Mixed Models with R: Non-Linear Models

Asymptotic Functions of Time

Georges Monette

random@yorku.ca

Longitudinal models for IQ recovery after coma

Data: IQ tests on 200 post coma patients at QEH

Number of observations per patient:

Number of observations	1	2	3	4	5	Total
Number of patients	107	61	27	4	1	200

Method of analysis: Mixed models for longitudinal da

Problem: Representing IQ recovery over time

Some functions of time:

- linear
- quadratic
- higher polynomials
- splines
- exponential growth, decay
- exponential asymptotic growth
- periodic functions

Polynomials:

Linear function:

$$E(IQ) = \beta_0 + \beta_1 T$$

Quadratic:

$$E(IQ) = \beta_0 + \beta_1 T + \beta_2 T^2$$

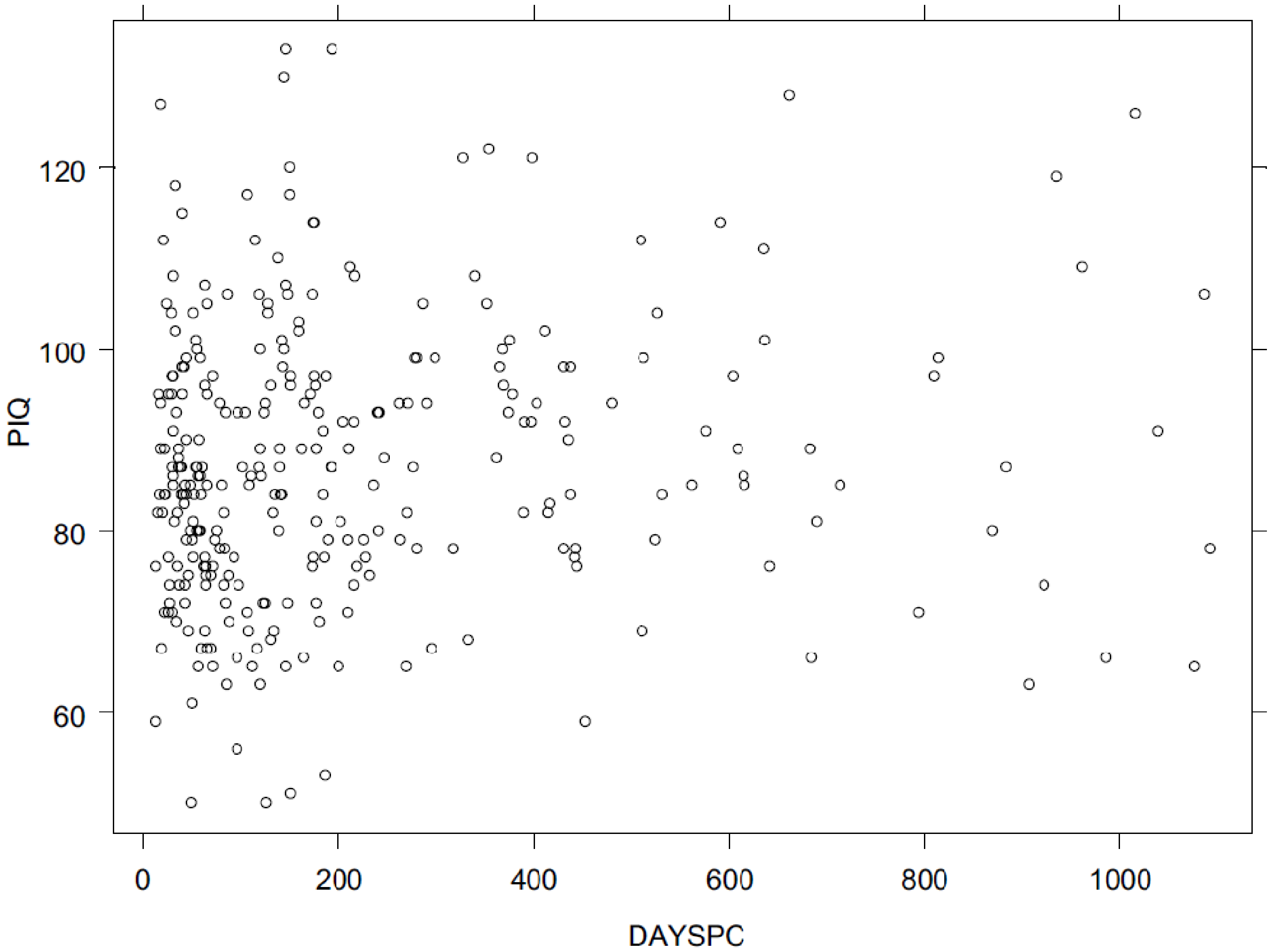
Cubic

$$E(IQ) = \beta_0 + \beta_1 T + \beta_2 T^2 + \beta_3 T^3$$

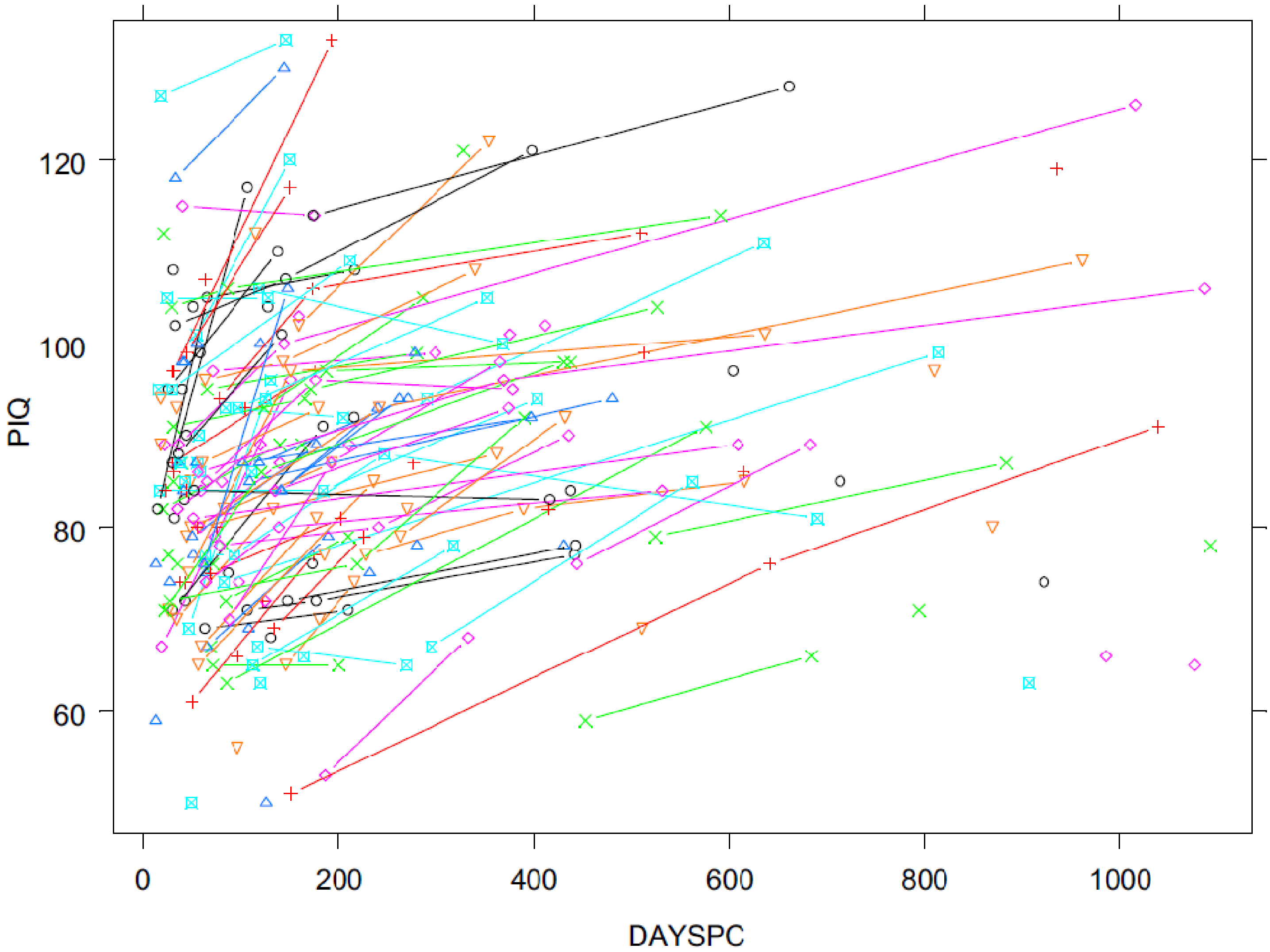
Quartic:

$$E(IQ) = \beta_0 + \beta_1 T + \beta_2 T^2 + \beta_3 T^3 + \beta_4 T^4$$

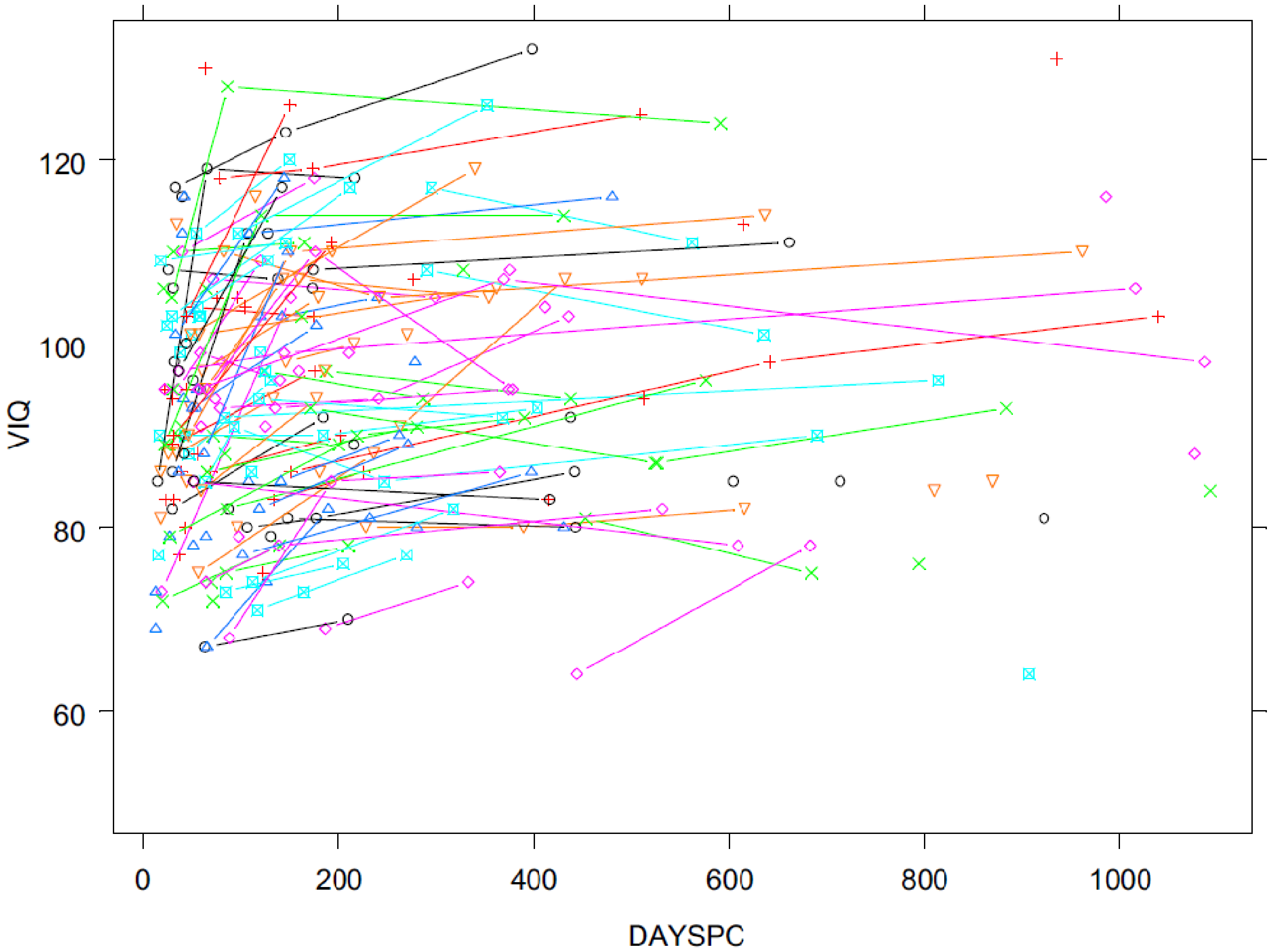
Note the change in meaning of parameters as we mo
Last parameter has global interpretation, previous pa
ters at time = 0



PIQ
DAYSPC
120
100
80
60
0
200
400
600
800
1000



PIQ
DAYSPC
0
200
400
600
800
1000
60
80
100
120



VIQ
DAYSPC
0
200
400
600
800
1000
60
80
100
120

Modeling individual trajectories

A good strategy in longitudinal data analysis is to start l
plausible model for individual trajectories even if there
data from any one individual to actually fit the model.
unbalanced and you are willing to assume that the betw
effect is close to the within-subject effect, then the estin
individual trajectories 'borrows strength' from the betwe
model.

Within the limits imposed by sample size, we try to con
model that:

1. captures the main theoretical properties of the phenon
2. preferably has interpretable parameters

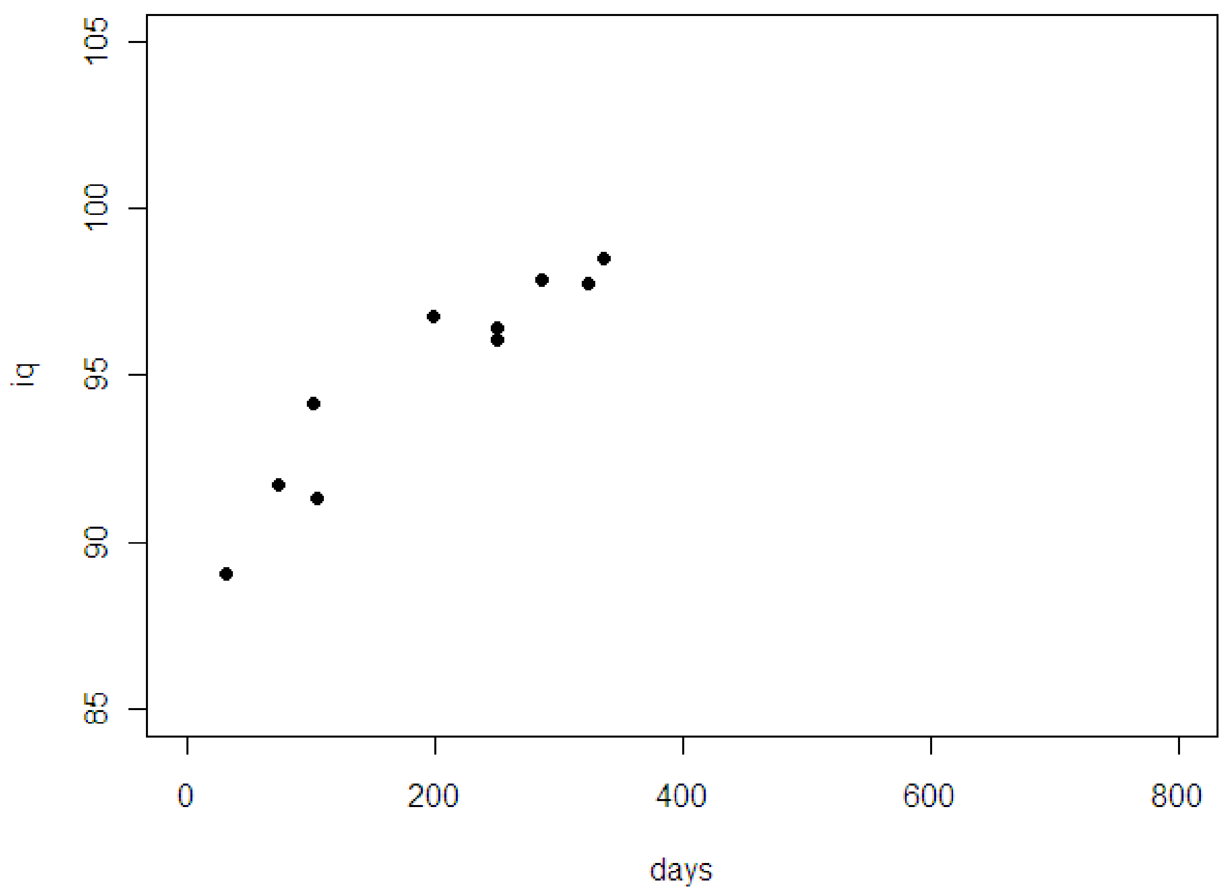
To experiment with your model don't hesitate to simula
some plausible data with `locator` and play with mode
Make an emply plotting surface, click to create some xy
the columns the names you want:

```
plot(0,0, xlim = c(0,800),
      ylim = c(85,105), type = 'n')
iqsim.ex <- locator( 10 , type = 'p')
iqsim.ex
iqsim.ex <- as.data.frame( iqsim.ex )
iqsim.ex
names( iqsim.ex ) <- c('days','iq')
iqsim.ex <- iqsim.ex[
           order(iqsim.ex$days),]
```

However, we'll use precooked data:

```
> data( iqsim )     # from spida
> iqsim
       days       iq
1   30.9375 89.07734
2   73.1250 91.74573
4  101.2500 94.12407
3  104.3750 91.28166
5  198.1250 96.73445
6  249.6875 96.03835
7  249.6875 96.44441
8  285.6250 97.89462
9  323.1250 97.72059
10 335.6250 98.47470
```

```
plot( iq ~ days ,
      iqsim, pch = 16, xlim = c(0,800),
      ylim = c(85,105) )
```



