ScEnSor-Kit_Biolector_Analysis_Markdown

Luca Torello Pianale

December 2022

SCOPE OF THE MARKDOWN

This code is used to analyse the data coming from a BioLector I run.
Filter used here were: biomass (E-OP-301), YFP (E-OP-315), CFP (E-OP-309) and RFP (E-OP-319, for mCherry!).
Input file should be a .xlsx file with a sheet with the summary of the plate (called "layout") and one sheet with all the data (called "analysis"). See example.
In my case, even if I try different gains on the same filter, I just analyse one gain, the best one (so the script is adapted to that).
Of course, changes can be made upon need.

SUMMARY PARAMETERS BIOLECTOR RUN

This is not necessary for the script, but it is good to keep track of the setup of the experiment.

RPM - 900
TEMPERATURE - 30*C
HUMUDITY - 85%
PLATE - 96-well plate (Greiner Bio-one)

n* - FILTER - GAIN - Uploaded here (Y/N):
1 - Biomass - 10 - Y
2 - mCherry - 60 - Y
3 - YFP - 40 - Y
4 - CFP - 50 - Y
5 - Not Used -
6 - Not Used -

LIBRARIES

Automated installation (if needed) and loading of libraries required for this script.
The packages "tidyverse" and "rstatix" are needed for smooth data and statistical analysis. The packages "readxl" and "writexl" are needed to import and export the data.
The packages "ggpubr" and "ggplot2" are needed for plotting. The packages "deSolve" and "growthrates" are needed for specific growth rates and lag phases.

```
requiredPackages <- c("tidyverse", "ggplot2", "ggpubr", "readxl", "writexl", "rstatix",
"deSolve", "growthrates")

ipak <- function(pkg){
```

```
        new.pkg <- pkg[!(pkg %in% installed.packages()[, "Package"])]
        if (length(new.pkg))
                install.packages(new.pkg, dependencies = TRUE)
        sapply(pkg, require, character.only = TRUE)
}

ipak(requiredPackages)
```

```
##   tidyverse     ggplot2      ggpubr      readxl     writexl     rstatix
##        TRUE        TRUE        TRUE        TRUE        TRUE        TRUE
##     deSolve growthrates
##        TRUE        TRUE
```

DATA PREPARATION

Load Data

Assign the file path in the computer to the variables "folder" and "file".
The R file should be in the same folder to use command getwd() in this context! Moreover, the file should end with "Analysis" and should be in .xlsx format. Note that, sometimes, the file will not load in R if it is open in excel!

```
folder <- getwd()
setwd(folder)
file <- paste0("/", list.files(folder)[endsWith(list.files(folder), "Analysis.xlsx")])


#Load the summary, which contains the information about the layout of the plate (strain,
sensor used, medium used, etc.) of the screening

summary <- read_excel(paste0(folder, file), sheet="layout")


#Data are loaded.
#In the "analysis" sheet, the first column should contain the wells, the second one the
channels and then one column for each timepoint.
#First row of the document should be: "well", "channel" and all the timepoints (i.e.
number representing the time in hours).

raw_data <- read_excel(paste0(folder, file), sheet = "analysis", col_names = TRUE) %>%
  as.data.frame() %>%
  gather("time", "value", -c(well, channel)) %>%  #One column with time info and a column
  with the value for that timepoint
  mutate_at(c('time', 'value'), as.numeric) %>%
  mutate(across(where(is.numeric), ~ round(.,3))) %>%
#  filter(!(channel %in% c(5, 6))) %>% #If you have channels you want to remove
  mutate(channel = case_when(
    channel == 1 ~ "biomass",
    channel == 2 ~ "rfp_raw",
    channel == 3 ~ "yfp_raw",
    channel == 4 ~ "cfp_raw")) #Pair the channel number with the desired name
```

Analysis

```
#Merge wells-layout information with the data.
#To omit a replicate, put "empty" in the sensor column in the summary

raw_data <- raw_data %>%
  right_join(., summary, by ="well") %>%
  filter(!(sensor == "empty")) %>%  #Filter out empty wells and unwanted replicates
  select(-well) %>%
  pivot_wider(names_from = "channel", values_from = "value") #One column for each channel


#Subtract the background fluorescence from the Parental strain for each medium in each
timepoint.

subtracted_data <- raw_data %>%
  group_by(medium, time) %>%
  mutate(across(.cols = ends_with("_raw"),
                .fns = ~ .- mean(.[sensor == "Parental"]),
                .names = "{.col}_sub")) %>%
  rename_with(~sub("_raw_sub", "", .x, fixed = TRUE))


#Computations on fluorescence data are made and added to the data-frame as new columns.

final_data <- subtracted_data %>%
  mutate(YR = yfp/rfp,
         CR = cfp/rfp) %>%
  #mutate(y = mx + q) %>%                          #Use here calibration curves if needed
  filter(time != min(subtracted_data$time)) %>% #Remove first timepoint
  group_by(sensor, medium, replicate) %>%       #Group using all the variables except
  time
  mutate(YR.Norm = YR/first(YR)) %>%            #Normalise YR by first time value
  mutate(CR.Norm = CR/first(CR))                #Normalise CR by first time value


#Computing mean and sd for all the desired columns (in vector columns_for_meansd).

columns_for_meansd <- setdiff(colnames(final_data), c(colnames(summary), "time"))

summary_data <- final_data %>%
  group_by(sensor, medium, time) %>%  #Group everything except replicate!
  transmute(across(.cols = any_of(columns_for_meansd),
                   .fns = list(mean = ~ mean(.x, na.rm = TRUE),
                               sd = ~ sd(.x, na.rm = TRUE)))) %>%
  distinct()


#Save the data frame now if you are not computing specific growth rates, etc.

sheets <- list("Data" = as.data.frame(final_data),
               "Summary_data" = as.data.frame(summary_data))
#write_xlsx(sheets, path = paste0(folder, "/Data_Analysed.xlsx"), use_zip64 = TRUE)
```

