**Growthcurver Workshop**

**Mariana Rius - 4/24/2020**

# Setup

1. Install on your own computer:

[R](https://www.r-project.org/)

[RStudio](https://rstudio.com/products/rstudio/download/)

1. In console run:

install.packages("readxl")

install.packages("rlist")

install.packages("growthcurver")

install.packages("dplyr")

install.packages(“ggplot2")

1. Download datasets or locate your own data you would like to use.

IMPORTANT: For the script to run properly an independent folder containing the data file is *required*

# Datasets provided:

## LAA-E2.xlsx

* OD data - Single reads per well
* LAA = Laby Antibiotic Assay
  + [plate setup](https://docs.google.com/presentation/d/1jRHy32pm0gParXaoHVKo8pyVgZAvbHmZJvZlcpzZYKc/edit?usp=sharing)
  + 96 wells
  + 16 time points
  + No media blank
* Excel file
  + sheets arranged reverse chronologically (most recent **first**)
  + OD reads start at row 27 (header = row 26)
  + Date and time on row 23
  + Last sheet is not blank

## NGE-E1-P3.xlsx

* OD Data - Multiple reads per well
* NGE = Nutrient Growth Experiment
  + [plate setup](https://docs.google.com/presentation/d/1Fe_pQ2ebTqtWHqC9wkUPmBWrXytWQm8loQvxyXw0EWs/edit?usp=sharing)
  + 96 wells
  + 6 time points
  + No media blank
* Excel file
  + Sheets arranged reverse chronologically (most recent **first**)
  + OD reads start at row 48 (header = row 47)
  + Date and time on row 25
  + Last sheet is blank

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

# Growthcurver Intro

# Growth phases of a culture

# 

Logistic equation (used by growthcurver)

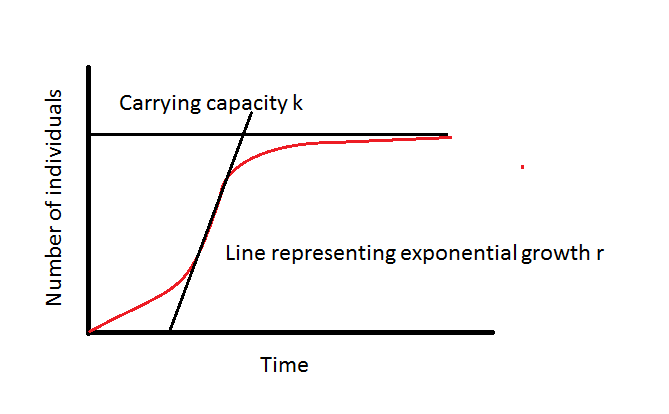
# 

*Nt* = population at time *t*

*K* = carrying capacity

*N0* = initial population size

*r* = intrinsic growth rate



# R/ RStudio Orientation

* Script, Console, Environment, Help
* R basics
  + making objects
  + object types
    - numeric, character, factor, vector, data.frame, list
  + for loop
  + functions
    - c() and trick to building a recursive vector
    - head() and tail()

# Pipeline

We will be following this vignette: [Growthcurver](https://cran.r-project.org/web/packages/growthcurver/vignettes/Growthcurver-vignette.html)

## Pipeline breakdown:

1. Setup script
2. Load data
3. Organize data for growthcurver
4. Run growthcurver
5. Plot growthcurver output
6. PCA & plot
7. Sigma histogram
8. Plot parameters

## 

## LAA-E2.xlsx

### Setup script

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

getwd()

setwd("C:/Users/maria/Downloads/LAA-E2")

options(stringsAsFactors = F)

#total wells

tw<-96

#filename

fn<-"LAA-E2.xlsx"

#replicates?

