growthrates R package

ESM 211 Winter 2024

Linus Blomqvist

2024-01-30

What does the package do?

- `growthrates` offers various ways of assessing growth rates (and other growth parameters) in populations
- For example, finding the r in an exponential growth function, or r_{max} and K in a logistic growth function
- Can also estimate dose response curves
- It doesn't have its own model-fitting routines, but rather serves as a convenient wrapper for other packages

Application

- The models can be applied to any type of population, from microbes to bison
- For the simplest applications, all you need is a time series of population size
- Can also handle multiple time series from a single experiment (see example)

Example

- I will use a dataset on bacterial growth that comes with the package
- Three strains of bacteria (D = Donor, R = Recipient, T = transconjugant)
- Different concentrations of the antibiotic tetracycline
- Read off at 31 different times (0:30)
- Two replicates

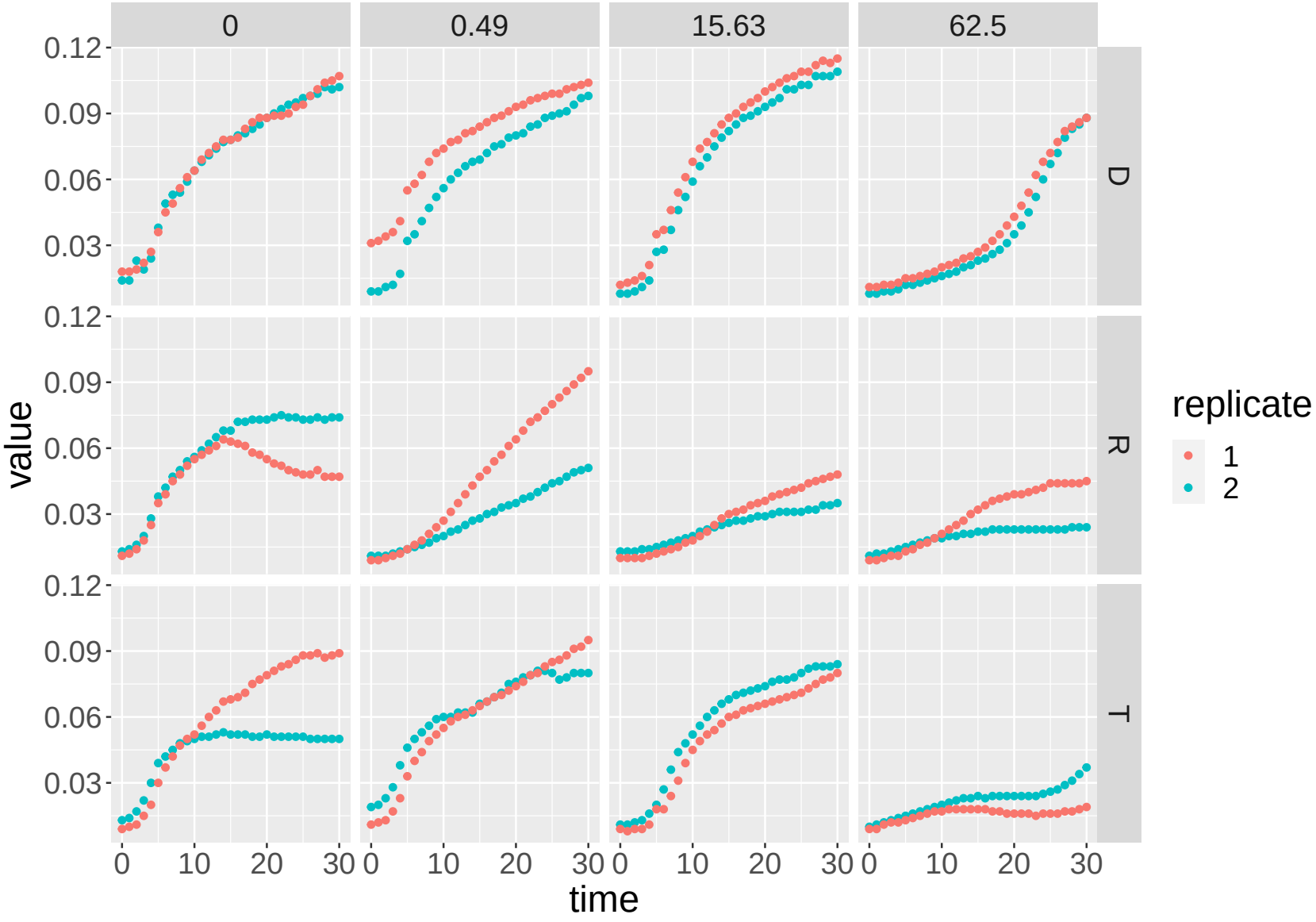
Load, inspect, subsample data

```r
# Load dataset and show summary
data(bactgrowth)
summary(bactgrowth)
```

```
##  strain    replicate        conc                time         value       
##  D:744   Min.   :1.0   Min.   :  0.0000   Min.   : 0   Min.   :0.008  
##  R:744   1st Qu.:1.0   1st Qu.:  0.8575   1st Qu.: 7   1st Qu.:0.019  
##  T:744   Median :1.5   Median :  5.8600   Median :15   Median :0.036  
##          Mean   :1.5   Mean   : 41.6467   Mean   :15   Mean   :0.045  
##          3rd Qu.:2.0   3rd Qu.: 39.0625   3rd Qu.:23   3rd Qu.:0.070  
##          Max.   :2.0   Max.   :250.0000   Max.   :30   Max.   :0.153
```

```r
# Create subsample to fit graphs on one page
bactgrowth_small <- bactgrowth %>%
  filter(conc %in% sample(conc, 4))
```

Plot data


0
0.49
15.63
62.5
D
R
T
value
0.12
0.09
0.06
0.03
0
10
20
30
time
replicate
1
2

Simplest case: single dataset (time series)

- Three functions: `fit_easylinear`, `fit_growthmodels`, and `fit_splines`
- For single dataset (just one growth curve), need to subset data

```
# Subset bactgrowth data
splitted.data <- multisplit(bactgrowth,
                            c("strain", "conc", "replicate"))
dat <- splitted.data[[1]]
```

