AUG 20, 2019

OPEN ACCESS

DOI:
dx.doi.org/10.17504/protocols.io.4fzgtp6

External link:
<https://www.ncbi.nlm.nih.gov/pubmed/31343979>

Protocol Citation: Norman Van Rhijn, Panagiotis Papastamoulis, Takanori Furukawa, Elaine Bignell, Mike Bromley, Magnus Rattray 2019. 96-well Growth Curve Analysis. **protocols.io** <https://dx.doi.org/10.17504/protocols.io.4fzgtp6>

MANUSCRIPT CITATION: Papastamoulis P, Furukawa T, van Rhijn N, Bromley M, Bignell E, Rattray M (2019). Bayesian detection of piecewise linear trends in replicated time-series with application to growth data modelling. The International Journal Of Biostatistics. DOI: 10.1515/ijb-2018-0052.

License: This is an open access protocol distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited

96-well Growth Curve Analysis

In 2 collections

Norman Van Rhijn¹, Panagiotis Papastamoulis²,
 Takanori Furukawa¹, Mike Elaine Bignell¹, Bromley¹,
 Magnus Rattray¹

¹University of Manchester;

²Athens University of Economics and Business, Greece

Norman Van Rhijn
 University of Manchester

ABSTRACT

Generally, growth assays for filamentous fungi have been performed on solid media, either as dilution series or spot tests. However, the solid media environment does not accurately mimic the environment encountered during infection (ie the mammalian lung). Previously, we have developed a methodology to perform liquid growth assays in time for *A. fumigatus* and other filamentous fungi including analysis via mathematical modelling.

This protocol is designed for *Aspergillus* species (and other filamentous fungi) to generate growth curves in liquid media in a 96-well plate. This can be done in high-throughput to generate 96 growth curves per run.

Protocol status: Working
We use this protocol and it's working

Created: Jun 20, 2019

Last Modified: Aug 20, 2019

PROTOCOL integer ID:
24793

Pre-processing

1 Import the file from the previous protocol into R studio.

5m

Software

R Studio Desktop

NAME

The R Studio, Inc.

DEVELOPER

Protocol

NAME

96-well Plate Growth Curve Setup

CREATED BY

Norman Van Rhijn

PREVIEW

1.1 Harvest and dilute spores in PBS+0.01% Tween to 4×10^5 spores/mL.

30m

1.1.1 Aliquot 5 uL of this spore stock per well of a CytoOne 96-well plate (Starlab) non-coated, this will be overlayed with 195 uL of liquid media to a total volume of 200 uL.

20m

Note

This equals to 2000 spores (absolute) per well. 1000 spores or 500 spores are possible to use, but result in more variance within replicates.

1.2 Fill the space between the wells with either media or H₂O. This will avoid an "edge effect", where the wells around the edge of the plate show increased growth compared to other wells.

5m

1.3 Cover the 96 well plate with a breathable cover to allow gas exchange.

2m

1.4 In the Manchester Fungal Infection Group we either use a Powerwave X-2 with KC software or a BioTek platereader. Set this up to run for 48:00:00 at 37 °C to measure OD₆₀₀ every 10 minutes. Take a blank reading before running the plate and preheat the machine. Run your experiment.

5m

This will result in a "growth curve".

Extract or manipulate the data to look like attached (txt file format).

1.5 See our next protocol for analysis.

2 You can find the script below that you need in order to read your raw data-files, format the file names like this:

10m

plate1-Rep1.txt plate1-Rep3.txt plate2-Rep2.txt plate3-Rep1.txt plate3-Rep3.txt plate4-Rep2.txt plate5-Rep1.txt plate5-Rep3.txt
plate1-Rep2.txt plate2-Rep1.txt plate2-Rep3.txt plate3-Rep2.txt plate4-Rep1.txt plate4-Rep3.txt plate5-Rep2.txt

Note

Make sure the different replicates are in different files. However, the location in the plate should be exactly the same (ie. A5 should be A5, B2 should be B2). If this is not the case or replicates are on one plate, transform the original files to match this format. Furthermore, titles of the wells should **exactly** be the same!

- 3 Apply the code below. Change the number of reps and plates to your experiment (top two lines). Make sure the data is formatted as above.

2m

The final object returned after applying the code is the list `myDataList`.

