growthcurver test

Patrick Thomas

2/1/2022

Notes:

- I don't like it
- it's super annoying
- it forces you to put your data in wide non-tidy format, so our data would have >3000 columns, not a huge problem just more unnecessary steps
- it only has the logistic function
- time points for different experimental units have to be exactly the same- so as far as I can tell it is impossible to include our data in their required format without looping through every single unique_ID we have
- need to rename time column to "time" or it doesn't work
- basically, it's only useful for well plate assays where growth is very uniform and taking the exact same time points for all your treatments is easy

```
knitr::opts_chunk$set(echo = TRUE, warning = FALSE, message = FALSE)
```

```
library(tidyverse)
library(growthcurver)
library(readxl)

d <- growthdata
```

Example from growthcurver vignette

Wide format needed:

```
head(d)
```

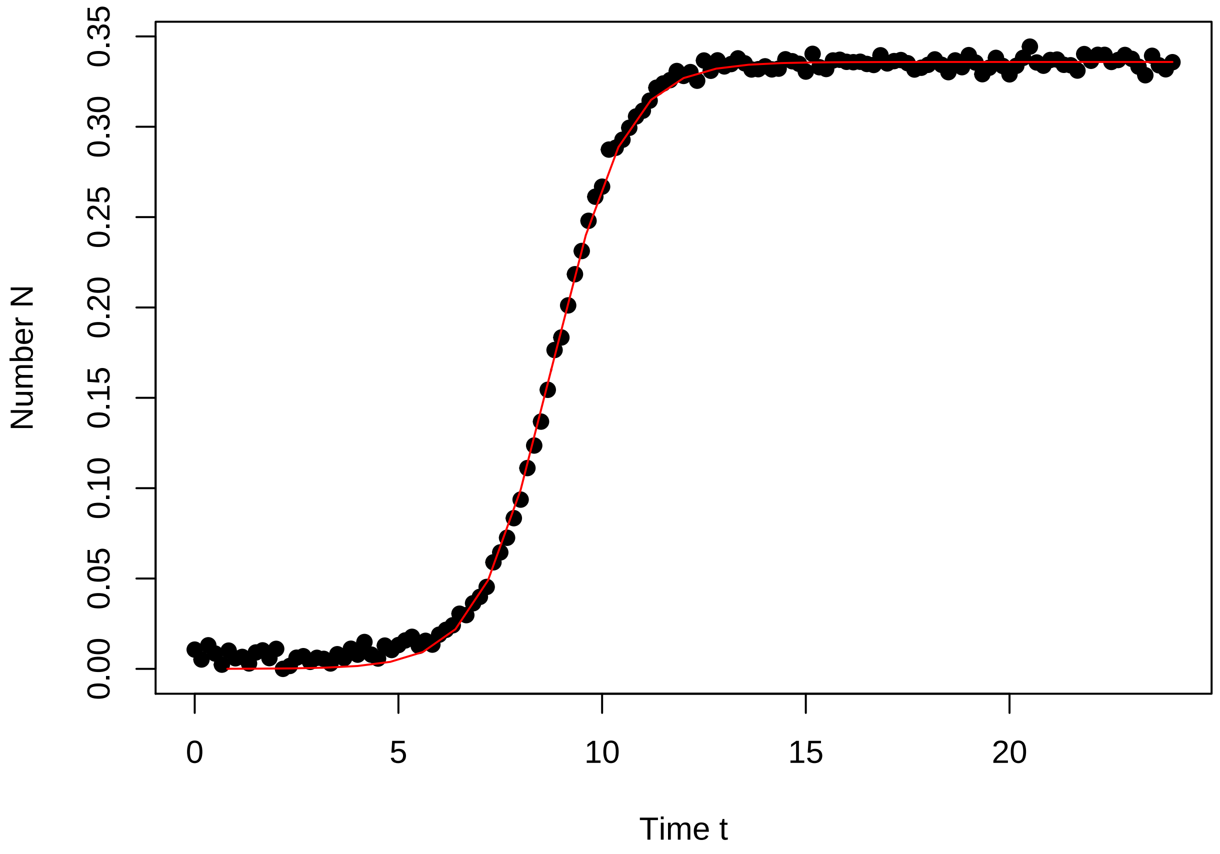
```
## # A tibble: 6 x 97
##    time     A1     B1     C1     D1     E1     F1     G1     H1     A2     B2
##   <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1 0     0.0535 0.0450 0.0525 0.0560 0.0556 0.0482 0.0480 0.0536 0.0543 0.0524
## 2 0.167 0.0480 0.0439 0.0514 0.0535 0.0498 0.0545 0.0508 0.0507 0.0454 0.0517
## 3 0.333 0.0559 0.0500 0.0471 0.0490 0.0548 0.0508 0.0462 0.0488 0.0481 0.0445
## 4 0.5   0.0513 0.0521 0.0482 0.0456 0.0461 0.0552 0.0460 0.0531 0.0488 0.0484
## 5 0.667 0.0452 0.0472 0.0474 0.0515 0.0528 0.0530 0.0462 0.0535 0.0503 0.0506
## 6 0.833 0.0529 0.0494 0.0512 0.0488 0.0466 0.0467 0.0506 0.0502 0.0491 0.0486
## # ... with 86 more variables: C2 <dbl>, D2 <dbl>, E2 <dbl>, F2 <dbl>, G2 <dbl>,
## #   H2 <dbl>, A3 <dbl>, B3 <dbl>, C3 <dbl>, D3 <dbl>, E3 <dbl>, F3 <dbl>,
## #   G3 <dbl>, H3 <dbl>, A4 <dbl>, B4 <dbl>, C4 <dbl>, D4 <dbl>, E4 <dbl>,
## #   F4 <dbl>, G4 <dbl>, H4 <dbl>, A5 <dbl>, B5 <dbl>, C5 <dbl>, D5 <dbl>,
## #   E5 <dbl>, F5 <dbl>, G5 <dbl>, H5 <dbl>, A6 <dbl>, B6 <dbl>, C6 <dbl>,
## #   D6 <dbl>, E6 <dbl>, F6 <dbl>, G6 <dbl>, H6 <dbl>, A7 <dbl>, B7 <dbl>,
## #   C7 <dbl>, D7 <dbl>, E7 <dbl>, F7 <dbl>, G7 <dbl>, H7 <dbl>, A8 <dbl>, ...
```