This includes only strain D, replicate 1, and concentration 0

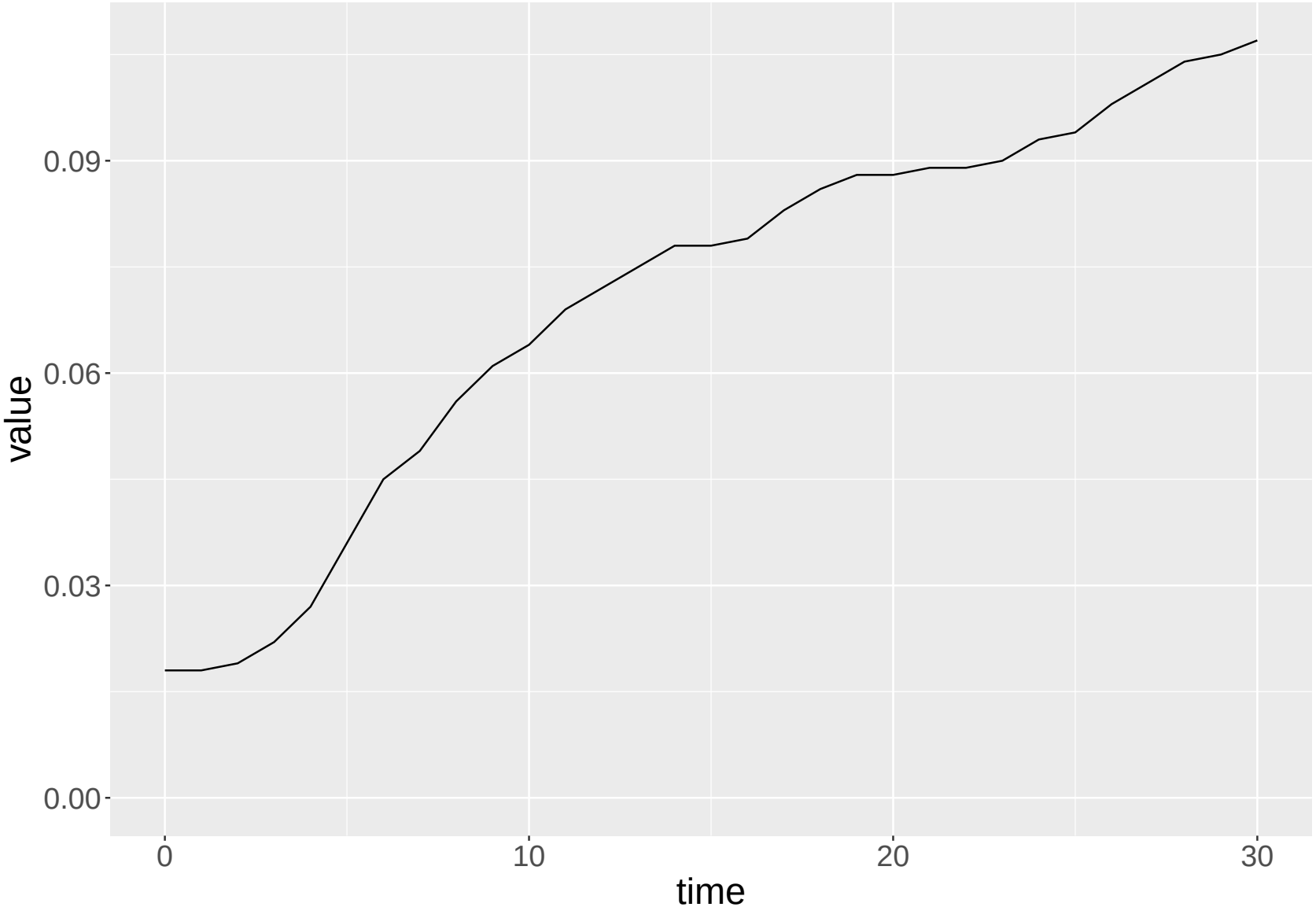
Inspect dataset for single growth curve

```
head(dat %>%
       select(time, value))
```

```
##      time value
## 1117    0 0.018
## 1118    1 0.018
## 1119    2 0.019
## 1120    3 0.022
## 1121    4 0.027
## 1122    5 0.036
```

This is all the data you need to find r, K, and other parameters. Make sure time starts at 0.

Plot single growth curve



Finding r

Exponential growth:

$$N_t = N_0 e^{rt}$$

Derivative of exponential growth:

$$ln(N_t) = ln(N_0) + rt$$

Can be evaluated as a linear function:

$$y = b_0 + b_1 t$$

where $b_0 = ln(N_0)$ and $b_1 = r$.

Using `growthrates` to find *r*

The function `fit_easylinear` finds the maximum growth rate on the exponential segment of the growth curve.

`fit_easylinear` takes four arguments:

```
fit_easylinear(time, y, h = 5, quota = 0.95)
```

- `h` = number of data points (see next slide)
- `quota` = how much of adjacent data to include

How `fit_easylinear`'s algorithm works

1. Fit linear regressions to all subsets of h consecutive data points. If for example $h = 5$, fit a linear regression to points 1 . . . 5, 2 . . . 6, 3. . . 7 and so on. The method seeks the highest rate of exponential growth, so the dependent variable is log-transformed.
2. Find the subset with the highest slope bmax and include also the data points of adjacent subsets that have a slope of at least $quota \cdot bmax$, e.g. all data sets that have at least 95% of the maximum slope.
3. Fit a new linear model to the extended data window identified in step 2.