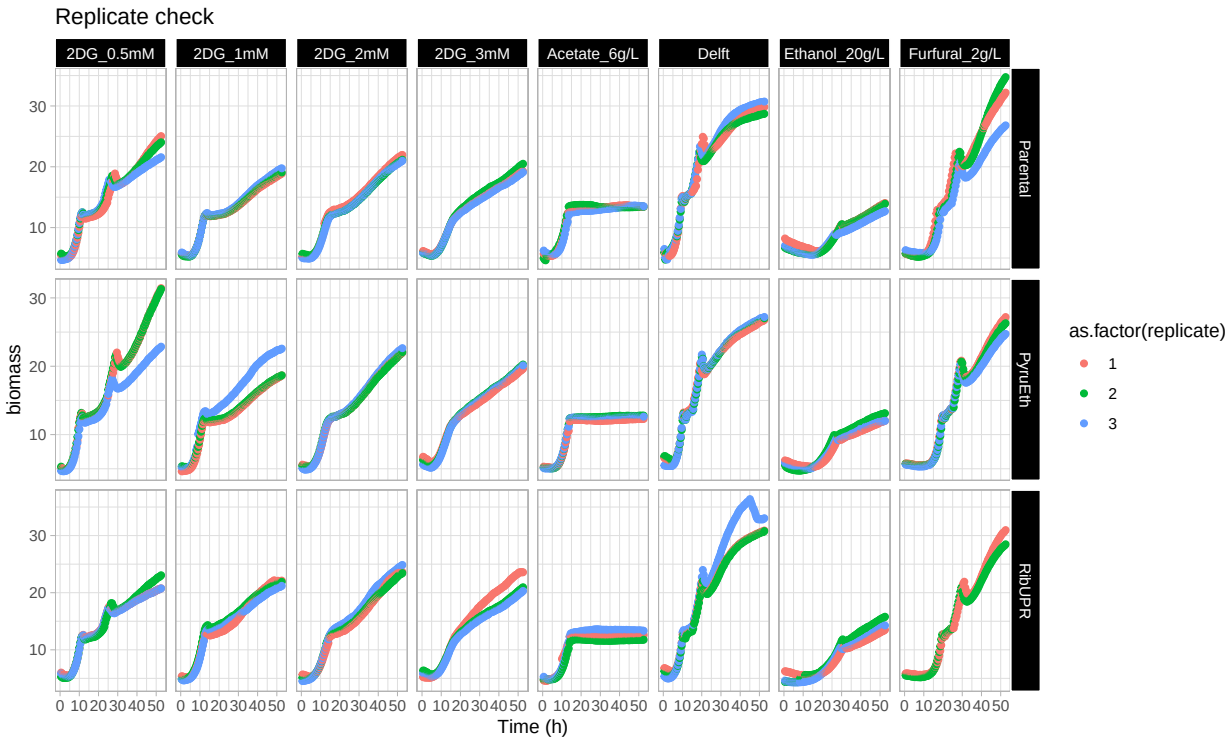
```
rm(subtracted_data, raw_data, sheets, columns_for_meansd, summary) #Clean environment
```

Replicate Check

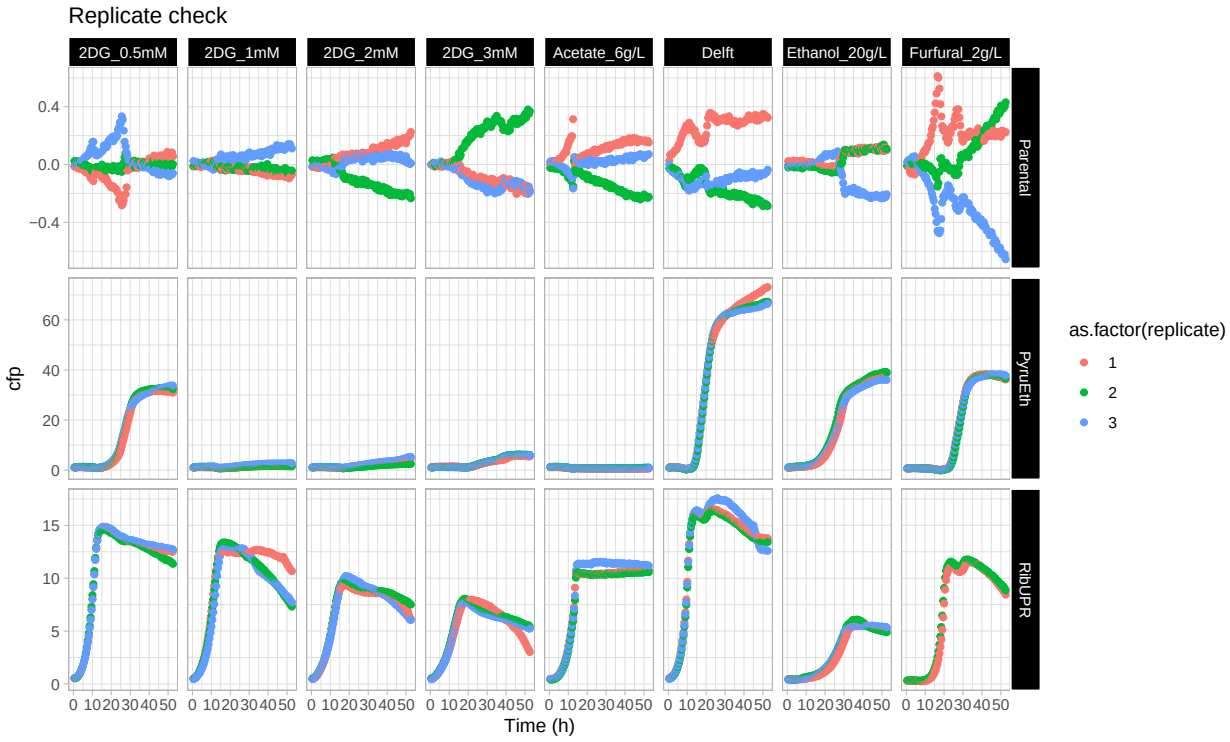
Section to verify the replicates and fluorescence outputs.

```
#Data frame rearrangement for easier plotting.

tmp_plot <- final_data %>%
  pivot_longer(!c(sensor, medium, time, replicate),
               names_to = "Parameter", values_to = "value")

for(i in c("biomass", "rfp", "cfp", "yfp")) { #Replace parameters if needed
  assign(paste0("plot_rep_", i),
         ggplot(data = subset(tmp_plot, Parameter == i)) +
           geom_point(aes(y = value, x = time, colour = as.factor(replicate))) +
           xlim(0, max(unique(final_data$time))) +
           labs(title = "Replicate check", y = i, x = "Time (h)") +
           facet_grid(rows = vars(sensor), cols = vars(medium), scales = "free") +
           theme_light()+
           theme(legend.position = "right",
                 strip.background = element_rect(colour = "white", fill = "black")))
}

mget(ls()[startsWith(ls(), "plot_rep_")]) #Show plots checking the replicates
```

