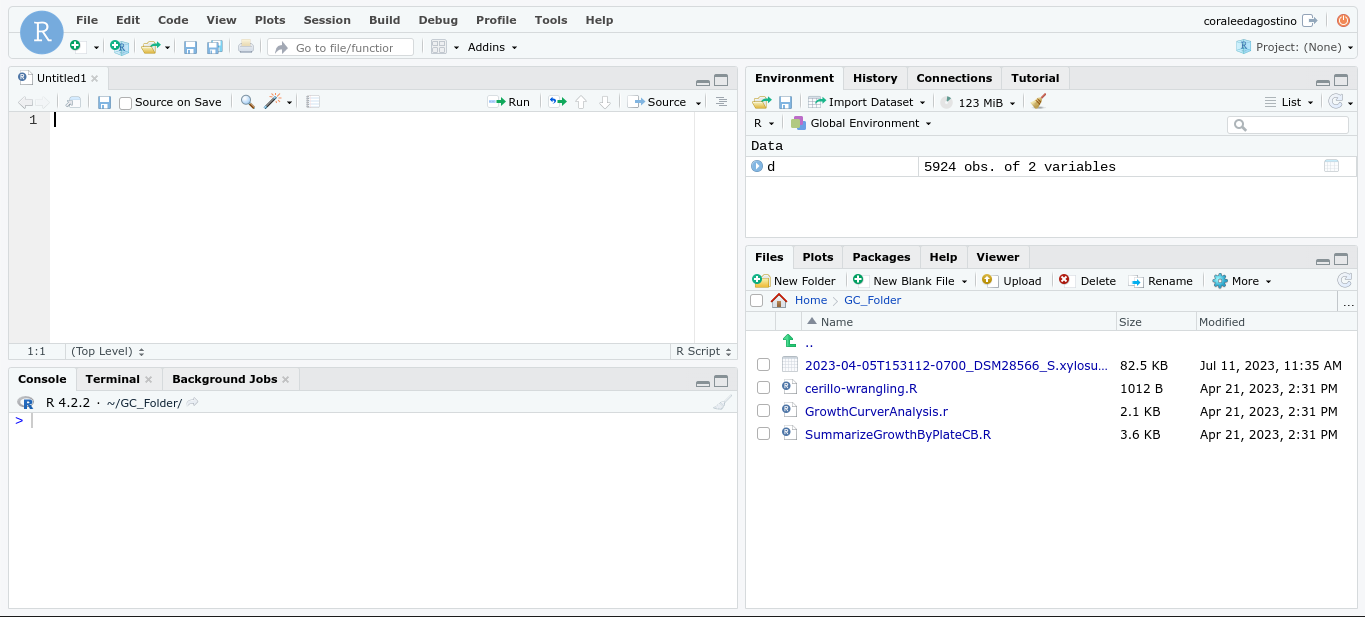
Growthcurver Protocol

July 24th, 2023 Coralee D’Agostino, Christina Guerrero

*The use of the growthcurver package requires the use of RStudio, an integrated development environment that utilizes coding languages R & Python. RStudio runs on Windows, Mac, and Linux (Linux requires extra work to install). If given permission by the PI, one can instead use an interactive session through* [*https://ood.hpc.arizona.edu/*](https://ood.hpc.arizona.edu/) *for a web-browser experience. The user should try to familiarize themselves with RStudio & the R language before running. This protocol includes how to run data from a Cerillo through growthcurver on RStudio as well as ways to fix and manage the code in cases of complex data or errors in computing.*

**Setting Up RStudio for Desktop**

Note: anything in the code that changes from a case to case situation will be shown in blue.

1. First, go to the following link and download the correct version according to your computer’s operating system for both R as well as RStudio: <https://posit.co/download/rstudio-desktop/>
2. Next, you must install the code files, titled “GrowthCurverAnalysis.R”, “SummarizeGrowthByPlateCB.R”, and “cerillo-wrangling.R”, at the following link: <https://github.com/cbaughan/CerilloWrangling> (Credit to Christina Guerrero)
   * On your desktop, drag these downloaded files into a folder that you’ll use specifically for growthcurver. For future reference, this will be called “GC\_Folder”. Make sure this folder is accessible to RStudio (differs per operating system)
   * If working with the online interactive session, these files must be uploaded into the files linked to your account. To do this, open HPC Arizona and then click “Files”>”Home Directory” on the top left bar. From there, press “New DIrectory” to create a new folder, and title it accordingly. For future reference, this will be called “GC\_Folder”. Open GC\_Folder and then click “Upload”. From here, select the desired data file.
3. Download the OD data from Cerillo and transfer this data into the GC\_Folder
4. Open RStudio. It should look like this (colors may vary based on user preference)
   * + *Note: The interface is split into quadrants. Top left is the source, this is where code is written. The bottom left is the console, which is an area where one can interactively execute code. The top right is the environment, which stores R and Python objects. Finally, the bottom right is where files, plots, and packages are located.*