Fit one growth curve and look at output:

```
gc_fit <- SummarizeGrowth(d$time, d$A1)

gc_fit
```

```
## Fit data to K / (1 + ((K - N0) / N0) * exp(-r * t)):
##      K   N0  r
##   val:   0.336   0   1.119
##   Residual standard error: 0.004685978 on 142 degrees of freedom
##
## Other useful metrics:
##   DT 1 / DT  auc_l   auc_e
##   0.62   1.6e+00 5.11    5.15
```

```
plot(gc_fit)
```



```
gc_fit$vals
```

```
## k    k_se    k_p n0  n0_se   n0_p
## 0.336    0.001   2e-244  0   0   6e-12
##
## r    r_se    r_p sigma   df  t_mid
## 1.119    0.015   4e-115  0.005   142 8.779
##
## t_gen    auc_l   auc_e
## 0.62 5.112   5.152
```

```
str(gc_fit$vals)
```

```
## List of 16
##  $ k    : num 0.336
##  $ k_se : num 0.000552
##  $ k_p  : num 1.65e-244
##  $ n0   : num 1.82e-05
##  $ n0_se: num 2.42e-06
##  $ n0_p : num 5.59e-12
##  $ r    : num 1.12
##  $ r_se : num 0.0151
##  $ r_p  : num 3.57e-115
##  $ sigma: num 0.00469
##  $ df   : num 142
##  $ t_mid: num 8.78
##  $ t_gen: num 0.62
##  $ auc_l: num 5.11
##  $ auc_e: num 5.15
##  $ note : chr ""
##  - attr(*, "class")= chr "gcvals"
```

```
gc_fit$vals$r
```

```
## [1] 1.118657
```

Fit one plate (or many curves) at a time and check for notes on bad fits:

```
gc_out <- SummarizeGrowthByPlate(d)

head(gc_out)
```

```
##   sample         k           n0         r     t_mid     t_gen    auc_l    auc_e
## 1     A1 0.3358696 1.823365e-05 1.1186573  8.779414 0.6196242 5.112115 5.151667
## 2     B1 0.4041318 1.544352e-05 1.0223886  9.949513 0.6779684 5.678234 5.718521
## 3     C1 0.3706032 1.759433e-05 0.9865605 10.090879 0.7025896 5.154747 5.191147
## 4     D1 0.3819837 1.948897e-05 1.0257106  9.635499 0.6757726 5.486987 5.538353
## 5     E1 0.3700136 1.504866e-05 1.1968446  8.447183 0.5791455 5.754741 5.780758
## 6     F1 0.2599559 2.117161e-05 0.9024019 10.433849 0.7681136 3.526579 3.566355
##         sigma note
## 1 0.004685978
## 2 0.004438284
## 3 0.004409867
## 4 0.005408965
## 5 0.003887069
## 6 0.004417922
```

```
gc_out %>% filter(note != "")
```

```
##  [1] sample k      n0     r      t_mid  t_gen  auc_l  auc_e  sigma  note
## <0 rows> (or 0-length row.names)
```