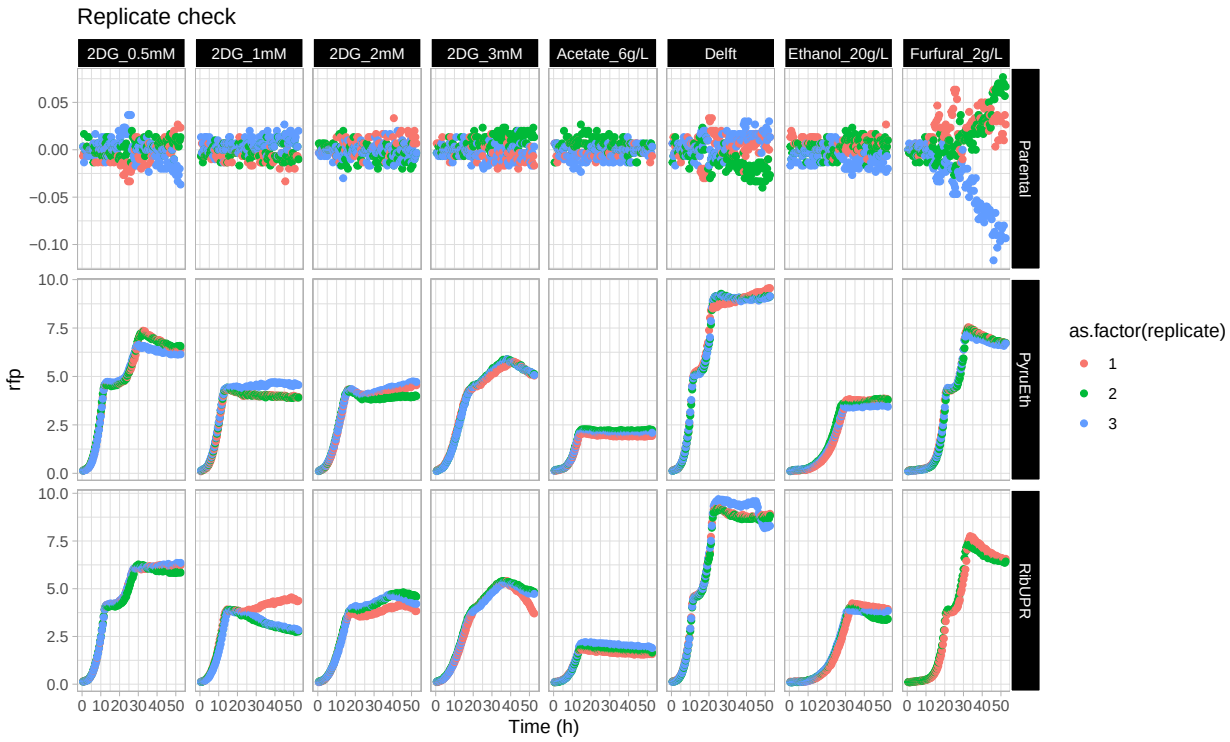
```
## $plot_rep_biomass
```



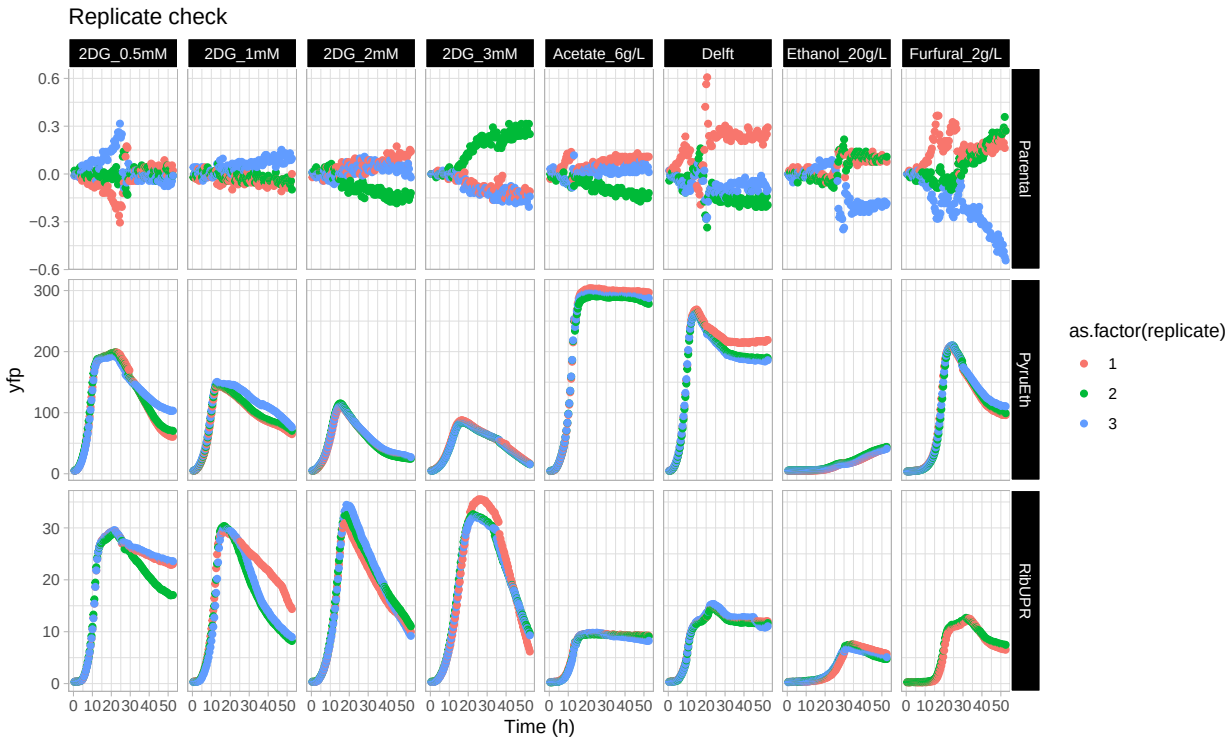

```
##
## $plot_rep_cfp
```



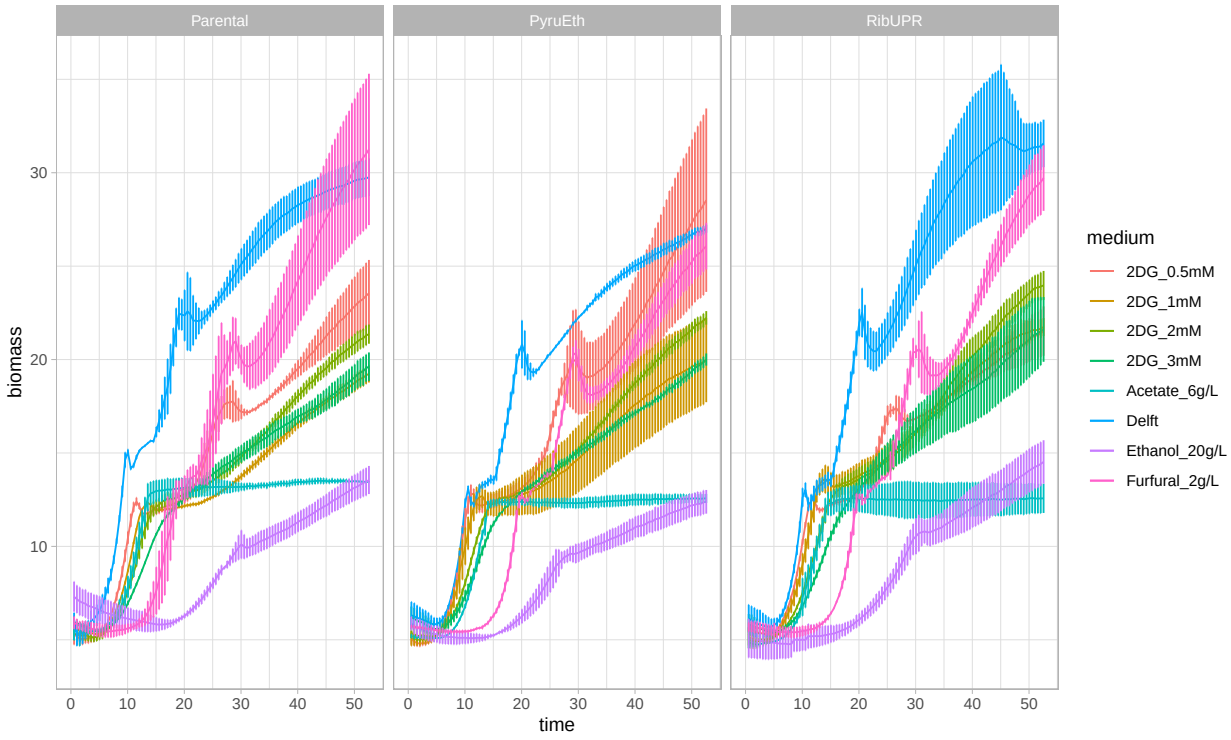

```
##
## $plot_rep_rfp
```


Replicate check
2DG_0.5mM
2DG_1mM
2DG_2mM
2DG_3mM
Acetate_6g/L
Delft
Ethanol_20g/L
Furfural_2g/L
Parental
PyruEth
RibUPR
rfp
Time (h)
as.factor(replicate)
1
2
3

```
##
## $plot_rep_yfp
```


