth = 16.02




### Investigate - How to go from normalized data back to data frame?

Now, I have a problem. I still want to associate each measured value with a time point and genotype. But, the matrix data type can only hold integer values. So, is there a way to move backwards from the quantile normalized matrix to the data frame with all plant ID information?

Some clues:

An example:

```
library(preprocessCore)
# the function expects a matrix create a
# matrix using the same example
mat <- matrix(c(5, 2, 3, 4, 4, 1, 4, 2, 3,
    4, 6, 8), ncol = 3)
mat
```

```
##      [,1] [,2] [,3]
## [1,]    5    4    3
## [2,]    2    1    4
## [3,]    3    4    6
## [4,]    4    2    8
```

```
norm_mat <- normalize.quantiles(mat)
norm_mat
```

```
##          [,1]     [,2]     [,3]
## [1,] 5.666667 5.166667 2.000000
## [2,] 2.000000 2.000000 3.000000
## [3,] 3.000000 5.166667 4.666667
## [4,] 4.666667 3.000000 5.666667
```

So, it appears that the order of the values is maintained in normalize.quantiles.

For example, the third column is increasing as the rows increase for both the original and normalized matrix.

```
invest <- cbind(as.data.frame(mat), as.data.frame(norm_mat))
colnames(invest) <- c("Original_1", "Original_2",
    "Original_3", "Normalized_1", "Normalized_2",
    "Normalized_3")

invest
```

```
##   Original_1 Original_2 Original_3 Normalized_1 Normalized_2 Normalized_3
## 1          5          4          3     5.666667     5.166667     2.000000
## 2          2          1          4     2.000000     2.000000     3.000000
## 3          3          4          6     3.000000     5.166667     4.666667
## 4          4          2          8     4.666667     3.000000     5.666667
```

So, in this case, I would replace the original columns with the normalized columns. The order would be the same.

```
invest %>% arrange(Original_1)
```

```
##   Original_1 Original_2 Original_3 Normalized_1 Normalized_2 Normalized_3
## 1          2          1          4     2.000000     2.000000     3.000000
## 2          3          4          6     3.000000     5.166667     4.666667
## 3          4          2          8     4.666667     3.000000     5.666667
## 4          5          4          3     5.666667     5.166667     2.000000
```

So, the order of Original_1 is always the same as the order of Normalized_1.

```
temp_test <- depi_all %>% filter(measurement ==
    "npq", month == "Jan", flat_number ==
    "1") %>% select(measurement, month, flat_number,
    measured_value, individual_plant_metadata)

temp_test <- cbind(temp_test, na.omit(jan_npq_normalize$flat_1))


colnames(temp_test)[6] <- "normalized_values"

head(temp_test)
```

```
##         measurement month flat_number measured_value individual_plant_metadata
## 1035531         npq   Jan           1         0.4679      0118_F_DEPI_SC_1_3_6
## 1148341         npq   Jan           1         0.4764      0118_F_DEPI_SC_1_5_7
## 1276171         npq   Jan           1         0.5322      0118_F_DEPI_SC_1_8_4
```

```
## 1840171         npq   Jan           1         0.5134    0118_F_DEPI_SC_1_18_10
## 1035541         npq   Jan           1         0.4811      0118_F_DEPI_SC_1_3_6
## 1148351         npq   Jan           1         0.4783      0118_F_DEPI_SC_1_5_7
##         normalized_values
## 1035531         0.4994250
## 1148341         0.5077958
## 1276171         0.5729684
## 1840171         0.5488284
## 1035541         0.5123259
## 1148351         0.5095708
```