Test using our data

> pivoting reveals some duplicated data points, not sure what's up with those, but they look identical so I will just collapse those into one when pivoting

```
test_wide <- test %>%
  select(unique_ID, time_days, RFU_platereader) %>%
  rename(time = time_days) %>%
  pivot_wider(names_from = unique_ID,
              values_fn = mean,
              values_from = RFU_platereader)
```

> also that didn't work at all >:(

```
gc_out <- SummarizeGrowthByPlate(test_wide)
head(gc_out)
```

```
##        sample k n0 r t_mid t_gen auc_l auc_e sigma            note
## 1 AH_2020_196 0  0 0     0     0     0     0     0 cannot fit data
## 2 AH_2020_197 0  0 0     0     0     0     0     0 cannot fit data
## 3 AH_2020_201 0  0 0     0     0     0     0     0 cannot fit data
## 4 AH_2020_202 0  0 0     0     0     0     0     0 cannot fit data
## 5 AH_2020_206 0  0 0     0     0     0     0     0 cannot fit data
## 6 AH_2020_207 0  0 0     0     0     0     0     0 cannot fit data
```

```
gc_out %>% filter(note != "")
```

```
##             sample k n0 r t_mid t_gen auc_l auc_e sigma            note
## 1      AH_2020_196 0  0 0     0     0     0     0     0 cannot fit data
## 2      AH_2020_197 0  0 0     0     0     0     0     0 cannot fit data
## 3      AH_2020_201 0  0 0     0     0     0     0     0 cannot fit data
## 4      AH_2020_202 0  0 0     0     0     0     0     0 cannot fit data
## 5      AH_2020_206 0  0 0     0     0     0     0     0 cannot fit data
## 6      AH_2020_207 0  0 0     0     0     0     0     0 cannot fit data
## 7      AH_2020_211 0  0 0     0     0     0     0     0 cannot fit data
## 8      AH_2020_212 0  0 0     0     0     0     0     0 cannot fit data
## 9      AH_2020_216 0  0 0     0     0     0     0     0 cannot fit data
## 10      AH_2020_217 0  0 0     0     0     0     0     0 cannot fit data
## 11      IG_B3_1_L10 0  0 0     0     0     0     0     0 cannot fit data
## 12      IG_C3_1_L10 0  0 0     0     0     0     0     0 cannot fit data
## 13      IG_D3_1_L10 0  0 0     0     0     0     0     0 cannot fit data
## 14      IG_E3_1_L10 0  0 0     0     0     0     0     0 cannot fit data
## 15      IG_F3_1_L10 0  0 0     0     0     0     0     0 cannot fit data
## 16      IG_G3_1_L10 0  0 0     0     0     0     0     0 cannot fit data
## 17      IG_B3_10_L6 0  0 0     0     0     0     0     0 cannot fit data
## 18      IG_C3_10_L6 0  0 0     0     0     0     0     0 cannot fit data
## 19      IG_D3_10_L6 0  0 0     0     0     0     0     0 cannot fit data
## 20      IG_F3_10_L6 0  0 0     0     0     0     0     0 cannot fit data
## 21     Patrick_p1A1 0  0 0     0     0     0     0     0 cannot fit data
## 22     Patrick_p1A2 0  0 0     0     0     0     0     0 cannot fit data
## 23     Patrick_p1A3 0  0 0     0     0     0     0     0 cannot fit data
## 24     Patrick_p1A4 0  0 0     0     0     0     0     0 cannot fit data
## 25     Patrick_p1A5 0  0 0     0     0     0     0     0 cannot fit data
## 26     Patrick_p1A6 0  0 0     0     0     0     0     0 cannot fit data
## 27     Patrick_p1B1 0  0 0     0     0     0     0     0 cannot fit data
## 28     Patrick_p1B2 0  0 0     0     0     0     0     0 cannot fit data
## 29     Patrick_p1B3 0  0 0     0     0     0     0     0 cannot fit data
## 30     Patrick_p1B4 0  0 0     0     0     0     0     0 cannot fit data
## 31   Vanessa_expA_1 0  0 0     0     0     0     0     0 cannot fit data
## 32  Vanessa_expA_10 0  0 0     0     0     0     0     0 cannot fit data
## 33  Vanessa_expA_11 0  0 0     0     0     0     0     0 cannot fit data
## 34  Vanessa_expA_12 0  0 0     0     0     0     0     0 cannot fit data
## 35   Vanessa_expA_2 0  0 0     0     0     0     0     0 cannot fit data
## 36   Vanessa_expA_3 0  0 0     0     0     0     0     0 cannot fit data
## 37   Vanessa_expA_4 0  0 0     0     0     0     0     0 cannot fit data
## 38   Vanessa_expA_5 0  0 0     0     0     0     0     0 cannot fit data
## 39   Vanessa_expA_6 0  0 0     0     0     0     0     0 cannot fit data
## 40   Vanessa_expA_7 0  0 0     0     0     0     0     0 cannot fit data
```