Command

This is to process 96-well growth data to match formatting required for the BEAST CRAN package. Different replicates from different files are merged into one dataframe.

```
number_of_plates <- 5
number_of_reps <- 3

raw_dataset <- vector("list", length = number_of_reps)

for(repl in 1:number_of_reps){
  rawData <- c()
  for(plate in 1:number_of_plates){
    fileName=paste0("plate",plate,"-Rep",repl,".txt")
    cat(paste0("Opening input connection: `", fileName,"` ..."), "\n")
    con=file(fileName,open="r")
    is.first = TRUE
    nline <- 0
    while ( length(line <- readLines(con, n = 1)) > 0 ) {
      nline <- nline + 1
      # if line is blank is skipped
      if (line == ""){
        cat(paste0(" blank line:", nline," skipped"),"\n")
        if(is.first == FALSE){
          rawData <- cbind(rawData, tmp)
        }
      }
    }
  }
}
```

```

}else{
# if line contains comment beginning with: "[Table" is skipped
if( substr(line, 1, 6) == "[Table" ){
  cat(paste0("  comment line: ",nline," skipped"),"\n")
}else{
  if( substr(line, 1, 4) == "Time" ){
    # reading header and discard first "Time" column
    myColNames <- strsplit(line, split = ";")[[1]][-1]
    # add plate info to column names:
    myColNames <- paste0("plate_", plate, "_", myColNames)
    # initialize empty array
    tmp <- array(data = NA, dim = c(0, length(myColNames)))
    colnames(tmp) <- myColNames
  }else{
    # reading data
    tmp <- rbind(tmp,
      as.numeric(strsplit(line, split = ";")[[1]][-1])
    )
    is.first = FALSE
  }

}

}

rawData <- cbind(rawData, tmp)
cat(paste0("Closing input connection: `", fileName,"`."), "\n\n")
close(con)
}
raw_dataset[[repl]] <- rawData
}

# the raw_dataset contains all data, with dimensions: 3 replicates x 289 time-points x 480
time-series

# clean raw_dataset by filtering out blanks and weirdos

averagedData <- array(data = NA, dim = dim(raw_dataset[[1]]))
n <- dim(raw_dataset[[1]])[2]
nTime <- dim(raw_dataset[[1]])[1]
for(i in 1:n){
  repl <- 1
  averagedData[,i] <- raw_dataset[[repl]][,i]
  for(repl in 2:number_of_reps){

```

```

averagedData[,i] <- averagedData[,i] + raw_dataset[[repl]][,i]
}

}
averagedData <- averagedData/number_of_reps
discardWells <- which(averagedData[nTime, ] < 0.015) # this threshold excludes all blank
wells, it should be adjusted accordingly to other applications. Here, a blank well is defined
as one where the average growth level at the last time-point (t = 289) is less than 0.015.
discardWells <- c(discardWells, c(36, 75)) # these two cases are also excluded because
they don't behave nice

myDataList <- lapply(raw_dataset, function(y)y[,-discardWells]) # final dataset with N =
411 time-series.
lapply(myDataList, function(y)dim(y))

```

Run Beast

- 4 Install BEAST software from CRAN into R studio.
<https://rdrr.io/cran/beast/man/beast-package.html>

Manual can be found: <https://cran.r-project.org/web/packages/beast/beast.pdf>

- 5 Run beast. The top is an example for data included in the package (**don't run** if you want to run your own data). Example dataset is attached to run.

Dataset

FungalGrowthDataset

NAME

https://cran.rstudio.com/bin/windows/contrib/3.7/beast_1.1.zip

LINK

If you want an example first, run the top example and reload your data with the code above.

For your own data run, only run the code from Run MCMC sampler and Print and Plot output

lines.

Assume that the observed data consists of N time-series, each one consisting of R variables (or replicates) measured at T consecutive time-points.

The input of the main function `beast` should be given in the form of a list `myDataList` with the following attributes:

- `length(myDataList)` should be equal to R , that is, the number of variables (or replicates)
- `dim(myDataList)[[1]], ..., dim(myDataList)[[R]]` should be all $T \times N$, that is, rows and columns should correspond to time-points and different series, respectively.

Then, a basic usage of the package consists of the following commands:

- `beastRun <- beast( myDataList = myDataList )`
- `print(beastRun)`
- `plot(beastRun)`

which correspond to running the MCMC sampler, printing and plotting output summaries, respectively.

Below a toy-example is given which reproduces Figure 1 of the paper.

```
> library('beast')      # load package
> data("FungalGrowthDataset") # load dataset
> myIndex <- c(392, 62, 3, 117) # run the sampler only for the
#                               specific subset of time-series
> set.seed(1)             # optional
# Run MCMC sampler with the default number of iterations (nIter = 70000):
> run_mcmc <- beast(myDataList = FungalGrowthDataset, subsetIndex = myIndex,
                    zeroNormalization = TRUE)
# Print output:
> print(run_mcmc)
# Plot output to file: "beast_plot.pdf"
> plot(run_mcmc, fileName = "beast_plot_fungal_data.pdf", timeScale=1/6, xlab = "hours", ylab =
"growth")
```

The complexity prior distribution with parameter `gammaParameter = 2` is the default prior assumption imposed on the number of change-points.

Smaller (larger) values of `gammaParameter` will a-priori support larger (respectively: smaller) number of change-points.

Full documentation of the package is given in <https://cran.r-project.org/web/packages/beast/beast.pdf>.

- 6 This should result in a PDF file containing all curves combined clustered in groups with different change-points and individual curves for each isolate.

Expected result