x<-3

#number of antibiotic concentrations

nc<-12

#number of strains #JUST AURLI

ns<-1

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

### Load data

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

library("readxl")

allSheets<-list()

for (i in 1:c((length(excel\_sheets(fn))))){ #This will read in each sheet of excel file 'fn', if there is a blank sheet at the end of excel file this line will be "for (i in 1:c((length(excel\_sheets(fn)))-1)){"

mySheet <- read\_excel(fn, sheet = i) #This will import a sheet as a dataframe

write.csv(mySheet, file = paste("t", i,".csv", sep = ""), row.names = FALSE) #This will save each sheet as a csv with the file name 't.x', x = sheet number

mySheet<-read.csv(paste("t", paste(i),".csv", sep= ""), header = T, skip = 25, stringsAsFactors=FALSE) #This will read in the data starting at the line number after 'skip='

mySheet<-mySheet[,2:13] #This will filter out your data [row,column], removing row names

allSheets[[i]]<-mySheet[1:3,] #preserved only first three rows (aurli) and stored data as a list element in allSheets

}

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

what does allSheets look like?

### Organize data for growthcurver

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

#reverse list order for excel files in reverse chronological order

library("rlist")

allSheets1<-list.reverse(allSheets)

#calculate difference in time

sto1.t<-list()

for (i in 1:c(length(excel\_sheets(fn)))) {

sto.t<-read.csv(paste("t", paste(i),".csv", sep= ""), header = F, skip = 22, nrow = 1, stringsAsFactors=FALSE)

#date & time

sto.t<-sto.t[1,2]

#strsplit date & replace "/" with "-"

date<-gsub("/", "-", paste((strsplit(sto.t, " ")[[1]][(1)]), collapse = " "))

#date format from m-d-y to y-m-d

y<-(strsplit(date, "-")[[1]][(3)])

m<-(strsplit(date, "-")[[1]][(1)])

d<-(strsplit(date, "-")[[1]][(2)])

tdate<-paste(y, m,d, sep = "-")

#put into list

sto1.t[[i]]<-tdate

#strsplit time

t<-paste((strsplit(sto.t, " ")[[1]][(2:3)]), collapse = " ")

#change 12H to 24H

ttime<-format(strptime(t, "%I:%M:%S %p"), format="%H:%M:%S")

sto1.t[[i]][2]<-ttime

}

sto1.t<-list.reverse(sto1.t)

tdif<-c()

for (i in 2:length(sto1.t)) {

dif<-difftime(paste(sto1.t[[i]][1],sto1.t[[i]][2], sep = " "), paste(sto1.t[[1]][1],sto1.t[[1]][2], sep = " "), units = "hours")

tdif<-c(tdif, dif)

}

tdif<-c(0,tdif)

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

what does tdif look like?

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

#create the data.frame structure and column names

u<-data.frame(time=c(tdif))

ro<-c("A","B","C")

co<-c(1:12)

df<-c() #makes a vector of all the well names used in 'ro' and 'co'

for (i in (unique(ro))) {

for (j in (unique(co))) {

df<-c(df,paste(i,j,sep=""))

}

}

for (i in 1:length(df)) { #creates each well name to be a vector in existence

assign(df[i], c(recursive = TRUE))

}

for (i in 1:length(df)) { #assigns vector of well name and column of u with data of each time point at well placement

for (j in 1:length(allSheets1)) {

assign(df[i], c(get(df[i]),as.numeric(c(allSheets1[[j]][1,],allSheets1[[j]][2,],allSheets1[[j]][3,]))[i])) # makes a vector of data to sleect based on df[i] // optimization required here for more than three rows

}

u[,(i+1)]<-get(df[i])

}

colnames(u)<-c("time",df) #give each column the corresponding well name

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

what does u look like?

### Run growthcurver

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

library("growthcurver")

gc\_out <- SummarizeGrowthByPlate(u)

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

what does gc\_out look like?