Replicate check
2DG_0.5mM
2DG_1mM
2DG_2mM
2DG_3mM
Acetate_6g/L
Delft
Ethanol_20g/L
Furfural_2g/L
Parental
PyruEth
RibUPR
yfp
Time (h)
as.factor(replicate)
1
2
3

```
rm(tmp_plot, list = ls()[startsWith(ls(), "plot_rep_")])


#Biomass data is plotted to assess the quality of the growth curves.

ggplot(data = final_data, aes(y = biomass, x = time, colour = medium)) +
  stat_summary(fun = mean, geom = "line") +
  stat_summary(fun.data = mean_sdl, geom = "errorbar",
               fun.args = list(mult = 1), width= 0.1) +
  xlim(0, max(unique(final_data$time))) +
  facet_wrap(~ sensor) +
  theme_light()
```



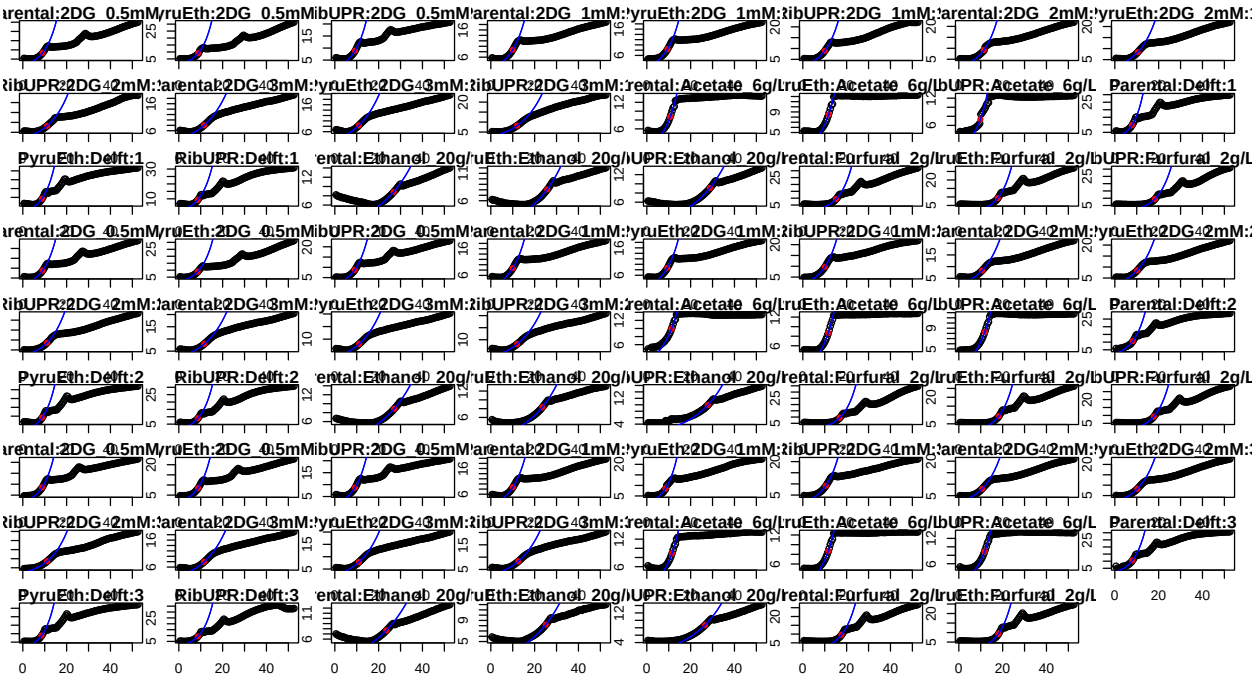
GROWTH CHECK

Computing specific growth rates (mumax) and lag phases + statistical analysis.

```
#Analysis of the growth curves for mumax. Note: in the output graphs the inflection point
and the fitting curve are shown.

many_spline_fits <- all_splines(biomass ~ time | sensor + medium + replicate,
                                data = final_data, spar = 0.45)
par(mfrow = c(12, 8))
par(mar = rep(1, 4))
plot(many_spline_fits)
growth_param <- results(many_spline_fits)   #Saving the results
```

```r
#Calculate lambda (lag phase): the coordinates of the inflection point have been
extracted and used to calculate the tangent and subsequently the x value corresponding to
the lag phase.

inf_points <- NULL

for (i in 1:length(many_spline_fits@fits)){
  tmp <- many_spline_fits@fits[[i]]@xy
  inf_points <- rbind(inf_points, tmp)
}

inf_points <- as.data.frame(inf_points)

growth_param <- bind_cols(growth_param, inf_points) %>%
  rename(., x = V1, y = V2) %>%
  mutate(lag = ((log10(y0) - log10(y)) / mumax) + x) %>%
  pivot_longer(c(lag, mumax), names_to = "Parameter", values_to = "value")


#Re-organise for plotting later on

growth_param$medium <- factor(growth_param$medium, levels = c("Delft", "2DG_0.5mM",
"2DG_1mM", "2DG_2mM", "2DG_3mM", "Furfural_2g/L", "Acetate_6g/L", "Ethanol_20g/L"))
growth_param$sensor <- factor(growth_param$sensor, levels = c("Parental", "PyruEth",
"RibUPR"))


#Statistical comparison of mumax and lag for each medium with respect to the parental
strain and comparison between different media with respect to the control "Delft".

Stats_strain_compare <- growth_param %>%
  group_by(medium, Parameter) %>%
  t_test(value ~ sensor, ref.group = "Parental") %>%
  add_significance("p.adj") %>%
  add_xy_position(x = "medium", step.increase = 0)

Stats_medium_compare <- growth_param %>%
  group_by(Parameter) %>%
  t_test(value ~ medium, ref.group = "Delft") %>%
  add_significance("p.adj") %>%
  add_xy_position(x = "medium", step.increase = 0)


#Export data in an excel sheet (.xlsx). Add to the list all the data frames to be saved
in each sheet.

sheets <- list("Data" = as.data.frame(final_data),
               "Summary_data" = as.data.frame(summary_data),
               "Growth_Param" = as.data.frame(growth_param),
               "Strain_Compare" = as.data.frame(Stats_strain_compare),
               "Medium_Compare" = as.data.frame(Stats_medium_compare))
write_xlsx(sheets, path = paste0(folder, "/Data_Analysed.xlsx"), use_zip64 = TRUE)
```

