Anthocyanen uit druiven als conserveermiddel?

Jurre Hageman

2022-11-30

# Inleiding

Anthocyanen zijn in water oplosbare flavonoïden. Anthocyanen hebben een diverse functie in planten. Ze lokken insecten zoals bijen voor bestuiving maar ze hebben ook antioxidant en antimicrobiële eigenschappen. Hoewel anthocyanen in de Europese Unie zijn goedgekeurd als kleurstof voor voedingsmiddelen en dranken, zijn ze nog niet goedgekeurd voor gebruik als voedseladditief omdat ze niet als veilig zijn geverifieerd bij gebruik als voedsel- of supplementingrediënten. Het bedrijf waar ik mee samenwerk (projectpartner) wil anthocyanen gebruiken als voedseladditief in voedingsmiddelen om bacteriebederf tegen te gaan. Er heeft een laboratoriumonderzoek plaatsgevonden waarbij de groei van een bacterie is onderzocht in media met verschillende concentraties anthocyanen uit rode en groene druiven.

# Onderzoeksvraag

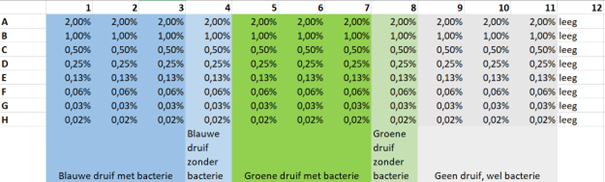
Werken anthocyanen uit rode druiven of groene druiven antimicrobieel?

# Het experiment

Anthocyanen zijn uit rode en groene druiven geextraheerd. Daarna is een experiment ingezet om te testen of de anthocyanen ingezet kunnen worden als conserveringsmiddel.

# Plate setup

De plate setup is hieronder te zien:

 # Analyse

## User Settings

# Set organism  
organism <- "Bacterie"  
  
# Set file and folder  
filename <- "2020-01-30.csv"  
excel\_file <- "2020-01-30.xlsx"  
folder <- "../sample\_data"  
  
# Set samples  
sample\_1 <- "Rode druif"  
sample\_2 <- "Groene druif"  
sample\_3 <- "Negatieve controle"  
  
# Set columns (sample#\_replicate).  
# This is for user input convenience.   
# Sample and replicate will be separated later using dplyr::separate  
col\_1 <- "1\_1"   
col\_2 <- "1\_2"   
col\_3 <- "1\_3"   
col\_4 <- "1\_c"   
col\_5 <- "2\_1"   
col\_6 <- "2\_2"   
col\_7 <- "2\_3"   
col\_8 <- "2\_c"   
col\_9 <- "3\_1"   
col\_10 <- "3\_2"   
col\_11 <- "3\_3"   
col\_12 <- "3\_c"   
  
# Set dilutions  
row\_1 <- 2  
row\_2 <- 1  
row\_3 <- 0.5  
row\_4 <- 0.25  
row\_5 <- 0.13  
row\_6 <- 0.06  
row\_7 <- 0.03  
row\_8 <- 0  
  
# Set line colors  
linecolors <- c("#FF3633", "#FF8933", "#FCFF33", "#62FF33", "#33FFEB", "#33A7FF", "#5F33FF", "#EA33FF")

## Imports

library(Rmisc)

## Loading required package: lattice

## Loading required package: plyr

library(tidyverse)

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ dplyr 1.1.4 ✔ readr 2.1.4  
## ✔ forcats 1.0.0 ✔ stringr 1.5.1  
## ✔ ggplot2 3.4.4 ✔ tibble 3.2.1  
## ✔ lubridate 1.9.3 ✔ tidyr 1.3.0  
## ✔ purrr 1.0.2

## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::arrange() masks plyr::arrange()  
## ✖ purrr::compact() masks plyr::compact()  
## ✖ dplyr::count() masks plyr::count()  
## ✖ dplyr::desc() masks plyr::desc()  
## ✖ dplyr::failwith() masks plyr::failwith()  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::id() masks plyr::id()  
## ✖ dplyr::lag() masks stats::lag()  
## ✖ dplyr::mutate() masks plyr::mutate()  
## ✖ dplyr::rename() masks plyr::rename()  
## ✖ dplyr::summarise() masks plyr::summarise()  
## ✖ dplyr::summarize() masks plyr::summarize()  
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

