Chapter 19 Mixed-Effects Models | Chapter 20 Non-Linear Regression

Qianqian Shan

June 9, 2017

The essence in deciding whether a categorical variable should be treated as a fixed effect or random effect:

1. Fixed effects influence only the **mean** of y.
2. Random effects influence only the **variance** of y.

Random effects arise from two contrasting kinds of circumstances:

1. observational studies with hierarchical structure,
2. designed experiments with different spatial or temporal scales.

Fundamental assumptions of linear mixed-effects models:

1. Within group errors are independent with mean 0 , variance σ^2 and are independent of the random effects.
2. The random effects are normally distributed with mean 0 and covariance matrix Ψ.
3. The random effects are independent in different groups.
4. The covariance matrix doesn't depend on the group.

Replication and pseudo-replication

Replicates properties:

1. Independent
2. NOT grouped together in one place as aggregation means that they are not spatially independent
3. Be of appropriate spatial scale
4. Repeated measures are not replicates.

Pseudo-replication occurs when the data has more degrees of freedom than really has:

1. temporal involving repeated measurements from the same individual
2. spatial involving several measurements taken from the same vicinity.

Ways to deal with pseudo-replication:

1. Average away the pseudo-replication and carry out analysis on the means
2. Carry out separate analysis for each time period
3. Use proper time series analysis or mixed effects models.

`lme` and `lmer` functions

- `lme` has separate fixed and random effects specification, $fixed = y\tilde{}1$, $random = \tilde{}1|a/b/c$, meaning that "there are three random effects with c nested within b and b is nested within a", $lme(fixed = y\tilde{}1, randome = \tilde{}1|a/b/c)$.

- `lmer` has fixed and random effects specified togher, $lmer(y\tilde{1} + (1|a/b/c)$.

Best linear unbiased predictors

In mixed-effects models, the correlation between the pseudo replicates within a group causes **shrinkage**, and the best linear unbiased predictor $a_i = (\overline{y_i} - \mu)(\frac{\sigma_a^2}{\sigma_a^2 + \sigma^2/n})$, where σ^2 is the residual variance, σ_a^2 is the between group variance which introduces the correlation between the pseudo replicates within each group.

More details on BLUP later, or refer https://dnett.github.io/S510/21BLUP.pdf.

Designed experiments with different spatial scales: Split plots

If we want to use `anova` to compare mixed models with different fixed effects structurs, we jmust use maximum likelihood: `method = "ML"` for `lme`,REML = FALSE`for`lmer‘.

For more details on `REML`, see https://dnett.github.io/S510/20REML.pdf.