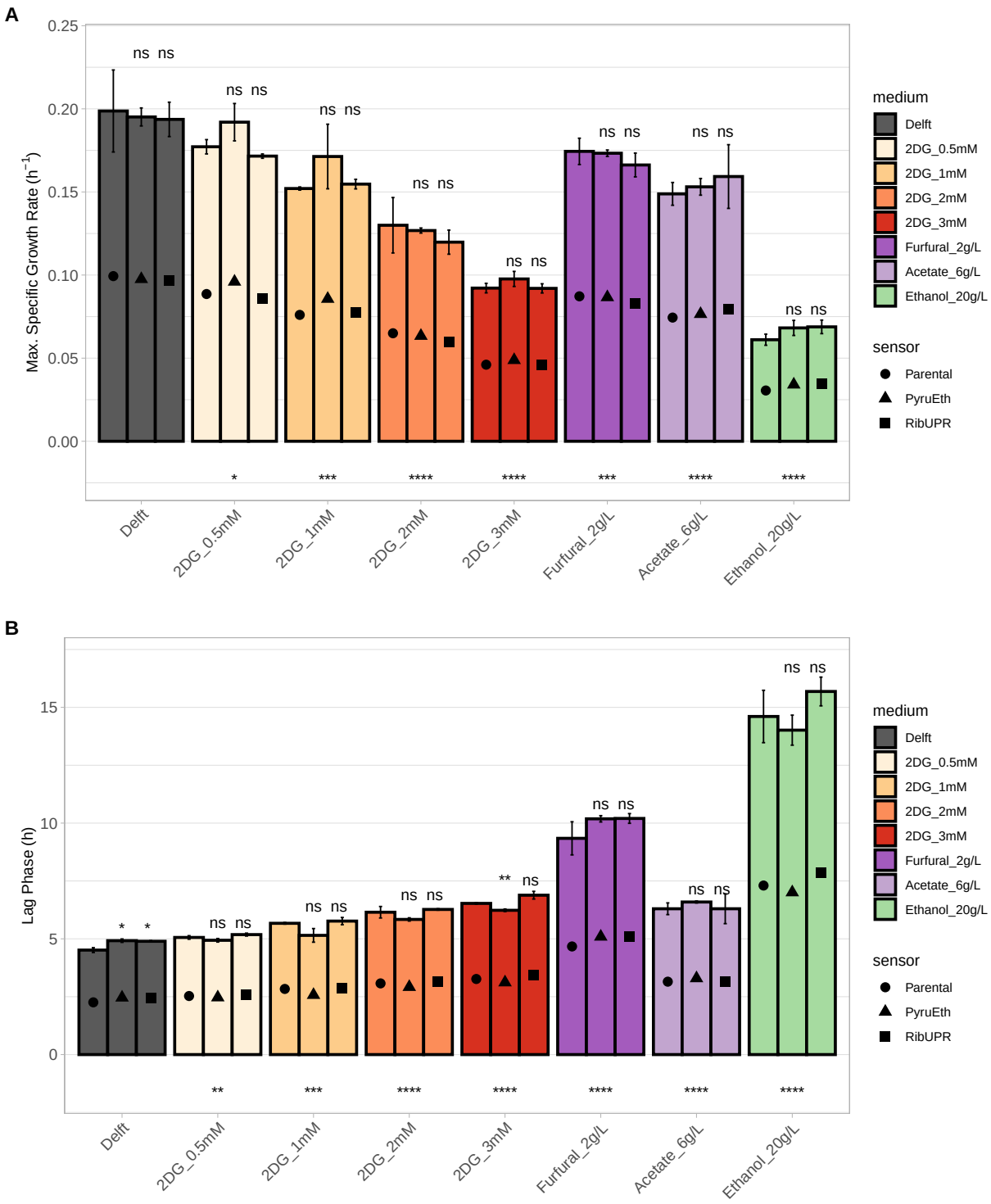
```
rm(sheets, many_spline_fits, tmp, inf_points, i, summary_data) #Clean the R environment
```



Plotting Growth Parameters (Supplementary Figure 1)

```
for(i in c("lag", "mumax")) {
  df1 <- subset(growth_param, Parameter == i)

  assign(paste0(i, "_bar"),
         ggplot(data = df1) + aes(x = medium, y = value, group = sensor) +
           stat_summary(fun = mean, geom = "bar", position = position_dodge(width = 0.9),
                        aes(fill = medium), color = "black", size = 1) +
           stat_summary(fun.data = mean_sdl, geom = "errorbar", fun.args = list(mult =
           1),
                        width = 0.1, position = position_dodge(width = 0.9)) +
           stat_summary(aes(x = medium, y = value/2, shape = sensor),
                        fun = mean, geom = "point", color = "black",
                        size = 3, position = position_dodge(width = 0.9)) +
           stat_pvalue_manual(data = subset(Stats_strain_compare, Parameter == i),
                              label = "p.adj.signif",
                              remove.bracket = T, vjust = -0.5) +
           stat_pvalue_manual(data = subset(Stats_medium_compare, Parameter == i),
```

```
                              label = "p.adj.signif",
                              remove.bracket = T, y.position = -max(df1$value)*0.1) +
          scale_y_continuous(limits = c(-max(df1$value)*0.1, max(df1$value)*1.05)) +
          scale_fill_manual(breaks = c("Delft", "2DG_0.5mM", "2DG_1mM", "2DG_2mM",
          "2DG_3mM", "Furfural_2g/L", "Acetate_6g/L","Ethanol_20g/L"),
                            values = c("#5A5A5A", "#fef0d9", "#fdcc8a", "#fc8d59",
                            "#d7301f", "#A45BBD", "#c2a5cf", "#a6dba0")) +
          theme_light() +
          theme(legend.position = "right",
                legend.box="vertical",
                axis.title.x = element_blank(),
                axis.text.x = element_text(size = 11, angle = 45, hjust = 1),
                axis.text.y = element_text(size = 11),
                panel.grid.minor.x = element_blank(),
                panel.grid.major.x = element_blank(),
                plot.margin = unit(rep(0.6, 4), "cm")))

  if(i == "mumax") {mumax_bar <- mumax_bar + labs(y = expression(paste("Max. Specific
  Growth Rate (h"^"-1", ")")))}
  if(i == "lag") {lag_bar <- lag_bar + labs(y = "Lag Phase (h)")}
}


#Saving final figure

Supp_fig1 <- ggarrange(mumax_bar, lag_bar, ncol = 1, nrow = 2, labels = c("A", "B"))

ggsave(filename = "Supplementary Figure 1.png",
       plot = Supp_fig1, device = "png", path = folder,
       width = 28, height = 22, unit = "cm", dpi = 300)

rm(i, df1, list = ls()[endsWith(ls(), "_bar")])
Supp_fig1
```


A
Max. Specific Growth Rate (h^{-1})
0.25
0.20
0.15
0.10
0.05
0.00
ns ns
*
***
****
Delft
2DG_0.5mM
2DG_1mM
2DG_2mM
2DG_3mM
Furfural_2g/L
Acetate_6g/L
Ethanol_20g/L
medium
sensor
Parental
PyruEth
RibUPR
B
Lag Phase (h)
15
10
5
0
**