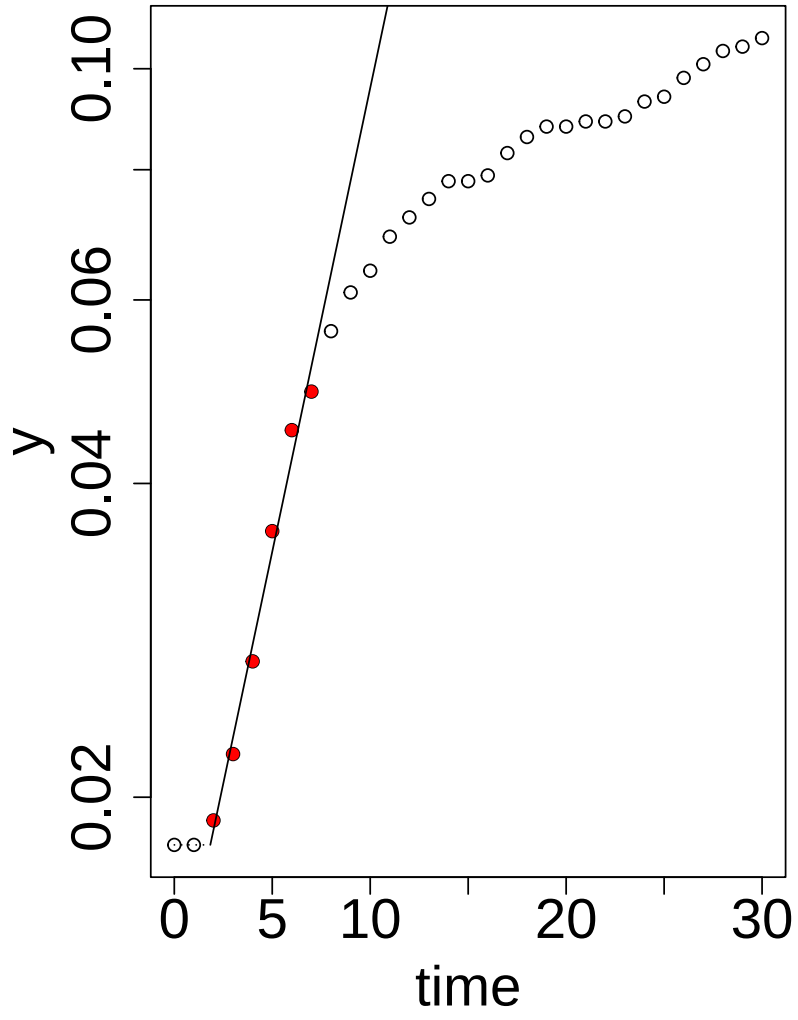
Applying `fit_easylinear`

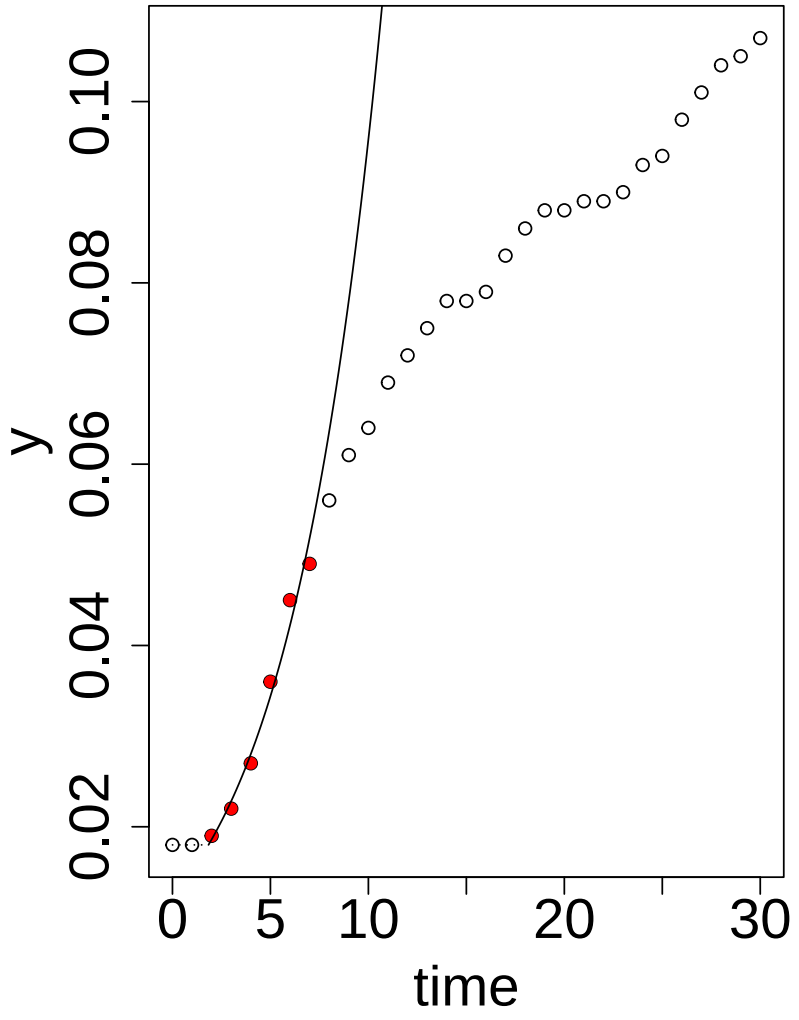
```
fit <- fit_easylinear(dat$time, dat$value)

# Keeping defaults for h and quota

coef(fit)
```

```
##         y0      y0_lm      mumax        lag
## 0.0180000 0.0123482 0.2048985 1.8392607
```

mumax is r.

Plotting the results





Using `fit_splines`

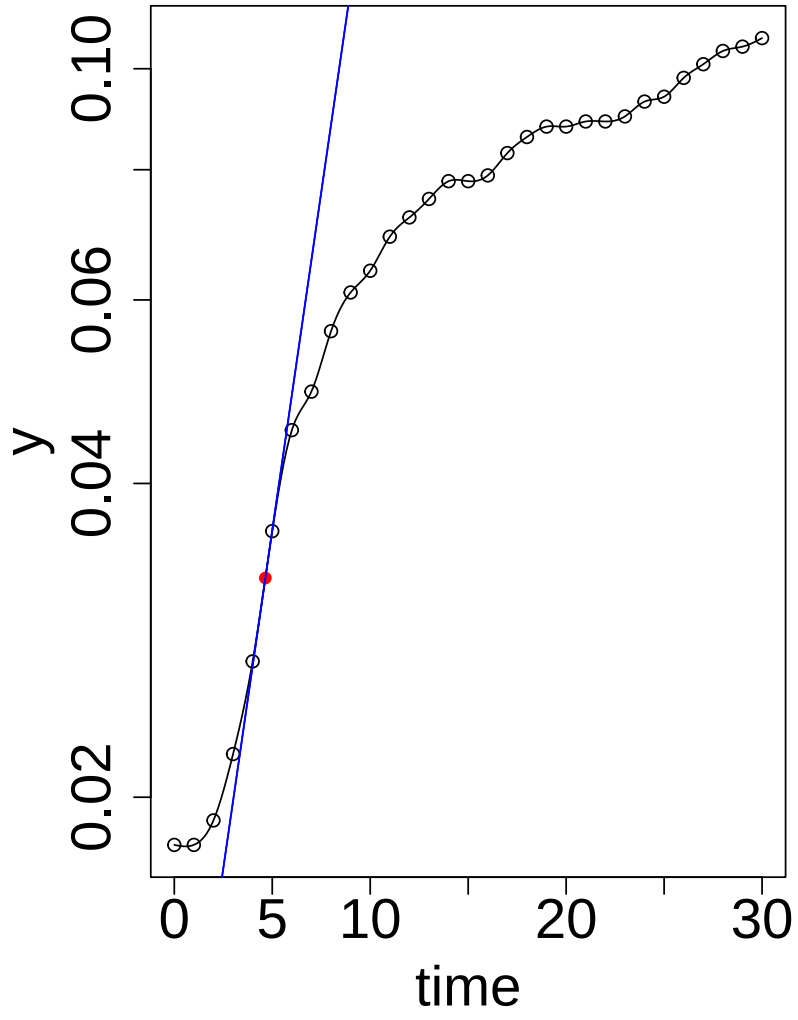
Splines fit models piecewise to a dataset, with a polynomial for each segment, and some fancy math to make it nice and smooth.