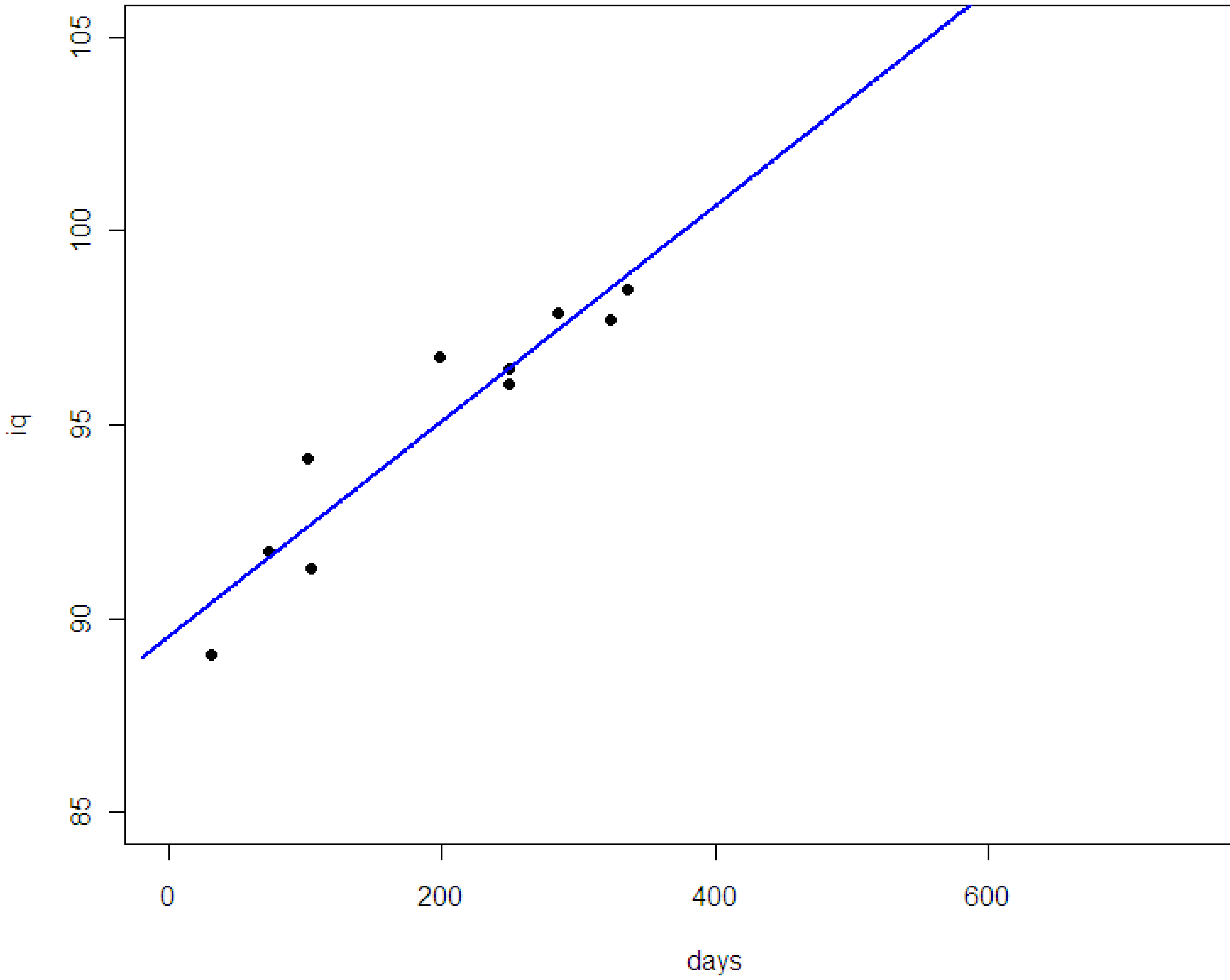
Fitting a line:

```
> fit.lin <- lm ( iq ~ days, iqsim )
> summary( fit.lin )
. . . . .

Coefficients:
             Estimate Std. Error t value Pr(>|t|
(Intercept) 89.540151   0.759021  117.97 2.98e-1
days         0.027739   0.003429    8.09 4.03e-0


Residual standard error: 1.133 on 8 degrees of f
Multiple R-squared: 0.8911,     Adjusted R-squar
F-statistic: 65.45 on 1 and 8 DF,  p-value: 4.02
```

Graphing a fitted line

We would like to show the predicted value over the wh
days in the graph, not just the values that were observed
straight line we could just use **abline**. With curved li
need a different approach. So we create a prediction dat
the one predictor variable.

```
> pred <- expand.grid( days = seq( -20, 850,
> pred$iq.lin <- predict( fit.lin, pred )
> some( pred )
    days    iq.lin
94    73  91.56510
137  116  92.75788
203  182  94.58865
.   .   .   .   .
746  725 109.65094
753  732 109.84511
> lines( iq.lin ~ days, pred,
             col = 'blue', lwd = 2)
```

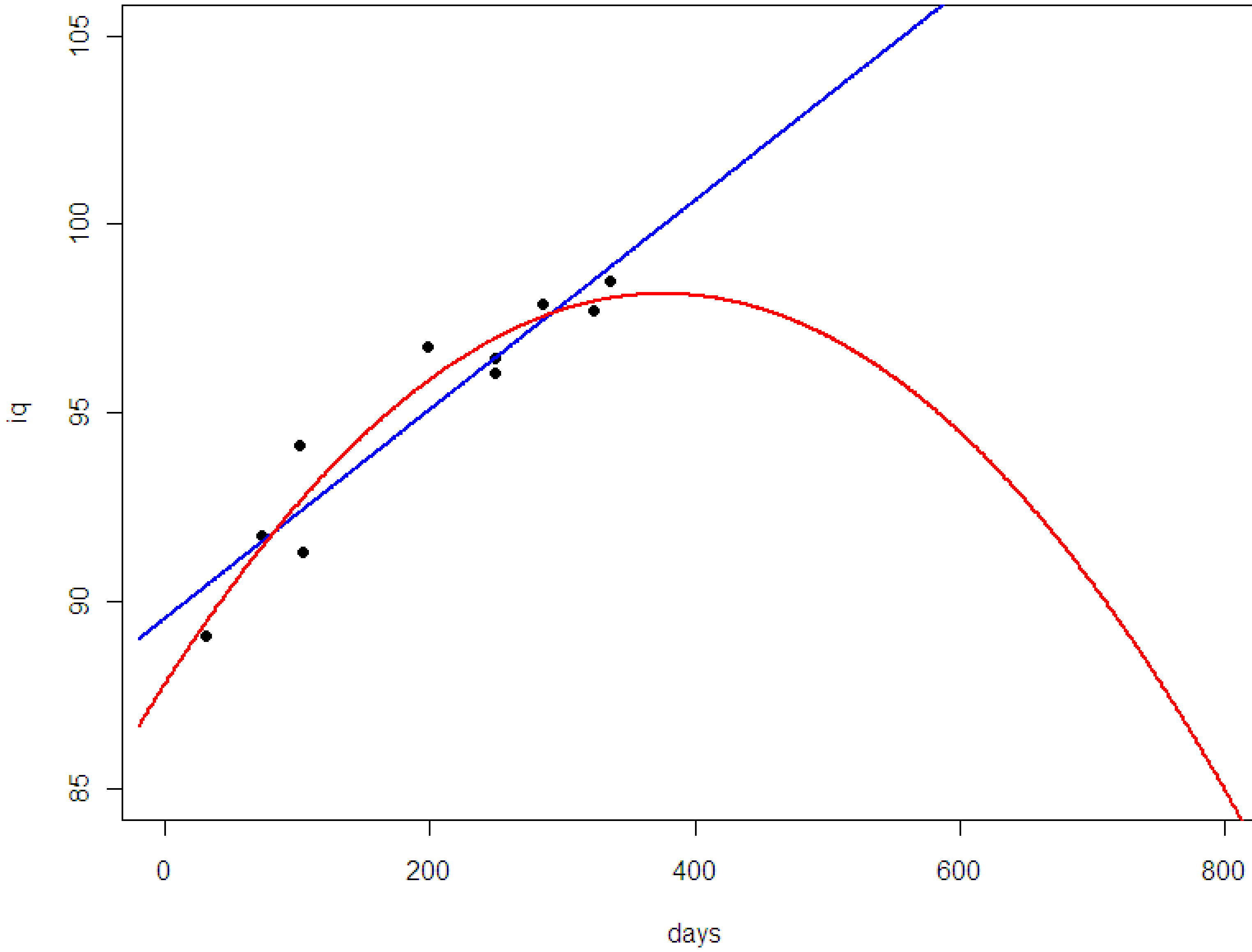


105
100
95
90
85
iq
0
200
400
600
days

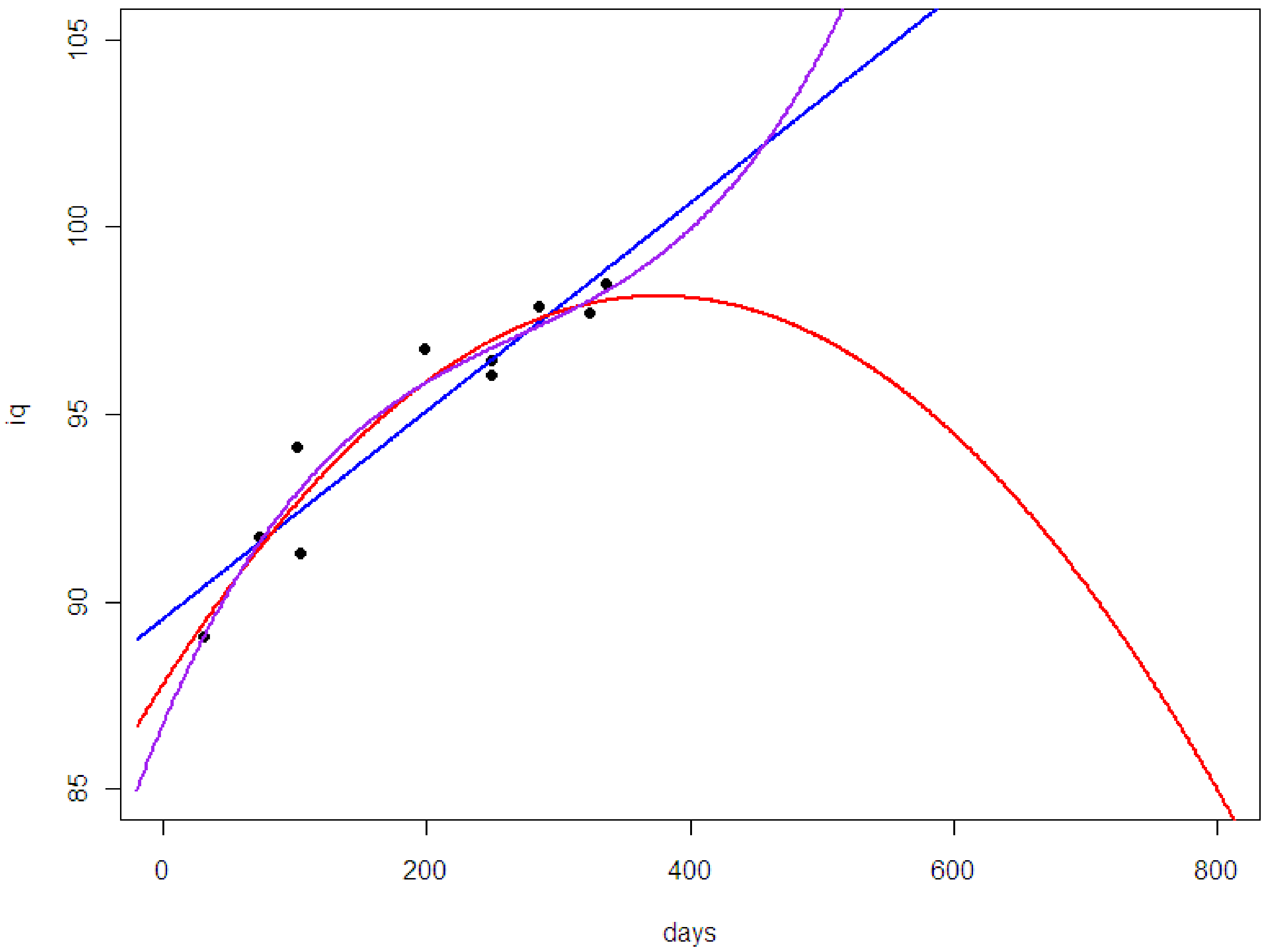
Doesn't make much sense!

Let's try a quadratic

```
>    fit.quad <- lm( iq ~ days + I(days ^2),
>    summary( fit.quad )
. . . .
Coefficients:
              Estimate Std. Error t value Pr
(Intercept)  8.780e+01  1.170e+00  75.020 1.
days         5.506e-02  1.535e-02   3.586
I(days^2)   -7.324e-05  4.036e-05  -1.815

Residual standard error: 0.9988 on 7 degrees of
Multiple R-squared: 0.9259,     Adjusted R-squar
F-statistic: 43.75 on 2 and 7 DF,  p-value: 0.00

>    pred$iq.quad <- predict( fit.quad, pred )
>    lines( iq.quad ~ days , pred, col = 'red',
>
```


iq
105
100
95
90
85
0
200
400
600
800
days

With a quadratic, what goes up must come down … the
went up! Maybe a cubic makes more sense:

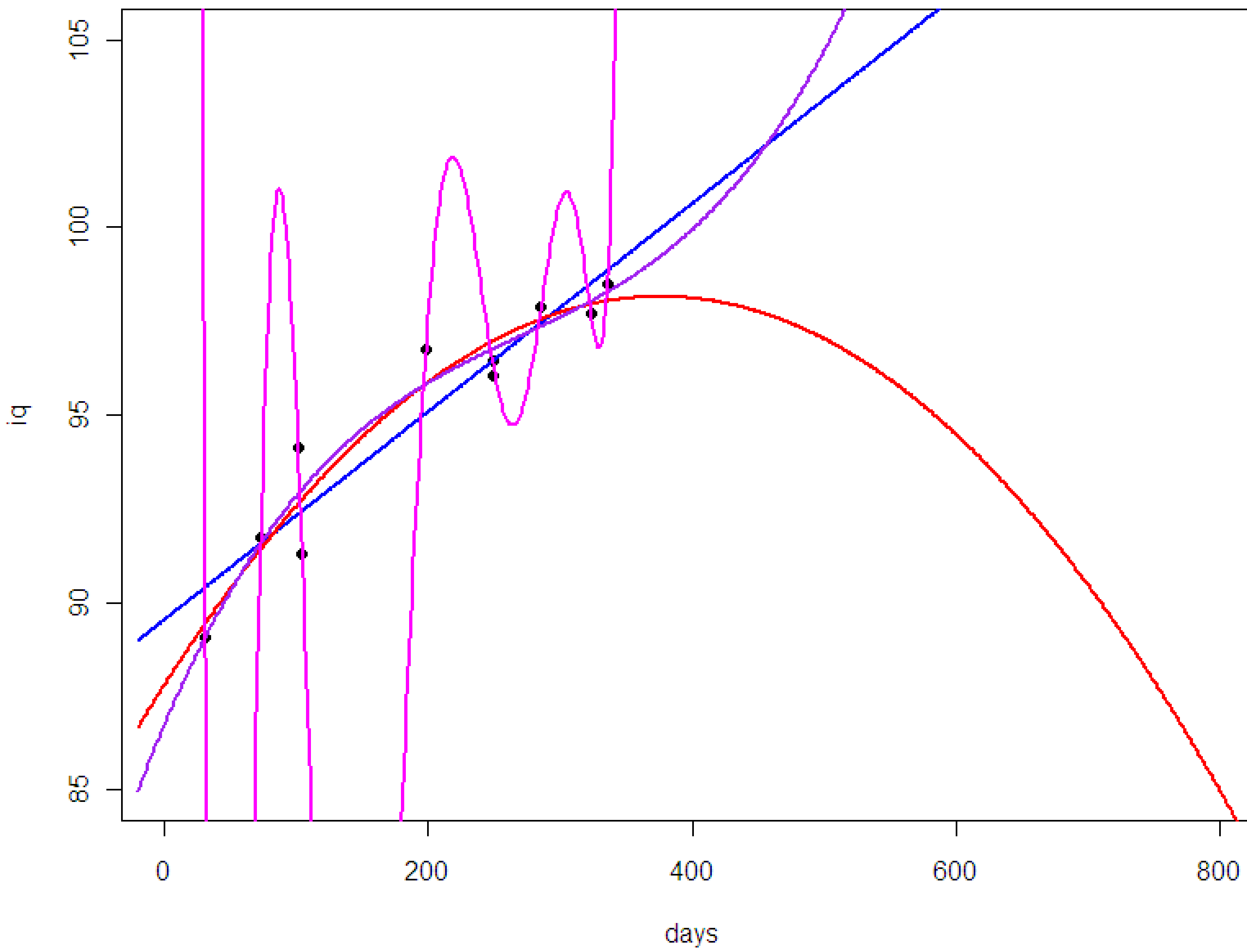


Exasperated we decided to go all the way with a polyno
degree 8:

Thanks to John, this option

```
>  p8 <- function( x ) poly( x, 8, raw = TRUE)
>  fit.high <- lm( iq ~ p8( days ), iqsim )
>  summary(fit.high)   # look at R-Squared!!
.   .   .   .
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.314e+03  1.665e+02   7.889   0.0803 .
p8(days)1   -9.135e+01  1.238e+01  -7.381   0.0857 .
p8(days)2    2.548e+00  3.436e-01   7.414   0.0854 .
p8(days)3   -3.596e-02  4.829e-03  -7.447   0.0850 .
p8(days)4    2.878e-04  3.846e-05   7.482   0.0846 .
p8(days)5   -1.362e-06  1.811e-07  -7.518   0.0842 .
p8(days)6    3.777e-09  4.999e-10   7.555   0.0838 .
p8(days)7   -5.674e-12  7.476e-13  -7.590   0.0834 .
p8(days)8    3.567e-15  4.679e-16   7.624   0.0830 .
---
Residual standard error: 0.2871 on 1 degrees of freedom
Multiple R-squared: 0.9991,     Adjusted R-squared: 0.9921
F-statistic: 142.8 on 8 and 1 DF,  p-value: 0.06463
```

An almost perfect fit!



19

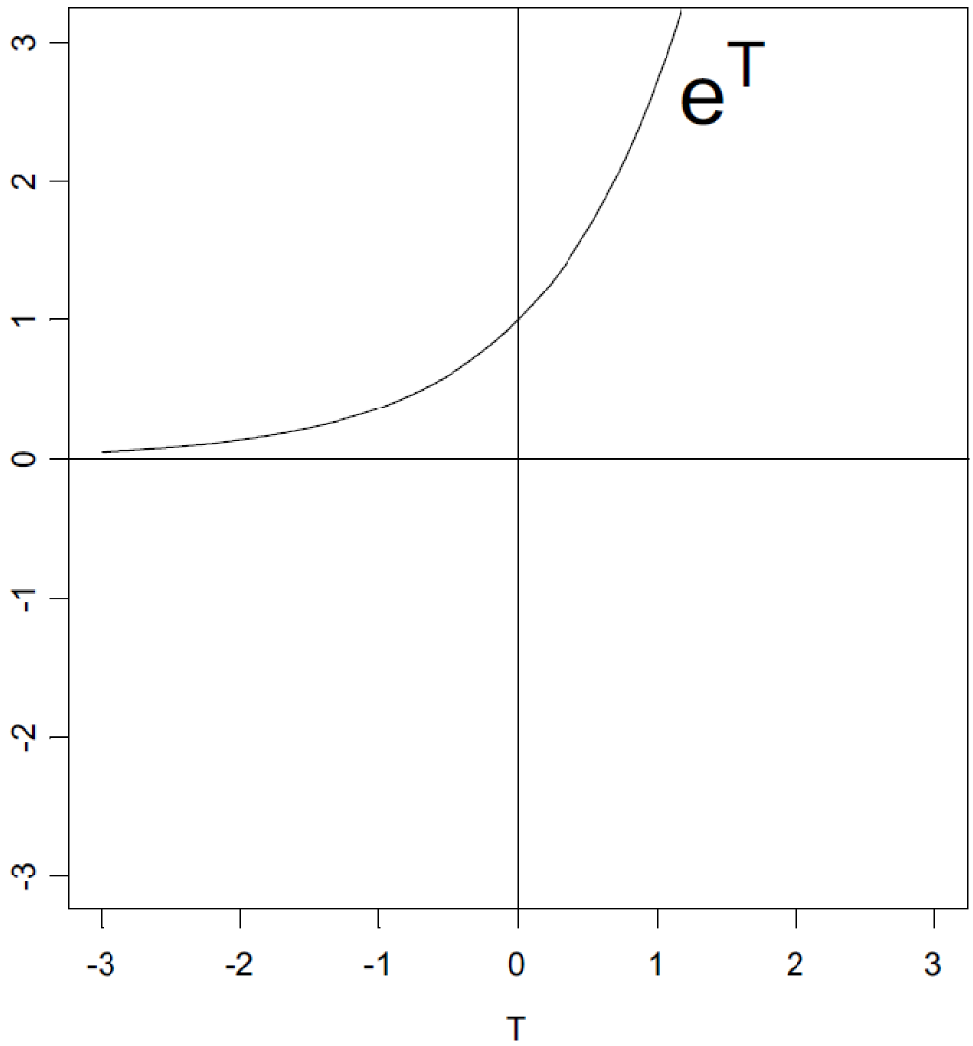
- A perfect fit to the data
- But a very poor fit to the 'population'
- An example of overfitting and loss of validity