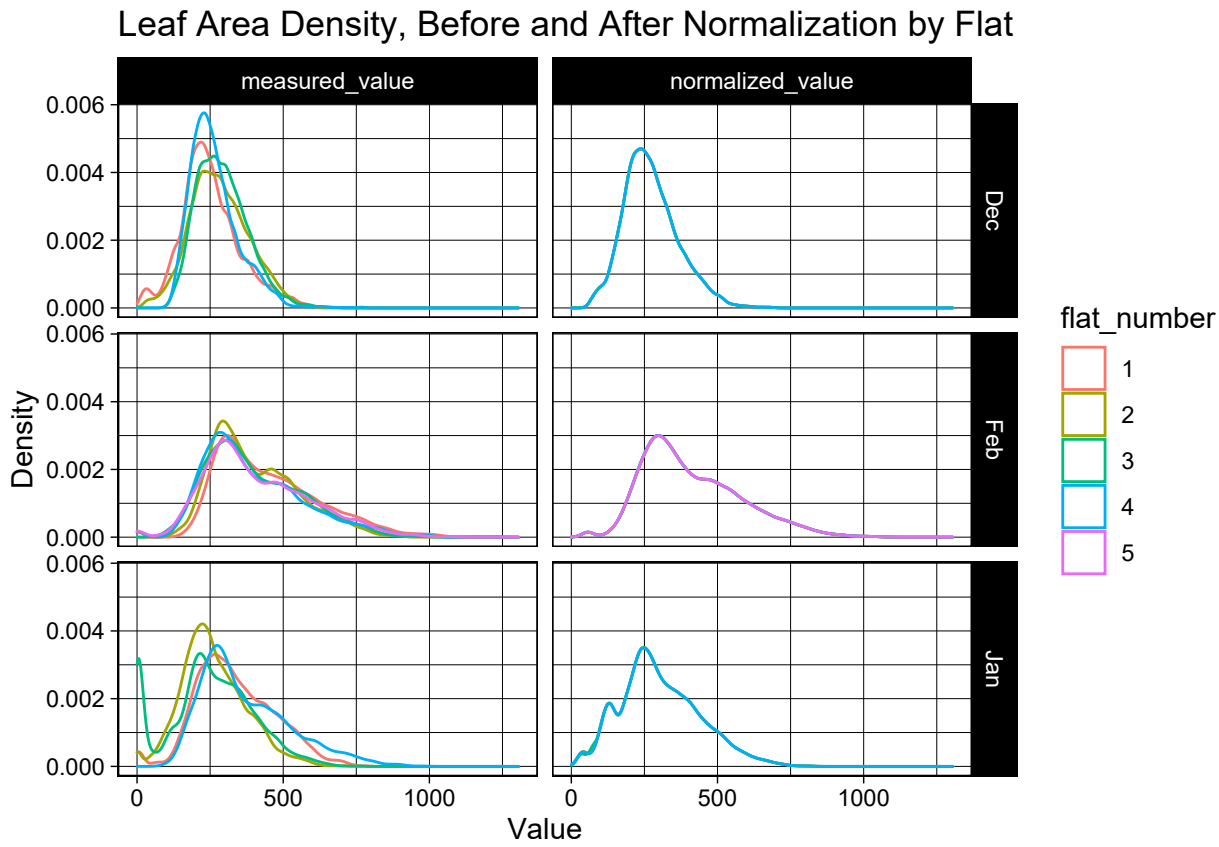
I still need to verify that this is actually associating each quantile normalized value with the correct measured value.

```
### Create a loop for each measurement and
### experiment
for (i in c("npq", "phi2", "leafarea")) {
    for (j in c("Dec", "Jan", "Feb")) {
        ### Filter to select each specific month
        ### and measured value
        temp_vector <- filter(depi_all, month ==
            j, measurement == i)
        ### Initialize an empty data frame
        temp_df <- data.frame()
        ### Loop through each flat
        for (k in 1:length(unique(temp_vector$flat_number))) {
            ### Create a temporary data frame - each
            ### column is the measured values for each
            ### flat
            temp <- filter(temp_vector, flat_number ==
                k)$measured_value
            temp_df <- addToDF(temp_df, temp)
        }
        ### Normalize across the flats
        temp_normalize <- as.data.frame(normalize.quantiles(as.matrix(temp_df))) %>%
            ### Add columns with the measurement and
        ### experiment to be certain there hasn't
        ### been any mix-ups
        mutate(measurement_verify = i, experiment_verify = j)
        ### Create a name to give the normalized
        ### data based on the measurement and
        ### experiment
        temp_name <- paste(tolower(j), "_",
            i, "_normalize", sep = "")
        ### Rename the columns
        if (ncol(temp_normalize) == 6) {
            temp_normalize <- temp_normalize %>%
                rename(flat_1 = V1, flat_2 = V2,
                  flat_3 = V3, flat_4 = V4)
        } else {
            temp_normalize <- temp_normalize %>%
                rename(flat_1 = V1, flat_2 = V2,
                  flat_3 = V3, flat_4 = V4,
                  flat_5 = 5)
```

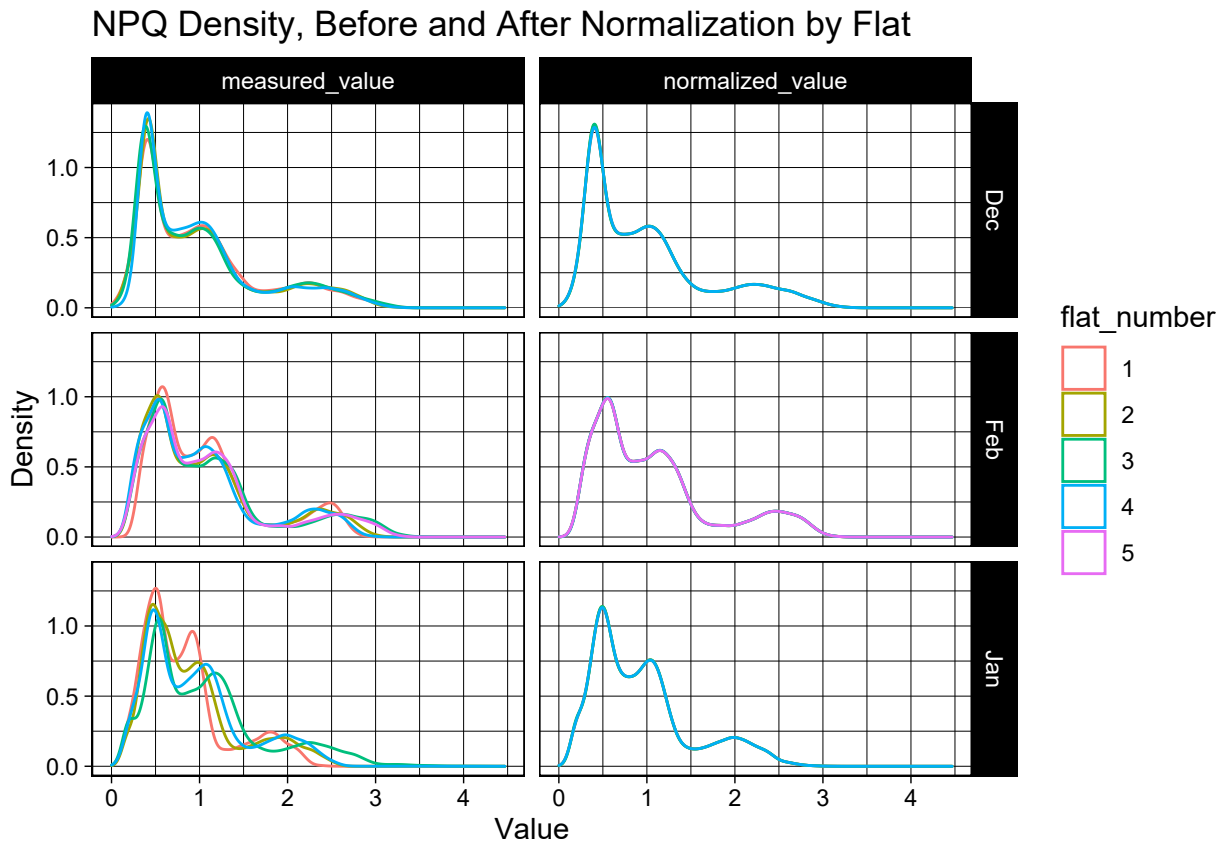
```
        }
        ### Assign the name to the data frame
        assign(temp_name, temp_normalize)
        ### Loop through each of the columns that
        ### have measured values for each flat
        for (num in 1:(ncol(temp_normalize) -
            2)) {
            ### Filter the relevant matching rows from
            ### the depi_all dataframe
            temp_depi_information <- depi_all %>%
                filter(measurement == unique(temp_normalize$measurement),
                  month == unique(temp_normalize$experiment),
                  flat_number == num)
            ### Add the column with the normalized data
            ### to the depi subset
            temp_merged <- cbind(temp_depi_information,
                na.omit(temp_normalize[num]))
            ### Create a name for this data frame -
            ### based on the measurement, month, and
            ### flat
            temp_name_final <- paste(temp_name,
                "_flat_", num, sep = "")

            names(temp_merged)[length(names(temp_merged))] <- "normalized_value"
            assign(temp_name_final, temp_merged)

        }
    }
}
### Combine all seperate data frames
quantile_normalize_all <- rbind(dec_leafarea_normalize_flat_1,
    dec_leafarea_normalize_flat_2, dec_leafarea_normalize_flat_3,
    dec_leafarea_normalize_flat_4, jan_leafarea_normalize_flat_1,
    jan_leafarea_normalize_flat_2, jan_leafarea_normalize_flat_3,
    jan_leafarea_normalize_flat_4, feb_leafarea_normalize_flat_1,
    feb_leafarea_normalize_flat_2, feb_leafarea_normalize_flat_3,
    feb_leafarea_normalize_flat_4, feb_leafarea_normalize_flat_5,
    dec_npq_normalize_flat_1, dec_npq_normalize_flat_2,
    dec_npq_normalize_flat_3, dec_npq_normalize_flat_4,
    jan_npq_normalize_flat_1, jan_npq_normalize_flat_2,
    jan_npq_normalize_flat_3, jan_npq_normalize_flat_4,
    feb_npq_normalize_flat_1, feb_npq_normalize_flat_2,
    feb_npq_normalize_flat_3, feb_npq_normalize_flat_4,
    feb_npq_normalize_flat_5, dec_phi2_normalize_flat_1,
    dec_phi2_normalize_flat_2, dec_phi2_normalize_flat_3,
    dec_phi2_normalize_flat_4, jan_phi2_normalize_flat_1,
    jan_phi2_normalize_flat_2, jan_phi2_normalize_flat_3,
    jan_phi2_normalize_flat_4, feb_phi2_normalize_flat_1,
    feb_phi2_normalize_flat_2, feb_phi2_normalize_flat_3,
    feb_phi2_normalize_flat_4, feb_phi2_normalize_flat_5)
```

```
plot_data <- quantile_normalize_all %>% gather(key = "type",
    value = "value", normalized_value, measured_value)
```