DATA PLOTTING

Data re-organisation

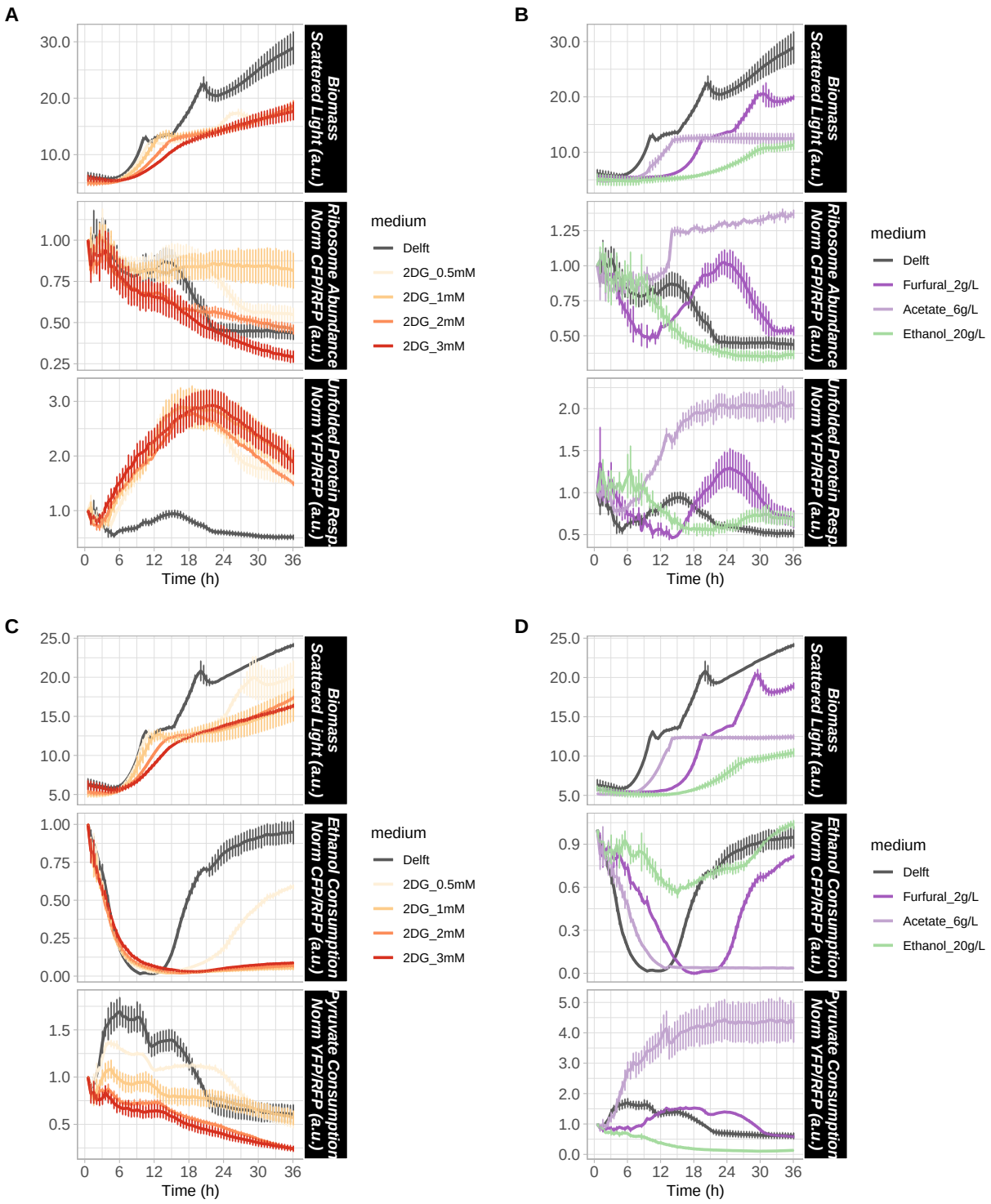
For more efficient plotting, the data frame with the data is being re-organised.

```
#Keep only the grouping columns (like medium, time, sensor and replicate) and the columns
you want to plot (here: biomass, CR.Norm and YR.Norm).
#Then, associate the names that will show up in the y axis in all the graphs.

plot_data <- final_data %>%
  filter(time < 36.5) %>%   #If the screening was longer than what you want to represent
  select(sensor, medium, time, replicate, biomass, CR.Norm, YR.Norm) %>%
  pivot_longer(!c(sensor, medium, time, replicate),
               names_to = "Parameter", values_to = "value") %>%
  mutate(IntraPar = case_when(
    Parameter == "biomass" ~ "Biomass\nScattered Light (a.u.)",
    Parameter == "CR.Norm" & sensor == "PyruEth" ~ "Ethanol Consumption\nNorm CFP/RFP
(a.u.)",
    Parameter == "YR.Norm" & sensor == "PyruEth" ~ "Pyruvate Consumption\nNorm YFP/RFP
(a.u.)",
    Parameter == "CR.Norm" & sensor == "RibUPR" ~ "Ribosome Abundance\nNorm CFP/RFP
(a.u.)",
    Parameter == "YR.Norm" & sensor == "RibUPR" ~ "Unfolded Protein Resp.\nNorm YFP/RFP
(a.u.)"))


#Re-organise the media sequence and create the desired groups (here based on the media
used).

plot_data$medium <- factor(plot_data$medium,levels = c("Delft", "2DG_0.5mM", "2DG_1mM",
"2DG_2mM", "2DG_3mM", "Furfural_2g/L", "Acetate_6g/L", "Ethanol_20g/L"))

DG_group <- c("Delft", "2DG_0.5mM", "2DG_1mM", "2DG_2mM", "2DG_3mM")
Stressor_group <- c("Delft", "Furfural_2g/L", "Acetate_6g/L", "Ethanol_20g/L")
sensors <- setdiff(unique(plot_data$sensor), "Parental")
```

Lineplots (Supplementary Figure 2)

```
#General aestethic for lineplots.

plot_aes = list(aes(x = time, y = value, colour = medium),
                stat_summary(fun = mean, geom = "line", size = 1),
                stat_summary(fun.data = mean_sdl, geom = "errorbar",
                             fun.args = list(mult = 1), width= 0.1),
                scale_x_continuous(breaks = seq(from = 0, to =
                max(unique(plot_data$time)), by = 6)),
                scale_y_continuous(labels = function(x) format(x, nsmall = 1)),
                scale_color_manual(breaks = c("Delft", "2DG_0.5mM", "2DG_1mM", "2DG_2mM",
                "2DG_3mM", "Furfural_2g/L", "Acetate_6g/L", "Ethanol_20g/L"),
                                   values = c("#5A5A5A", "#fef0d9", "#fdcc8a", "#fc8d59",
                                   "#d7301f", "#A45BBD", "#c2a5cf", "#a6dba0")),
```

```
                labs(y = "", x = "Time (h)"),
                facet_grid(rows = vars(IntraPar), scales = "free"),
                theme_light(),
                theme(legend.position = "right",
                      axis.title = element_text(size = 11),
                      axis.text = element_text(size = 11),
                      strip.background = element_rect(colour = "white", fill = "black"),
                      strip.text = element_text(size = 11, face = "bold.italic"),
                      plot.margin = unit(c(0.6, 0.6, 0.6, 0.6), "cm")))

#Lineplots

for(i in sensors) {
  assign(paste0(i, "_DG"), ggplot(data = subset(plot_data, sensor == i & medium %in%
  DG_group)) + plot_aes)
  assign(paste0(i, "_Stress"), ggplot(data = subset(plot_data, sensor == i & medium %in%
  Stressor_group)) + plot_aes)
}

#Lineplots Overview

Supp_fig2 <- ggarrange(RibUPR_DG, RibUPR_Stress, PyruEth_DG, PyruEth_Stress,
                       ncol = 2, nrow = 2, labels = c("A", "B", "C", "D"))

ggsave(filename = "Supplementary Figure 2.png",
       plot = Supp_fig2, device = "png", path = folder,
       width = 35, height = 35, unit = "cm", dpi = 300)

rm(list = ls()[endsWith(ls(), "_Stress")]); rm(list = ls()[endsWith(ls(), "_DG")])
Supp_fig2
```