The remedy:

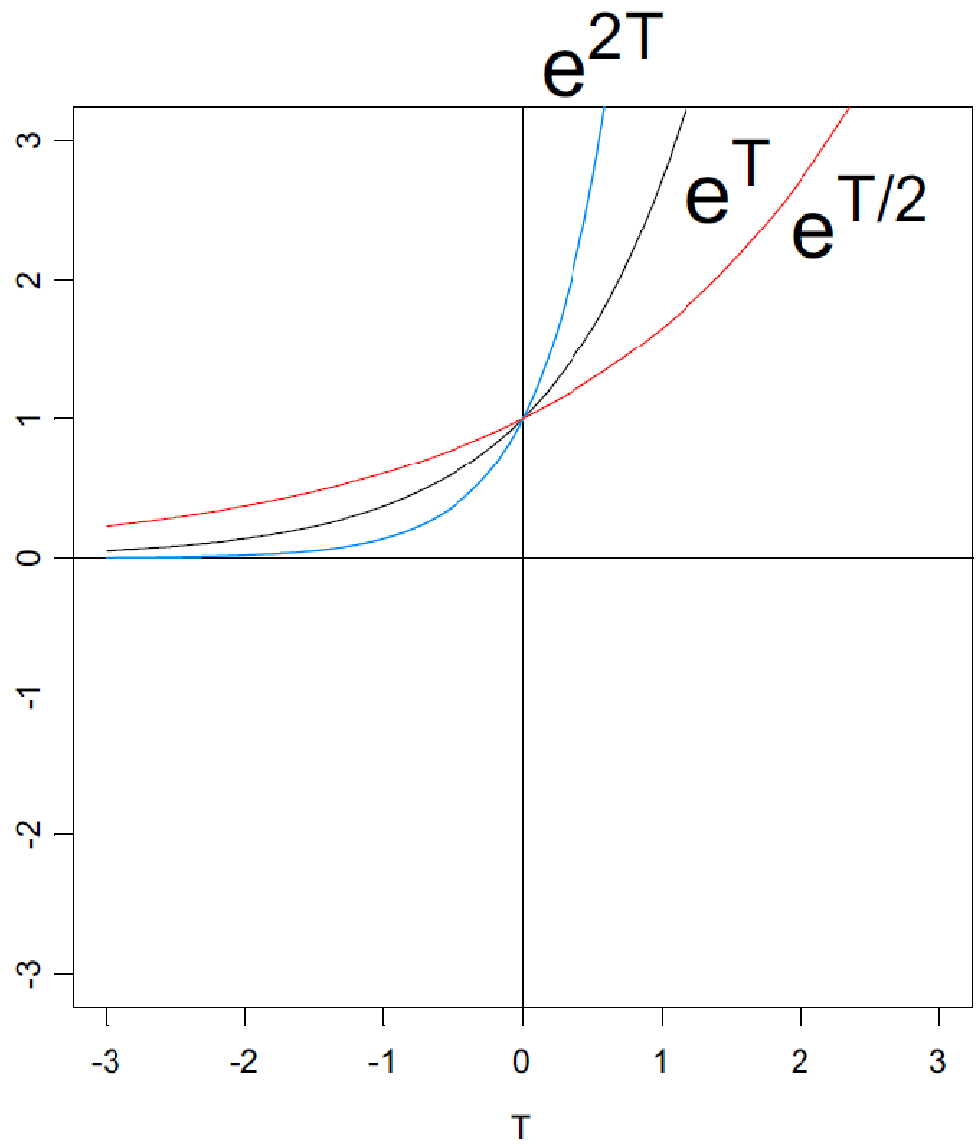
Use a model that captures characteristics of the process
Don't just use a high order polynomial to get a good em

Presumably, under typical circumstances, recovery reac
after a while. We need a model that rises at first and th
out.