Additional parameters provided in gc\_out

t\_mid *= t, ½ K,* the time at which the population density reaches half the carrying capacity

t\_gen = doubling time, the least amount of time required to double the population

auc\_l = the area under the modeled logistic curve (integral of the logistic equation)

auc\_e = the area under the curve obtained from the optical density readings data

sigma = residual standard deviation, the estimated standard deviation of the errors // residual sum of squares from the fit of the logistic curve to the data, so larger values mean poorer fits, a parameter used to evaluate the 'goodness of fit'

gc\_out$note? unique(gc\_out$note)

save gc\_out?

write.csv(gc\_out,file="aurli\_growthcurver\_output.csv")

save(gc\_out,file="aurli\_growthcurver\_output.rda")

### Plot growthcurver output

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

d<-u #renames our data to match growthcurver's provided script

num\_analyses <- length(names(d)) - 1

d\_gc <- data.frame(sample = character(num\_analyses),

k = numeric(num\_analyses),

n0 = numeric(num\_analyses),

r = numeric(num\_analyses),

t\_mid = numeric(num\_analyses),

t\_gen = numeric(num\_analyses),

auc\_l = numeric(num\_analyses),

auc\_e = numeric(num\_analyses),

sigma = numeric(num\_analyses),

stringsAsFactors = FALSE)

trim\_at\_time<-tdif[8] #set our preferred plotting range based on tdif

#pdf("LAA-E2\_growthcurver\_r.pdf", height = 8.5, width = 11) #to print a pdf remove the hashtag at this line AND at line below = 'dev.off()'

par(mfrow = c(8,12))

par(mar = c(0.25,0.25,0.25,0.25))

y\_lim\_max <- max(d[,setdiff(names(d), "time")]) - min(d[,setdiff(names(d), "time")])

n <- 1 # keeps track of the current row in the output data frame

for (col\_name in names(d)) {

# Don't process the column called "time".

# It contains time and not absorbance data.

if (col\_name != "time") {

# Create a temporary data frame that contains just the time and current col

d\_loop <- d[, c("time", col\_name)]

# Do the background correction.

# Background correction option 1: subtract the minimum value in a column

# from all measurements in that column

min\_value <- min(d\_loop[, col\_name])

d\_loop[, col\_name] <- d\_loop[, col\_name] - min\_value

# Background correction option 2: subtract the mean value of blank wells

# over the course the experiment

# (Replace B2, D8, G11 with the column

# names of your media-only wells)

#d$blank <- apply(d[, c("B2", "D8", "G11")], 1, mean)

#d$A1 <- d$A1 - d$blank

# Now, call Growthcurver to calculate the metrics using SummarizeGrowth

gc\_fit <- SummarizeGrowth(data\_t = d\_loop[, "time"],

data\_n = d\_loop[, col\_name],

t\_trim = trim\_at\_time,

bg\_correct = "none")

# Now, add the metrics from this column to the next row (n) in the

# output data frame, and increment the row counter (n)

d\_gc$sample[n] <- col\_name

d\_gc[n, 2:9] <- c(gc\_fit$vals$k,

gc\_fit$vals$n0,

gc\_fit$vals$r,

gc\_fit$vals$t\_mid,

gc\_fit$vals$t\_gen,

gc\_fit$vals$auc\_l,

gc\_fit$vals$auc\_e,

gc\_fit$vals$sigma)

n <- n + 1

# Finally, plot the raw data and the fitted curve

# Here, I'll just print some of the data points to keep the file size smaller

n\_obs <- length(gc\_fit$data$t)

idx\_to\_plot <- 1:20 / 20 \* n\_obs

plot(gc\_fit$data$t[idx\_to\_plot], gc\_fit$data$N[idx\_to\_plot],

pch = 20,

xlim = c(0, trim\_at\_time),

ylim = c(0, y\_lim\_max),

cex = 0.6, xaxt = "n", yaxt = "n")

text(x = trim\_at\_time / 4, y = y\_lim\_max, labels = col\_name, pos = 1)

if (gc\_fit$model=="") {

gc\_fit$model<-rep(0,length(tdif[1:grep(trim\_at\_time, tdif)])-1)

lines(gc\_fit$data$t, rep(0,length(tdif[1:grep(trim\_at\_time, tdif)])-1), col = "indianred3")

}

else #I added 'if...else' for data where no model was fit plots line at y=0

lines(gc\_fit$data$t, predict(gc\_fit$model), col = "indianred3")