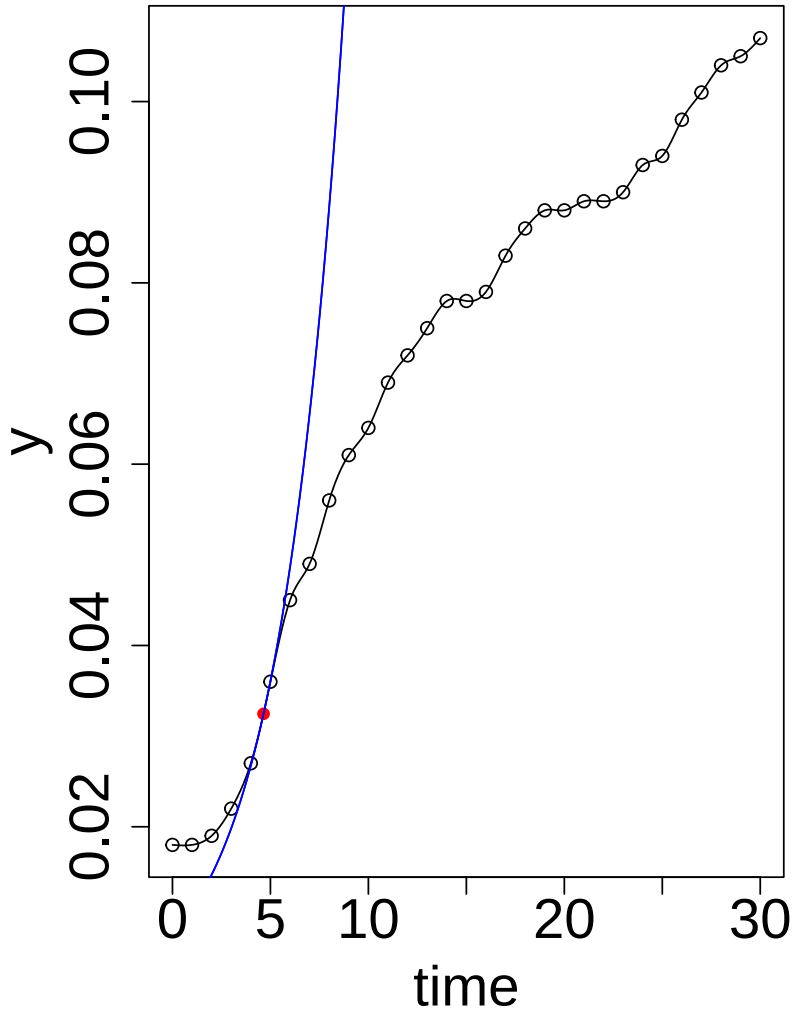
`growthrates` has the function `fit_splines` that can find r using this method

```
res <- fit_spline(dat$time, dat$value)

coef(res)
```

```
##           y0       mumax
## 0.008092223 0.298829681
```

Plotting the result





Fitting parametric nonlinear models

- `growthrates` can also be used to fit parametric nonlinear models with the `fit_growthmodel` function

```
fit_growthmodel(
FUN,
p,
time,
y,
lower = -Inf,
upper = Inf,
which = names(p),
method = "Marq",
transform = c("none", "log"), control = NULL,
...)
```

Applying `fit_growthmodel`

`FUN` defines the growth model; we will use `FUN = grow_logistic`

$$\frac{dN}{dT} = r_{max}\frac{(K-N)}{K}N$$

`p` sets start values for the parameters `y_0`, `mumax`, and `K`, where `mumax` is equivalent to r_{max}. Make your best guess based on an inspection of the data.

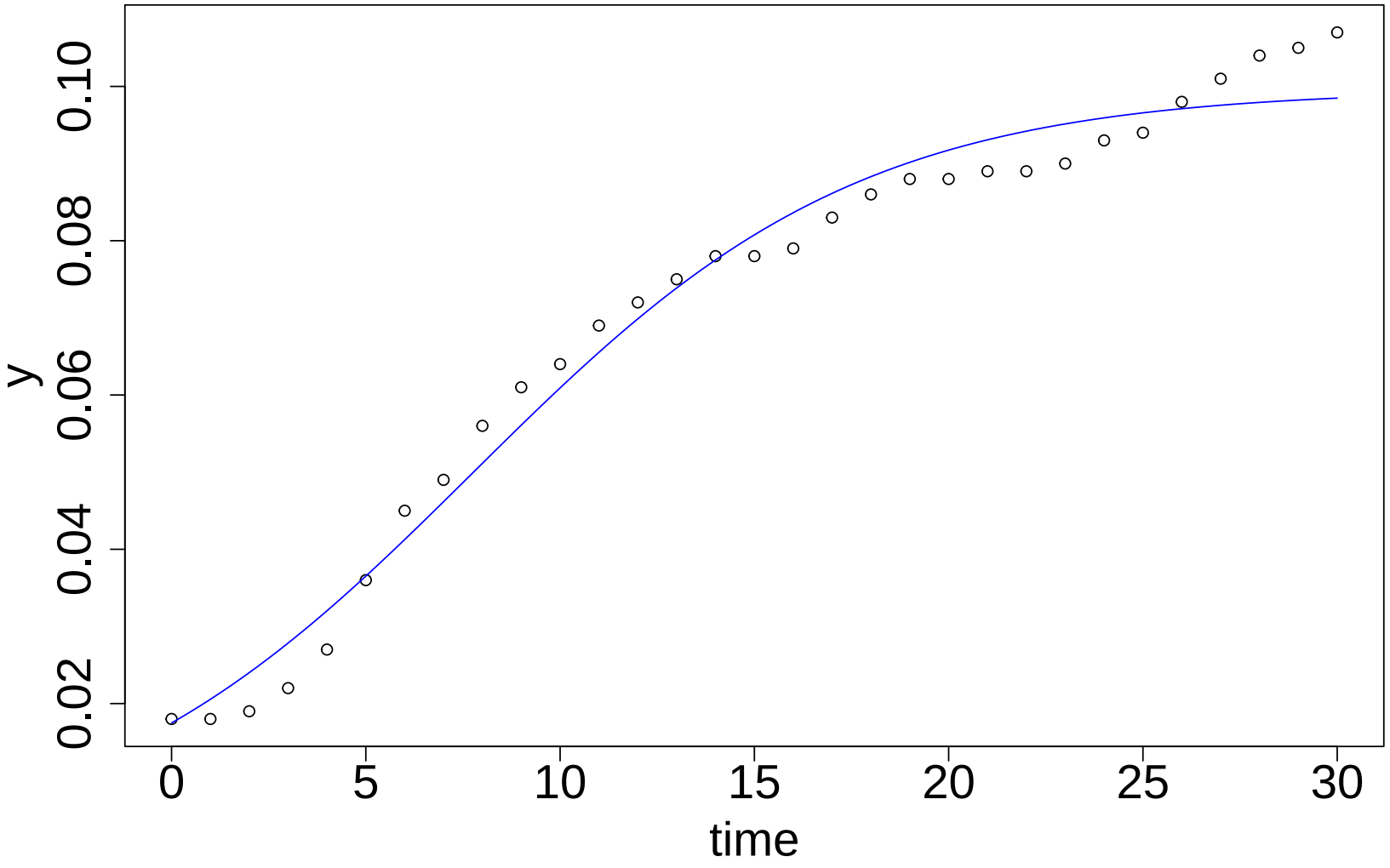
The other variables have defaults and we won't worry about them now.

Applying fit_growthmodel

```
p <- c(y0 = 0.01, mumax = 0.2, K = 0.1)

fit1 <- fit_growthmodel(FUN = grow_logistic,
                        p = p,
                        dat$time,
                        dat$value)

coef(fit1)
```

```
##         y0      mumax          K
## 0.0174826 0.2000701 0.0996260
```

Plotting results of logistic growth model



Two-step differential equation model

Cells (or individuals) can be either inactive or active. For animals, inactive could mean an individual is not of reproductive age.