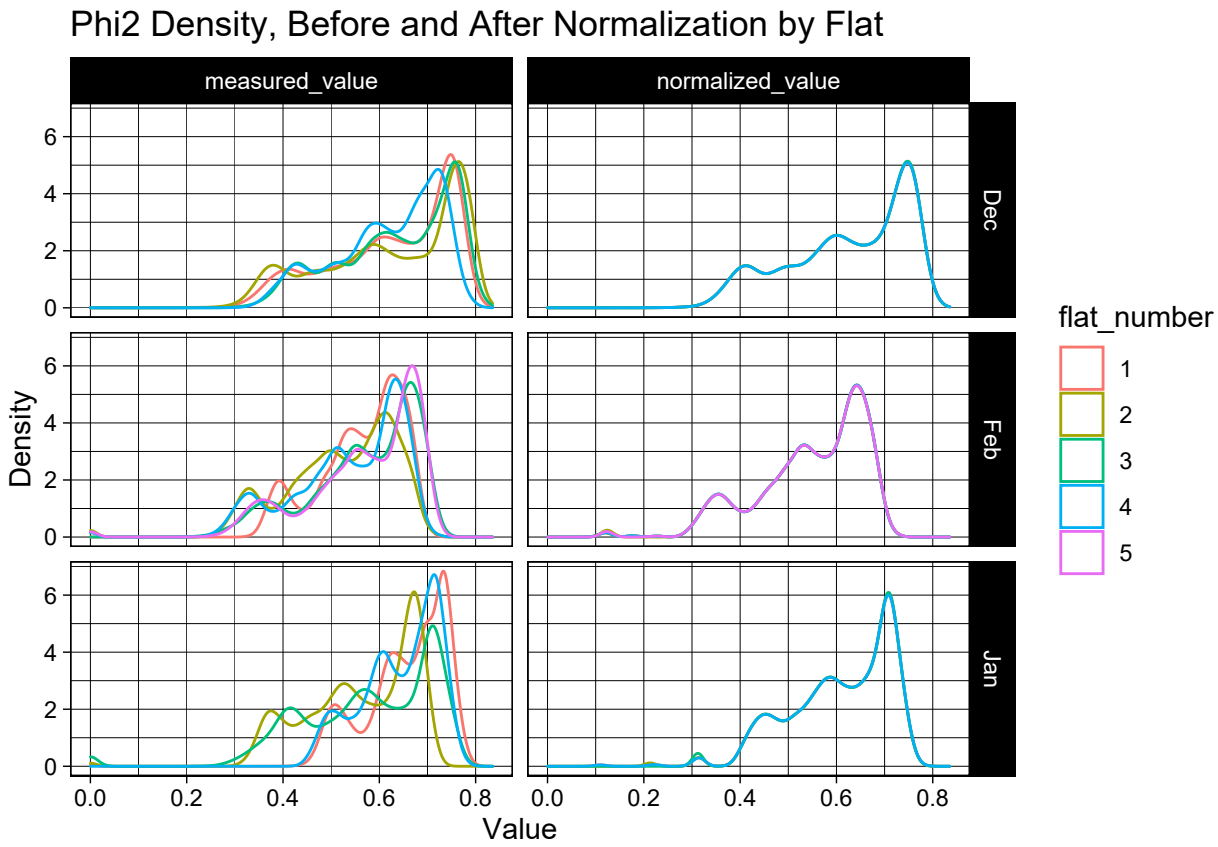
```
plot_data$flat_number <- as.factor(plot_data$flat_number)

plot_data_temp <- plot_data %>% filter(measurement ==
    "leafarea")
plot <- ggplot(data = plot_data_temp, aes(x = value,
    color = flat_number)) + geom_density() +
    facet_grid(month ~ type) + theme_linedraw() +
    labs(title = "Leaf Area Density, Before and After Normalization by Flat",
        x = "Value", y = "Density")
print(plot)
```




```
plot_data_temp <- plot_data %>% filter(measurement ==
    "npq")
plot <- ggplot(data = plot_data_temp, aes(x = value,
    color = flat_number)) + geom_density() +
    facet_grid(month ~ type) + theme_linedraw() +
    labs(title = "NPQ Density, Before and After Normalization by Flat",
        x = "Value", y = "Density")
print(plot)
```




```
plot_data_temp <- plot_data %>% filter(measurement ==
    "phi2")
plot <- ggplot(data = plot_data_temp, aes(x = value,
    color = flat_number)) + geom_density() +
    facet_grid(month ~ type) + theme_linedraw() +
    labs(title = "Phi2 Density, Before and After Normalization by Flat",
        x = "Value", y = "Density")
print(plot)
```