}

}

#dev.off() #to print a pdf remove the hashtag at this line AND at 'pdf(...)' line above

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

### PCA & plot

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

# Load dplyr, ggplot2, and the sample data

library(dplyr)

library(ggplot2)

pca\_gc\_out <- as\_data\_frame(gc\_out)

# Prepare the gc\_out data for the PCA

rownames(pca\_gc\_out) <- pca\_gc\_out$sample

# Do the PCA

pca.res <- prcomp(pca\_gc\_out %>% select(k:sigma), center=TRUE, scale=TRUE)

# Plot the results

as\_data\_frame(list(PC1=pca.res$x[,1],

PC2=pca.res$x[,2],

samples = rownames(pca.res$x))) %>%

ggplot(aes(x=PC1,y=PC2, label=samples)) +

geom\_text(size = 3)

# Do the PCA with percentages in axes

pca.res <- prcomp(pca\_gc\_out %>% select(k:sigma), center=TRUE, scale=TRUE)

percentage <- round(pca.res$sdev / sum(pca.res$sdev) \* 100, 2)

df\_out <- as.data.frame(pca.res$x)

percentage <- paste( colnames(df\_out), "(", paste( as.character(percentage), "%", ")", sep="") )

# Plot the results

as\_data\_frame(list(PC1=pca.res$x[,1],

PC2=pca.res$x[,2],

samples = rownames(pca.res$x))) %>%

ggplot(aes(x=PC1,y=PC2, label=samples)) + geom\_text(size = 3) + xlab(percentage[1]) + ylab(percentage[2])

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

### Sigma histogram

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

par(mfrow=c(1,1),mar = c(3.8,3.8,2,2))

# Plot a histogram of the sigma values in order to check for outliers

hist(gc\_out$sigma, breaks= 30,main = "Histogram of sigma values", xlab = "sigma", col = "palegreen3", xlim = c(0, max(gc\_out$sigma)))

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

### Plot parameters

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

par(mfrow = c(8,12))

par(mar = c(0.25,0.25,0.25,0.25))

y<-c(min(gc\_out$k), max(gc\_out$k))

for (i in 1:12) {

boxplot(gc\_out$k[c(i,i+12, i+24)], col="maroon", ylim=y, yaxt="n")

}

y<-c(min(gc\_out$r), max(gc\_out$r))

for (i in 1:12) {

boxplot(gc\_out$r[c(i,i+12, i+24)], col="light coral", ylim=y,yaxt="n")

}

y<-c(min(gc\_out$t\_mid), max(gc\_out$t\_mid))

for (i in 1:12) {

boxplot(gc\_out$t\_mid[c(i,i+12, i+24)], col="goldenrod3", ylim=y,yaxt="n")

}

y<-c(min(gc\_out$t\_gen), max(gc\_out$t\_gen))

for (i in 1:12) {

boxplot(gc\_out$t\_gen[c(i,i+12, i+24)], col="midnightblue", ylim=y,yaxt="n")

}

y<-c(min(gc\_out$auc\_l), max(gc\_out$auc\_l))

for (i in 1:12) {

boxplot(gc\_out$auc\_l[c(i,i+12, i+24)], col="mediumpurple3", ylim=y,yaxt="n")

}

y<-c(min(gc\_out$auc\_e), max(gc\_out$auc\_e))

for (i in 1:12) {

boxplot(gc\_out$auc\_e[c(i,i+12, i+24)], col="tomato3", ylim=y,yaxt="n")

}

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

## 

## NGE-E1-P3.xlsx

### Setup script

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

options(stringsAsFactors = F)

setwd("C:/Users/maria/Downloads/NGE-E1")

getwd()

#total wells

tw<-96

#filename

fn<-"NGE-E1-P3.xlsx"

#replicates?

x<-1

#number of conditions

nc<-96

#number of strains

ns<-1

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

### Load data

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

#excel file sheets to csv

library("readxl")

allSheets<-list()

for (i in 1:c((length(excel\_sheets(fn)))-1)){

mySheet <- read\_excel(fn, sheet = i) #This will import a sheet as a dataframe starting at datapoints.