Each type has its own growth equation:

$$\frac{dy_i}{dt} = -k_w \cdot y_i$$

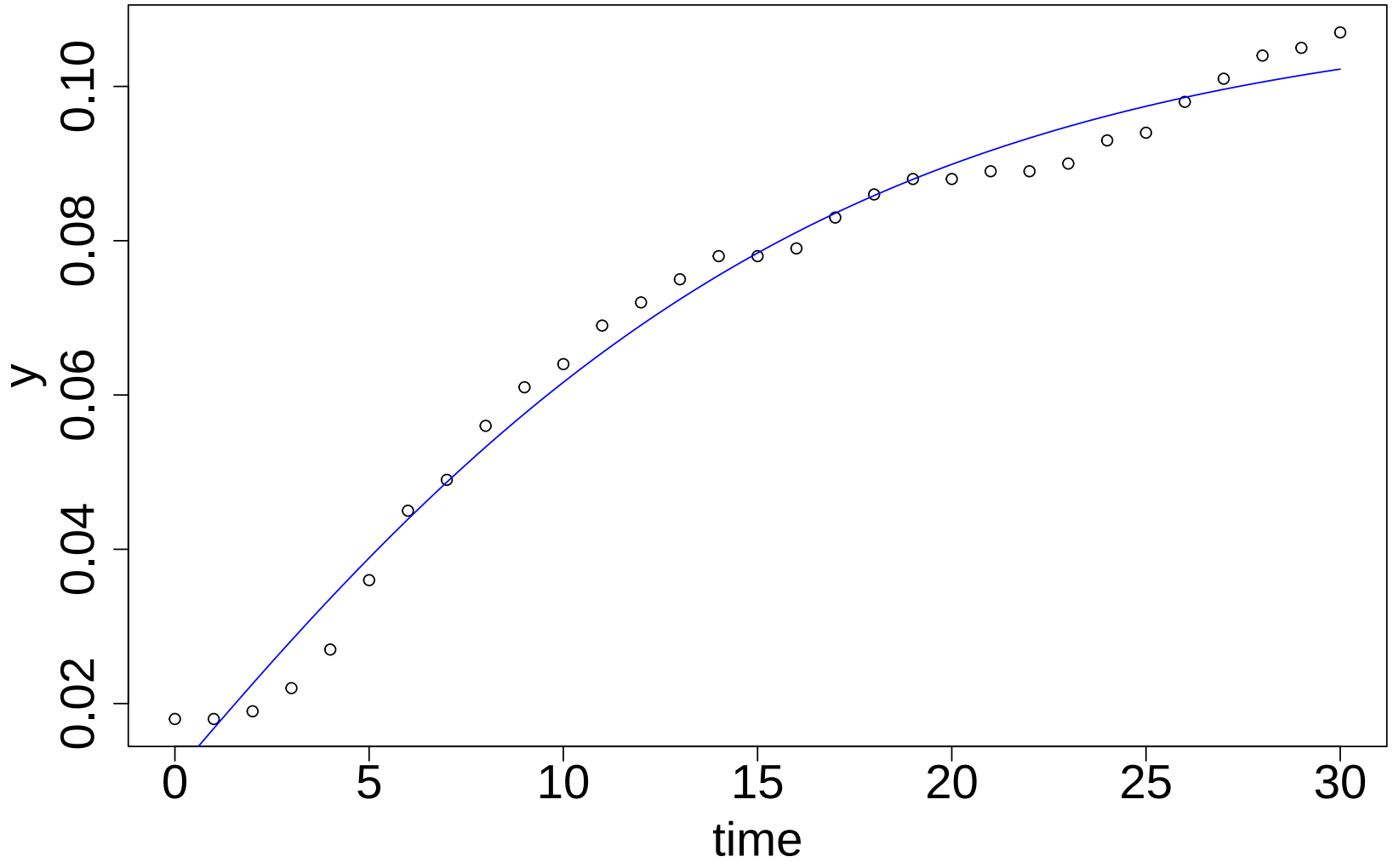
$$\frac{dy_a}{dt} = k_w \cdot y_i + \mu_{max} \cdot y_a \cdot \left(1 - \frac{y_a + y_i}{K}\right)$$

Applying the two-step differential equation model

```
p <- c(yi = 0.02, ya = 0.001, kw = 0.1, mumax = 0.2, K = 0.1)

fit2 <- fit_growthmodel(FUN = grow_twostep,
                        p = p,
                        time = dat$time,
                        y = dat$value)

coef(fit2)
```

```
##          yi          ya          kw       mumax           K
## -0.08745055  0.09830086  0.02172016  0.06748615  0.11067716
```