1. If the experiment is balanced and there are no missing values, use `aov` with `Error` term do describe the structure of the spatial pseudoreplication. See Chapter 11.4 for example.
2. If the experiment is not balanced, need to use `lme` or `lmer` for model simplication to estimate the p values of the significant interaction terms.

```
# Linear model for a split-plot experiment

yields <- read.table("splityield.txt", header = TRUE)
attach(yields)
head(yields)
```

```
##   yield block irrigation density fertilizer
## 1    90     A    control     low          N
## 2    95     A    control     low          P
## 3   107     A    control     low         NP
## 4    92     A    control  medium          N
## 5    89     A    control  medium          P
## 6    92     A    control  medium         NP
```

```
library(nlme)

model <- lme(yield ~ irrigation * density * fertilizer,
             random = ~ 1 | block/irrigation/density)
summary(model) # the only significant effect is irrigation
```

```
## Linear mixed-effects model fit by REML
##  Data: NULL
##        AIC      BIC    logLik
##   481.6212 525.3789 -218.8106
##
## Random effects:
##  Formula: ~1 | block
##          (Intercept)
## StdDev: 0.0006619703
##
##  Formula: ~1 | irrigation %in% block
##         (Intercept)
```

```
## StdDev:    1.982465
##
##  Formula: ~1 | density %in% irrigation %in% block
##         (Intercept) Residual
## StdDev:    6.975552 9.292805
##
## Fixed effects: yield ~ irrigation * density * fertilizer
##                                                 Value Std.Error DF
## (Intercept)                                     80.50  5.893741 36
## irrigationirrigated                             31.75  8.335008  3
## densitylow                                       5.50  8.216281 12
## densitymedium                                   14.75  8.216281 12
## fertilizerNP                                     5.50  6.571006 36
## fertilizerP                                      4.50  6.571006 36
## irrigationirrigated:densitylow                 -39.00 11.619576 12
## irrigationirrigated:densitymedium              -22.25 11.619576 12
## irrigationirrigated:fertilizerNP                13.00  9.292805 36
## irrigationirrigated:fertilizerP                  5.50  9.292805 36
## densitylow:fertilizerNP                          3.25  9.292805 36
## densitymedium:fertilizerNP                      -6.75  9.292805 36
## densitylow:fertilizerP                          -5.25  9.292805 36
## densitymedium:fertilizerP                       -5.50  9.292805 36
## irrigationirrigated:densitylow:fertilizerNP      7.75 13.142011 36
## irrigationirrigated:densitymedium:fertilizerNP   3.75 13.142011 36
## irrigationirrigated:densitylow:fertilizerP      20.00 13.142011 36
## irrigationirrigated:densitymedium:fertilizerP    4.00 13.142011 36
##                                                 t-value p-value
## (Intercept)                                    13.658558  0.0000
## irrigationirrigated                             3.809235  0.0318
## densitylow                                      0.669403  0.5159
## densitymedium                                   1.795216  0.0978
## fertilizerNP                                    0.837010  0.4081
## fertilizerP                                     0.684827  0.4978
## irrigationirrigated:densitylow                 -3.356405  0.0057
## irrigationirrigated:densitymedium              -1.914872  0.0796
## irrigationirrigated:fertilizerNP                1.398932  0.1704
## irrigationirrigated:fertilizerP                 0.591856  0.5576
## densitylow:fertilizerNP                         0.349733  0.7286
## densitymedium:fertilizerNP                     -0.726368  0.4723
## densitylow:fertilizerP                         -0.564953  0.5756
## densitymedium:fertilizerP                      -0.591856  0.5576
## irrigationirrigated:densitylow:fertilizerNP     0.589712  0.5591
## irrigationirrigated:densitymedium:fertilizerNP  0.285344  0.7770
## irrigationirrigated:densitylow:fertilizerP      1.521837  0.1368
## irrigationirrigated:densitymedium:fertilizerP   0.304367  0.7626
##  Correlation:
##                                                (Intr) irrgtn dnstyl dnstym
## irrigationirrigated                            -0.707
## densitylow                                     -0.697  0.493
## densitymedium                                  -0.697  0.493  0.500
## fertilizerNP                                   -0.557  0.394  0.400  0.400
## fertilizerP                                    -0.557  0.394  0.400  0.400
## irrigationirrigated:densitylow                  0.493 -0.697 -0.707 -0.354
## irrigationirrigated:densitymedium               0.493 -0.697 -0.354 -0.707
```

```