Exponential growth or decay

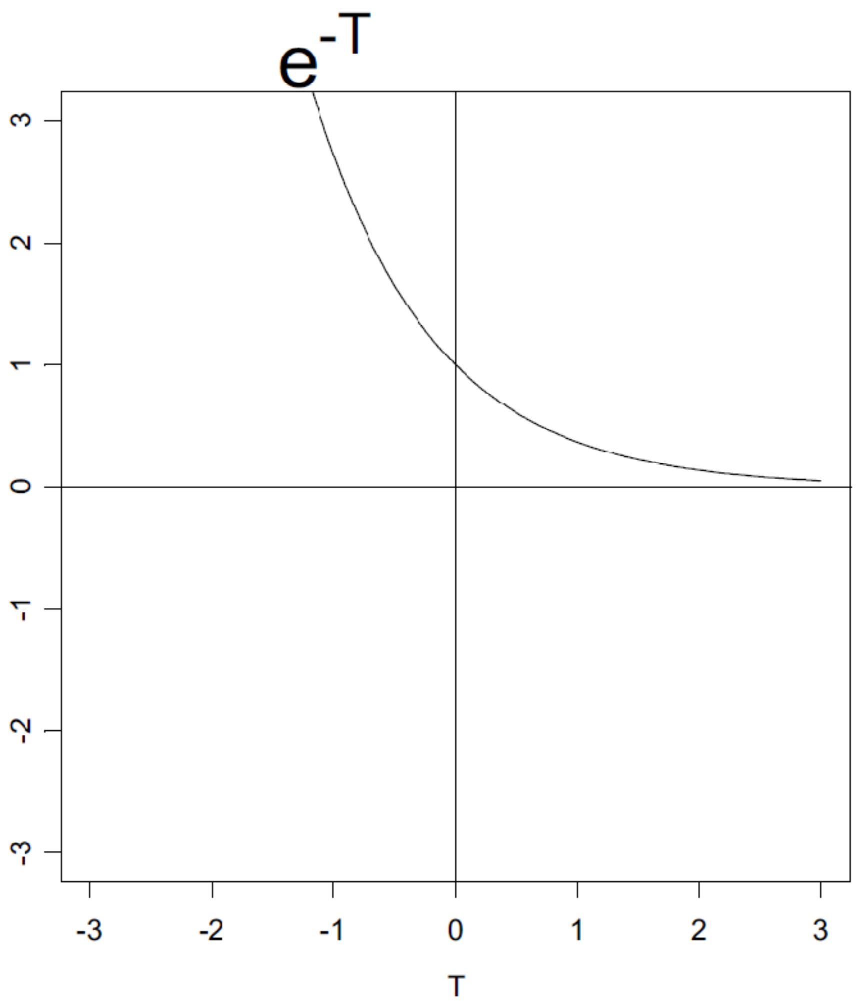


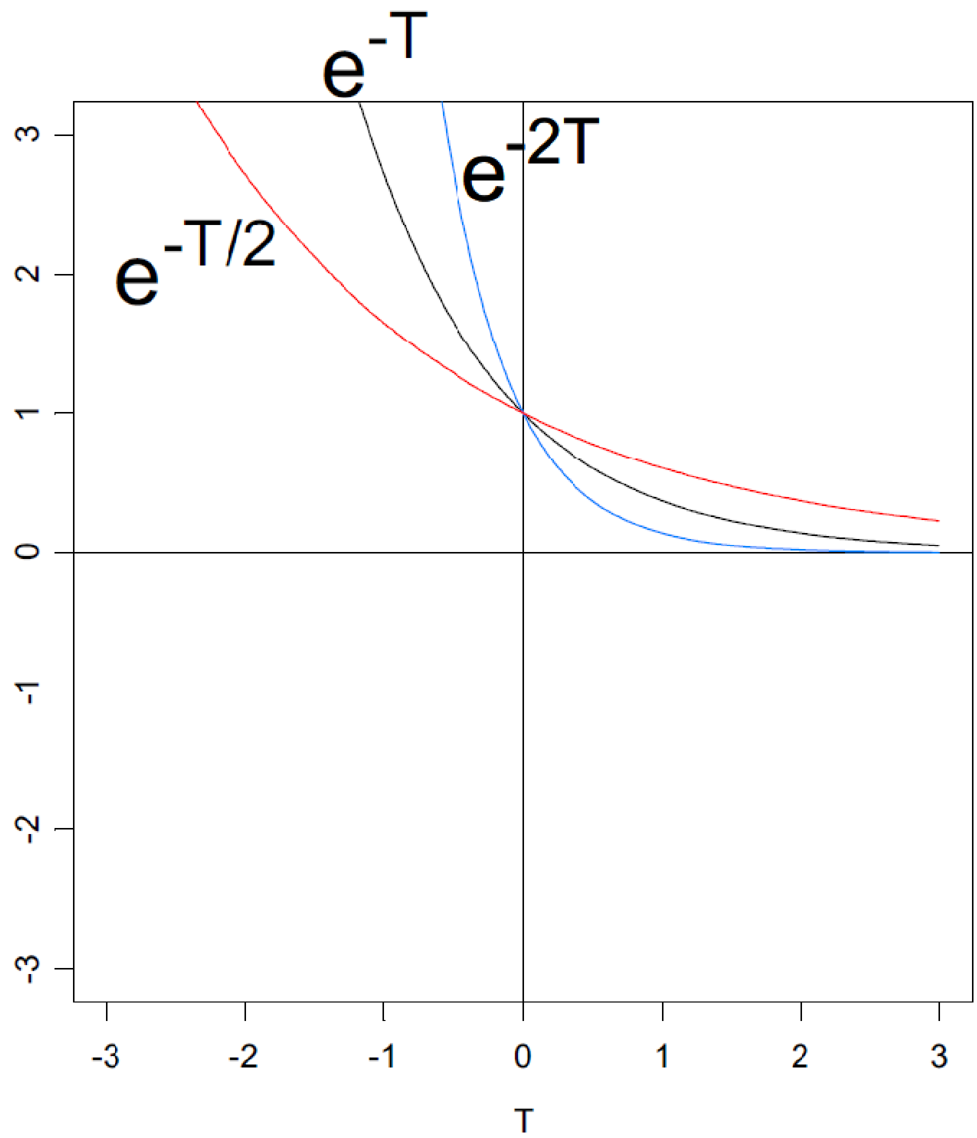
21



e^{2T}
e^T
e^{T/2}
3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3
T

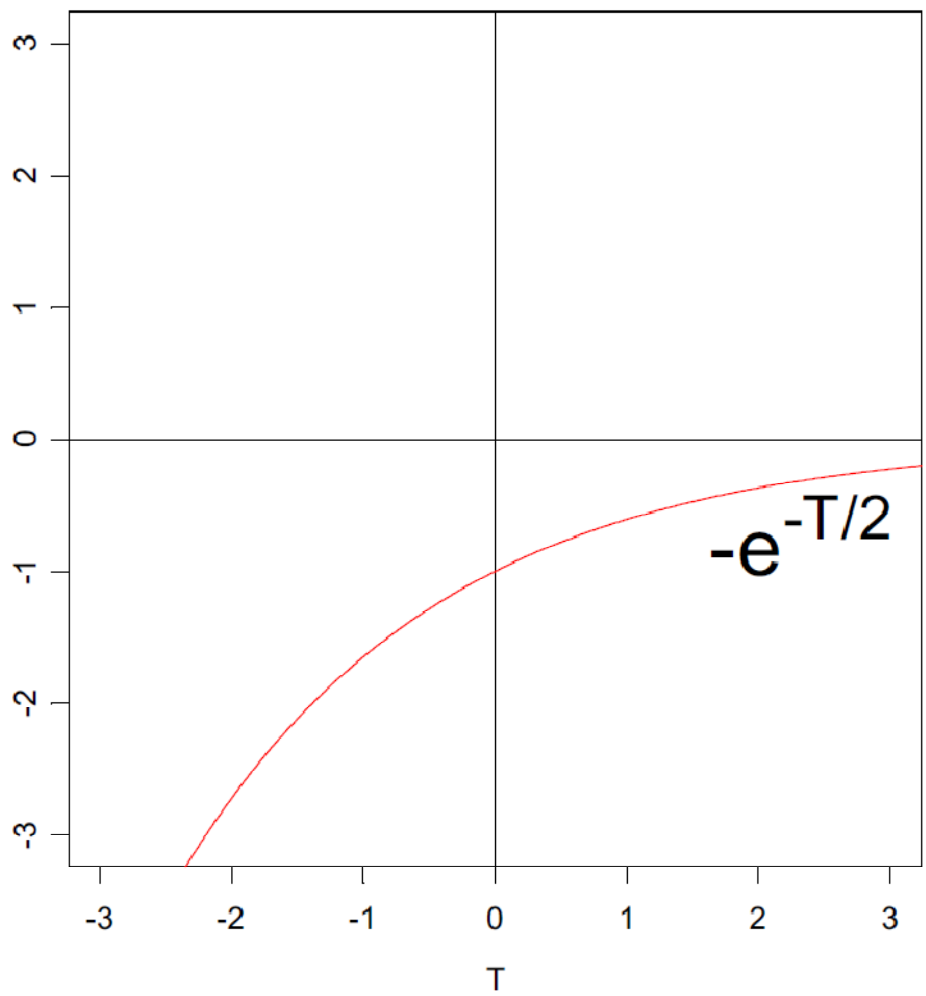
Exponential decay

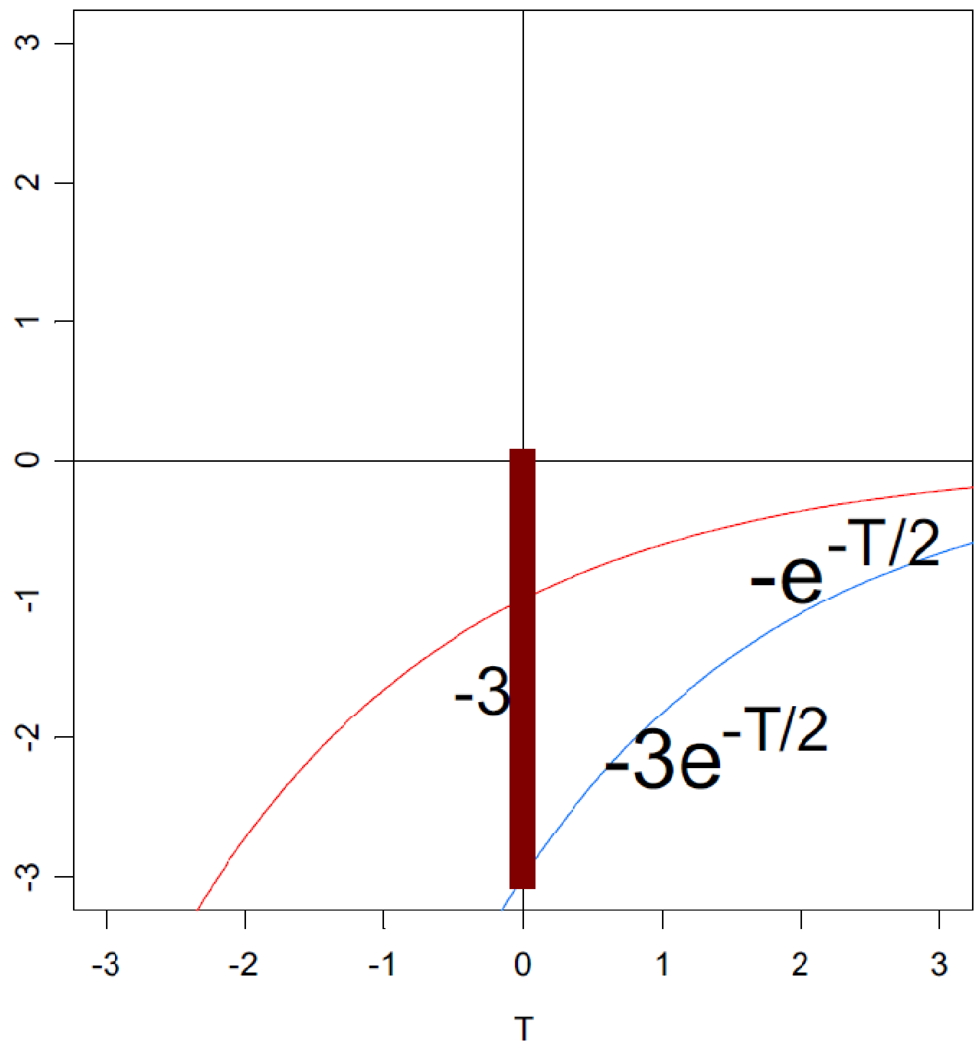


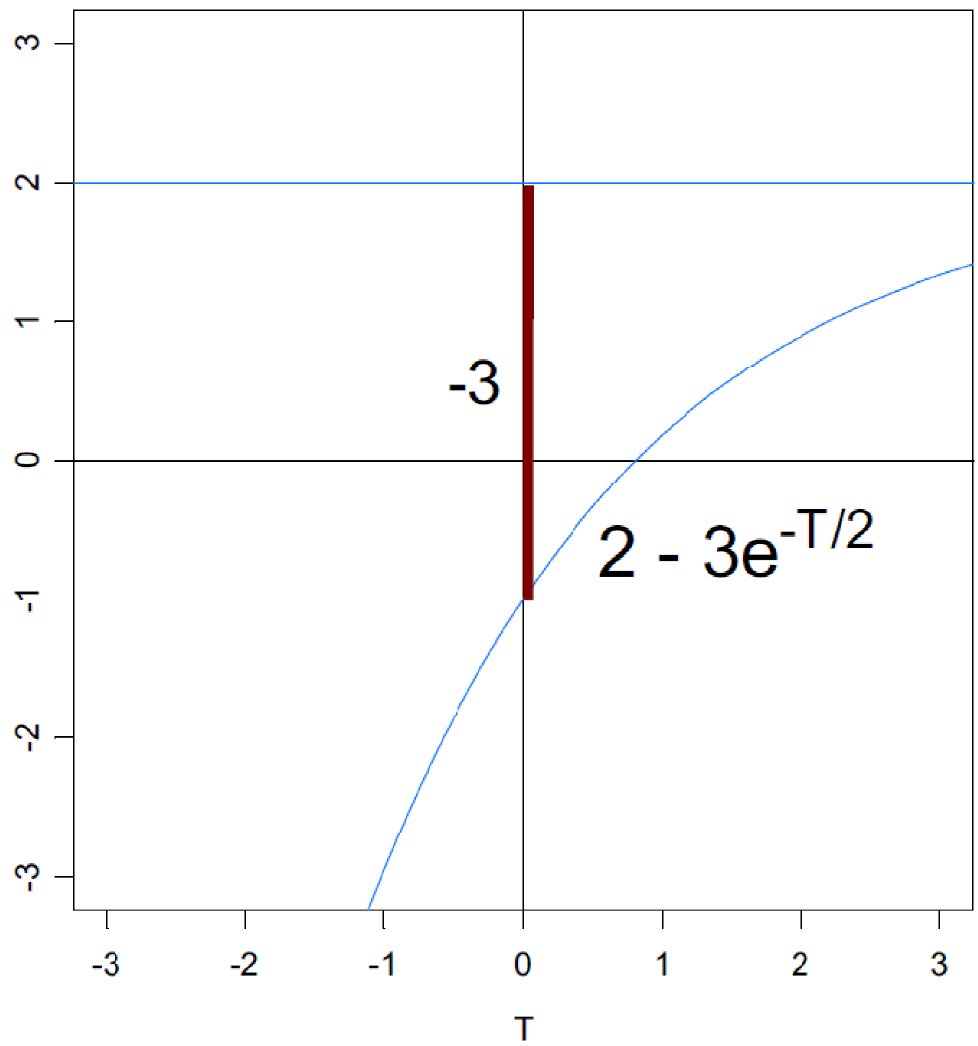


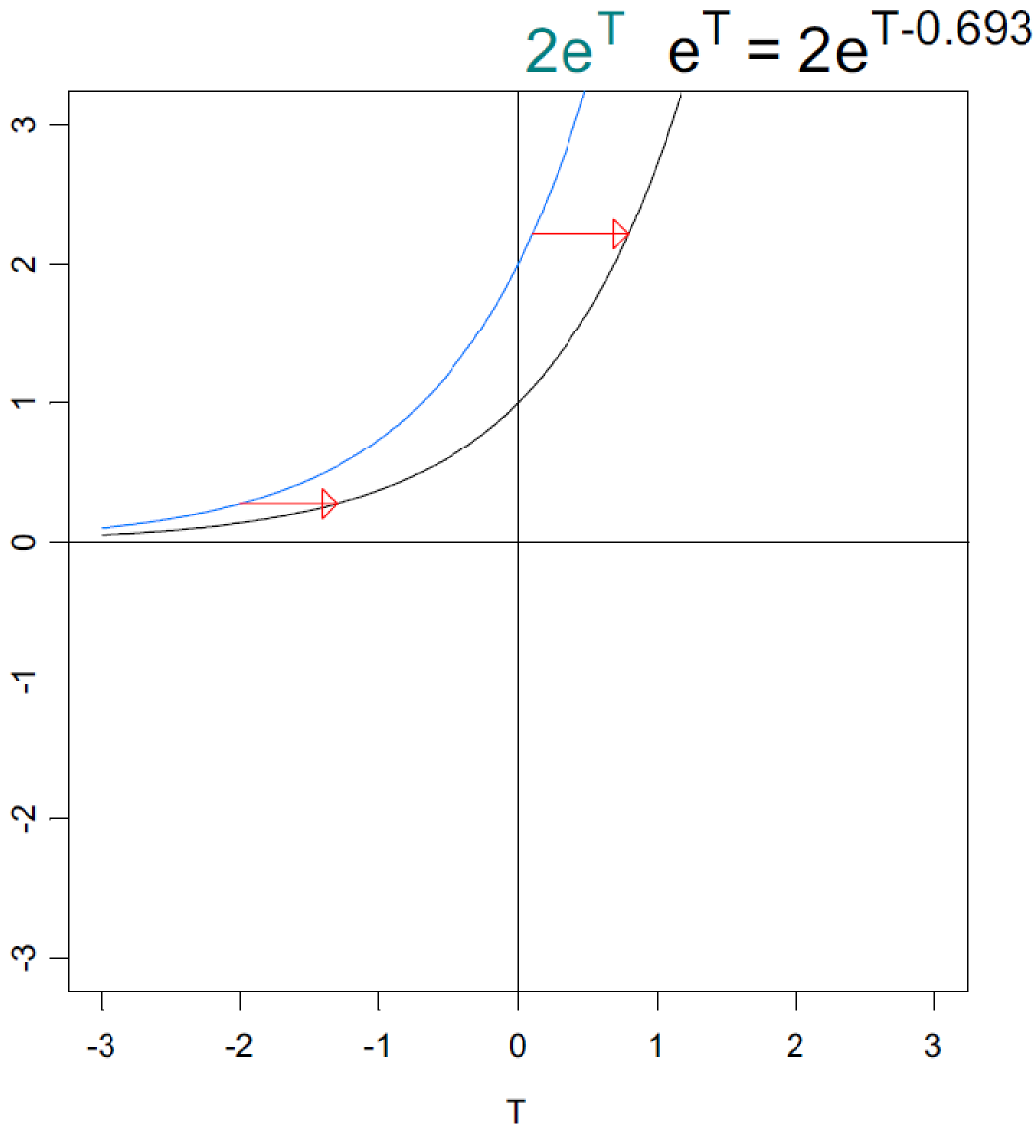
e^{-T}
e^{-2T}
$e^{-T/2}$
3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3
T

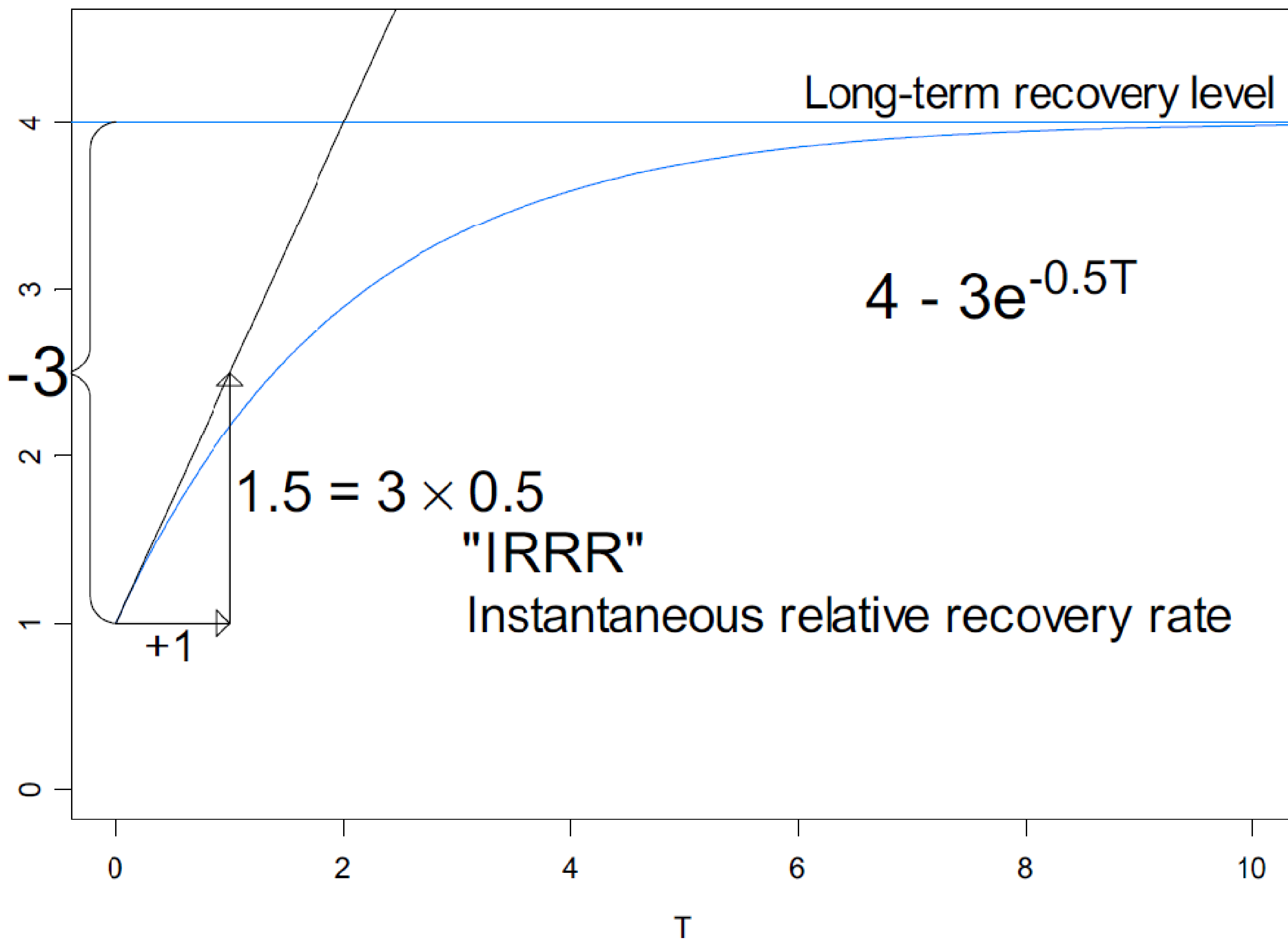
Exponential asymptotic growth

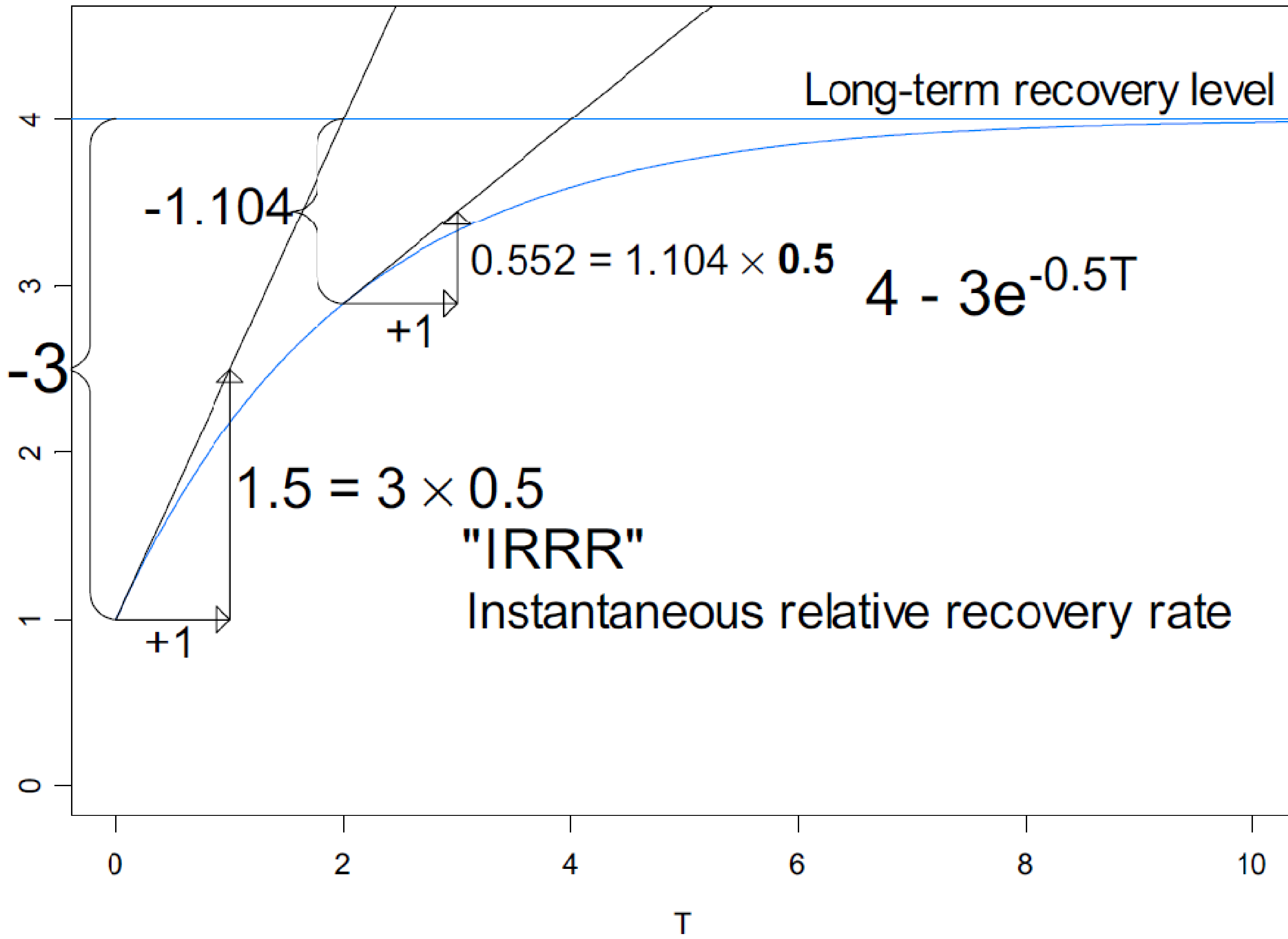



3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3
T
$-e^{-T/2}$
-3
$-3e^{-T/2}$


3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3
T
-3
2 - 3e^{-T/2}

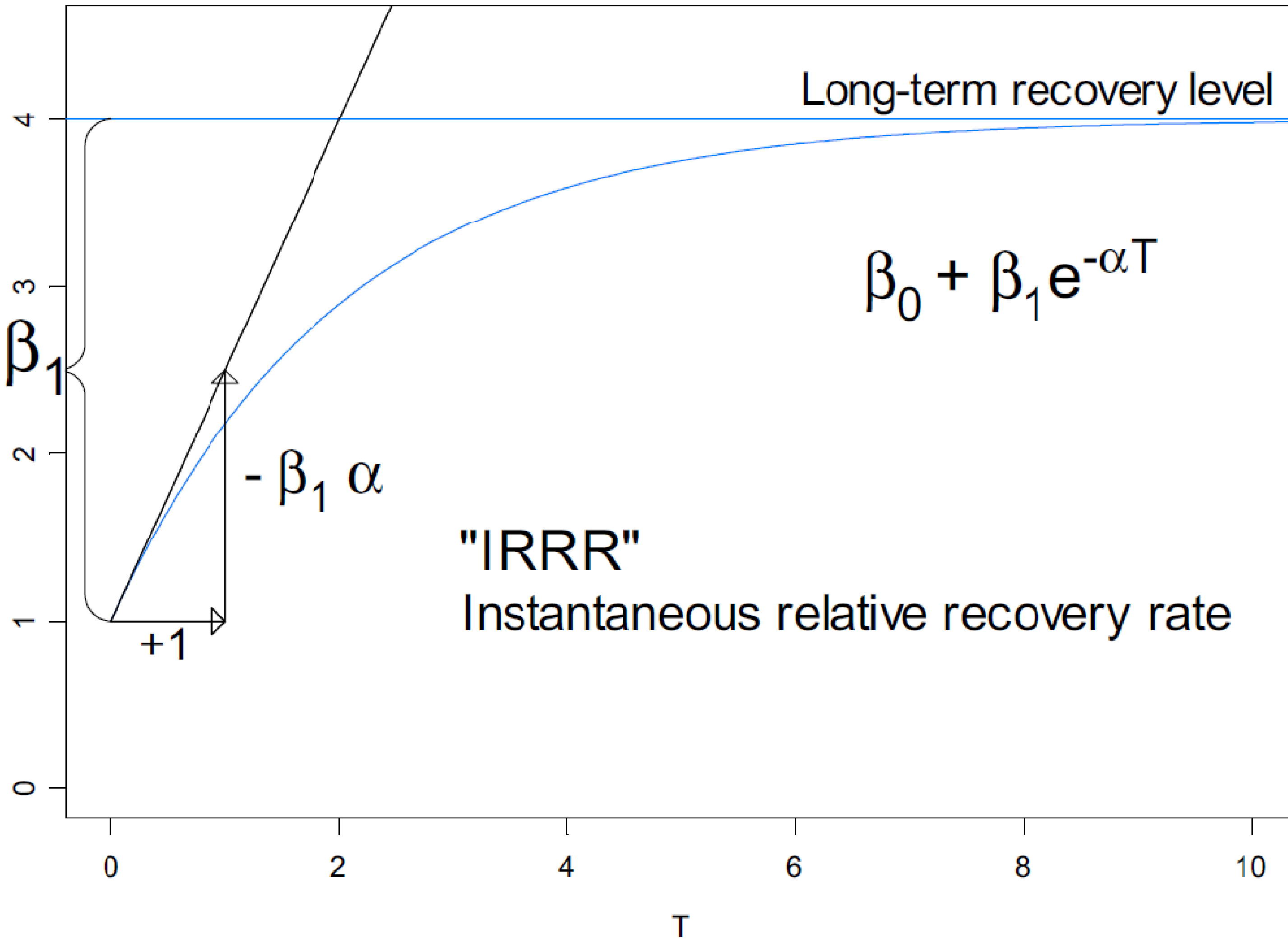

2e^T
e^T = 2e^T-0.693
3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3
T

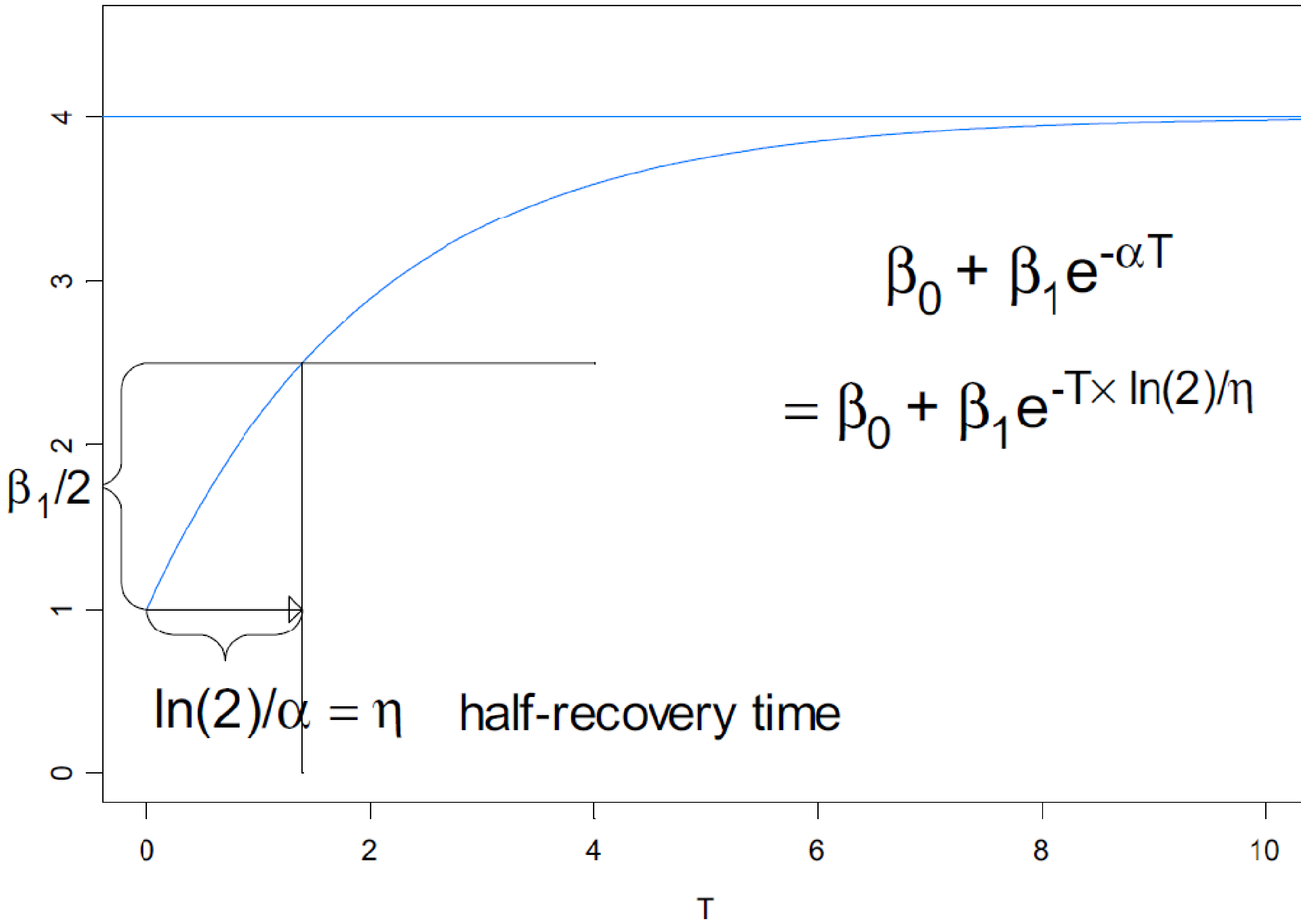

Long-term recovery level
4 - 3e^-0.5T
-3
1.5 = 3 × 0.5
"IRRR"
Instantaneous relative recovery rate
+1
T
0
1
2
3
4
0
2
4
6
8
10



Long-term recovery level
-1.104
0.552 = 1.104 × 0.5
4 - 3e^-0.5T
+1
-3
1.5 = 3 × 0.5
"IRRR"
Instantaneous relative recovery rate
+1
0
1
2
3
4
0
2
4
6
8
10
T

Using half-recovery time instead (half-life)





$\beta_0 + \beta_1 e^{-\alpha T}$
$= \beta_0 + \beta_1 e^{-T\times \ln(2)/\eta}$
$\beta_1/2$
$\ln(2)/\alpha = \eta$ half-recovery time
T

Fitting a non-linear growth curve model:

$$Y_t = \beta_0 + \beta_1 \exp(-\alpha T_i) + \varepsilon_i \quad \varepsilon_i \text{ independent } N(0,\sigma$$

Later we will use nlme for longitudinal data with more
subject. With just one subject we use 'nls', which is to 'n
'lm' is to 'lme'.