Plotting result of two-step model

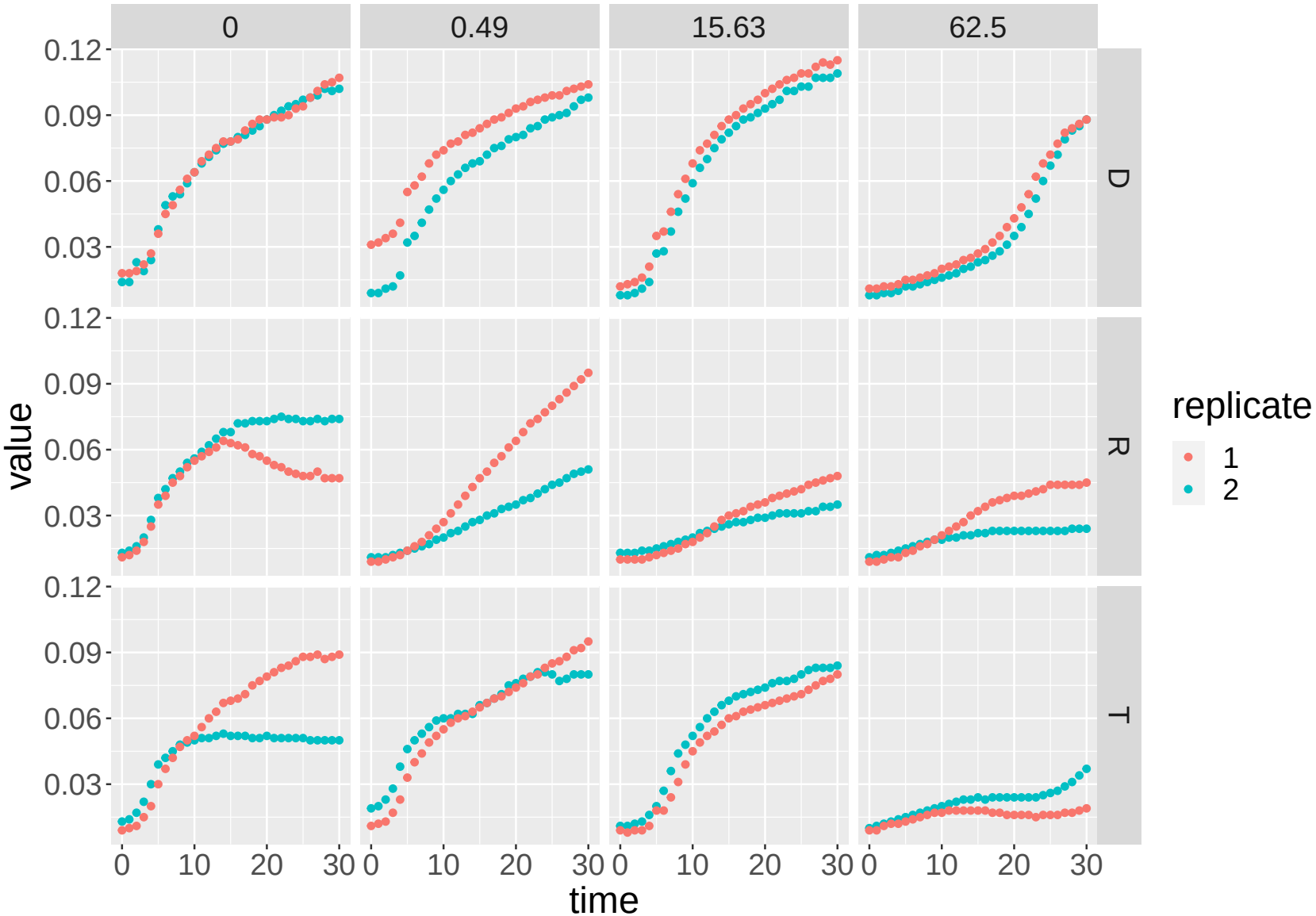


Fitting models to multiple datasets

This applies when you have more than one growth curve. In `bactgrowth` we have a total of 72 growth curves, a combination of three strains, two replicates, and 12 different concentrations.

Full `bactgrowth` dataset

Remember this plot of a subset with only four concentrations:



Analysis of multiple datasets

One question we might ask ourselves is: how does the concentration of an antibiotic affect growth rates of different strains and replicates? This is known as dose response curves. The same logic applies to, for example, environmental toxins.

We start with a spline fit. `spar` is a parameter that determines smoothness of the spline fit. The output gives us estimates for y_0 and r_{max}.

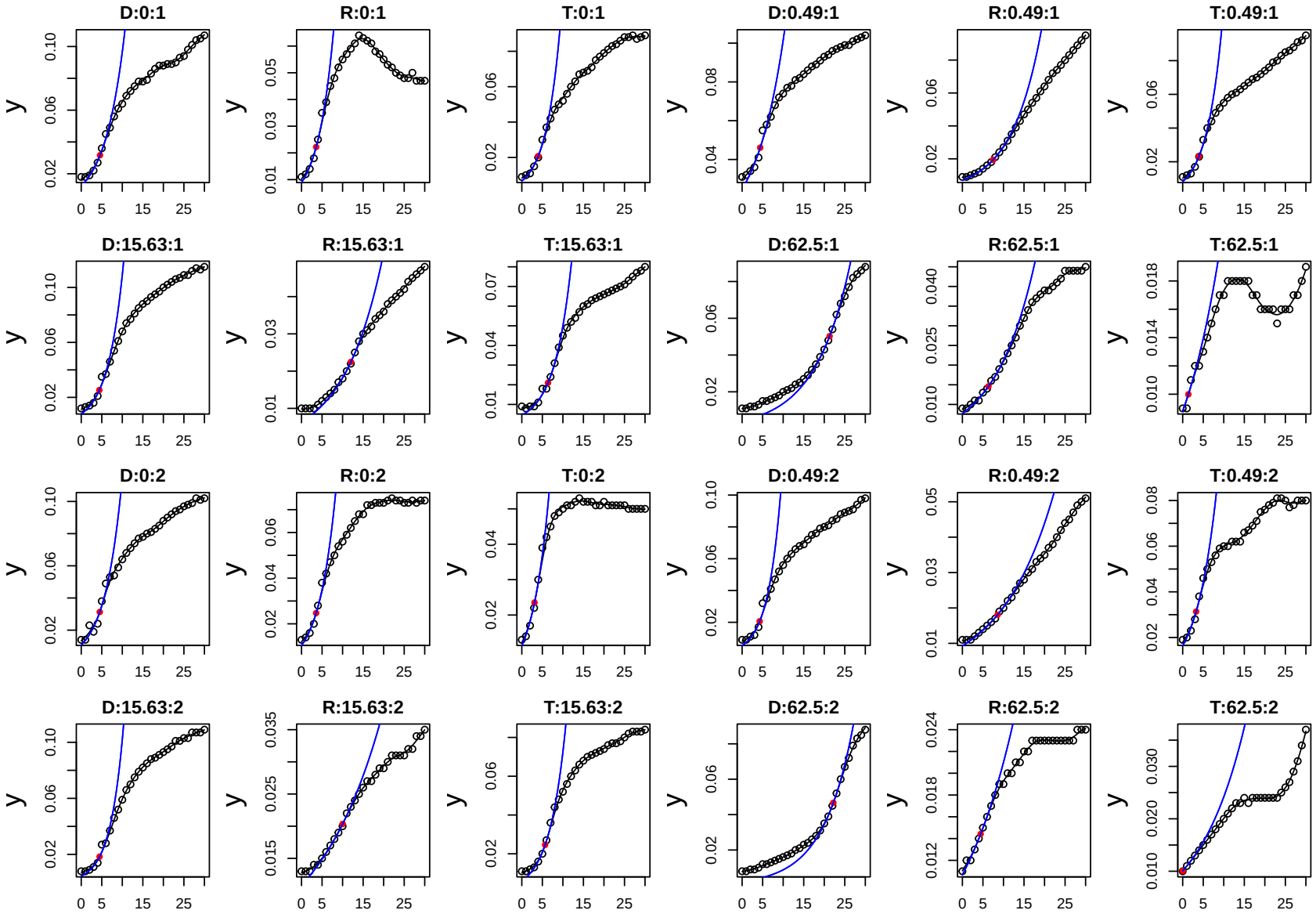
```
many_spline_fits <- all_splines(value ~ time |
                                  strain + conc + replicate,
                                data = bactgrowth, spar = 0.5)
```

Look at coefficients

```
head(coef(many_spline_fits), 5)
```

```
##                      y0      mumax
## D:0:1     0.012349620 0.20570733
## R:0:1     0.008911212 0.25621722
## T:0:1     0.006727476 0.28233842
## D:0.24:1 0.016084542 0.17270804
## R:0.24:1 0.013925666 0.05419363
```