**Running Cerillo Data through Growthcurver Package**

1. Using the files quadrant, find and open the GC\_Folder. Double click on “GrowthCurverAnalysis.R” to open the code in the source.
2. Install the packages. These packages are required to provide additional functions. Install these by clicking on the line of code you intend on running (either by line or click and drag to select entire chunks of text) and hit ctrl+enter. Run the following code:

###### install.packages(“caret”)

###### install.packages(“tidyverse”)

###### install.packages(“readr”)

###### install.packages(“FSA”)

###### install.packages(“growthcurver”)

1. Load the libraries that are associated with these packages, as well as some that are already included in RStudio. These must be loaded every single time you start or reload RStudio. Run the following code:

###### library(reshape2)

###### library(tidyr)

###### library(stringr)

###### library(caret)

###### library(dplyr)

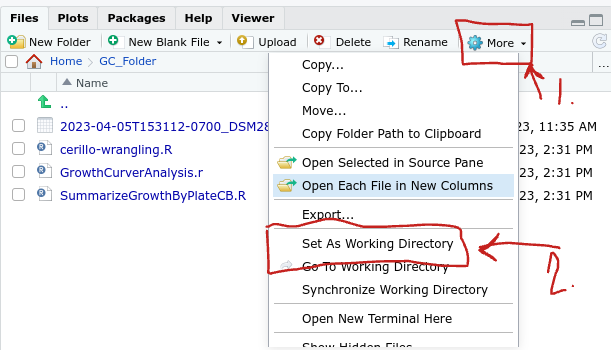
###### library(ggplot2)

###### library(tidyverse)

###### library(readr)

###### library(FSA)

1. Set the working directory. This is very important, as it tells RStudio where it can find, input, and work with files. If not already opened, open the GC\_Folder using the files panel, and then click the “More” button with the cog to open the pull down menu. Then select “Set as Working Directory.”

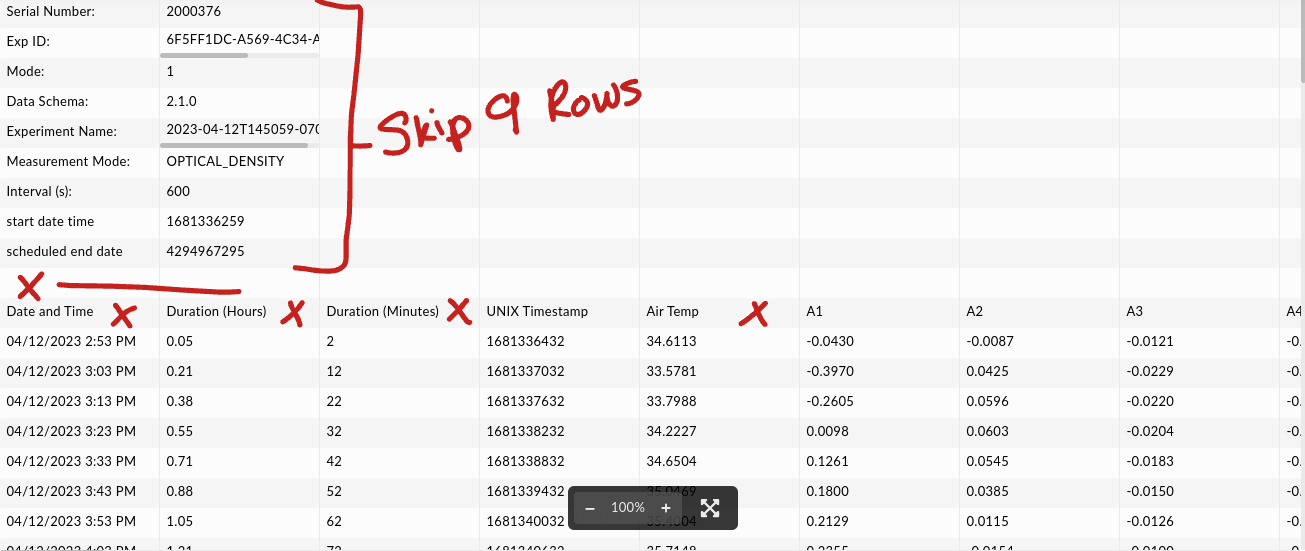


1. Load the source codes. These files should already be in the working directory for it to pull from. Run the following code:

