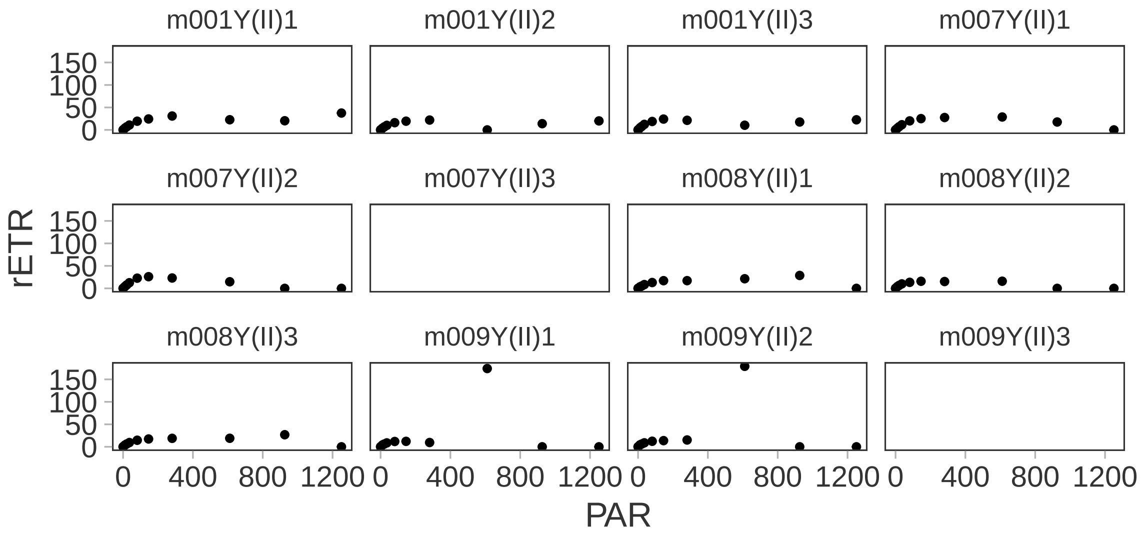
Data extraction, wrangling, and analyses https://github.com/gerard-ricardo/PAM-RLC-analysis.git

Data import and wrangling——————————————————————-

```r
## Load in datasheets.
#Define a function to read CSV files and add a column with the filename
read_csv_filename1 <- function(filename) {
  ret <- read_csv(filename) # Read the CSV file
  ret$disc <- filename # Add filename as a new column
  ret
}
# List all CSV files in the specified directory
filenames <- list.files(path = "./data", full.names = TRUE)
# Read and combine all CSV files into one dataframe
data1 <- ldply(filenames, read_csv_filename1)
data1$disc
# Extract the ID from the filename and add it as a new column
data1 <- data1 %>%
  separate(disc, c("a", "b", "c", "d")) %>%
  mutate(disc1 = c)

### Load in meta data
env_fact <- read.table(file = "https://raw.githubusercontent.com/gerard-ricardo/data/master/postset%20t
# Rename the 'disc' column in environmental data to 'disc1'
#head(env_fact)
env_fact <- env_fact %>% dplyr::rename(disc1 = disc)

## Combine datasheets with meta data
data_2 <- left_join(data1, env_fact, by = 'disc1')
# Select only necessary columns
data_3 <- select(data_2, c('disc1', 'PAR', 'dli', 'spec', 'Y(II)1', 'Y(II)2', 'Y(II)3'))
# Pivot data to long format
data1_long <- data_3 %>%
  pivot_longer(-c(disc1, PAR, dli, spec), names_to = "rep", values_to = "meas")
# Create individual ID for each RLC
data1_long$id <- paste(data1_long$disc1, data1_long$rep, sep = "")

## Create new variables
data1_long$rETR <- data1_long$PAR * data1_long$meas  # Calculate rETR
```

Data exploration and cleaning——————————————————————

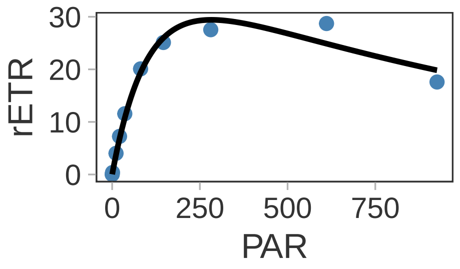
```r
# Raw data plot, faceted by id
p0 <- ggplot(data1_long) +
  geom_point(aes(x = PAR, y = rETR), size = 1) +
  facet_wrap(~id)
p0
```



```
# Remove rETR anomalies
data1_l <- data1_long %>% filter(rETR < 100)
data1_l <- data1_l %>% filter(!(PAR > 400 & rETR < 5))
# Replace non-positive PAR and rETR values
data1_l$PAR <- ifelse(data1_l$PAR <= 0, 0.1, data1_l$PAR)
data1_l$rETR <- ifelse(data1_l$rETR <= 0, 0.01, data1_l$rETR)
```

Single model fit

```
# Extract data by individual ID
data1_s <- subset(data1_l, id %in% 'm007Y(II)1')
# Get starting values for nonlinear fitting
start <- unname(getInitial(rETR ~ SSPlatt.my(PAR, alpha, beta, Pmax), data1_s))
# Fit nonlinear model
md1 <- nlsLM(rETR ~ Pmax * (1 - exp(-alpha * PAR / Pmax)) * (exp(-beta * PAR / Pmax)),
             start = list(Pmax = start[3], alpha = start[1], beta = start[2]),
             data = data1_s, control = nls.lm.control(maxiter = 100))
# Set up new data frame for predictions
df_x <- data.frame(PAR = seq(0.1, 926, length = 1000))
p1 <- ggplot(data1_s) +
  geom_point(aes(x = PAR, y = rETR), size = 2, col = 'steelblue') +
geom_line(df_x, mapping = aes(PAR, predict(md1, df_x)), size = 1)
p1
```



Batch fit

```
# Get starting values for all IDs
starts <- data1_l %>%
  group_by(id) %>%
  do(broom::tidy(stats::getInitial(rETR ~ SSPlatt.my(PAR, alpha, beta, Ys), data = .))) %>%
  pivot_wider(names_from = names, values_from = x, names_prefix = "") %>%
  select(-c('NA'))
# Remove invalid data
starts <- NaRV.omit(starts)
colnames(starts) <- c("id", "alpha_s", 'beta_s', 'Pmax_s')
# Fit nonlinear model for each ID and extract model parameters
batch_fit <- data1_l %>%
  right_join(starts, by = 'id') %>%
  group_by(id) %>%
  do({
    model_try <- tryCatch({
      model <- nlsLM(rETR ~ Pmax * (1 - exp(-alpha * PAR / Pmax)) * (exp(-beta * PAR / Pmax)),
                     start = list(Pmax = mean(.$Pmax_s), alpha = mean(.$alpha_s), beta = mean(.$beta_s)
                     data = ., control = nls.lm.control(maxiter = 100))
      broom::tidy(model)
    }, error = function(e) {
      tibble(term = character(), estimate = double(), std.error = double(), statistic = double(), p.valu
    })
  })
# Convert model parameters to wide format
df_param <- select(batch_fit, c(id, term, estimate)) %>%
  pivot_wider(names_from = term, values_from = estimate)
# Calculate additional parameters
df_param <- df_param %>%
  mutate(ETRm = Pmax * (alpha / (alpha + beta)) * ((beta / (alpha + beta))^(beta / alpha)),
         Ek = ETRm / alpha,
         Em = (Pmax / alpha) * log((alpha + beta) / beta)) %>%
  na.omit()
```