All of these plots look good!! The normalized plot that looks the worst is January Phi2. But, it looks like the minimum value for each flat is pretty variant - this could explain why the quantile normalization isn't as smooth (flats 1 and 4 behave similarly, and flats 2 and 3 behave similarly, but each pair is variant to the other).

Okay, now the final step of this normalize process is to divide by the median value of Col0 on each flat. **I am dividing the normalized values by the median normalized value of Col0 - make sure this is correct!**

```
test <- quantile_normalize_all %>% group_by(month,
    measurement, flat_number) %>% mutate_each(funs(./median(.[genotype ==
    "Col0"])), normalized_value)
```

```
quantile_normalize_all %>% filter(time_point ==
    "0", measurement == "npq") %>% group_by(flat_number,
    genotype, month) %>% tally() %>% filter(genotype ==
    "Col0")
```

```
## # A tibble: 13 x 4
## # Groups:   flat_number, genotype [5]
##    flat_number genotype month     n
##          <int> <chr>    <chr> <int>
##  1           1 Col0     Dec       1
##  2           1 Col0     Feb       2
##  3           1 Col0     Jan       4
##  4           2 Col0     Dec       9
```

```
##  5           2 Col0     Feb       5
##  6           2 Col0     Jan       4
##  7           3 Col0     Dec       1
##  8           3 Col0     Feb       3
##  9           3 Col0     Jan       4
## 10           4 Col0     Dec       2
## 11           4 Col0     Feb       3
## 12           4 Col0     Jan       1
## 13           5 Col0     Feb       3
```

I think we acknowledged that this was a problem before - there isn't always a wild type plant on each flat! And, in some cases there is only 1 wild type plant per flat. So, this isn't going to work.

```
depi_all %>% filter(time_point == "0", measurement ==
    "npq") %>% group_by(month, flat_number) %>%
    summarize(length(unique(individual_plant_metadata)))
```

```
## # A tibble: 13 x 3
## # Groups:   month [3]
##    month flat_number `length(unique(individual_plant_metadata))`
##    <chr>       <int>                                       <int>
##  1 Dec             1                                         104
##  2 Dec             2                                         129
##  3 Dec             3                                         135
##  4 Dec             4                                         107
##  5 Feb             1                                         109
##  6 Feb             2                                         130
##  7 Feb             3                                         124
##  8 Feb             4                                         130
##  9 Feb             5                                         108
## 10 Jan             1                                         110
## 11 Jan             2                                         135
## 12 Jan             3                                         135
## 13 Jan             4                                         108
```

Note that we excluded sublines 2, 3, and 4 of Col0, and genotypes B1, B3, B1B3, ftsz-1, ftsz-2, and ftsz-dbl, which explains why the flats appear less full than they actually were.

One possible cause of this could be because I filtered to only include subline 1 of Col0. Maybe ask Melissa if we could include other sublines - might make it possible to divide each flat by the median value of Col0 then.

```
unique(filter(depi_all, genotype == "Col0")$subline)
```

```
## [1] 4 2 1 3
```

## Summary

We expect to see each flat having the same summary statistics for each month and measurment!!!

```
quantile_normalize_all %>% group_by(month,
    measurement, flat_number) %>% summarize(min(normalized_value),
    max(normalized_value), median(normalized_value),
    quantile(normalized_value, 0.25))
```

```
## # A tibble: 39 x 7
## # Groups:   month, measurement [9]
##    month measurement flat_number `min(normalized~ `max(normalized~
##    <chr> <chr>              <int>            <dbl>            <dbl>
##  1 Dec   leafarea               1          49.2             700
##  2 Dec   leafarea               2          49.2             700
##  3 Dec   leafarea               3          49.2             700
##  4 Dec   leafarea               4          49.2             700
##  5 Dec   npq                    1           0.0133            3.36
##  6 Dec   npq                    2           0.0133            3.36
##  7 Dec   npq                    3           0.0133            3.36
##  8 Dec   npq                    4           0.0133            3.36
##  9 Dec   phi2                   1           0.153             0.827
## 10 Dec   phi2                   2           0.153             0.827
## # ... with 29 more rows, and 2 more variables:
## #   `median(normalized_value)` <dbl>, `quantile(normalized_value, 0.25)` <dbl>
```

## Normalization - Experiment (not flat) level

Here, instead of normalizing by the flat level, I will normalize by the experiment level.