Barplots

In selecting the data for the barplots, specific growth phases will be taken into account. "Early_lag" is the second timepoint of the screening. "Late_lag" is the end of the lag phase (computed previously). "Exponential" is the inflection point of the log phase (point from where the mumax was taken). "Stationary"

is the last timepoint of the screening.

```
#Select the desired data.

early_lag <- subset(plot_data,
                    sensor != "Parental" & time == unique(plot_data$time)[2]) %>%
  mutate(Phase = "early_lag")

late_lag <- distinct(plot_data[, c("sensor", "medium", "time")]) %>%
  filter(sensor != "Parental") %>%  #No need of Parental strain for fluorescence
  merge(., subset(growth_param, Parameter == "lag")[, c("sensor", "medium", "value")])
  %>%
  mutate(diff = abs(time - value)) %>%
  group_by(sensor, medium) %>%
  filter(diff == min(diff)) %>%     #Closest timepoints to lag phase lebgths
  select(-c(value, diff)) %>%
  merge(., plot_data) %>%           #Add the fluorescence information
  mutate(Phase = "late_lag")

exp <- distinct(plot_data[, c("sensor", "medium", "time")]) %>%
  filter(sensor != "Parental") %>%  #No need of Parental strain for fluorescence
  merge(., distinct(growth_param[, c("sensor", "medium", "x")])) %>%
  mutate(diff = abs(time - x)) %>%
  group_by(sensor, medium) %>%
  filter(diff == min(diff)) %>%     #Closest timepoints to inflection point
  select(-c(x, diff)) %>%
  merge(., plot_data) %>%           #Add the fluorescence information
  mutate(Phase = "exponential")

statio <- subset(plot_data,
                 sensor != "Parental" & time == max(unique(plot_data$time))) %>% #last
point in the screening
  mutate(Phase = "stationary")


#Merge and re-organise the data.

barplot_data <- rbind(early_lag, late_lag, exp, statio) %>%
  mutate(across(where(is.numeric), ~ round(.,2)))

barplot_data$Phase <- factor(barplot_data$Phase,levels = c("early_lag", "late_lag",
"exponential", "stationary"))
barplot_data$medium <- factor(barplot_data$medium,levels = c("Delft", "2DG_0.5mM",
"2DG_1mM", "2DG_2mM", "2DG_3mM", "Furfural_2g/L", "Acetate_6g/L", "Ethanol_20g/L"))

#Modify one value to allow for statistical analysis (from 0.07 to 0.071)

barplot_data$value[barplot_data$sensor == "PyruEth" & barplot_data$replicate == 3 &
barplot_data$medium == "2DG_0.5mM" & barplot_data$Phase == "exponential" &
barplot_data$Parameter == "CR.Norm"] <- 0.071

rm(early_lag, late_lag, exp, statio)
```

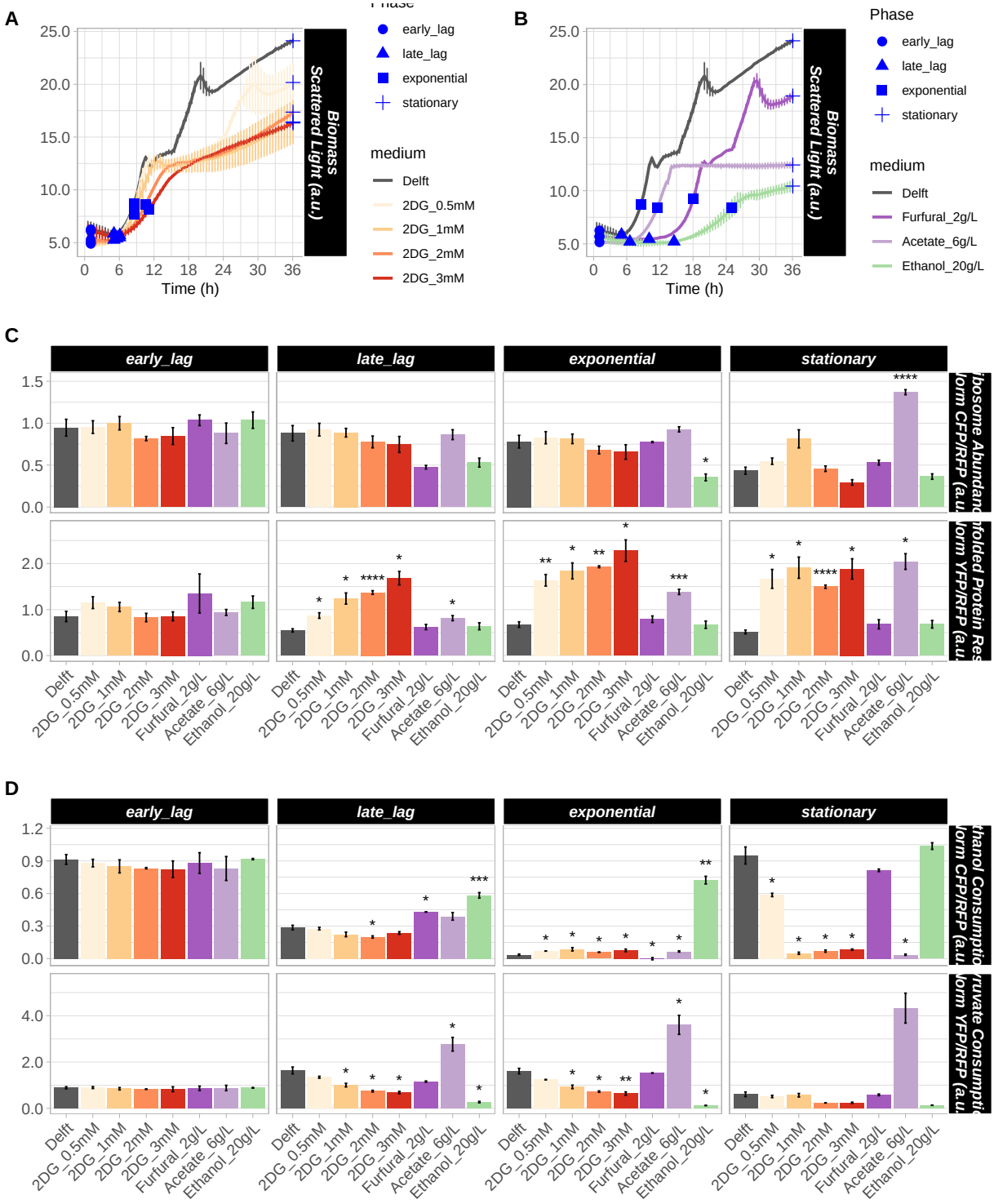
```r
#Statistics comparing the different media (for each biosensor).

Stats_barplot_compare <- barplot_data %>%
  group_by(sensor, Parameter, IntraPar, Phase) %>%
  t_test(value ~ medium, ref.group = "Delft") %>%
  add_significance("p.adj") %>%
  rename(medium = "group2")

barplot_data <- merge(barplot_data,
                      Stats_barplot_compare[, c("sensor", "Parameter", "IntraPar",
"Phase", "p.adj.signif", "medium")],
                      all.x = TRUE)  #Add significance to the data frame for plotting

barplot_data$p.adj.signif[barplot_data$p.adj.signif == "ns"] <- NA


#Barplot aesthetic

barplot_aes <- list(aes(x = medium, y = value),
                    stat_summary(fun = mean, geom = "bar", aes(fill = medium),
                                 size = 1, position = position_dodge(width = 0.9)),
                    stat_summary(fun.data = mean_sdl, geom = "errorbar",
                                 fun.args = list(mult = 1),
                                 width = 0.1, position = position_dodge(width = 0.9)),
                    stat_summary(aes(label = p.adj.signif, vjust = -0.5),
                                 fun = max, geom = "text"),
                    scale_y_continuous(expand = expansion(mult = c(0.05, 0.15)),
                                       labels = function(x) format(x, nsmall = 1)),
                    scale_fill_manual(breaks = c("Delft", "2DG_0.5mM", "2DG_1mM",
                    "2DG_2mM", "2DG_3mM", "Furfural_2g/L",
                    "Acetate_6g/L","Ethanol_20g/L"),
                                      values = c("#5A5A5A", "#fef0d9", "#fdcc8a",
                                      "#fc8d59", "#d7301f", "#A45BBD", "#c2a5cf",
                                      "#a6dba0")),
                    facet_grid(rows = vars(IntraPar), cols = vars(Phase),
                               scales = "free_y"),
                    theme_light(),
                    theme(legend.position = "none", legend.box="vertical",
                          axis.title = element_blank(),
                          axis.text.x = element_text(angle = 45, hjust = 1, size = 11),
                          axis.text.y = element_text(size = 11),
                          strip.background = element_rect(colour = "white",
                                                          fill = "black"),
                          strip.text = element_text(size = 11, face = "bold.italic"),
                          panel.grid.minor.x = element_blank(),
                          panel.grid.major.x = element_blank(),
                          plot.margin = unit(c(0.6, 0.6, 0.6, 0.6), "cm")))

#Barplots

for(i in sensors) {
  assign(paste0(i, "_bar"),
         ggplot(data = subset(barplot_data, Parameter != "biomass" & sensor == i)) +
```

```
          barplot_aes)
  }
```