###### source(“SummarizeGrowthByPlateCB.R”)

###### source(“cerillo-wrangling.R

1. Next, choose the data set to work on. Open this data set OUTSIDE of RStudio to view the full set of data. The data should be formatted as seen in the figure below. Red annotations included for further understanding of how the “cerillo-wrangling.R” function will modify the data file.



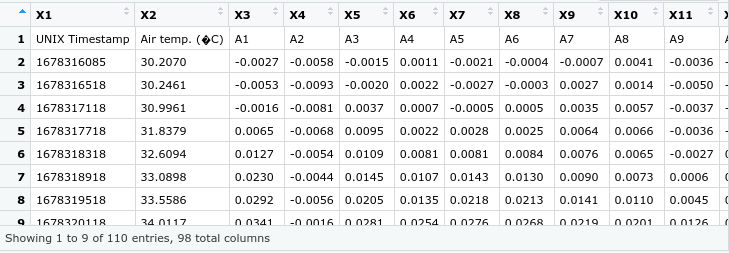
* + Note: This protocol requires that the first 9 rows are skipped (Step 11) and the “Date and Time”, “Duration (Hours)”, “Duration (Minutes)”, and “Air Temp” columns (Step 13). If changes occur in Cerillo outputting, you will likely have to alter the code in these steps.

1. Run the following code, which will open a menu that shows the working directory. Pick the data set you wish to work with. It will also skip the first 9 lines which consist of the settings of the cerillo.

###### d<-read\_csv(file.choose(),

###### col\_names = FALSE, skip = 9)

* + - *If you view your data in RStudio, then it should appear as following*

**

* + - *Note: if this doesn’t work, go to the file quadrant and click on the data set. Select “Import dataset”. This will pull up a view of the dataset. From here, you can manipulate the import options til the data preview looks similar to the one above. Based on the manipulations, it will generate a code in the code preview. You may copy and paste the code from the code preview into the spot of the above code. This should fix the problem.*
    - *Note: Once this step is complete, you can view the data you’re working with at any time in the environment quadrant. Underneath Data, click on “d”. This will show the data that the code is actively working with.*

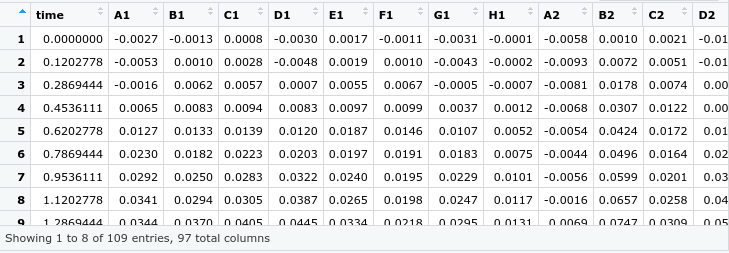
1. Change the output file name. Run the following code and insert the name of the file in “yourfilename” below. This is generally the exact name of the data file.

###### outputPath <- 'yourfilename'

1. Format the data frame by using the cerillo\_wrangling function. Run the following code:

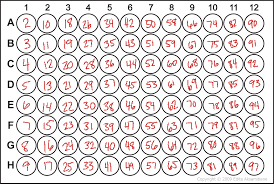
###### d<-cerillo\_wrangling(d)

* + - *If you view your data after running this function, then it should appear similar to this, with time and well names in the top shaded row, with the wells in A1, B1, C1… pattern. This function should delete the“Date and Time”, “Duration (Hours)”, “Duration (Minutes)”, and “Air Temp” columns and convert the Linux Time to time since beginning.*



* + *Note: if experiencing an error with this function, open up “cerillo-wrangling.R” from the files quadrant. This will show you the series of code that it runs when you enter the above code. In order to find the error, manually run each line of code in this function until you find the error. The fix will depend on where the error lies.*

1. Select the blank columns to run a quick check. Run the following code to only select the blank wells. The second line of code *(blanks<-blanks[,-c(2,81)])* will remove columns 2 through 81 (which correspond to wells A1-H10 in a 96 well plate) from the blanks category (leaving columns 11 & 12 as the selected blanks). Column 1 is considered the time. Refer to the diagram below for selection.