The syntax for fitting a non-linear model is very simila
linear model with three differences.

1. With a linear model we only need to specify the
We don't need to say anything about the ***parame***
it is understood that there is exactly one paramet
regressor (some predictors will have more than o
and each parameter multiplies its regressor. The
model formula for a non-linear model needs to s

the parameters and the regressor.

2. The algorithm for fitting is iterative and needs st
which you generally need to supply.

3. In non-linear mixed effects models – with `nlme`
in the non-linear model are themselves be model
linear models potentially based on other predicto
allows the non-linear model to be simpler since i
to capture the essentially non-linear aspects of th
Another advantage is that this formulation is eas
numerically, i.e. it's less work for the computer.

Growth curve model:

$$Y_i = \beta_0 + \beta_1 \exp\left(-\alpha T_i\right) + \varepsilon_i \qquad \varepsilon_i \text{ i.i.d. } N(0,\sigma^2)$$

Non-linear model formula:

```
iq ~ b0 + b1*exp(-alpha*days)
```

The formula contains references to data: `iq, days` th
found in the `iq` data frame.

Parameters: `b0, b1, alpha` that need starting val

Finding starting values: best way: sketch and undertand
and infer plausible parameters.

From graphs I would guess:

```
list( b0 = 100, b1 = -20, alpha = 0
```

How did we get these values?

b0 is the long-run level, **b1** is the relative deficit at
alpha is the daily proportion of lost iq recovered. I
might take 100 days for a half recovery of 0.5, so di
suggests roughly 0.005 per day.

Call in R:

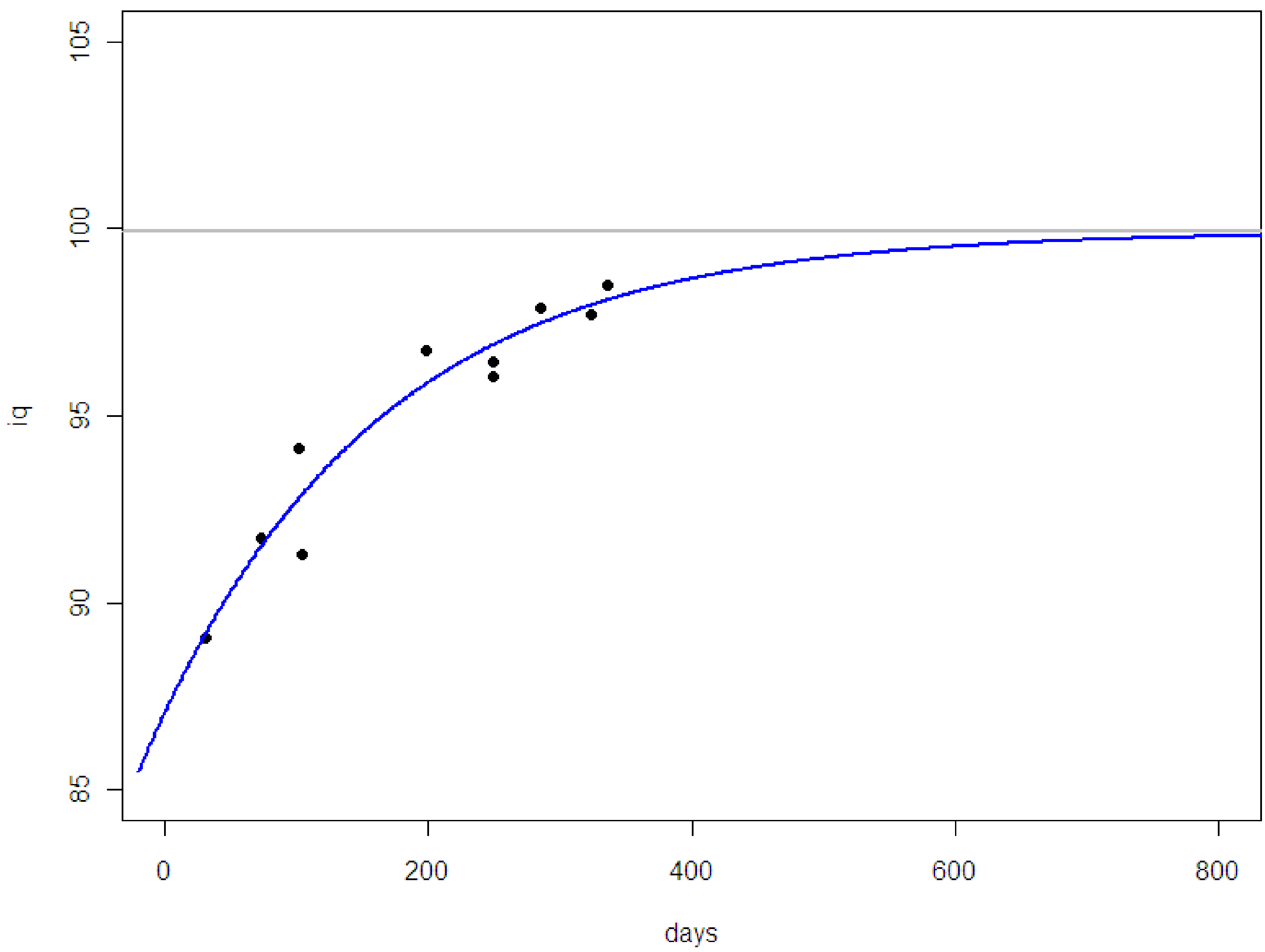
```
nls( iq ~ b0 + b1*exp(-alpha*days),
  start = list(  b0 = 100, b1 = -20
        alpha = 0.005 ) )
```

R code and output:

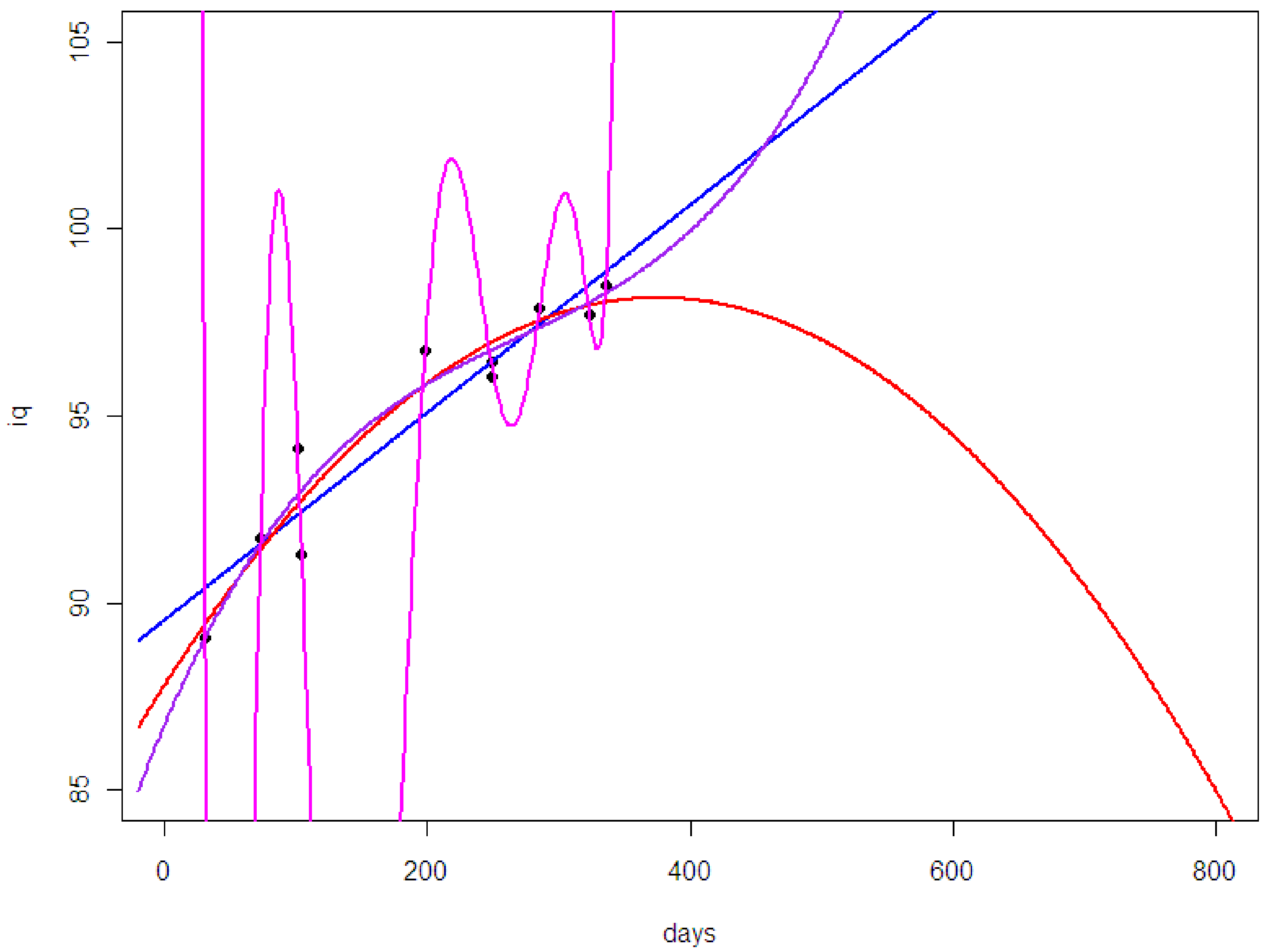
```
>  fit.nl <-nls ( iq ~ b0 + b1*exp( -a*days
+          start= list(b0 = 100, b1 = -30, a
>  summary( fit.nl )

Formula: iq ~ b0 + b1 * exp(-a * days)

Parameters:
     Estimate Std. Error t value Pr(>|t|)
b0  99.906891   2.393470  41.741 1.18e-09 **
b1 -12.847352   1.620520  -7.928 9.66e-05 **
a    0.005820   0.002956   1.969   0.0897 .
---
Residual standard error: 0.9738 on 7 degrees of

Number of iterations to convergence: 4
Achieved convergence tolerance: 7.917e-06
```

```
> pred$iq.nl <- predict( fit.nl, pred )
> plot( iq ~ days , iqsim, pch = 16,
+     xlim = c(0,800), ylim = c(85,105) )
> lines( iq.nl ~ days , pred, col = 'black', lwd
> coef( fit.nl )
           b0            b1             a
 99.906891397 -12.847351723   0.005819758
>    abline( h = coef(fit.nl)[1], col = 'gray',
```

Asymptotic growth curve:



Polynomials:



Alternative: Transforming Time

What difference does it make if we turn our non-linear
linear model by transforming time:

$$\text{ttime} = \exp\{-0.0056 \times \text{days}\}$$

As days $\rightarrow \infty$, ttime(days) $\rightarrow 0$, as days $\rightarrow 0$, ttime(da

```
>    ttime <- function( x ) exp( -0.0
>    fit.lin <- lm( iq ~ ttime( days
>    summary(fit.lin)
```

```
Coefficients:
             Estimate Std. Error t value
(Intercept)   99.9224     0.5628  177.54
ttime(days) -12.8535     1.2508  -10.28

Residual standard error: 0.9109 on 8 degrees of
Multiple R-squared: 0.9296,     Adjusted R-squar
F-statistic: 105.6 on 1 and 8 DF,  p-value: 6.92
```

Compare coefficients from tranformed fit and from no

```
Coefficients: (transformed)
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   99.9224   0.5628   177.54 1.13e-15
ttime(days)  -12.8535   1.2508   -10.28 6.92e-06

Parameters: (non-linear)
     Estimate Std. Error t value Pr(>|t|)
b0  99.906891   2.393470  41.741 1.18e-09 ***
b1 -12.847352   1.620520  -7.928 9.66e-05 ***
a    0.005820   0.002956   1.969   0.0897 .
```

- The **estimated parameters** are almost the same but the linear fit using tran reports **much smaller SEs** than the non-linear fit.
- Why? The linear fit is not taking into account the uncertainty stemming fro not known.
- Note that the biggest difference in SE occurs for the asymptote. Unless you the asymptote – where the curve gets very flat – the estimate of the asympt on the estimate of curvature.
- When reviewing work that used transformations consider whether a non-lin have been more honest. Note that the transformation is free if it is an intent scale: e.g. log(Salary).

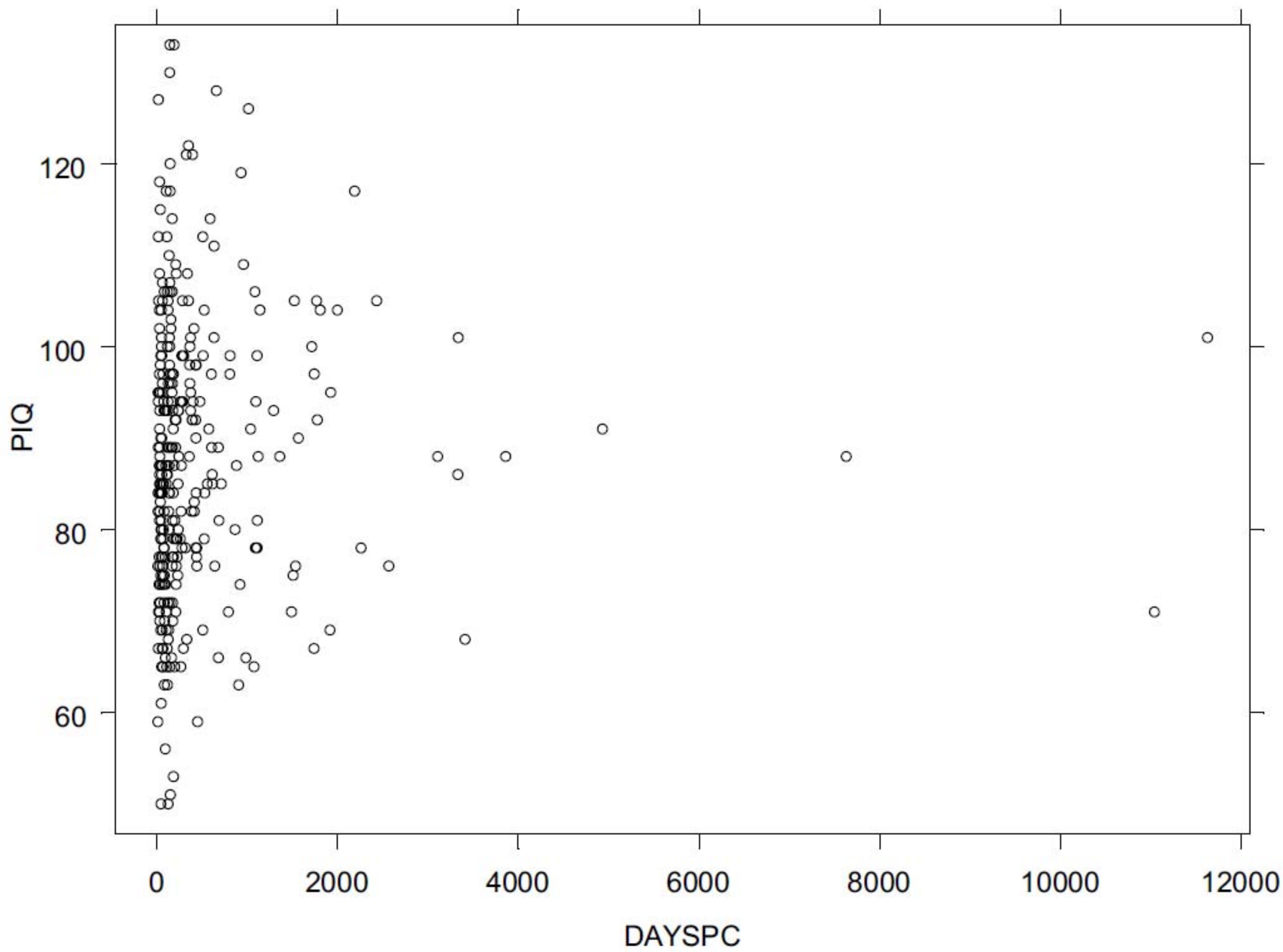
Summary:

We fit an asymptotic non-linear growth curve to PIQ as a fu
post Coma.

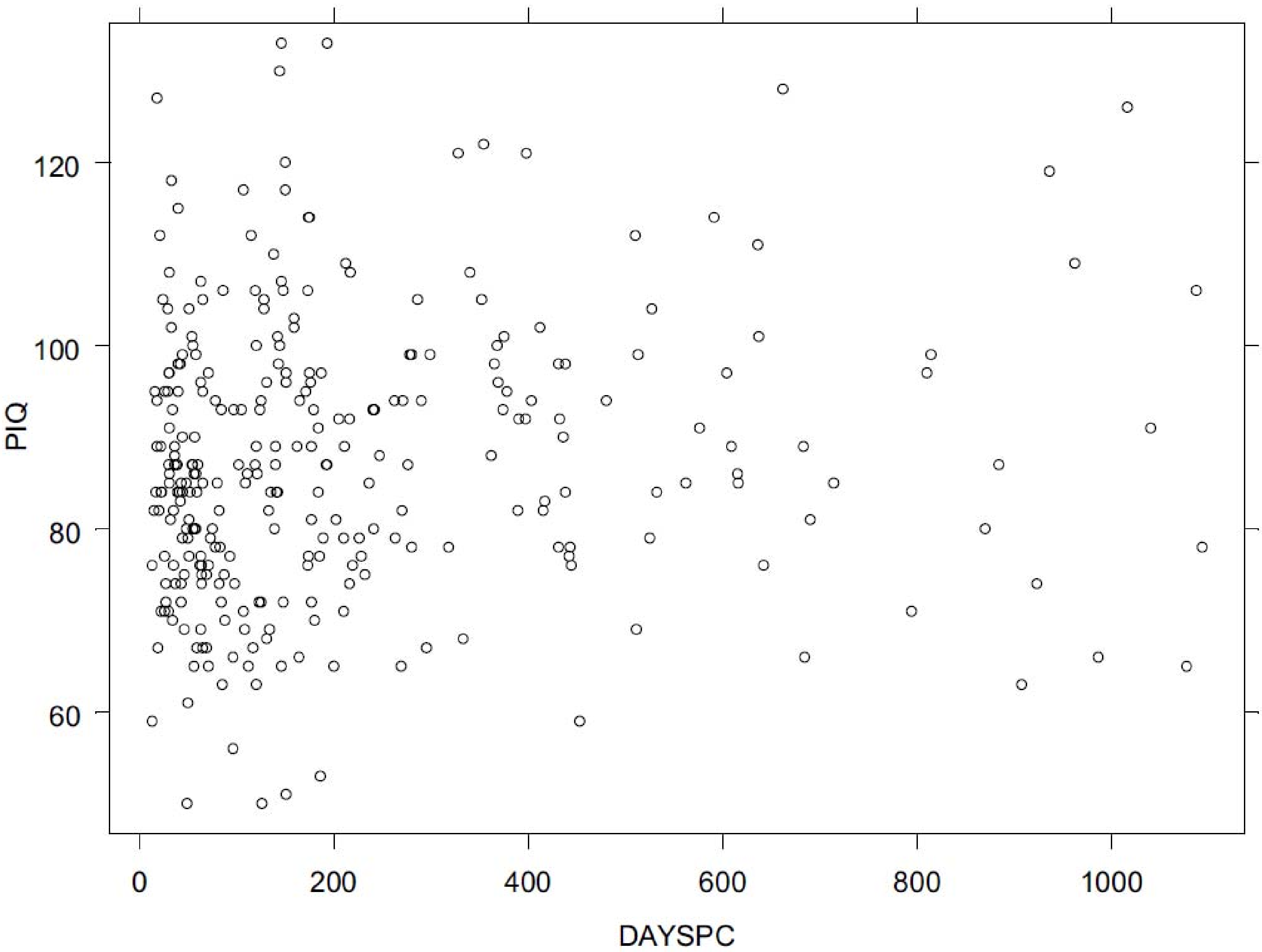
We then try the same model with VIQ in the hope of compa
models, but the VIQ model does not converge. We explore
remedies of non convergence and eventually decide on a mi
reparametrization. It works! We then use the same model on
compare the two models.