Barplots with Growth Curves (Figure 2)

Figure 2 from technical note.

```
DG_fig2 <- ggplot(data = subset(plot_data, Parameter == "biomass" & sensor == "PyruEth" &
medium %in% DG_group)) +
  plot_aes +
  stat_summary(data = subset(barplot_data, Parameter == "biomass" & sensor == "PyruEth" &
  medium %in% DG_group),
               fun = mean, geom = "point", size = 3, color = "blue",
               aes (x = time, y = value, shape = Phase, group = medium))

Stress_fig2 <- ggplot(data = subset(plot_data,
                                    Parameter == "biomass" & sensor == "PyruEth" & medium
%in% Stressor_group)) +
  plot_aes +
  stat_summary(data = subset(barplot_data, Parameter == "biomass" & sensor == "PyruEth" &
  medium %in% Stressor_group),
               fun = mean, geom = "point", size = 3, color = "blue",
               aes (x = time, y = value, shape = Phase, group = medium))

fig2 <- ggarrange(ggarrange(DG_fig2, Stress_fig2,
                            ncol = 2, nrow = 1, labels = c("A", "B")),
                  RibUPR_bar, PyruEth_bar,
                  ncol = 1, nrow = 3, labels = c("", "C", "D"), heights = c(0.7, 1, 1))

ggsave(filename = "Figure 2.png", plot = fig2, device = "png",
       path = folder, width = 30, height = 40, unit = "cm", dpi = 300)

rm(list = ls()[endsWith(ls(), "_bar")]); rm(list = ls()[endsWith(ls(), "_fig2")])
fig2
```



CITATIONS

Citations of R Studio and packages used.

```
print(citation(), style = "text")
```

```
## R Core Team (2021). _R: A Language and Environment for Statistical
## Computing_. R Foundation for Statistical Computing, Vienna, Austria.
## <URL: https://www.R-project.org/>.
```

```
for(i in c("rmarkdown", requiredPackages)) {
  print(i); print(citation(i), style = "text"); cat('\n')
}
```

```
## [1] "rmarkdown"
## Allaire J, Xie Y, McPherson J, Luraschi J, Ushey K, Atkins A, Wickham
## H, Cheng J, Chang W, Iannone R (2022). _rmarkdown: Dynamic Documents
## for R_. R package version 2.17, <URL:
## https://github.com/rstudio/rmarkdown>.
##
## Xie Y, Allaire J, Grolemund G (2018). _R Markdown: The Definitive
## Guide_. Chapman and Hall/CRC, Boca Raton, Florida. ISBN 9781138359338,
## <URL: https://bookdown.org/yihui/rmarkdown>.
##
## Xie Y, Dervieux C, Riederer E (2020). _R Markdown Cookbook_. Chapman
## and Hall/CRC, Boca Raton, Florida. ISBN 9780367563837, <URL:
## https://bookdown.org/yihui/rmarkdown-cookbook>.
##
## [1] "tidyverse"
## Wickham H, Averick M, Bryan J, Chang W, McGowan LD, François R,
## Grolemund G, Hayes A, Henry L, Hester J, Kuhn M, Pedersen TL, Miller E,
## Bache SM, Müller K, Ooms J, Robinson D, Seidel DP, Spinu V, Takahashi
## K, Vaughan D, Wilke C, Woo K, Yutani H (2019). "Welcome to the
## tidyverse." _Journal of Open Source Software_, *4*(43), 1686. doi:
## 10.21105/joss.01686 (URL: https://doi.org/10.21105/joss.01686).
##
## [1] "ggplot2"
## Wickham H (2016). _ggplot2: Elegant Graphics for Data Analysis_.
## Springer-Verlag New York. ISBN 978-3-319-24277-4, <URL:
## https://ggplot2.tidyverse.org>.
##
## [1] "ggpubr"
## Kassambara A (2020). _ggpubr: 'ggplot2' Based Publication Ready Plots_.
## R package version 0.4.0, <URL:
## https://CRAN.R-project.org/package=ggpubr>.
##
## [1] "readxl"
## Wickham H, Bryan J (2022). _readxl: Read Excel Files_. R package
## version 1.4.1, <URL: https://CRAN.R-project.org/package=readxl>.
##
## [1] "writexl"
## Ooms J (2022). _writexl: Export Data Frames to Excel 'xlsx' Format_. R
## package version 1.4.1, <URL:
## https://CRAN.R-project.org/package=writexl>.
##
## [1] "rstatix"
```

```
## Kassambara A (2021). _rstatix: Pipe-Friendly Framework for Basic
## Statistical Tests_. R package version 0.7.0, <URL:
## https://CRAN.R-project.org/package=rstatix>.
##
## [1] "deSolve"
## Soetaert K, Petzoldt T, Setzer RW (2010). "Solving Differential
## Equations in R: Package deSolve." _Journal of Statistical Software_,
## *33*(9), 1-25. doi: 10.18637/jss.v033.i09 (URL:
## https://doi.org/10.18637/jss.v033.i09).
##
## [1] "growthrates"
## Petzoldt T (2022). _growthrates: Estimate Growth Rates from
## Experimental Data_. R package version 0.8.4, <URL:
## https://CRAN.R-project.org/package=growthrates>.
```