Does this mean I want the distributions of each experiment to be the same? Ask Melissa.

```
### Create a loop for each measurement and
### experiment
for (i in c("npq", "phi2", "leafarea")) {
    ### Filter to select each specific measured
    ### value
    temp_vector <- filter(depi_all, measurement ==
        i)
    ### Initialize an empty data frame
    temp_df <- data.frame()
    ### Loop through each flat
    month_list <- c("Dec", "Feb", "Jan")

    for (k in 1:length(unique(temp_vector$month))) {
        ### Create a temporary data frame - each
        ### column is the measured values for each
        ### flat
        temp <- filter(temp_vector, month ==
            month_list[k])$measured_value
        temp_df <- addToDF(temp_df, temp)
    }
    ### Normalize across the flats
    temp_normalize <- as.data.frame(normalize.quantiles(as.matrix(temp_df))) %>%
        ### Add columns with the measurement and
    ### experiment to be certain there hasn't
    ### been any mix-ups
```

```
    mutate(measurement_verify = i)
    ### Create a name to give the normalized
    ### data based on the measurement and
    ### experiment
    temp_name <- paste(i, "_normalize", sep = "")
    ### Rename the columns
    temp_normalize <- temp_normalize %>%
        rename(Dec = V1, Feb = V2, Jan = V3)
    ### Assign the name to the data frame
    assign(temp_name, temp_normalize)

    month_list <- c("Dec", "Feb", "Jan")
    ### Loop through each of the columns that
    ### have measured values for each flat
    for (num in 1:(ncol(temp_normalize) -
        1)) {
        ### Filter the relevant matching rows from
        ### the depi_all dataframe
        temp_depi_information <- depi_all %>%
            filter(measurement == unique(temp_normalize$measurement_verify),
                month == month_list[num])
        ### Add the column with the normalized data
        ### to the depi subset
        temp_merged <- cbind(temp_depi_information,
            na.omit(temp_normalize[num]))
        ### Create a name for this data frame -
        ### based on the measurement, month, and
        ### flat
        temp_name_final <- paste(temp_name,
            "_", tolower(month_list[num]),
            sep = "")

        names(temp_merged)[length(names(temp_merged))] <- "normalized_value"
        assign(temp_name_final, temp_merged)
    }
}

normalize_all_exp <- rbind(leafarea_normalize_jan,
    leafarea_normalize_dec, leafarea_normalize_feb,
    npq_normalize_jan, npq_normalize_dec,
    npq_normalize_feb, phi2_normalize_jan,
    phi2_normalize_feb, phi2_normalize_dec)
```

Now, check to make sure this actually did what we wanted it to. We should see the same summary statistics for each measurement across the three experiments!

```
normalize_all_exp %>% group_by(measurement,
    month) %>% arrange(measurement) %>% summarize(min(normalized_value),
    median(normalized_value), max(normalized_value),
    quantile(normalized_value, 0.25), quantile(normalized_value,
        0.75))
```

```
## # A tibble: 9 x 7
```

```
## # Groups:   measurement [3]
##   measurement month `min(normalized~ `median(normali~ `max(normalized~
##   <chr>       <chr>            <dbl>            <dbl>            <dbl>
## 1 leafarea    Dec             1.67            308             1026
## 2 leafarea    Feb             4.67            307.            1026
## 3 leafarea    Jan             9.33            307.            1017.
## 4 npq         Dec             0.0117            0.836            3.85
## 5 npq         Feb             0.0117            0.836            3.85
## 6 npq         Jan             0.0117            0.836            3.85
## 7 phi2        Dec             0                 0.618            0.797
## 8 phi2        Feb             0.101             0.618            0.797
## 9 phi2        Jan             0.103             0.618            0.797
## # ... with 2 more variables: `quantile(normalized_value, 0.25)` <dbl>,
## #   `quantile(normalized_value, 0.75)` <dbl>
```

Everything looks acceptable - although the minimum values are fairly variant for leafarea.

Now, I need to make sure that the quantile normalized values are correctly merging with the rest of the data from the depi data frame.

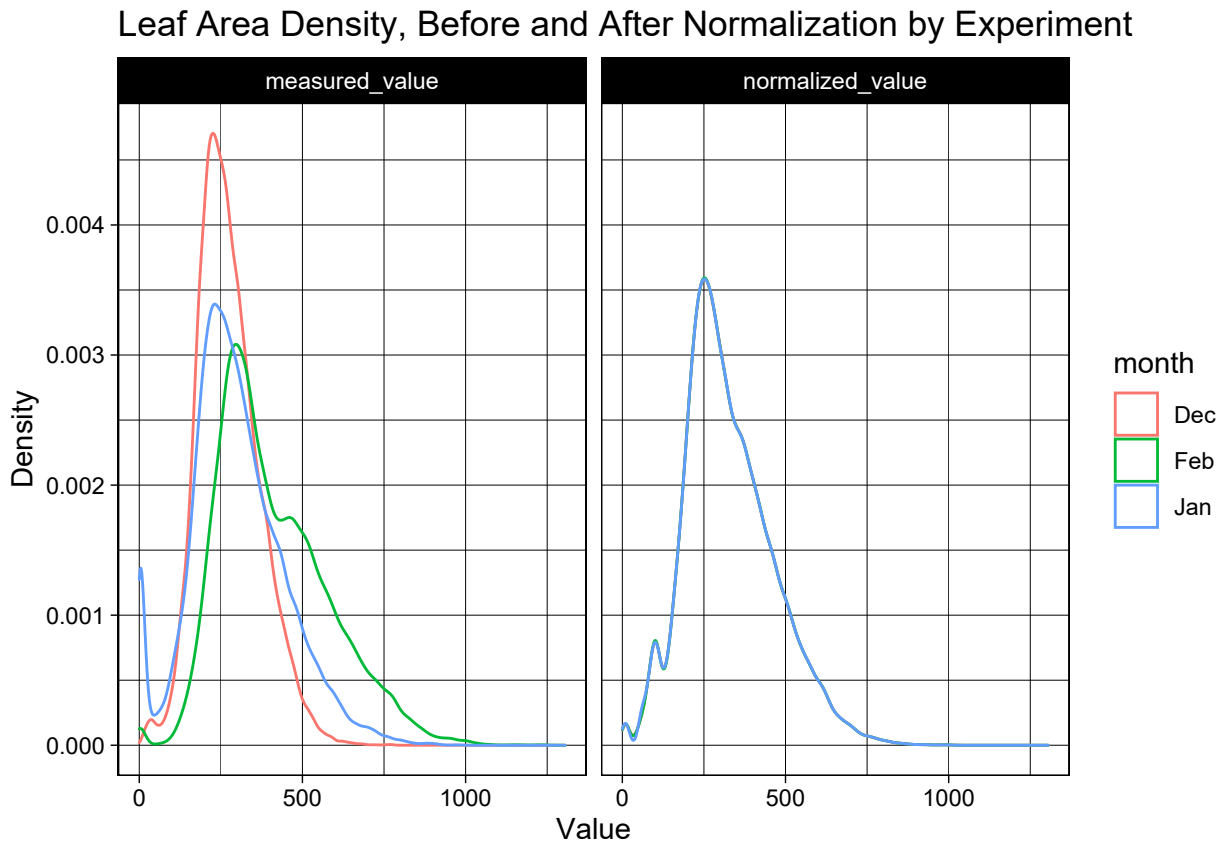
Again, to check this, I'll filter to each measurement and month, and then sort the measured values. If I merged everything correctly, the quantile normalized values should automatically sort as well!

```
for (i in unique(normalize_all_exp$measurement)) {
    for (j in unique(normalize_all_exp$month)) {
        temp <- filter(normalize_all_exp,
            measurement == i, month == j) %>%
            arrange(measured_value)
        print(is.unsorted(temp$measured_value))
        print(is.unsorted(temp$normalized_value))
    }
}
```