Plotting results for multiple splines



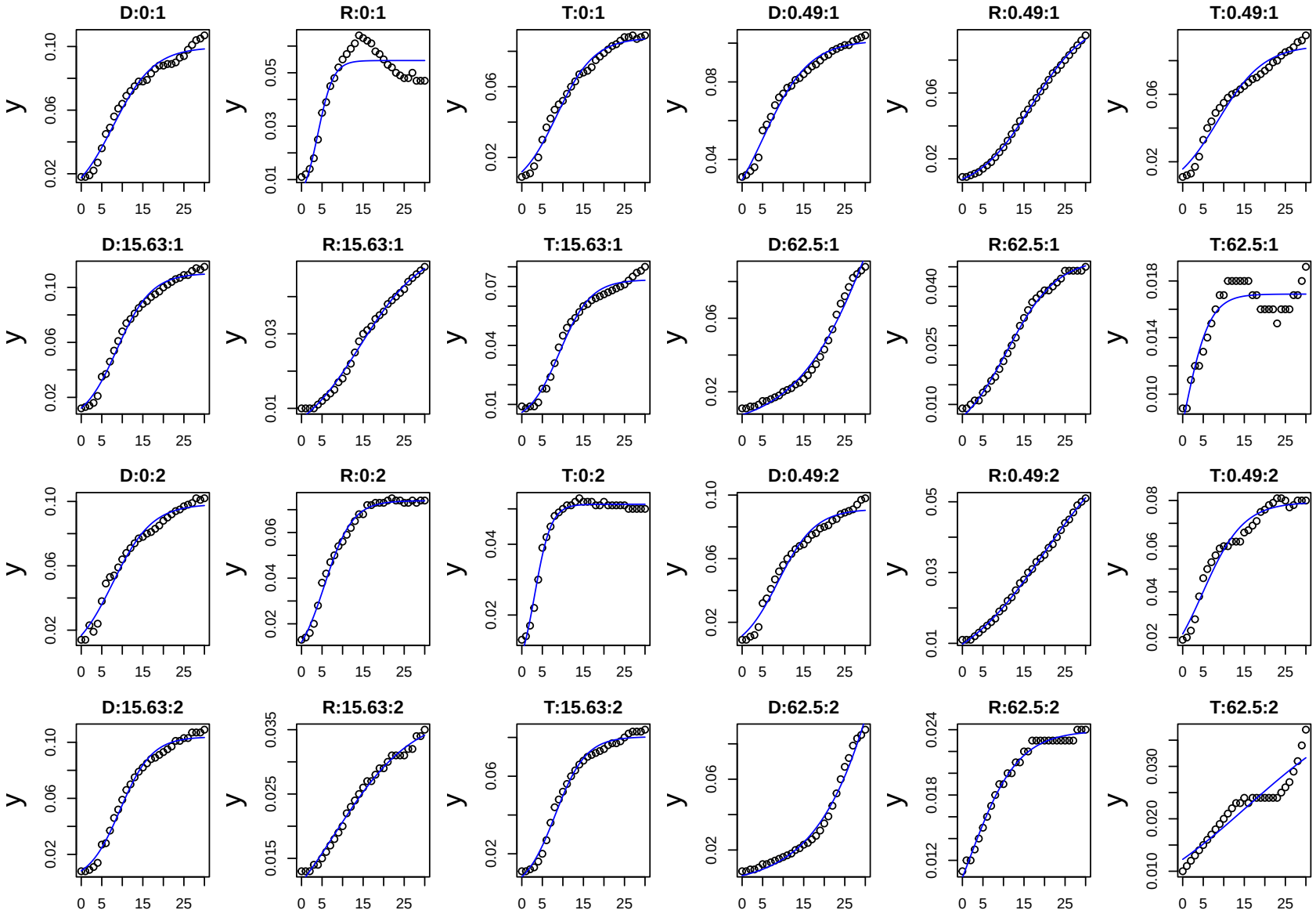
Logistic growth models for multiple datasets

```
all_growthmodels(
formula,
data,
p,
lower = -Inf,
upper = Inf,
which = names(p),
FUN = NULL,
method = "Marq",
transform = c("none", "log"),
...,
subset = NULL,
ncores = detectCores(logical = FALSE)
)
```

Applying all_growthmodels with logistic growth

```
p   <- c(y0 = 0.03, mumax = .1, K = 0.1)

many_logistic <- all_growthmodels(
                    value ~ grow_logistic(time, parms) |
                      strain + conc + replicate,
                    data = bactgrowth,
                    p = p)
```