even with subset of one person's data it doesn't work - only seems to work when observations are all for the exact same time points

```
A_wide <- A %>%
  select(unique_ID, time_days, RFU_platereader) %>%
  #filter(unique_ID %in% c("AH_2020_217", "AH_2020_221", "AH_2020_222")) %>%
  rename(time = time_days) %>%
  pivot_wider(names_from = unique_ID,
              values_fn = mean,
              values_from = RFU_platereader)

gc_out <- SummarizeGrowthByPlate(A_wide)
head(gc_out)
```

```
##        sample k n0 r t_mid t_gen auc_l auc_e sigma            note
## 1 AH_2020_196 0  0 0     0     0     0     0     0 cannot fit data
## 2 AH_2020_197 0  0 0     0     0     0     0     0 cannot fit data
## 3 AH_2020_201 0  0 0     0     0     0     0     0 cannot fit data
## 4 AH_2020_202 0  0 0     0     0     0     0     0 cannot fit data
## 5 AH_2020_206 0  0 0     0     0     0     0     0 cannot fit data
## 6 AH_2020_207 0  0 0     0     0     0     0     0 cannot fit data
```

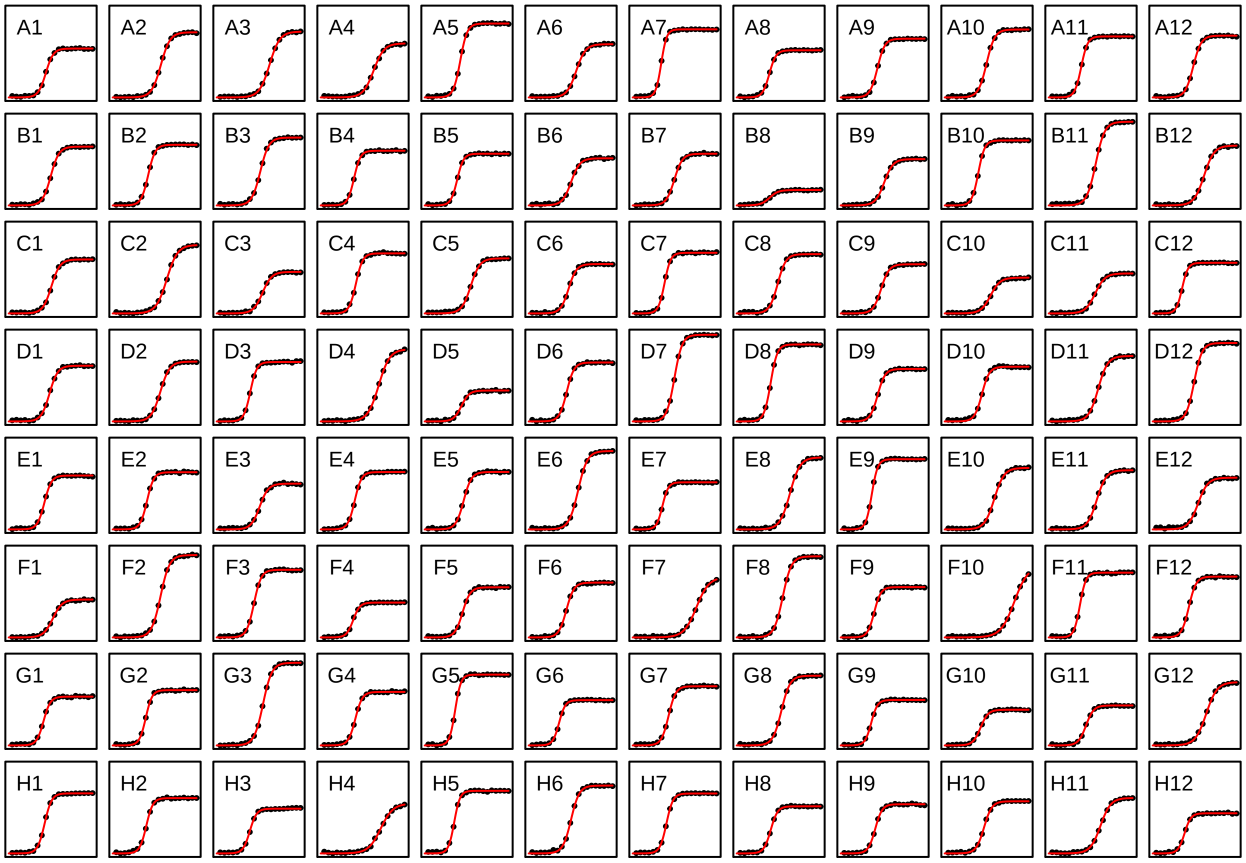
This is the only way I got it to work- when I used a tiny subset of data with the exact same time points corresponding to several experimental units...

```r
A_wide <- A %>%
  select(unique_ID, time_days, RFU_platereader) %>%
  filter(unique_ID %in% c("AH_2020_217", "AH_2020_221", "AH_2020_222")) %>%
  rename(time = time_days) %>%
  pivot_wider(names_from = unique_ID,
              values_fn = mean,
              values_from = RFU_platereader)

gc_out <- SummarizeGrowthByPlate(A_wide)
head(gc_out)
```

```
##        sample         k           n0         r     t_mid     t_gen    auc_l
## 1 AH_2020_217 160.78981 1.9443072305 0.3973259 11.081651 1.7445308 1914.963
## 2 AH_2020_221  82.84668 0.0005924022 1.3222261  8.960880 0.5242274 1163.094
## 3 AH_2020_222 113.02587 0.0022848419 1.1358695  9.516107 0.6102349 1524.027
##    auc_e    sigma note
## 1 1971.5 14.37721
## 2 1260.0 19.00232
## 3 1651.0 25.61631
```