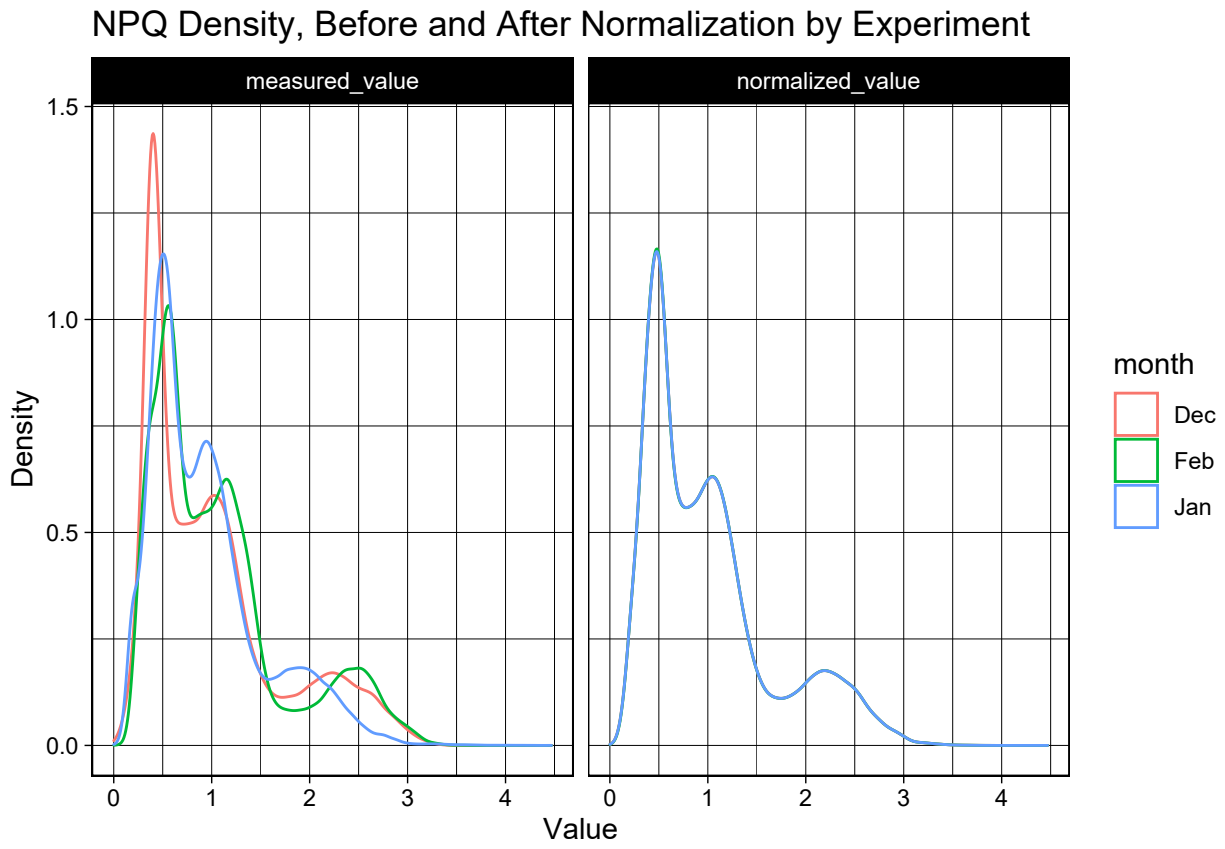
```
## [1] FALSE
## [1] FALSE
## [1] FALSE
## [1] FALSE
## [1] FALSE
## [1] FALSE
## [1] FALSE
## [1] FALSE
## [1] FALSE
## [1] FALSE
## [1] FALSE
## [1] FALSE
## [1] FALSE
## [1] FALSE
## [1] FALSE
## [1] FALSE
## [1] FALSE
## [1] FALSE
## [1] FALSE
```

Okay - so when the measured values are sorted, the normalized values are sorted as well. (The FALSE seems counterintuitive. A FALSE for is.unsorted means that the data is sorted - which is the result we want. There is no is.sorted function.)

