Maximum Growth Rates Estimation

with

\textit{R}-package \texttt{growthrates}

Thomas Petzoldt, David Kneis and Claudia Seiler

Dresden University of Technology, Faculty of Environmental Sciences, Dresden (Germany)
The growth rate of a population is a measure of fitness.
- applicable to animals, plants, bacteria
- test substances / model organisms
  - resources (nutrients)
  - toxic substances (micro-pollutants, pharmaceuticals, antibiotics)
  - physical environment (high temperatures)
- growth rate $\sim$ resource or stressor
Microwell plate with 384 wells

- modern biology → large amounts of data
- data analysis → bottleneck
Data analysis bottleneck

Limitations of proprietary laboratory software

- high manual effort with “user friendly” Windows or Mac programs
- data import/export between different software packages

R can do it. What’s the problem?

Not all biologists “want to spend the rest of their lives as programmers”\(^1\):

- Babylonian diversity of parametric models
- → difficulty to compare approaches
- → automatisation: need to fit lots of growth rates in a batch

Fitting growth rates was a FAQ for me.

---

\(^1\) Geoff Patton at ECOLOG-L, https://listserv.umd.edu/cgi-bin/wa?A2=ind0505d&L=ecolog-l&D=0&P=5326
Create a package for USErS

Different approaches – unique user interface

- *growth rates made easy*-approach (Hall et al. 2013, MolbEv)
- smoothing spline approach (Kahm et al. 2010, JSS)
- parametric models

Generic data structure, generic methods *(S3/S4 OOP)*

- formula interface to fit several data sets at once
- functions to extract results
- visualization functions

Small set of built-in models – extensible by the user

- given in closed form or as system of ODEs
- numerical integration of ODE models
- option to use compiled code
Database-like and NOT cross tables

```r
library("growthrates")
data(bactgrowth)
head(bactgrowth)
```

<table>
<thead>
<tr>
<th>strain</th>
<th>replicate</th>
<th>conc</th>
<th>time</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0.013</td>
</tr>
<tr>
<td>T</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0.014</td>
</tr>
<tr>
<td>T</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0.017</td>
</tr>
<tr>
<td>T</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>0.022</td>
</tr>
<tr>
<td>T</td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>0.030</td>
</tr>
<tr>
<td>T</td>
<td>2</td>
<td>0</td>
<td>5</td>
<td>0.039</td>
</tr>
</tbody>
</table>

- independent variable: time
- dependend variable: value (counts, optical density, . . .)
- arbitrary number of factors (strain, replicate, treatment, . . .)
“Easylinear”, Spline, Parametric fits

- **Easylinear**
  - $m_{\text{max}} = 0.86$

- **Spline**
  - $m_{\text{max}} = 0.72$

- **Logistic**
  - $m_{\text{max}} = 0.76$

- **Huang**
  - $m_{\text{max}} = 0.69$

- **User Defined**
  - $m_{\text{max}} = 0.66$
Heuristic approach: “growth rates made easy”

Proposed by Hall et al. 2013

1. log transform the data
2. select all sets of 5 consecutive points
3. fit linear regressions
4. take the one with the steepest slope
5. include more points as long as slope is $\geq 95\%$ of maximum slope

Shiny Demo: http://limno-live.hydro.tu-dresden.de/apps/grow-easy
Heuristic approach: “growth rates made easy”

\[
\text{splitted.data} \leftarrow \text{multisplit(}
\begin{align*}
\text{value} & \sim \text{time}|\text{strain:conc:replicate, data=bactgrowth} \\
\text{dat} & \leftarrow \text{splitted.data[[1]]} \\
\text{fit} & \leftarrow \text{fit\_easylinear(dat[,\text{time}], dat[,\text{value}])} \\
\text{plot(fit); plot(fit, log="y")}
\end{align*}
\]

- multisplit with formula interface
- plot, summary, coef, deviance, ..., results
Smoothing spline method

```r
fit2 <- fit_spline(dat$time, dat$value)
plot(fit2); plot(fit2, log="y")
```

- $\mu_{max} = \text{maximum of 1st derivative}$
- degree of smoothing:
  - cross validation (by smooth.spline)
  - fixed value spar, plate reader data are typically auto-correlated
Parametric models

In closed form

- exponential, logistic, Gompertz, Baranyi, Huang
- package \texttt{FME} for (constrained) nonlinear optimization

As differential equation (ODE)

- twostep, generalized logistic
- package \texttt{deSolve} for numerical integration