## Converting user input

samples <- c(sample\_1, sample\_2, sample\_3)  
columns <- c("row", col\_1, col\_2, col\_3, col\_4, col\_5, col\_6, col\_7, col\_8, col\_9, col\_10, col\_11, col\_12)  
dilution\_series <- factor(c(row\_1, row\_2, row\_3, row\_4, row\_5, row\_6, row\_7, row\_8))  
filepath <- file.path(folder, filename)  
filepath\_excel <- file.path(folder, excel\_file)  
group1 <- c('1\_1', '1\_2', '1\_3', '1\_c')  
group2 <- c('2\_1', '2\_2', '2\_3', '2\_c')  
group3 <- c('3\_1', '3\_2', '3\_3', '3\_c')

## Loading the data

## Read from Excel in xlsx format

library("readxl")  
df <- read\_excel(filepath\_excel, sheet = 2) %>% slice(-1:-1) #drop the first row

## New names:  
## • `` -> `...1`  
## • `` -> `...2`  
## • `` -> `...3`  
## • `` -> `...4`  
## • `` -> `...5`  
## • `` -> `...6`  
## • `` -> `...7`  
## • `` -> `...8`  
## • `` -> `...9`  
## • `` -> `...10`  
## • `` -> `...11`  
## • `` -> `...12`  
## • `` -> `...13`  
## • `` -> `...14`  
## • `` -> `...15`  
## • `` -> `...16`  
## • `` -> `...17`  
## • `` -> `...18`  
## • `` -> `...19`  
## • `` -> `...20`  
## • `` -> `...21`  
## • `` -> `...22`  
## • `` -> `...23`  
## • `` -> `...24`  
## • `` -> `...25`  
## • `` -> `...26`  
## • `` -> `...27`  
## • `` -> `...28`  
## • `` -> `...29`  
## • `` -> `...30`  
## • `` -> `...31`  
## • `` -> `...32`  
## • `` -> `...33`  
## • `` -> `...34`  
## • `` -> `...35`  
## • `` -> `...36`  
## • `` -> `...37`  
## • `` -> `...38`  
## • `` -> `...39`  
## • `` -> `...40`  
## • `` -> `...41`  
## • `` -> `...42`

skip <- 14 #each time, 14 rows of plate info that will be skipped  
collect <- 8 #there are 8 rows of OD data that I will keep  
reps <- nrow(df) %/% (skip + collect) #number of repetitions  
index <- rep(0:(reps - 1), each = collect) \* (skip + collect) + (1 : collect) + skip  
df <- df[index, ] #collect the blocks of OD data  
df <- df[, 1:13]  
colnames(df) <- columns  
df[2:13] <- lapply(df[2:13], function(x) as.numeric(as.character(x)))  
df

## # A tibble: 872 × 13  
## row `1\_1` `1\_2` `1\_3` `1\_c` `2\_1` `2\_2` `2\_3` `2\_c` `3\_1` `3\_2`  
## <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 A 0.465 0.400 0.399 0.366 0.131 0.146 0.137 0.118 0.101 0.144   
## 2 B 0.321 0.350 0.349 0.328 0.130 0.141 0.138 0.162 0.118 0.109   
## 3 C 0.260 0.243 0.236 0.216 0.131 0.135 0.125 0.129 0.0950 0.125   
## 4 D 0.172 0.189 0.176 0.164 0.117 0.0998 0.141 0.127 0.101 0.120   
## 5 E 0.166 0.169 0.160 0.149 0.128 0.105 0.127 0.113 0.104 0.112   
## 6 F 0.138 0.148 0.154 0.141 0.0976 0.113 0.110 0.116 0.109 0.124   
## 7 G 0.147 0.131 0.112 0.130 0.104 0.105 0.107 0.0980 0.112 0.113   
## 8 H 0.117 0.123 0.104 0.0980 0.0993 0.0974 0.126 0.104 0.106 0.111   
## 9 A 0.348 0.263 0.277 0.259 0.101 0.109 0.102 0.0952 0.0825 0.0861  
## 10 B 0.267 0.291 0.293 0.270 0.107 0.104 0.109 0.119 0.0850 0.0800  
## # ℹ 862 more rows  
## # ℹ 2 more variables: `3\_3` <dbl>, `3\_c` <dbl>

## Read from a csv file.

# df <- read\_csv2(filepath,  
# col\_types = cols(row = 'c', .default = col\_double()),  
# col\_names = columns) %>% #Using col\_names also removes the empty columns  
# slice(-1:-2) #drop the first two rows  
#   
# skip <- 14 #each time, 14 rows of plate info that will be skipped  
# collect <- 8 #there are 8 rows of OD data that I will keep  
# reps <- nrow(df) %/% (skip + collect) #number of repetitions  
# index <- rep(0:(reps - 1), each = collect) \* (skip + collect) + (1 : collect) + skip  
# df <- df[index, ] #collect the blocks of OD data  
# df