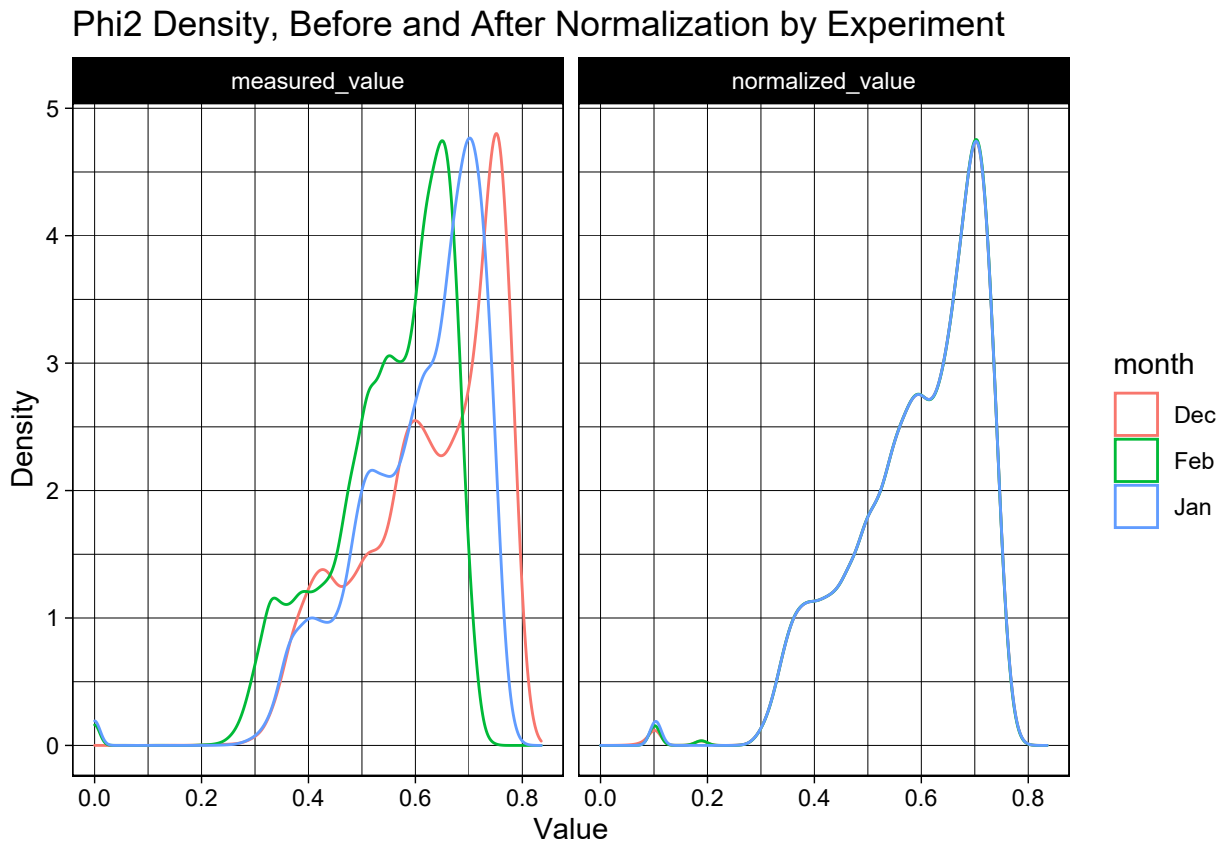
```
plot_data <- normalize_all_exp %>% gather(key = "type",
    value = "value", normalized_value, measured_value)

plot_data_temp <- plot_data %>% filter(measurement ==
    "leafarea")
ggplot(data = plot_data_temp, aes(x = value,
    color = month)) + geom_density() + facet_grid(~type) +
    theme_linedraw() + labs(title = "Leaf Area Density, Before and After Normalization by Experiment",
    x = "Value", y = "Density")
```




```
plot_data_temp <- plot_data %>% filter(measurement ==
    "npq")
ggplot(data = plot_data_temp, aes(x = value,
    color = month)) + geom_density() + facet_grid(~type) +
    theme_linedraw() + labs(title = "NPQ Density, Before and After Normalization by Experiment",
    x = "Value", y = "Density")
```




```
plot_data_temp <- plot_data %>% filter(measurement ==
    "phi2")
ggplot(data = plot_data_temp, aes(x = value,
    color = month)) + geom_density() + facet_grid(~type) +
    theme_linedraw() + labs(title = "Phi2 Density, Before and After Normalization by Experiment",
    x = "Value", y = "Density")
```




Comments:

The phi2 plot is really interesting!!! It looks like each experiment has the same density shape, just shifted from each other on the x-axis.

Note that it looks like we can only see one month plotted on the right (January) instead of all three. This makes sense - after we normalize, the density for all three experiments should be the same, which is why all three lines will lie on top of each other.

Now, the next step is to divide each measured value by the median value of wild type at the same time point on a flat scale.

But, we run into a problem - there aren't always Col0 on a flat. And, in some cases, there is only one wild type plant on a flat.

```
depi_all %>% filter(time_point == "0", measurement ==
    "npq") %>% group_by(month, genotype,
    flat_number) %>% summarize(length(unique(individual_plant_metadata)))
```

```
## # A tibble: 473 x 4
## # Groups:   month, genotype [113]
##    month genotype flat_number `length(unique(individual_plant_metadata))`
##    <chr> <chr>          <int>                                        <int>
##  1 Dec   Col0               1                                            1
##  2 Dec   Col0               2                                            9
##  3 Dec   Col0               3                                            1
##  4 Dec   Col0               4                                            2
##  5 Dec   mpk1               1                                            3
```

```
##  6 Dec   mpk1              2                                          3
##  7 Dec   mpk1              3                                          3
##  8 Dec   mpk1              4                                          4
##  9 Dec   mpk1-13           1                                          1
## 10 Dec   mpk1-13           2                                          4
## # ... with 463 more rows
```

Just how many plants are on each flat? I would expect each flat to have at least one wildtype plant.

```
depi_all %>% group_by(month, flat_number) %>%
    filter(time_point == "0", measurement ==
        "npq") %>% summarize(length(unique(individual_plant_metadata)))
```

```
## # A tibble: 13 x 3
## # Groups:   month [3]
##    month flat_number `length(unique(individual_plant_metadata))`
##    <chr>       <int>                                       <int>
##  1 Dec             1                                         104
##  2 Dec             2                                         129
##  3 Dec             3                                         135
##  4 Dec             4                                         107
##  5 Feb             1                                         109
##  6 Feb             2                                         130
##  7 Feb             3                                         124
##  8 Feb             4                                         130
##  9 Feb             5                                         108
## 10 Jan             1                                         110
## 11 Jan             2                                         135
## 12 Jan             3                                         135
## 13 Jan             4                                         108
```

Okay - I think each flat holds 200 flats. But, we removed Col0 2-X, and all gene familes that were not mpk (ftsz and B1B3). So, this could partly explain why the flats appear to only be a little over half full.