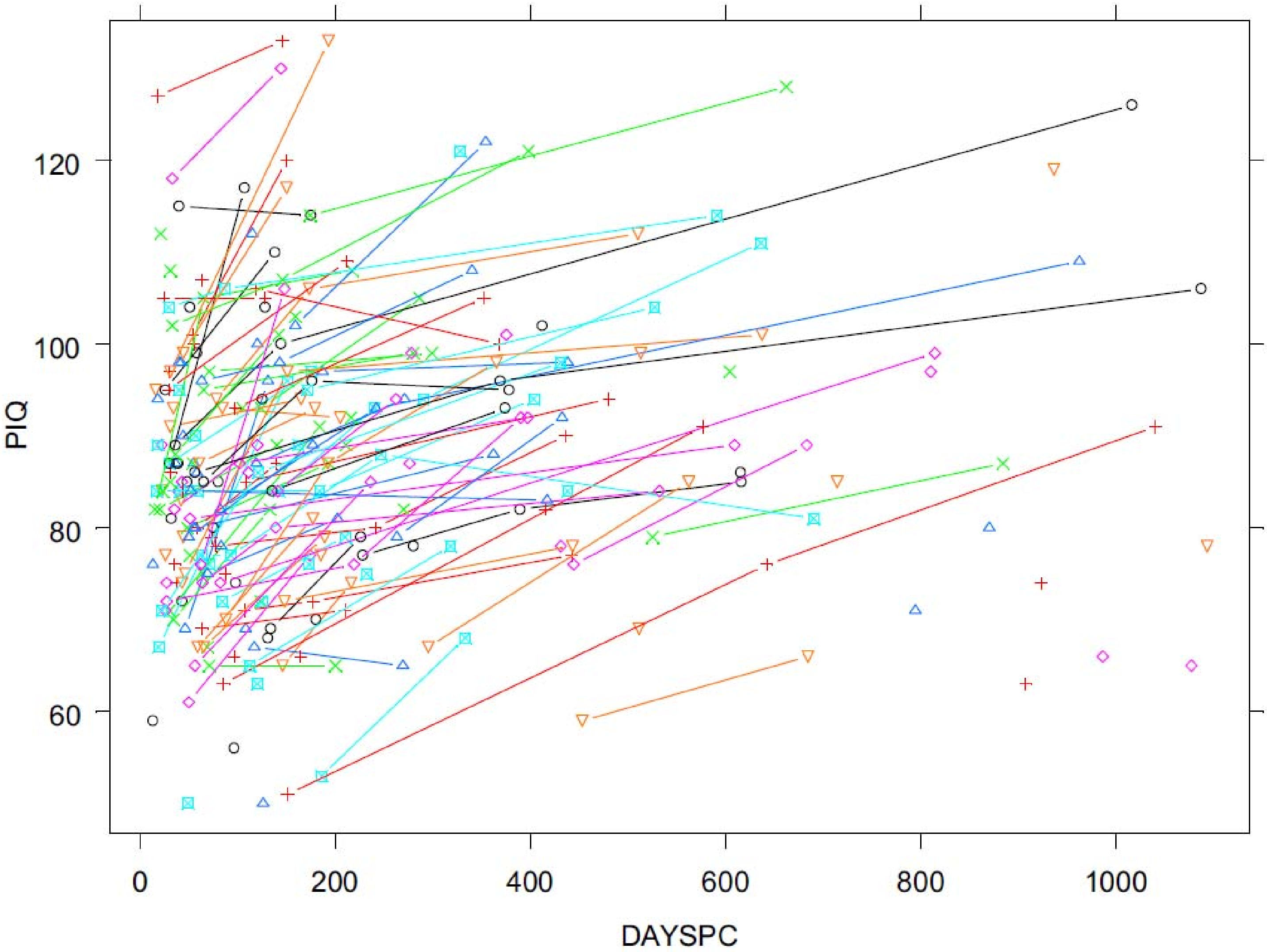
These two models are 'univariate' multivariate models, looki
response at a time. To get p-values in the comparison of the
two responses, we need to do more. One possibility is boots
we don't explore. The other is to exploit multilevel (with 3 l
modeling in nlme to fit something close to (but not exactly)
model. This is done at the end of the Lab script.

Recovery of post-coma IQ



First 3 years:




PIQ
DAYSPC
0
200
400
600
800
1000
60
80
100
120

Using nlme

The nlme model is specified like a hierarchical model, e
you can mix variable levels.
Example:

```
nlme( piq ~ b0 + b1*exp(-a*dayspc),
   data = iq,
   fixed = list( b0 ~ 1+sqrt( dcoma
      b1 ~ 1,
      a ~ 1),
   random = list( id = b0 ~ 1),
   start = list( fixed = c(
      100, 0,-20.,.05)),
   control = list( maxIter = 100,
      returnObject = TRUE),
   verbose = T)
```

The code one line at a time:

```
piq ~ b0 + b1*exp(-a*dayspc)
```

A ***non-linear model formula*** with ***regressors*** and pa
this example, it's the Level 1 model. In general, you
Level 2 ***regressors*** in this model. If you want to use
need to use it through its dummies: e.g.

```
b.sex * (sex=="Female")
```

where `b.sex` is the parameter multiplying the indic
Female.

```
data = iq,
```

data frame as usual

```
fixed = list(
       b0 ~ 1+sqrt( dcoma ),
       b1 ~ 1+sqrt( dcoma ),,
       a ~ 1)
```

A list of ***linear model formulas***, one for each param
the parameter `a` is assumed to have the same value
population, `b0` and `b1` are assumed to depend throu
model on `sqrt(dcoma)`. This transformation inc
assumption that an extra day of coma after, say, 3 da
greater impact than an extra day after 50 days. The s
transformation was chosen by examining visual plot
somewhat arbitrary. Also it is an oversimplification
that `a` is a constant across the population.

```
random = list( id = list( b0 ~ 1, b1
```

Specify the parameters that are assumed to vary ran
id to id. Note that `b0` is the asymptotic level but it is
constant added to all observations.

```
start = list( fixed =
   c( 100, -10, -20.,-1,.05)))
```

This is the challenging part that rewards a good und
the paramters of the model. Recall the fixed portion
above:

```
       fixed = list(
              b0 ~ 1+sqrt( dcoma ),
              b1 ~ 1+sqrt( dcoma ),
              a ~ 1)
```

The starting values are listed in the same order as th
of the 'fixed' portion of the model. Generally, it is go
have plausible starting values. Draw a sketch and m
guesses Here, our starting model is:

```
    b0 = 100 – 10 * sqrt(dcoma)
    b1 = –20 – 1 * sqrt(dcoma)
    a = 0.05
```

```
control = list( maxIter = 100,
       returnObject = TRUE)
```

Increases the default number of iterations from 50 to
returns the last fit even if there is no convergence. W
to use this shortly.

```
verbose = T
```

This shows information on each **PNLS** and **LME** st
Ctrl-W in the R console to get unbuffered output an
watch a frequently exciting show.

Fitting the model:

```
> fit.nlme <- nlme(
+       piq ~ b0 + b1*exp(-a*dayspc),
+       data = iq,
+       fixed = list(
+             b0 ~ 1 + sqrt(dcoma) ,
+             b1 ~ 1 + sqrt(dcoma) ,
+             a ~ 1),
+       random = list( id  = list( b0 ~ 1, b1~ 1
+       control = list( maxIter = 200, returnObje
+       start = list(
+              fixed = c(100, -10, -10, 0,.05)),
+       control = list( maxIter = 100, returnObje
+       verbose = TRUE)
```

```
. . . . [Omitting Output on Iterations 1 to 3]
**Iteration 4
LME step: Loglik: -1287.679 , nlm iterations: 1
reStruct  parameters:
      id1       id2       id3
 1.515913  0.949715 23.921793

PNLS step: RSS =  15018.94
 fixed effects:97.0948  -1.24521  -11.1453  -3.24829
 iterations: 7
```

This was pretty quick co

```
Convergence:
       fixed     reStruct
1.312661e-06 7.021607e-04


> summary( fit.nlme )
Nonlinear mixed-effects model fit by maximum lik
  Model: piq ~ b0 + b1 * exp(-a * dayspc)
 Data: iq
       AIC      BIC    logLik
  2593.358 2627.577 -1287.679
```

```
Random effects:
 Formula: list(b0 ~ 1, b1 ~ 1)
 Level: id
 Structure: General positive-definite, Log-Cholesky parametrization
               StdDev    Corr
b0.(Intercept) 13.769293 b0.(I)
b1.(Intercept)  2.605835 -0.994
Residual        6.736055
```

worries me a bit
I might try to reparametri

```
Fixed effects: list(b0 ~ 1 + sqrt(dcoma), b1 ~ 1
sqrt(dcoma), a ~ 1)
                   Value Std.Error  DF  t-value
b0.(Intercept)  97.09476  2.036582 127 47.67536
b0.sqrt(dcoma)  -1.24521  0.480486 127 -2.59157
b1.(Intercept) -11.14530  3.208072 127 -3.47414
b1.sqrt(dcoma)  -3.24829  1.076749 127 -3.01676
a                0.00825  0.001651 127  4.99579


 Correlation:
               b0.(I) b0.s() b1.(I) b1.s()
b0.sqrt(dcoma) -0.724
b1.(Intercept) -0.596  0.463
b1.sqrt(dcoma)  0.463 -0.455 -0.789
a              -0.309  0.013  0.092 -0.380
```

```
Standardized Within-Group Residuals:
         Min           Q1          Med           Q3
-3.332408193 -0.365688335  0.009002275  0.382738703  2.30311
```

```
Number of Observations: 331
Number of Groups: 200
```

> ***An interesting calculation:***

Between subject SD of 'true' IQ	13.769
Within subject between test SD of IQ	6.736
Population SD of IQ	$\sqrt{13.769^2 + 6.736^2} =$
Test-retest reliability of IQ $= \dfrac{\text{Variance in True Score}}{\text{Variance of Observed Score}}$	$\dfrac{13.769^2}{15.328^2} = 0.8$

How does fitting work?

See Pinheiro and Bates (2000) and Lindstrom and B for a description. It's a clever blend of available tools. B Watts (1988) *Non-linear regession analysis and its app* deals with non-linear models for independent data whic adapted to situation where the variance-covariance is kn So we have we have tools for non-linear models when t is known. And we have tools for linear mixed models (l have estimates of parameters in a non-linear model we c an approximating linear model.

The algorithm keeps repeating 2 steps until convergenc

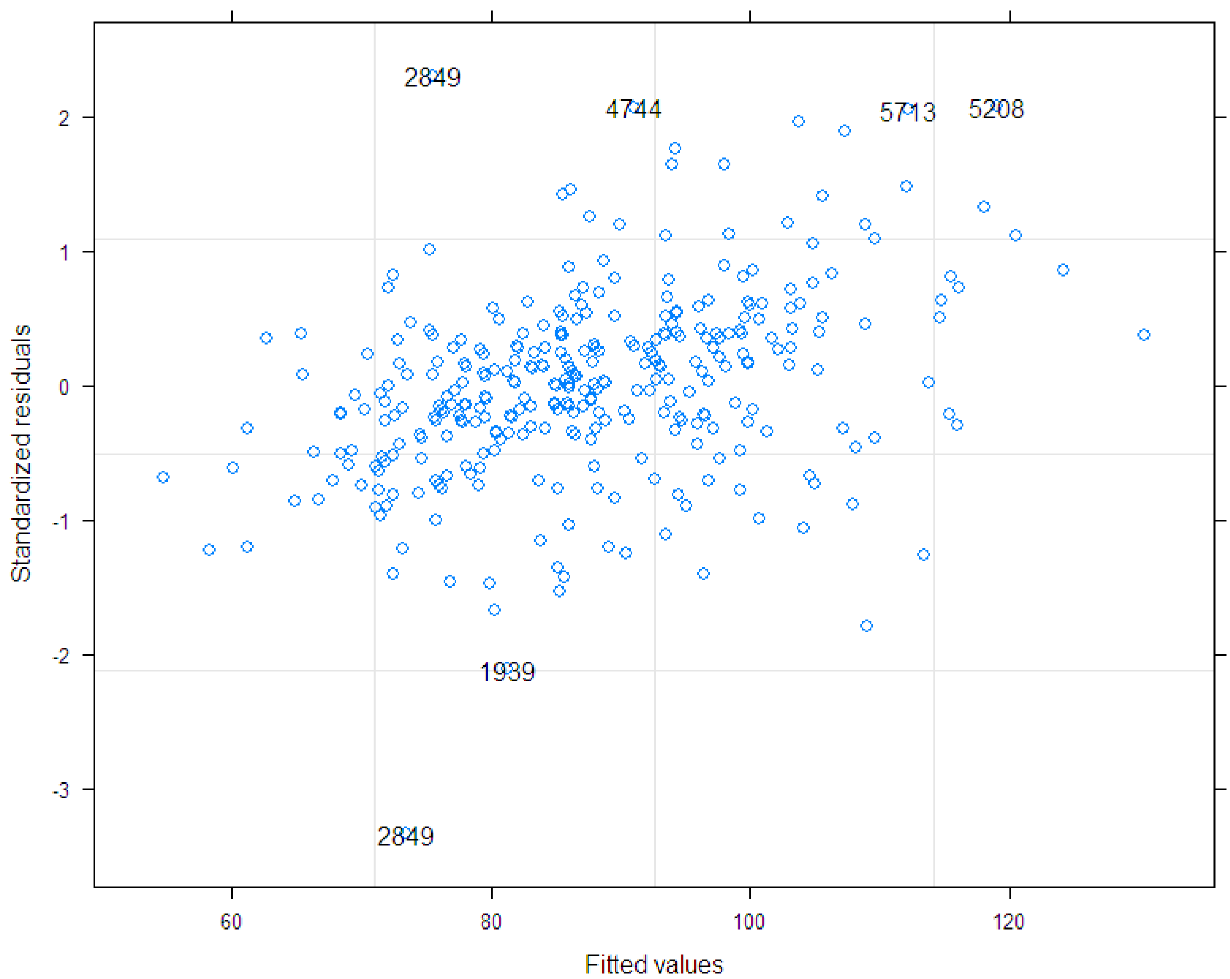
1) PNLS step: Given an estimate of G and R, estimate f parameters and random effects using a ***p***enalized ***n***on-li ***s***quares algorithm.

2) LME step: Given estimates of fixed parameters and r
effects, construct an approximating linear model and es
R with lme.

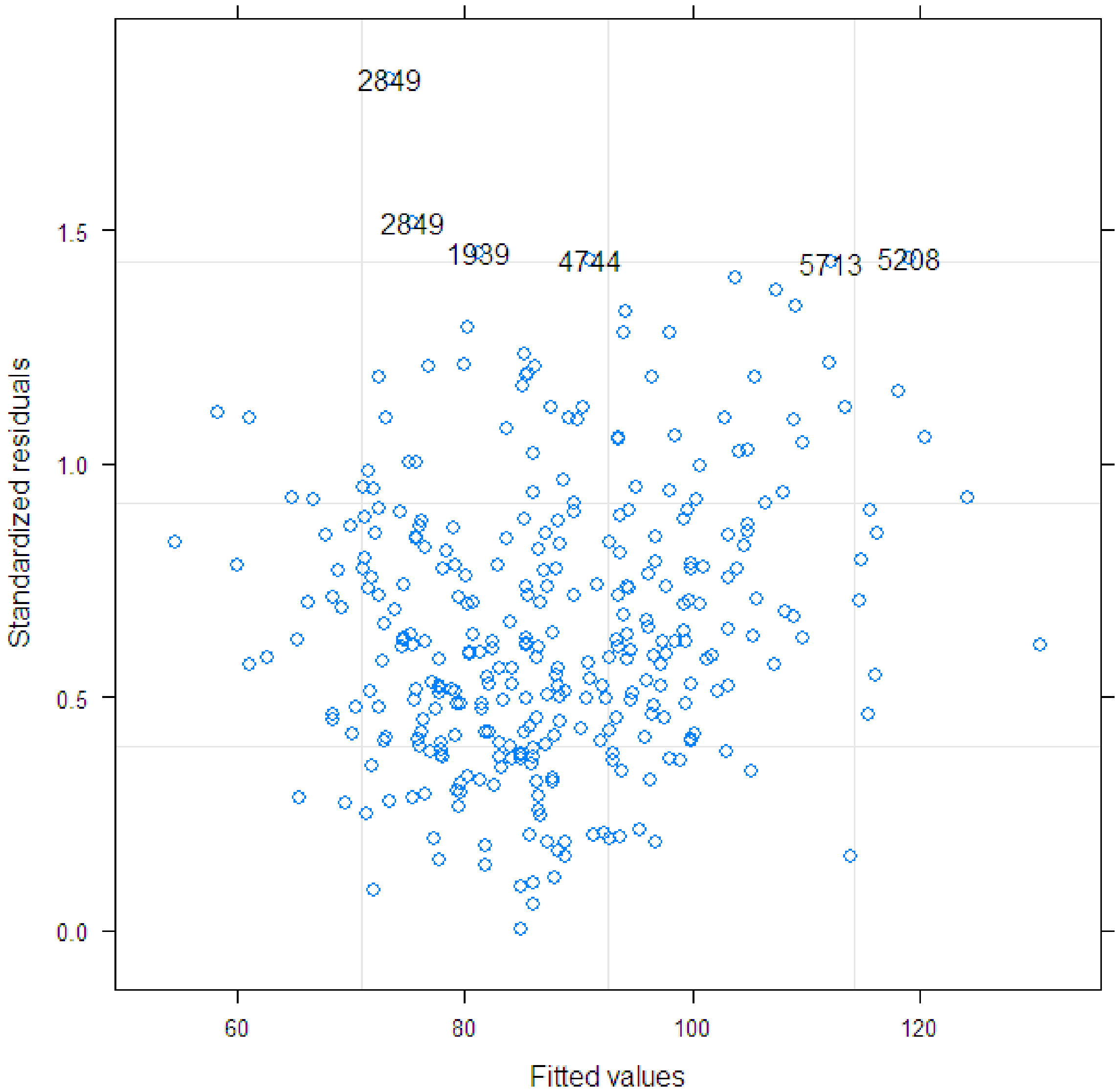
Keep repeating (1) and (2) until the estimates don't char

Some diagnostics for PIQ

```
> plot( fit.nlme, resid(. , type = 'p') ~ fi
+          id = .05)
```

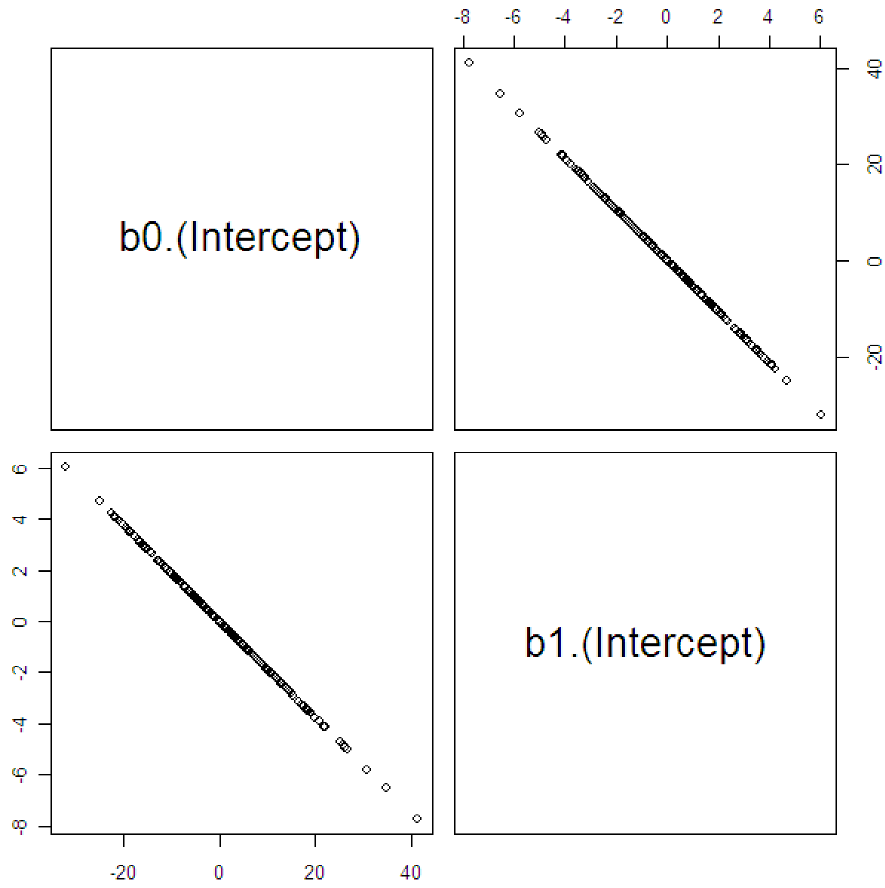


```
> plot( fit.nlme, sqrt( abs( resid(. ,type='
+          ~fitted(.), id = .05)
```




```
>  plot( ranef( fit.nlme )) # output omitted
```

```
> pairs( ranef( fit.nlme ))
```



This plot shows the near singularity of the G matrix sho
the strong correlation between **b0** and **b1**. It suggests th
recovery level, possibly related to pre-trauma intelligen
confer so large a benefit in the early stages of recovery.