User-defined

- in closed form or as ODE
- plain R or with packages \texttt{cOde (C)} or \texttt{rodeo (Fortran)}
Parametric models: An example

antibiotic <- readRDS("antibiotic.Rda")
dat <- subset(antibiotic, conc==0.078 & repl=="R4")
parms <- c(y0=0.01, mumax=0.2, K=0.5)
fit <- fit_growthmodel(grow_logistic, parms, dat$time, dat$value)
plot(fit); plot(fit, log="y")

Alternatives to the standard logistic function in a Shiny Demo:
http://limno-live.hydro.tu-dresden.de/apps/grow-generic
grow_userdefined <- function(time, parms) {
    with(as.list(parms), {
        y <- ((K + dK*time) * y0) /
            (y0 + ((K + dK*time) - y0) * exp(-mumax * time))
        as.matrix(data.frame(time = time, y = y, log_y = log(y)))
    })
}

parms <- c(y0=0.01, mumax=0.2, K=0.5, dK=0.01)
fit2 <- fit_growthmodel(grow_userdefined, parms, dat$time, dat$value)
Fit a complete experiment

```r
fits <- all_growthmodels(
  value ~ grow_userdefined(time, parms) | conc + repl,
  data = antibiotic, p = parms, ncores = 4)
head(results(fits))
```

<table>
<thead>
<tr>
<th>conc</th>
<th>repl</th>
<th>y0</th>
<th>mumax</th>
<th>K</th>
<th>dK</th>
<th>r2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0:R3</td>
<td>0.000</td>
<td>0.0033510</td>
<td>0.9616455</td>
<td>0.3197630</td>
<td>0.0070695</td>
<td>0.9963226</td>
</tr>
<tr>
<td>0.002:R3</td>
<td>0.002</td>
<td>0.0026947</td>
<td>1.0302559</td>
<td>0.3281906</td>
<td>0.0068864</td>
<td>0.9960925</td>
</tr>
<tr>
<td>0.005:R3</td>
<td>0.005</td>
<td>0.0025441</td>
<td>1.0416826</td>
<td>0.3235049</td>
<td>0.0070012</td>
<td>0.9968601</td>
</tr>
<tr>
<td>0.01:R3</td>
<td>0.010</td>
<td>0.0031749</td>
<td>0.9735880</td>
<td>0.3311022</td>
<td>0.0069075</td>
<td>0.9961030</td>
</tr>
<tr>
<td>0.02:R3</td>
<td>0.020</td>
<td>0.0035295</td>
<td>0.9330783</td>
<td>0.3389016</td>
<td>0.0068939</td>
<td>0.9953515</td>
</tr>
<tr>
<td>0.039:R3</td>
<td>0.039</td>
<td>0.0035379</td>
<td>0.8530223</td>
<td>0.3455108</td>
<td>0.0063871</td>
<td>0.9961332</td>
</tr>
</tbody>
</table>

- options: box-constraints (lower, upper), selection of parameters (which), tuning of optimization methods, multi-core
- see also: all_easylinear, all_splines
Fit a functional response curve

```
res <- subset(results(fits), r2 > 0.9)
m <- nls(mumax ~ a*exp(b*conc) + c,
        data=res, start=c(a=1, b=-0.1, c=0.05))
```

- standard plotting functions
- confidence intervals with package `propagate` (Spiess, 2014)
Summary

Package growthrates

- three methods with unique interface
- plain data structure
- single and multiple fits

Convenient and Extensible

- fit..., all... summary, plot, ..., results
- pre- and user-defined growth models
- numerically solved ODE models

→ building blocks for complex statistical and mechanistic models

More reading

- https://github.com/tpetzoldt/growthrates
- https://cran.r-project.org/package=growthrates
Nonlinear mixed effects models with \texttt{lme4} can fit both, growth rate and functional response in \textit{one step}, given we have a good overall model, reasonable start parameters for it, and an in-depth understanding of both the methodology the data set.

Package \texttt{growthrates} can help to explore data and models and prepare us for this next step . . .
library("lme4")

f_custom <- function(time, conc, y0, K, dK, a, b)
  ((K + dK*time) * y0) / (y0 + ((K + dK*time) - y0) *
   exp((-a*exp(-b*conc) + 0.01) * time))

## analytical derivative
d_custom <- deriv(  
  body(f_custom),  
  namevec = c("y0", "K", "dK", "a", "b"),  
  function.arg = f_custom  
)

mm <- nlmer(value ~ d_custom(time, conc, y0, K, dK, a, b)  
  ~ (K|conc:repl) + (dK|conc:repl) + (y0|conc:repl),  
  data = antibiotic,  
  start = c(y0=0.02, K=0.3, dK=0.005, a=0.8, b=3),  
  nlmerControl(optimizer="bobyqa"))

summary(mm)
lme4 works :)