For more "advanced users" to customize things, this huge chunk of code is required. At this point, what's the points of even having a package to help? It also requires data to be in the same extremely

```r
# As in the simple example, load the package and the data.
library(growthcurver)
d <- growthdata

# Let's create an output data frame to store the results in.
# We'll create it so that it is the right size (it's faster this way!),
# but leave it empty.
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)

# Truncate or trim the input data to observations occuring in the first 20 hours.
# Remember that the times in these sample data are reported in hours. To use
# minutes (or to trim at a different time), change the next line of code.
# For example, if you still would like to trim at 20 hours, but your time data
# are reported in minutes use: trim_at_time <- 20 * 60
trim_at_time <- 20

# Now, loop through all of the columns in the data frame. For each column,
# run Growthcurver, save the most useful metrics in the output data frame,
# and make a plot of all the growth curve data and their best fits.

# First, create a plot for each of the wells in the 96-well plate.
# Uncomment the next line to save the plots from your 96-well plate to a
# pdf file in the working directory.
# pdf("growthcurver.pdf", height = 8.5, width = 11)
par(mfcol = 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)
     lines(gc_fit$data$t, predict(gc_fit$model), col = "red")
  }
}
```


A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12
B1 B2 B3 B4 B5 B6 B7 B8 B9 B10 B11 B12
C1 C2 C3 C4 C5 C6 C7 C8 C9 C10 C11 C12
D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12
E1 E2 E3 E4 E5 E6 E7 E8 E9 E10 E11 E12
F1 F2 F3 F4 F5 F6 F7 F8 F9 F10 F11 F12
G1 G2 G3 G4 G5 G6 G7 G8 G9 G10 G11 G12
H1 H2 H3 H4 H5 H6 H7 H8 H9 H10 H11 H12