###### blanks<- d[seq(1, nrow(d), 3), ] #Take every nth timepoint to save time while plotting

###### blanks<-blanks[,-c(2:81)]

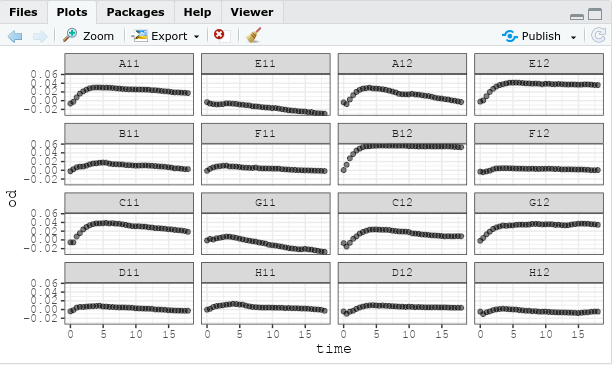
###### blanks<-melt(blanks, id.vars = "time", variable.name = "well", value.name = "od")

1. Run the following code to pull up the plots of the growth curve of the blanks. The plots will show in the files/plots quadrant of the interface.

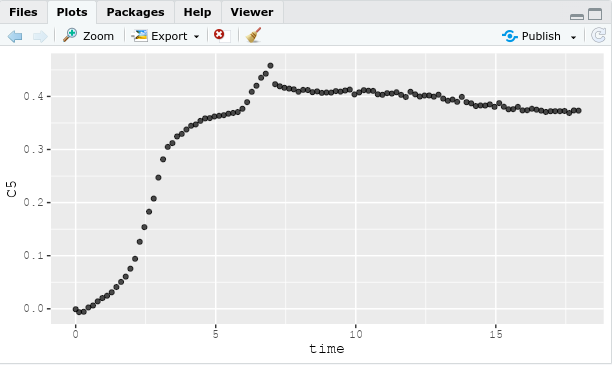
###### ggplot(blanks, aes(x=time, y=od)) + geom\_point(aes(), alpha=0.5) + facet\_wrap(~well, ncol = 4, dir = c("v")) + theme\_bw()

1. From the previously generated blanks file, determine which wells will be used for background correction. Manual select the well used to blank column and fill these selections into the code below .

###### d$blank<-apply(d[, c("H10", "H11", "H12")], 1, mean) #creates a blank column from the means of blank wells

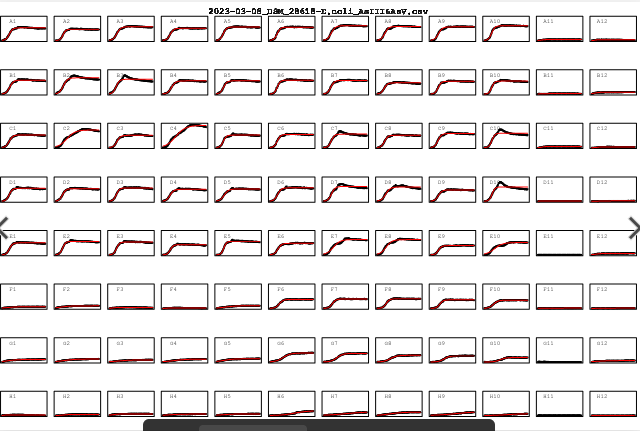
1. Check one well to determine if the data needs to be trimmed. Run the following code to view one of the non-blank wells. It will generate a plot of the growth of the well. Data can be trimmed if the end of the plot looks as if it has plateaued (For example, I can trim past time=15 on the graph below)

###### ggplot(d, aes(x = time, y = C5)) + geom\_point(alpha=0.7)



1. Growthcurver. If trimming is required, insert the time you wish to trim from in the spot of the 0 in the code below. Run the code below to generate and upload a pdf file of the growth curve plots to your working directory file. PDF will look similar to the image attached below.

###### gc\_out<-SummarizeGrowthByPlateCB(d, t\_trim = 0, bg\_correct = "blank", plot\_fit = TRUE, plot\_file = paste(outputPath, "\_gc.pdf", sep = ''))



1. Save data to a csv. Run the following code to generate a table of the data and upload it as a .csv file into your working directory file.

###### write.table(gc\_out, file = paste(outputPath,"\_gc\_table.csv", sep = ""), sep = ",", row.names = FALSE)