## Add timepoint and dillution columns

num\_of\_timepoints <- nrow(df) / 8 - 1  
timepoints\_actual <- seq(0, (10 \* num\_of\_timepoints), by = 10)  
timepoints <- rep(timepoints\_actual, each = 8)  
dilutions <- rep(dilution\_series, times = num\_of\_timepoints + 1)  
  
df <- mutate(df, timepoint = timepoints, dilution = dilutions) %>%  
 select(1, timepoint, dilution, everything())  
#View(df)

## Make tidy

#Use dplyr to make tidy df tables  
#Need to gather at multiple levels, hence the function.  
generateTidy <- function(df\_t, selector){  
 df\_t <- df\_t %>%   
 select(row, timepoint, dilution, all\_of(selector)) %>%  
 gather(selector[1], selector[2], selector[3], key = 'sample', value = 'OD\_S') %>%  
 separate(sample, into = c('sample', 'replicate')) %>%  
 select(1:3, sample, replicate, 'OD\_S', selector[4]) %>%  
 dplyr::rename('OD\_C' = selector[4]) %>%  
 mutate(OD\_N = OD\_S - OD\_C)  
 return(df\_t)  
}  
  
df\_tidy\_s1 <- generateTidy(df, group1)  
df\_tidy\_s2 <- generateTidy(df, group2)  
df\_tidy\_s3 <- generateTidy(df, group3)  
#View(df\_tidy\_s1)

## Summary Data

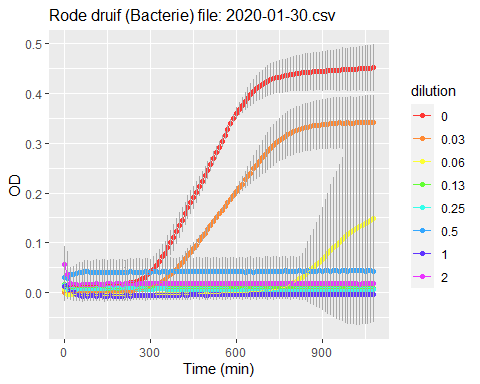
#Use the summarySE package to calculate the mean, SD, SE and CI data at once.  
vars <- c("dilution", "timepoint")  
stat.sample1 <- summarySE(df\_tidy\_s1, measurevar="OD\_N", groupvars=vars)  
stat.sample2 <- summarySE(df\_tidy\_s2, measurevar="OD\_N", groupvars=vars)  
stat.sample3 <- summarySE(df\_tidy\_s3, measurevar="OD\_N", groupvars=vars)  
#View(stat.sample3)

# Visualisatie

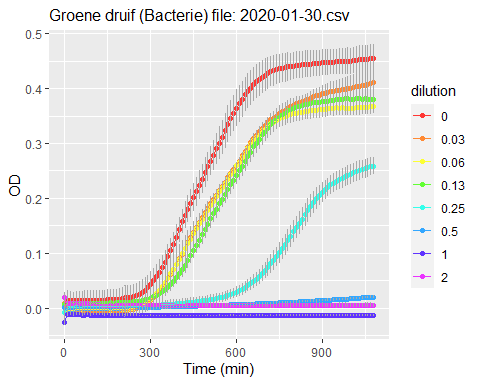
## Set-up plots

## Generate Plots

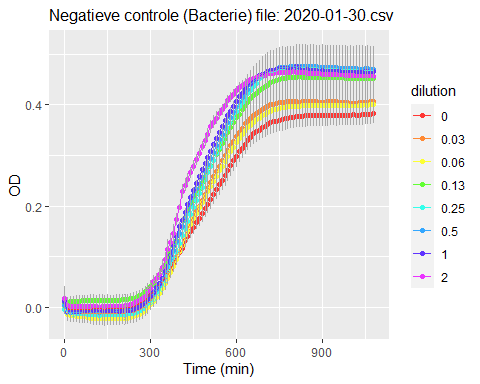
## Plot 1



## Plot 2



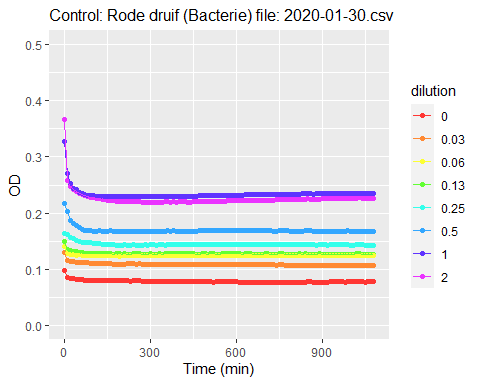
## Plot 3



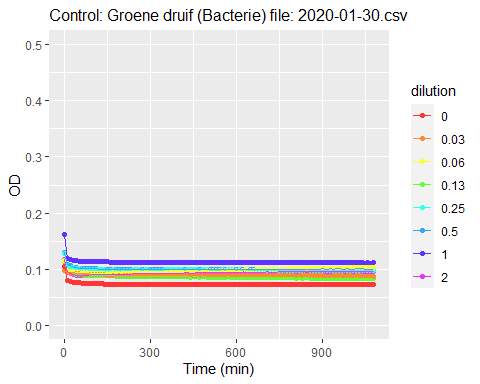
## Controls (can be used to find a contamination)