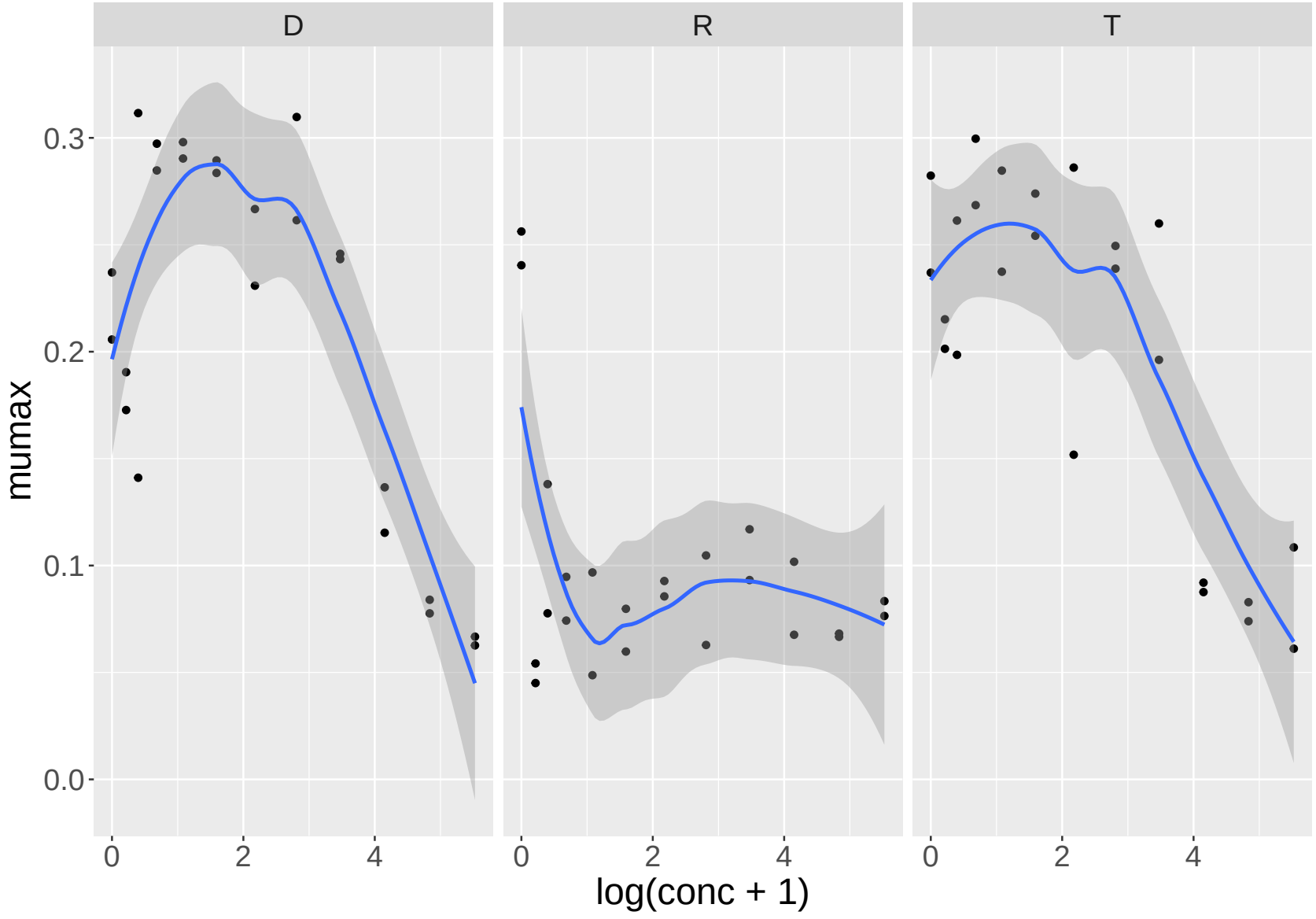
Plotting results



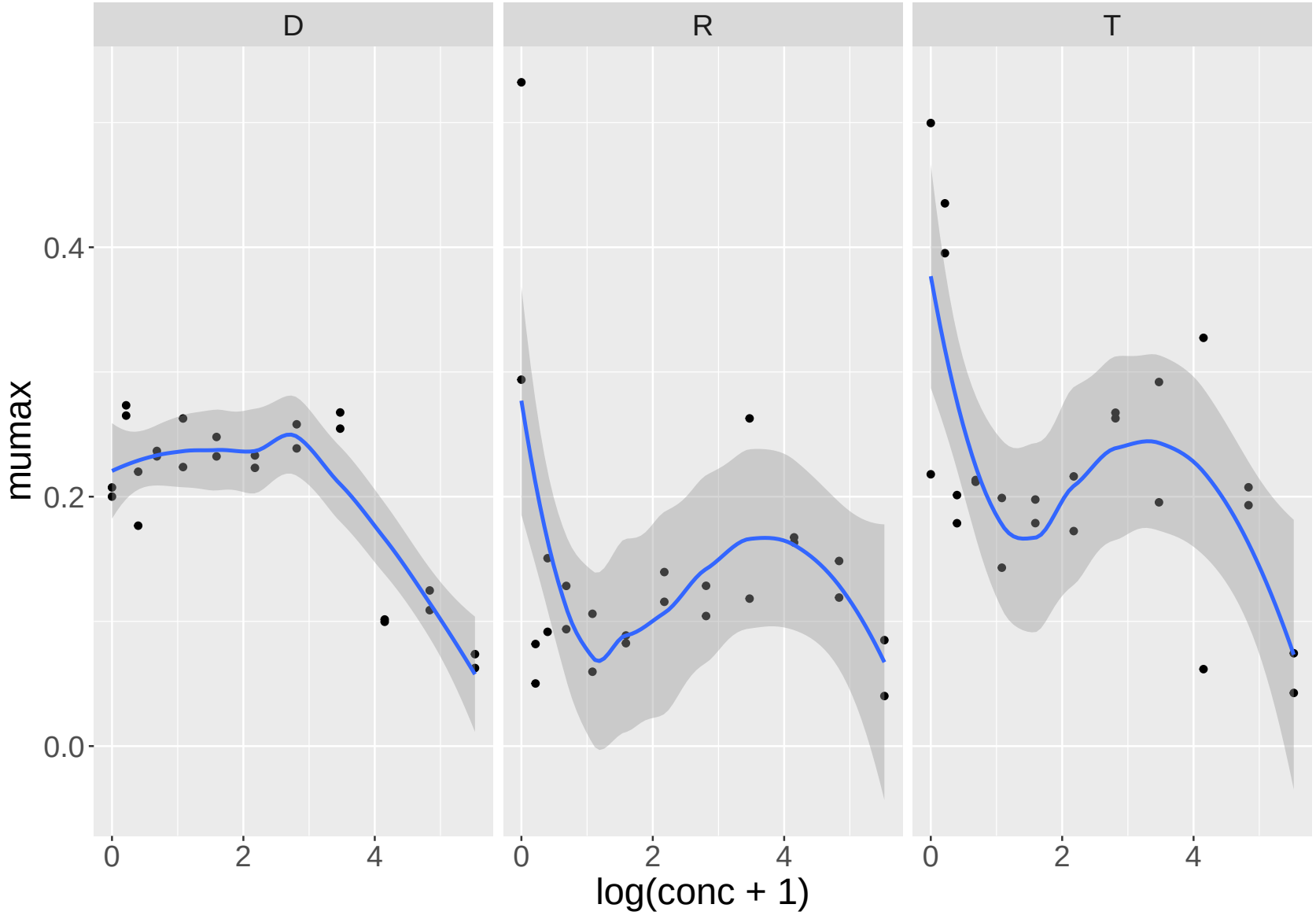
Using results to estimate dose response curves

```
many_spline_res   <- results(many_spline_fits)
many_logistic_res <- results(many_logistic)
```

Plotting dose response curves for splines



Plotting dose response curves for logistic



Unique insight

You can write your own functions!

Example for a shift model (won't go into details of the model)

$$y(t) = \frac{K \cdot y_0}{y_0 + (K - y_0) \cdot e^{-\mu_{max} t}} + y_{shift}$$

Code

```r
# Define model
grow_logistic_yshift <- function(time, parms) {
  with(as.list(parms), {
    y <- (K * y0) / (y0 + (K - y0) * exp(-mumax * time)) + y_shift
    as.matrix(data.frame(time = time, y = y))
  })
}

# Convert to class `growthmodel`
grow_logistic_yshift <- growthmodel(grow_logistic_yshift,
                                    c("y0", "mumax", "K", "y_shift"))

# Now use `growthrates` function to fit model
fit <- fit_growthmodel(grow_logistic_yshift,
                       p = c(y0 = 1, mumax = 0.1, K = 10, K = 10,
                             y_shift = 1),
                       time = x, y = y)
```

Advantages of `growthrates` package

- Adapted to problems of biological growth
- A lot simpler to use than the packages that contain the fitting routines
- Very flexible – can fit several different models; each function can be adapted
- Seems relatively fast
- Allows for user-defined functions
- Has built-in plotting function but also works well with ggplot
- Well documented

Limitations of the package

- Would be even easier if some of the user-defined functions (like the shift model) were integrated into the package
- Overall the package does what it sets out to do, and I have no major complaints

Sources

https://tpetzoldt.github.io/growthrates/doc/Introduction.html
https://cran.r-project.org/web/packages/growthrates/growthrates.pdf
https://github.com/tpetzoldt/growthrates

My code:

https://github.com/linusblomqvist/ESM_211/tree/main/growthrates

Portfolio assignment

Find r and K for the bison dataset using at least three different models (e.g. exponential, logistic, spline).

Hint: try functions `fit_easylinear`, `fit_growthmodels`, and `fit_splines`.

Do the different models agree on the parameter values?