Fitting VIQ

```
> fit.nlme.viq <- nlme(
+     viq ~ b0 + b1*exp(-a*dayspc),
+     data = iq,
+     fixed = list( b0 ~ 1 + sqrt(dcom
+                   b1 ~ 1 + sqrt(dcom
+                   a ~ 1),
+     random = list( id  =
+         list( b0 ~ 1, b1~ 1 )),
+     start = list(
+         fixed = c(100, -.3, -10, -5,
+     control = list( maxIter = 100,
+         returnObject = T),
+     verbose = T)
```

```
**Iteration 1

LME step: Loglik: -1246.109 , nlm iterations: 20
reStruct  parameters:
        id1         id2         id3
-0.31704425 -0.03919651  1.58004210

PNLS step: RSS =  8190.372
 fixed effects:98.3784  -0.398053  -11.1171  -3.81443
 iterations: 7

Convergence:
    fixed  reStruct
0.6045864 0.3340419
```

We hope these numbers will get very small

Then it repeats:

```
**Iteration 2
LME step: Loglik: -1242.047 , nlm iterations: 10
reStruct  parameters:
       id1        id2        id3
-0.6005839 -1.1484260  0.3314853

PNLS step: RSS =  29302.31
```

```
 fixed effects:93.3904  -0.500735  -2.79404  -5.93326
 iterations: 7

Convergence:
   fixed reStruct
2.978855 2.355074
```

Much, much later:

```
**Iteration 97
LME step: Loglik: -1223.248 , nlm iterations: 11
reStruct  parameters:
       id1        id2        id3
-0.3076645 -0.9642632  0.4746019

PNLS step: RSS =  13520.03
 fixed effects:96
 iterations: 7

Convergence:
    fixed  reStruct
0.6185455 1.3370712
```

```
**Iteration 98
LME step: Loglik: -1239.934 , nlm iterations: 11
reStruct  parameters:
       id1        id2        id3
-0.6920771 -0.9315449  0.3989076

PNLS step: RSS =  9035.715
 fixed effects:98.3663  -0.345367  -8.12821  -2.46351
 iterations: 7

Convergence:
   fixed reStruct
1.619486 0.572147

**Iteration 99
LME step: Loglik: -1223.284 , nlm iterations: 11
reStruct  parameters:
       id1        id2        id3
-0.3084674 -0.9647008  0.4741239

PNLS step: RSS =  13512.85
 fixed effects:96.0224  -0.381974  -9.02973  -5.45431
 iterations: 7
```

These are the esti
effects at iteratio

```
Convergence:
    fixed  reStruct
0.6181109 1.3332673

**Iteration 100
LME step: Loglik: -1239.936 , nlm iterations: 11
reStruct  parameters:
       id1        id2        id3
-0.6921102 -0.9316034  0.3988804

PNLS step: RSS =  9033.863
 fixed effects:98.3667  -0.345443  -8.13618  -2.46378
 iterations: 7

Convergence:
    fixed  reStruct
1.6167060 0.5714619
Warning message:
In nlme.formula(viq ~ b0 + b1 * exp(-a * dayspc), data
= list(b0 ~  :
  Maximum number of iterations reached without converg
```

These numbers are just bouncing around and not getting smaller

Have a close look
number have been
past few iterations

Our model didn't converge.

What to do?

Looking carefully at the output we notice that:

1) The convergence criteria are not slowly getting sma
   bouncing around, and
2) Some of the fixed parameters are stuck in an oscilla
   2.

Possible actions:

1) Increase maxIter and come back much later. In this
   help,
2) Simplify the model, particularly the RE model. Here
   and it would work – but it would require making ass
   would prefer to avoid for now.
3) Consider whether the model is well identified. Are t
   parameters whoses estimates are far from expected?
   constantly changing in a given direction as iteration

3) When the process is stuck in a cycle, as it is here, wh
parameters oscillate and how do they oscillated toge
visualize what this implies about the fitted surface. A
mean that the leverage and residuals of some points
from iteration to iteration. Such points pull the fit to
themselves in one iteration but in the new fit they lo
and let go of the fitted surface for the next iteration.
happen with OLS because leverage does not depend
But the linear approximation in non-linear models d
previous fit so leverage can change from one fit to th
points involved tend to be outliers.

Finding problematic points:

1) Use maxIter to stop the iteration at each point in the
2-cycle, use an odd iteration and an even iteration. S
object in each case. Plot the residuals of one fit agai

residuals of the other. The points with very different
likely to be the culprits. You can also compare the f
each fit attempts to approximate the data.

2) Use your knowledge of the data to isolate some kno

We note that the distribution of `dcoma` is highly skewe
try dropping anyone with `dcoma > 100`. Note dcoma
subject variable and extreme values are likely to have h
With the asymptotic model for `dayspc`, influence may
with small values being very influential and large value
After further experimenting we also see that `b1`, the de
`dayspc = 0` is bound with the estimate of `a`. A larg
`a` produces a very negative value of `b1`. This leads us
realization that attempting to estimate deficit immediate
arousal is not feasible since there is little IQ data that ea
easily move the origin to say one month after arousal by

changing **dayspc** to **dayspc - 30** in the non-linear
this works we can do the same for PIQ to have compara

```
> fit.nlme.viq2 <-nlme( viq ~ b0 + b1*exp(-a*(da
+          data = iq,
+          fixed = list( b0 ~ 1 + sqrt(dcoma) ,
+                        b1 ~ 1 + sqrt(dcoma) ,
+                        a ~ 1),
+          random = list( id  = list( b0 ~ 1, b1~
+          start = list(
+          fixed = c(100, -.3, -10, 0,.3)),
+          control = list( maxIter = 100, returnO
+          verbose = T,
+          subset = dcoma < 100)
```

which converges in 3 iterations:

```
.  .  .  .  .  .
```

70

```
**Iteration 3
LME step: Loglik: -1225.668 , nlm iterations
reStruct  parameters:
       id1         id2         id3
-0.8286352 12.2429072 59.7420069

PNLS step: RSS =  9725.302
 fixed effects:99.2084  -0.561859  -6.79895
0.0214789
 iterations: 7

Convergence:
       fixed     reStruct
1.054542e-07 1.588532e-01
>
> summary( fit.nlme.viq2 )
Nonlinear mixed-effects model fit by maximum
likelihood
```

```
  Model: viq ~ b0 + b1 * exp(-a * (dayspc -
 Data: iq
  Subset: dcoma < 100
       AIC      BIC    logLik
  2469.336 2503.363 -1225.668

Random effects:
 Formula: list(b0 ~ 1, b1 ~ 1)
 Level: id
 Structure: General positive-definite, Log-C
parametrization
               StdDev       Corr
b0.(Intercept) 1.254731e+01 b0.(I)
b1.(Intercept) 2.640292e-05 0
Residual       5.478719e+00
```

```
Fixed effects: list(b0 ~ 1 + sqrt(dcoma), b1
sqrt(dcoma), a ~ 1)
                     Value Std.Error  DF  t-val
b0.(Intercept) 99.20845 1.7262297 196 57.471
b0.sqrt(dcoma) -0.56186 0.4940133 123 -1.137
b1.(Intercept) -6.79895 2.2442154 123 -3.029
b1.sqrt(dcoma) -1.87309 0.7804948 123 -2.399
a               0.02148 0.0044938 123  4.779


 Correlation:
               b0.(I) b0.s() b1.(I) b1.s()
b0.sqrt(dcoma) -0.777
b1.(Intercept) -0.418  0.332
b1.sqrt(dcoma)  0.312 -0.343 -0.860
a              -0.148 -0.057  0.096 -0.058

Standardized Within-Group Residuals:
         Min           Q1          Med           Q3
-2.157276598 -0.389534045  0.007514936  0.366479340  1

Number of Observations: 324
```

```
Number of Groups: 197
```

We have no evidence of a long-term effect of duration o
only weak evidence of an effect at the end of 30 days.

Refitting PIQ with a similar model:

```
>  fit.nlme.piq2 <-nlme( piq ~ b0 + b1*exp(-a*(d
+       data = iq,
+       fixed = list( b0 ~ 1 + sqrt(dcoma) ,
+                     b1 ~ 1 + sqrt(dcoma) ,
+                     a ~ 1),
+       random = list( id  = list( b0 ~ 1, b1~ 1
+       start = list(
+              fixed = c(100, -.3, -10, 0,.1)),
+       control = list( maxIter = 100, returnObje
+       verbose = T,
+       subset = dcoma < 100)
```

which also converges quickly.

```
> summary( fit.nlme.piq2 )
. . . .
Random effects:
 Formula: list(b0 ~ 1, b1 ~ 1)
 Level: id
 Structure: General positive-definite, Log-C
parametrization
                  StdDev       Corr
b0.(Intercept) 1.303845e+01 b0.(I)
b1.(Intercept) 1.288991e-04 0.001
Residual       6.640711e+00

Fixed effects: list(b0 ~ 1 + sqrt(dcoma), b1 ~ 1
sqrt(dcoma), a ~ 1)
                    Value Std.Error  DF  t-value
b0.(Intercept)  98.45118 2.1594531 123 45.59079
b0.sqrt(dcoma)  -1.52286 0.5784392 123 -2.63270
b1.(Intercept) -10.71808 2.6410052 123 -4.05833
b1.sqrt(dcoma)  -2.06639 0.8298037 123 -2.49022
a                0.00707 0.0014827 123  4.76892
```

These results contrast interestingly with VIQ.

EXERCISE: Note that **b1** has very small variability in
What happens if you refit without a random effect for **b**

Comparison of half-recovery times

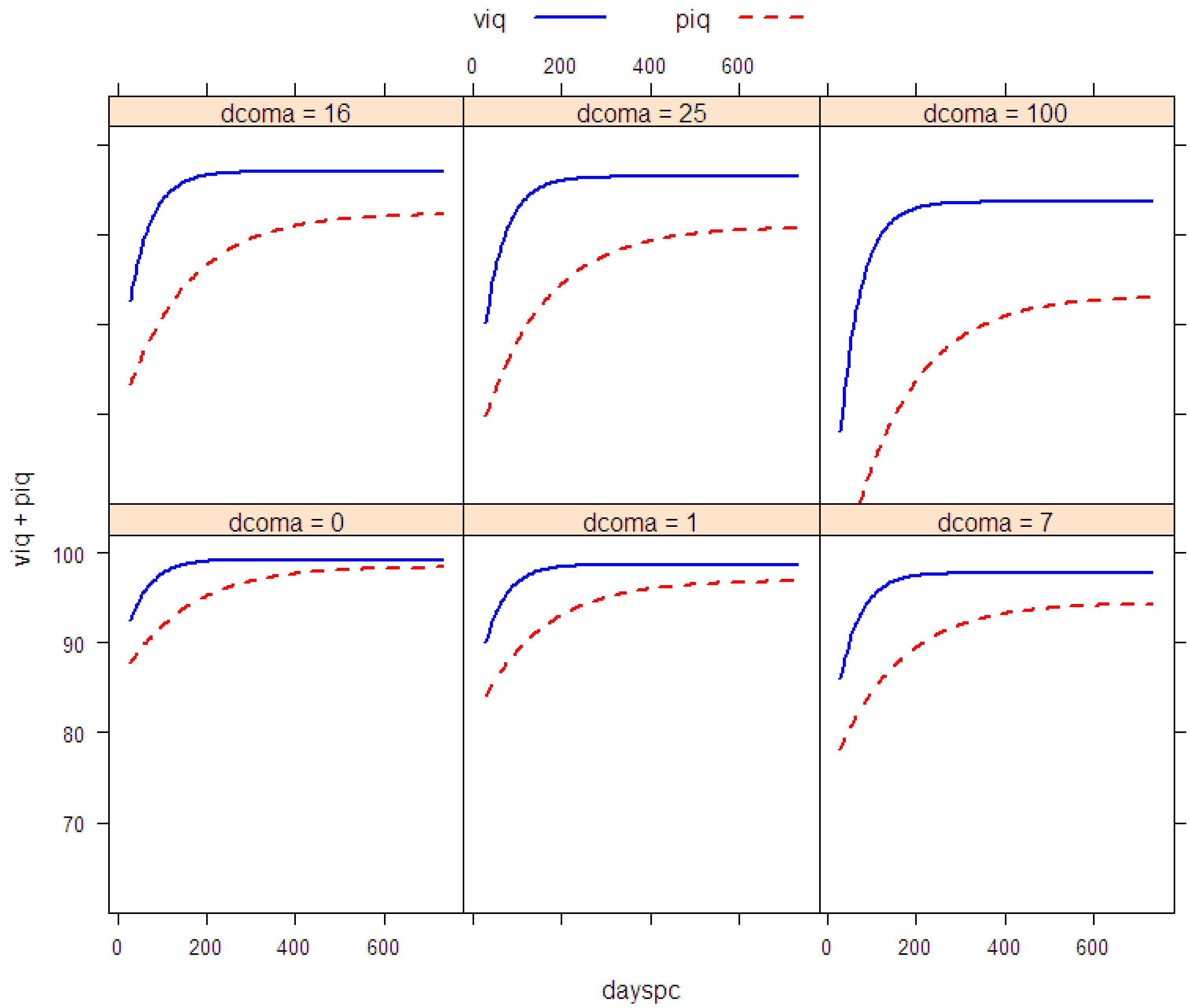
$$\text{Half-recovery time} = \frac{1}{\alpha \times \ln(2)}$$

IQ	α	half-recovery time
VIQ	0.02148	67 days
PIQ	0.00707	204 days

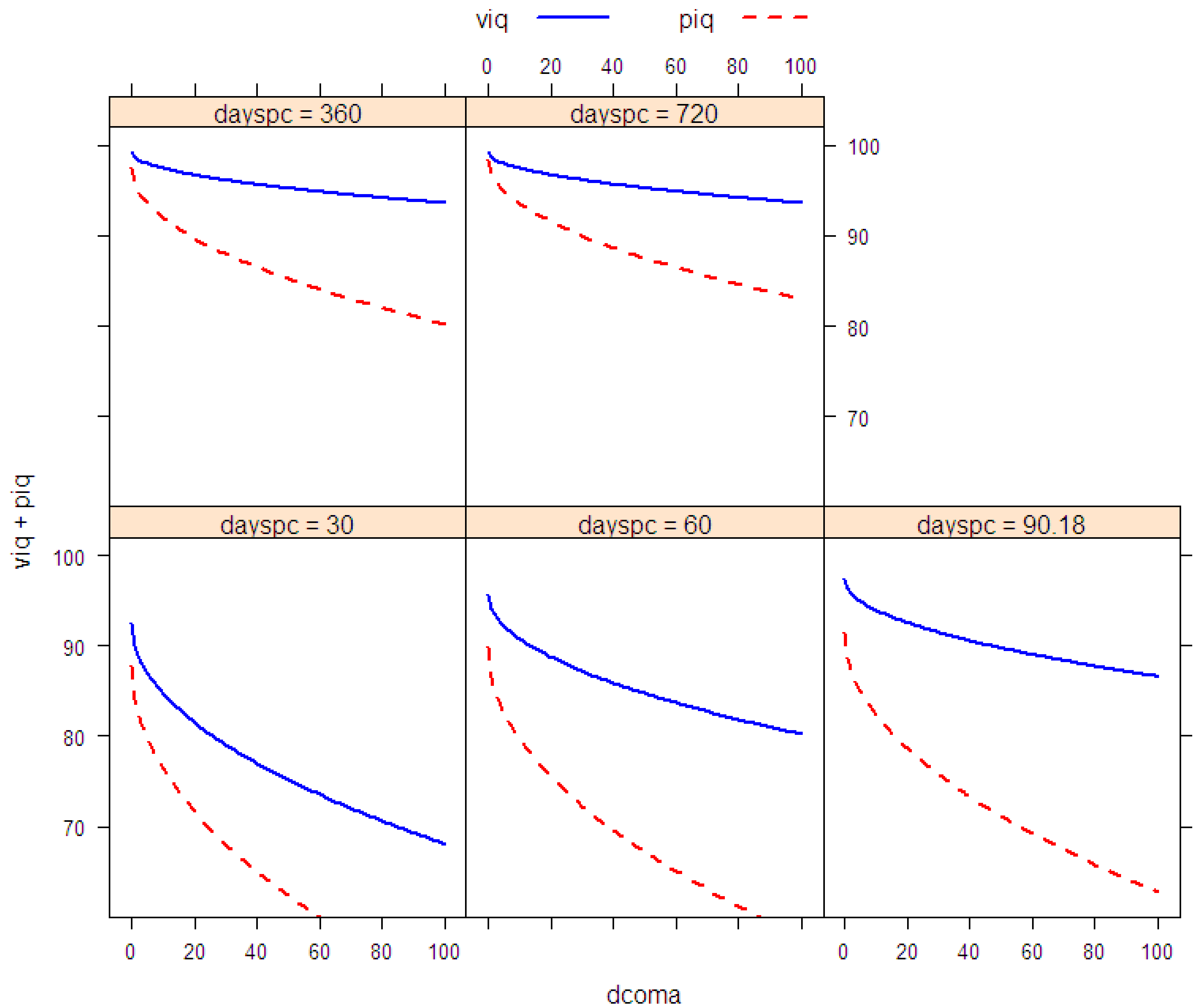
EXERCISE: Reparametrize the non-linear model form
half-life instead of IRRR in the model. Are the results c