generatePlotControl <- function(sample, title.string){  
 ggplot(data=sample, mapping = aes(x = timepoint, y = OD\_C, group = dilution)) +   
 geom\_point(aes(color = dilution)) +   
 geom\_line(aes(color = dilution)) +  
 scale\_color\_manual(values = linecolors) +  
 labs(x = "Time (min)", y = "OD", title = paste("Control:", title.string, sep = " ")) +  
 ylim(0, 0.5) +  
 theme(plot.title = element\_text(size=12))  
}

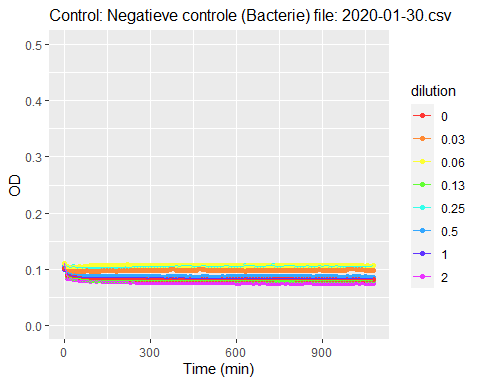
## Plot 4



## Plot 5



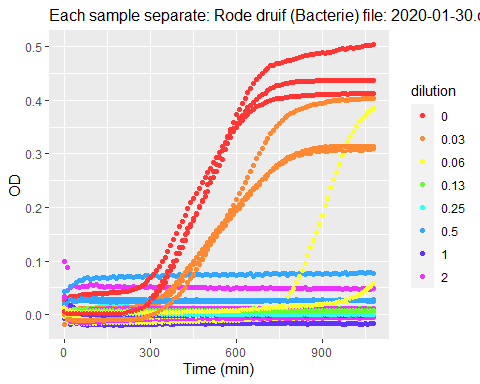
## Plot 6



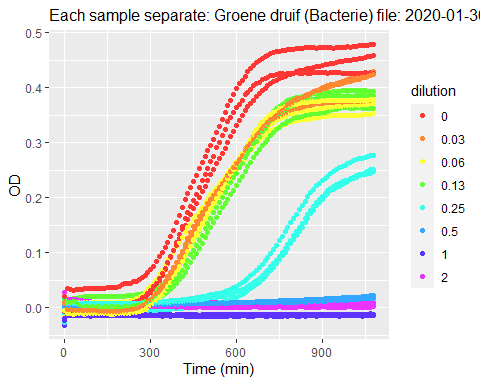
## Each data series separate (can be used to find a contamination)

generatePlotEachLine <- function(sample, title.string){  
 ggplot(data=sample, mapping = aes(x = timepoint, y = OD\_N, group = dilution)) +   
 geom\_point(aes(color = dilution)) +   
 scale\_color\_manual(values = linecolors) +  
 labs(x = "Time (min)", y = "OD", title = paste("Each sample separate:", title.string, sep = " ")) +  
 theme(plot.title = element\_text(size=12))  
}

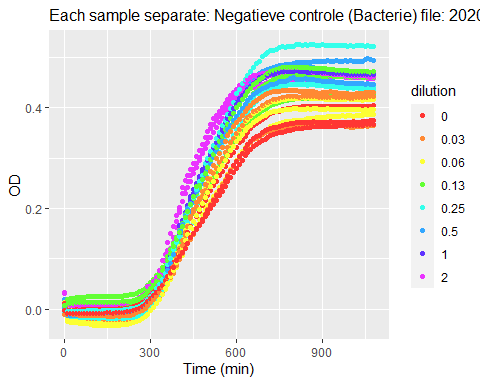
## Plot 7



## Plot 8



## Plot 9



# Conclusie

De steriliteitscontroles zijn allemaal negatief voor bacteriegroei.

Uit de data blijkt verder dat de anthocyanen van de rode druif het sterkst antimicrobieel werken. De groene druif werkt ook antimicrobieel maar de negatieve controle niet. Er is een duidelijke dosis-afhankelijke remming van de groei waar te nemen bij de anthocyanen van de rode en de groene druif.

The end…