write.csv(mySheet, file = paste("t", i,".csv", sep = ""), row.names = FALSE) #This will save each sheet with the file name Sheetx, x being whichever sheet number it is

mySheet<-read.csv(paste("t", paste(i),".csv", sep= ""), header = T, skip = 46, stringsAsFactors=FALSE)

mySheet<-mySheet[,1:3]

allSheets[[i]]<-mySheet[1:tw,]

}

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

### Organize data for growthcurver

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

#calculate difference in time

sto1.t<-list()

for (i in 1:c(length(excel\_sheets(fn))-1)) {

sto.t<-read.csv(paste("t", paste(i),".csv", sep= ""), header = F, skip = 24, nrow = 1, stringsAsFactors=FALSE)

#date & time

sto.t<-sto.t[1,2]

#strsplit date & replace "/" with "-"

date<-gsub("/", "-", paste((strsplit(sto.t, " ")[[1]][(1)]), collapse = " "))

#date format from m-d-y to y-m-d

y<-(strsplit(date, "-")[[1]][(3)])

m<-(strsplit(date, "-")[[1]][(1)])

d<-(strsplit(date, "-")[[1]][(2)])

tdate<-paste(y, m,d, sep = "-")

#put into list

sto1.t[[i]]<-tdate

#strsplit time

t<-paste((strsplit(sto.t, " ")[[1]][(2:3)]), collapse = " ")

#change 12H to 24H

ttime<-format(strptime(t, "%I:%M:%S %p"), format="%H:%M:%S")

sto1.t[[i]][2]<-ttime

}

sto1.t<-list.reverse(sto1.t)

tdif<-c()

for (i in 2:length(sto1.t)) {

dif<-difftime(paste(sto1.t[[i]][1],sto1.t[[i]][2], sep = " "), paste(sto1.t[[1]][1],sto1.t[[1]][2], sep = " "), units = "hours")

tdif<-c(tdif, dif)

}

tdif<-c(0,tdif)

#create the data.frame structure and column names

u<-data.frame(time=c(tdif))

ro<-c("A","B","C","D","E","F","G","H")

co<-c(1:12)

df<-c() #makes a vector of all the well names used in 'ro' and 'co'

for (i in ro) {

for (j in co) {

df<-c(df,paste(i,j,sep=""))

}

}

for (i in 1:length(df)) { #creates each well name to be a vector in existence

assign(df[i], c(recursive = TRUE))

}

for (i in 1:length(df)) { #assigns vector of well name and column of u with data of each time point at well placement

for (j in 1:length(allSheets)) {

assign(df[i], c(get(df[i]),as.numeric(allSheets[[j]][i,2])))#,allSheets1[[j]][2,],allSheets1[[j]][3,]))[i]))#optimization required here for more than three rows

}

u[,(i+1)]<-get(df[i])

}

colnames(u)<-c("time",df) #give each column the corresponding well name

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

1. Run growthcurver

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

1. Plot growthcurver output

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

1. PCA & plot
2. Sigma histogram
3. Plot parameters

library("readxl")

allSheets<-list()

for (i in 1:c((length(excel\_sheets(fn)))-1)){ #This will read in each sheet of excel file 'fn', if there is a blank sheet at the end of excel file this line will be "for (i in 1:c((length(excel\_sheets(fn)))-1)){"

mySheet <- read\_excel(fn, sheet = i) #This will import a sheet as a dataframe

write.csv(mySheet, file = paste("t", i,".csv", sep = ""), row.names = FALSE) #This will save each sheet as a csv with the file name 't.x', x = sheet number

mySheet<-read.csv(paste("t", paste(i),".csv", sep= ""), header = T, skip = 25, stringsAsFactors=FALSE) #This will read in the data starting at the line number after 'skip='

mySheet<-mySheet[1:3,] #This will filter out your data [row,column], removing row names

allSheets[[i]]<-mySheet[1:tw,] #preserved only first three rows (aurli) and stored data as a list element in allSheets