Visual comparisons of PIQ and VIQ

```
#

windows( height = 7, width = 8.5)

pred <- expand.grid( dcoma = c(0,1,7,16,25,100),
                  dayspc = seq(30,365*2,5))
pred$piq <- predict( fit.nlme.piq2, pred, level = 0 )
pred$viq <- predict( fit.nlme.viq2, pred, level = 0 )

zz <- factor( paste( 'dcoma =', pred$dcoma))
pred$dcoma.lab <- reorder( zz, pred$dcoma)

td( col = c('blue','red'), lwd = 2)
xyplot( viq + piq ~ dayspc | dcoma.lab, pred, type =
    ylim = c(60,102),
    lwd = 2, auto.key = list(columns = 2, points = F,
```


viq
piq
dcoma = 16
dcoma = 25
dcoma = 100
dcoma = 0
dcoma = 1
dcoma = 7
viq + piq
100
90
80
70
0
200
400
600
dayspc

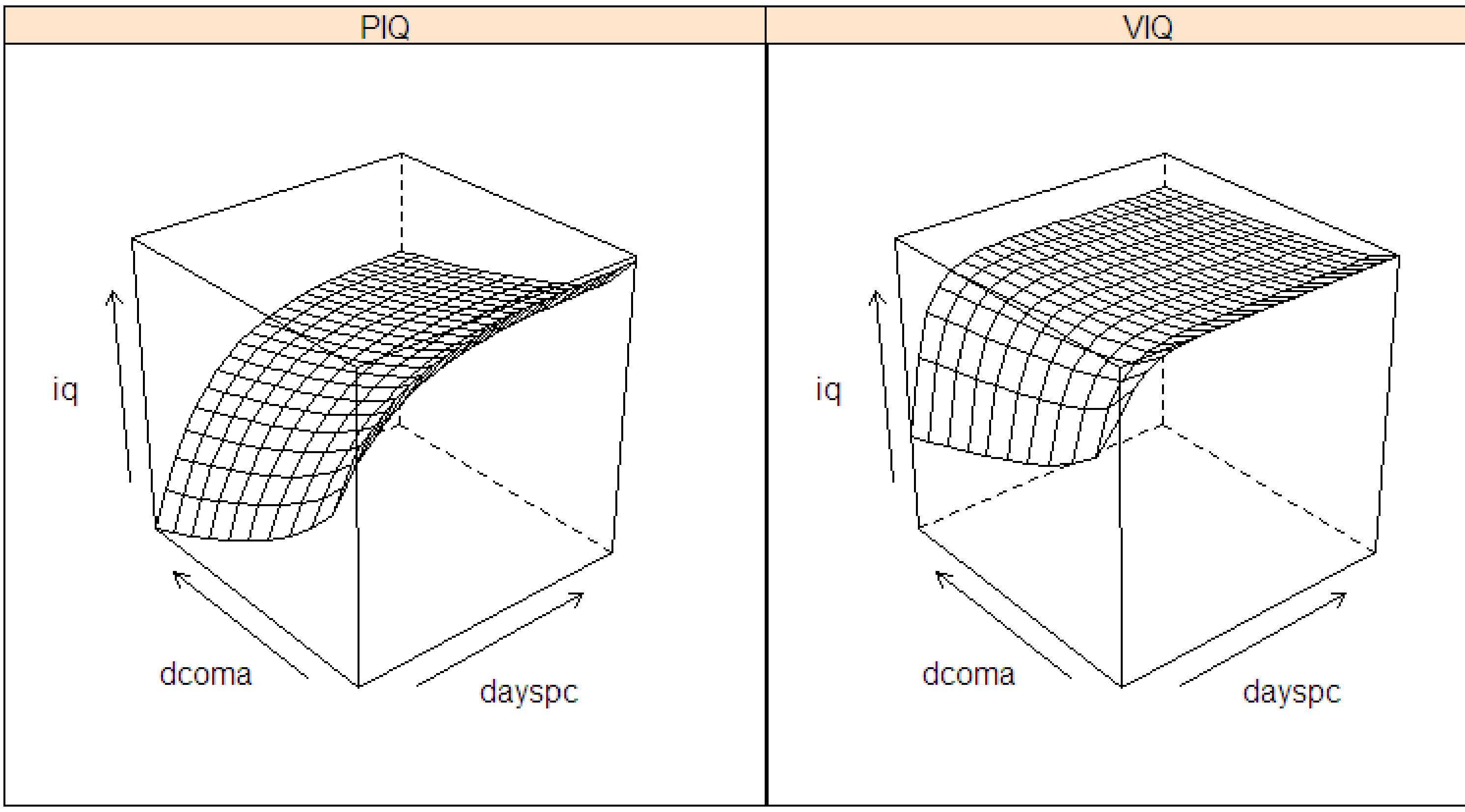
```
pred2 <- expand.grid( dcoma = 0:100,
                     dayspc = c(30,60,90.180,360,720))
pred2$piq <- predict( fit.nlme.piq2, pred2, level = 0
pred2$viq <- predict( fit.nlme.viq2, pred2, level = 0

zz <- factor( paste( 'dayspc =', pred2$dayspc))
pred2$dayspc.lab <- reorder( zz, pred2$dayspc)


xyplot( viq + piq ~ dcoma | dayspc.lab, pred2, type =
          ylim = c(60,102),
          lwd = 2,
          auto.key = list(columns = 2, points = F, li
```


viq
piq
dayspc = 360
dayspc = 720
dayspc = 30
dayspc = 60
dayspc = 90.18
viq + piq
dcoma
0
20
40
60
80
100
70
80
90
100

```
predviq <- expand.grid( dcoma = seq(0,100,10),
                    dayspc = seq(30,720,30))
predpiq <- predviq
predviq $ iq <- predict( fit.nlme.viq2, predviq, leve
predpiq $ iq <- predict( fit.nlme.piq2    , predpiq, 
predpiq$type <- factor( "PIQ" )
predviq$type <- factor( "VIQ" )
```

```
wireframe( iq ~ dayspc + dcoma | type, Rbind( predpiq,
```

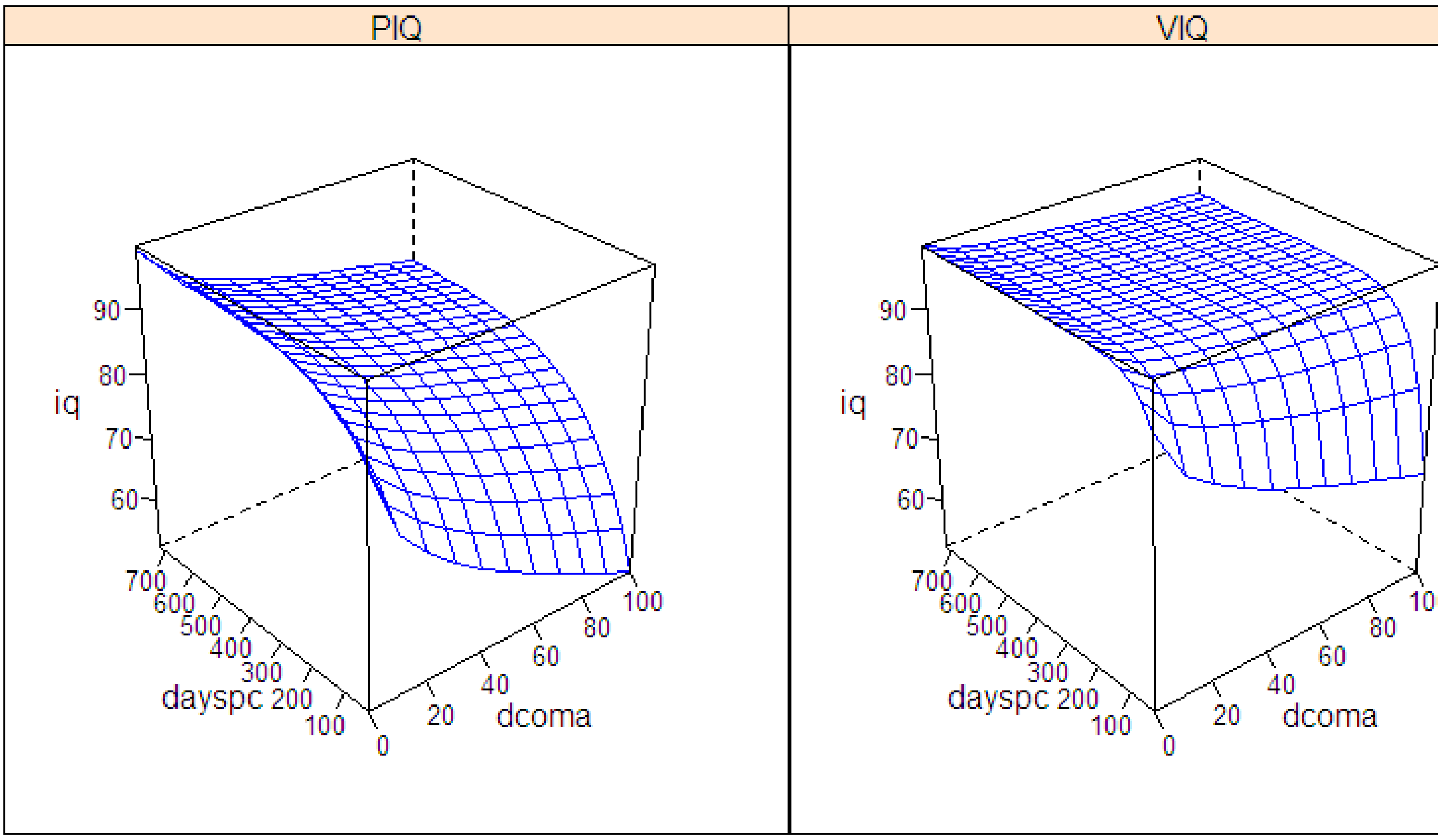
Default options and orientation for a wireframe plot:


PIQ
VIQ
iq
dcoma
dayspc
iq
dcoma
dayspc

```
wireframe( iq ~ dcoma + dayspc | type, Rbind( predpiq,
          scales = list( arrows = F), col = 'blue')
```

Exchange axes for a more natural presentation of daysp
variable) and dcoma (Level 2)

Suppress arrows and get axis with values and tickmarks

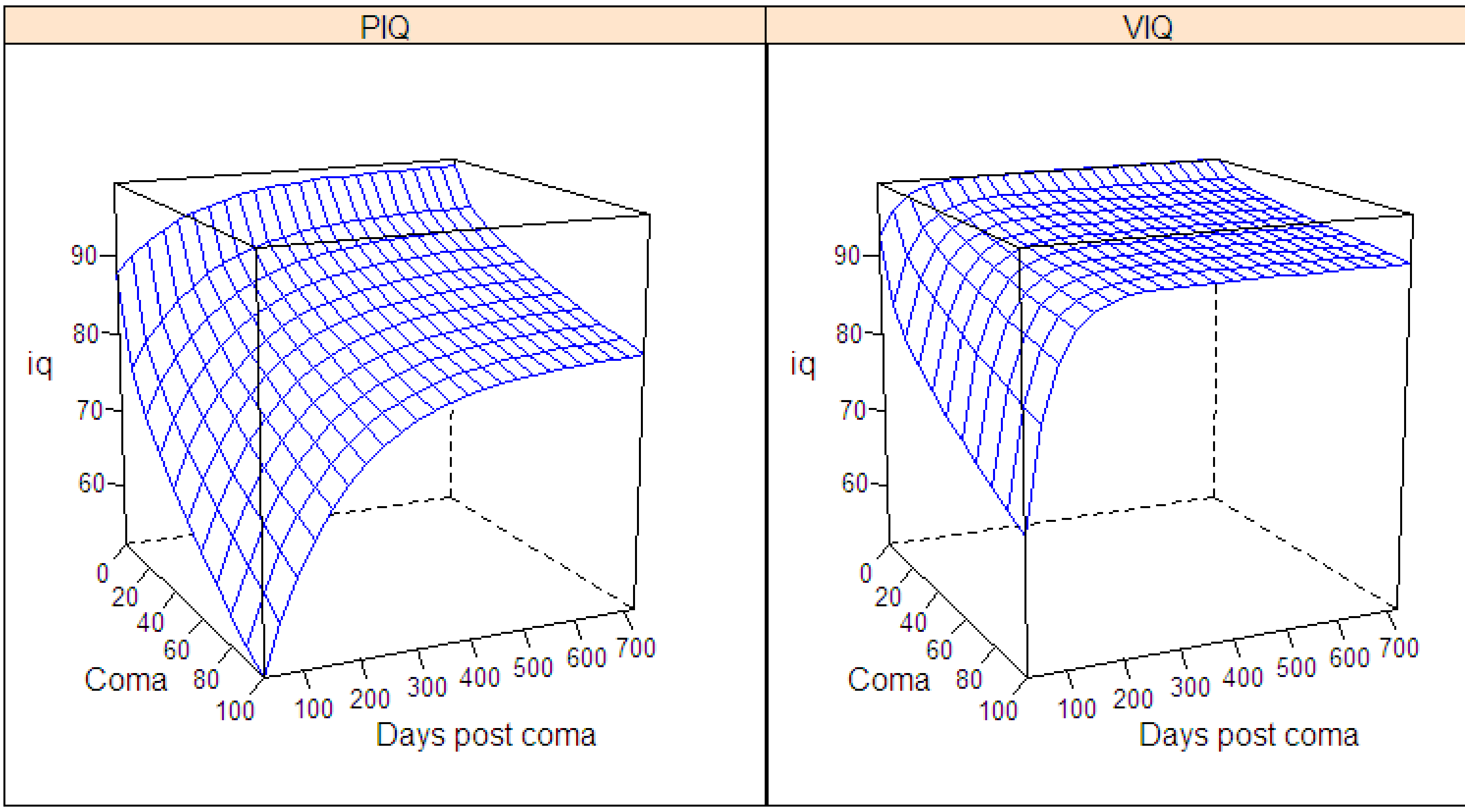


PIQ
VIQ
iq
90
80
70
60
700
600
500
400
300
dayspc 200
100
0
20
40
60
80
100
dcoma

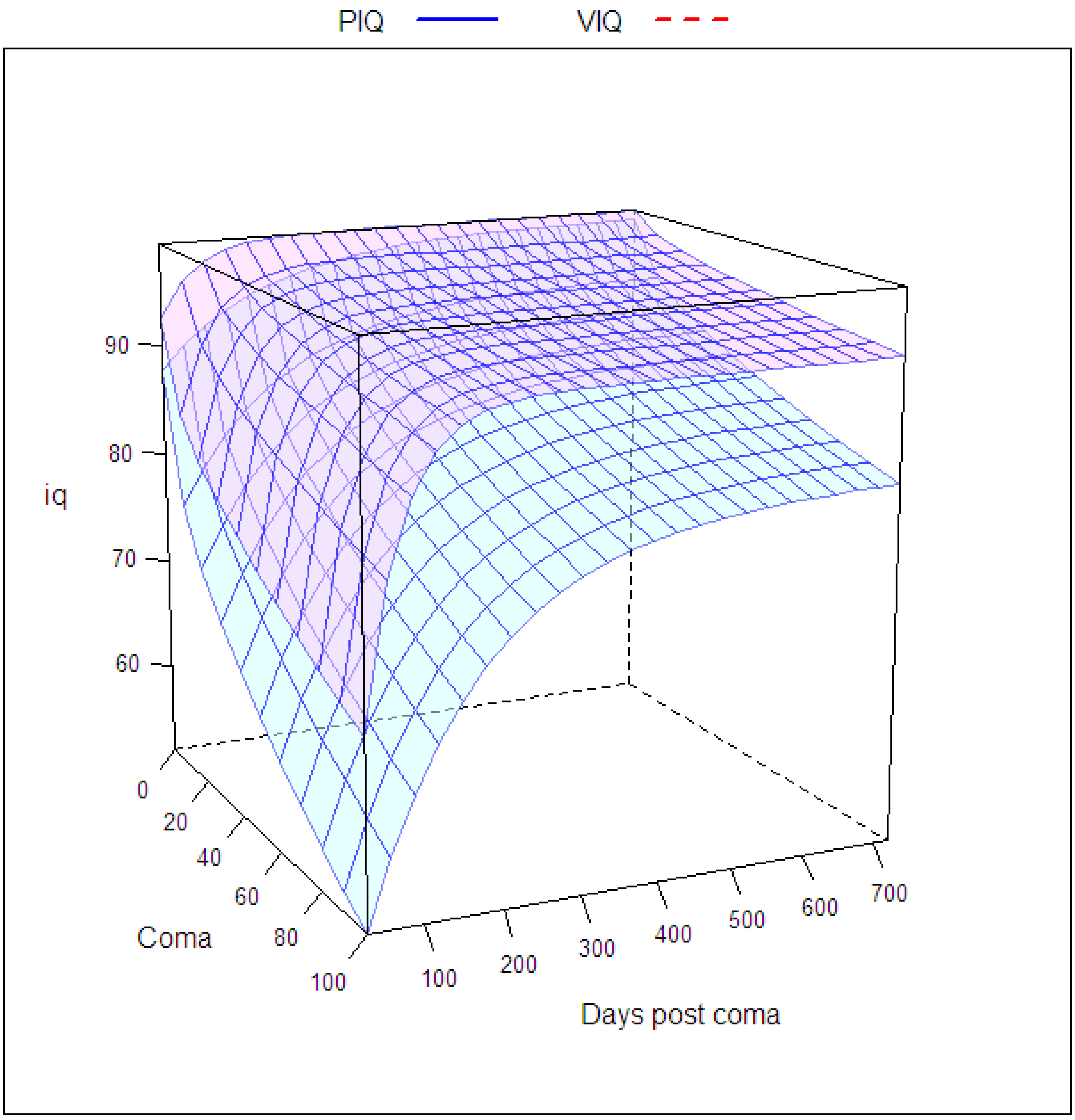
```
wireframe( iq ~ dcoma + dayspc | type, Rbind(predpiq,
         scales = list( arrows = F), col = 'blue',
         xlab = 'Coma',
         ylab = 'Days post coma',
         screen = list( z = -65, x = -75 ))
```

With the 'screen' parameter, you can control the orienta
graph. Here, the z axis is tha vertical axis, the x axis, th
axis in the surface of the screen and y, the horizontal ax
into the screen.

Rotation in the z axis of -65 degrees results in clockwis
+65 degrees from the top and x-axis rotation of -75, tilt
up by 75 degrees.


PIQ
VIQ
iq
90
80
70
60
Coma
0
20
40
60
80
100
Days post coma
100
200
300
400
500
600
700

```
wireframe( iq ~ dcoma + dayspc , Rbind(predpiq, predvi
         groups = type,
         scales = list( arrows = F), col = 'blue',
         xlab = 'Coma',
         ylab = 'Days post coma',
         screen = list( z = -65, x = -75 ),
         auto.key = list(columns=2, lines = T, points
         alpha = .5)
```


PIQ
VIQ
iq
90
80
70
60
0
20
40
60
80
100
Coma
100
200
300
400
500
600
700
Days post coma

See the Non-Linear Lab script for a multivariate model
compares the parameters for PIQ and VIQ.