##Trouble Shooting

What is the problem?

I was subsetting the data as a test by only selecting Col0. But, I didn't get rid of this test, and I assigned it to a dataframe that I used to normalize. So, I was only normalizing Col0 genotypes, and this led to flat disappearing.

**Fixed!! Just made a really dumb mistake...**

```
unique(filter(depi_all, month == "Jan")$flat_number)
```

```
## [1] 1 2 3 4
```

```
unique(filter(depi_all, month == "Feb")$flat_number)
```

```
## [1] 1 2 3 4 5
```

```
unique(filter(depi_all, month == "Dec")$flat_number)
```

```
## [1] 1 2 3 4
```

Before I filtered to the correct sublines of Col0, January had 4 flats! Where did this missing flat go?

```
unique(jan_data$flat_number)
```

```
## [1] 1 2 3 4
```

```
unique(feb_data$flat_number)
```

```
## [1] 1 2 3 4 5
```

```
unique(dec_data$flat_number)
```

```
## [1] 1 2 3 4
```

Okay - so even the Jan data before we rbind to create depi_all is saying that there are only three flats...

I suspect I made a mistake when merging the jan_data with the individual plant metadata.

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

```
## [1] "Col1_1" "Col1_3" "Col2_2" "Col2_4" "Col1_4" "Col2_1" "Col1_2" "Col2_3"
```

```
depi_jan <- depi_jan %>% filter(full_subline_information %in%
    c("Col1_1", "Col1_3", "Col1_4"))
unique(filter(depi_dec_feb, genotype == "Col0")$line)
```

```
## [1] "Col0"
```

```
unique(depi_jan$flat_number)
```

```
## [1] 1 2 3 4
```

Okay - so before I filter to get the genotypes I want, I have 4 flats.

There is a problem with one disappearing!!

Where does this disappear?

```
jan_data <- depi_jan %>% 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"))) %>%
    filter(border == FALSE) %>% select(individual_plant_metadata,
    genotype, flat_number, measurement, time_point,
    measured_value, border, subline, full_subline_information) %>%
    mutate(month = "Jan")
```

```r
unique(depi_jan$flat_number)
```

```
## [1] 1 2 3 4
```

Okay - so this is still good!

```r
unique(filter(depi_all, month == "Jan")$flat_number)
```

```
## [1] 1 2 3 4
```

But, when I rbind depi_jan to depi_all, the fourth flat disappears...

```r
### Merge the two data frames, only using
### the columns they have in common
depi_all <- rbind(dec_data, jan_data, feb_data) %>%
    arrange(genotype, time_point, month)

unique(filter(depi_all, month == "Jan")$flat_number)
```

## Notes from 8/6 Meeting with Melissa

No Col0 2-anything!! I have to go back through and fix this. (Don't filter by the subline column - look at the entire subline information.)

Get rid of outliers when looking for associations - do it with, and without, and see how they are different.

Next step - divide all genotypes by Col0 (normalize by the normalized median measured value for each flat for each time point.)

I don't have random effects - only fixed.

Anova (capital A!), type 3.

Make Powerpoint for next meeting with Shinhan.

## Linear Model

Another way to normalize the data is to create a linear model, using the experiment and flat number as factors in the model.

```r
library(car)
library(lme4)
# contrasts must be set to contr.sum so
# that Type III Anova is calculated
# correctly
options(contrasts = c("contr.sum", "contr.poly"))

npq_subset <- filter(depi_all, measurement ==
    "npq")

test <- lmer(measured_value ~ (genotype +
    genotype:flat_number + genotype:month +
    genotype:month:flat_number), data = npq_subset)

Anova(test)
```

## Scratch Work

```
for (i in unique(depi_all$genotype)) {
    for (j in unique(depi_all$measurement)) {
        for (k in unique(depi_all$month)) {
            temp <- filter(depi_all, genotype ==
                i, measurement == j, month ==
                k)
            for (l in unique(temp$time_point)) {
                temp <- filter(time_point ==
                  l)
                lmMeasuredValue <- lm(measured_value ~
                  genotype + genotype * flat_number +
                    genotype * month + flat_number *
                    month + genotype * flat_number *
                    month, data = temp)  #Create the linear regression

            }
        }
    }
}
```

Notes on using lm:

Anova(SN.lm, type=3)

Interactions? Don't use type 1

Investigate this:

Error in Anova.III.lm(mod, error, singular.ok = singular.ok, ...) : there are aliased coefficients in the model

```
library(car)
options(contrasts = c("contr.sum", "contr.poly"))
for (i in unique(depi_all$day)) {



    temp <- depi_all %>% filter(time_point ==
        i)
    temp$flat_number <- as.factor(temp$flat_number)
    temp_name <- paste("lmMeasuredValue_",
        i, sep = "")

    temp_lmMeasuredValue <- lm(measured_value ~
        (genotype + genotype * flat_number +
            genotype * month + flat_number *
            month + genotype * flat_number *
            month), data = temp, type = 3)  #Create the linear regression

    Anova(temp_lmMeasuredValue)
    temp_summary <- summary(temp_lmMeasuredValue)
    assign(temp_name, Anova(temp_lmMeasuredValue,
        type = 3))
```

```
}
```