## irrigationirrigated:fertilizerNP                0.394 -0.557 -0.283 -0.283
## irrigationirrigated:fertilizerP                 0.394 -0.557 -0.283 -0.283
## densitylow:fertilizerNP                         0.394 -0.279 -0.566 -0.283
## densitymedium:fertilizerNP                      0.394 -0.279 -0.283 -0.566
## densitylow:fertilizerP                          0.394 -0.279 -0.566 -0.283
## densitymedium:fertilizerP                       0.394 -0.279 -0.283 -0.566
## irrigationirrigated:densitylow:fertilizerNP    -0.279  0.394  0.400  0.200
## irrigationirrigated:densitymedium:fertilizerNP -0.279  0.394  0.200  0.400
## irrigationirrigated:densitylow:fertilizerP     -0.279  0.394  0.400  0.200
## irrigationirrigated:densitymedium:fertilizerP  -0.279  0.394  0.200  0.400
##                                                frtlNP frtlzP
## irrigationirrigated
## densitylow
## densitymedium
## fertilizerNP
## fertilizerP                                     0.500
## irrigationirrigated:densitylow                 -0.283 -0.283
## irrigationirrigated:densitymedium              -0.283 -0.283
## irrigationirrigated:fertilizerNP               -0.707 -0.354
## irrigationirrigated:fertilizerP                -0.354 -0.707
## densitylow:fertilizerNP                        -0.707 -0.354
## densitymedium:fertilizerNP                     -0.707 -0.354
## densitylow:fertilizerP                         -0.354 -0.707
## densitymedium:fertilizerP                      -0.354 -0.707
## irrigationirrigated:densitylow:fertilizerNP     0.500  0.250
## irrigationirrigated:densitymedium:fertilizerNP  0.500  0.250
## irrigationirrigated:densitylow:fertilizerP      0.250  0.500
## irrigationirrigated:densitymedium:fertilizerP   0.250  0.500
##                                                irrgtnrrgtd:dnstyl
## irrigationirrigated
## densitylow
## densitymedium
## fertilizerNP
## fertilizerP
## irrigationirrigated:densitylow
## irrigationirrigated:densitymedium               0.500
## irrigationirrigated:fertilizerNP                0.400
## irrigationirrigated:fertilizerP                 0.400
## densitylow:fertilizerNP                         0.400
## densitymedium:fertilizerNP                      0.200
## densitylow:fertilizerP                          0.400
## densitymedium:fertilizerP                       0.200
## irrigationirrigated:densitylow:fertilizerNP    -0.566
## irrigationirrigated:densitymedium:fertilizerNP -0.283
## irrigationirrigated:densitylow:fertilizerP     -0.566
## irrigationirrigated:densitymedium:fertilizerP  -0.283
##                                                irrgtnrrgtd:dnstym irr:NP
## irrigationirrigated
## densitylow
## densitymedium
## fertilizerNP
## fertilizerP
## irrigationirrigated:densitylow
## irrigationirrigated:densitymedium
```

```
## irrigationirrigated:fertilizerNP                0.400
## irrigationirrigated:fertilizerP                 0.400              0.500
## densitylow:fertilizerNP                         0.200              0.500
## densitymedium:fertilizerNP                      0.400              0.500
## densitylow:fertilizerP                          0.200              0.250
## densitymedium:fertilizerP                       0.400              0.250
## irrigationirrigated:densitylow:fertilizerNP    -0.283             -0.707
## irrigationirrigated:densitymedium:fertilizerNP -0.566             -0.707
## irrigationirrigated:densitylow:fertilizerP     -0.283             -0.354
## irrigationirrigated:densitymedium:fertilizerP  -0.566             -0.354
##                                                irrg:P dnstyl:NP dnstym:NP
## irrigationirrigated
## densitylow
## densitymedium
## fertilizerNP
## fertilizerP
## irrigationirrigated:densitylow
## irrigationirrigated:densitymedium
## irrigationirrigated:fertilizerNP
## irrigationirrigated:fertilizerP
## densitylow:fertilizerNP                         0.250
## densitymedium:fertilizerNP                      0.250  0.500
## densitylow:fertilizerP                          0.500  0.500     0.250
## densitymedium:fertilizerP                       0.500  0.250     0.500
## irrigationirrigated:densitylow:fertilizerNP    -0.354 -0.707    -0.354
## irrigationirrigated:densitymedium:fertilizerNP -0.354 -0.354    -0.707
## irrigationirrigated:densitylow:fertilizerP     -0.707 -0.354    -0.177
## irrigationirrigated:densitymedium:fertilizerP  -0.707 -0.177    -0.354
##                                                dnstyl:P dnstym:P
## irrigationirrigated
## densitylow
## densitymedium
## fertilizerNP
## fertilizerP
## irrigationirrigated:densitylow
## irrigationirrigated:densitymedium
## irrigationirrigated:fertilizerNP
## irrigationirrigated:fertilizerP
## densitylow:fertilizerNP
## densitymedium:fertilizerNP
## densitylow:fertilizerP
## densitymedium:fertilizerP                       0.500
## irrigationirrigated:densitylow:fertilizerNP    -0.354   -0.177
## irrigationirrigated:densitymedium:fertilizerNP -0.177   -0.354
## irrigationirrigated:densitylow:fertilizerP     -0.707   -0.354
## irrigationirrigated:densitymedium:fertilizerP  -0.354   -0.707
##                                                irrgtnrrgtd:dnstyl:NP
## irrigationirrigated
## densitylow
## densitymedium
## fertilizerNP
## fertilizerP
## irrigationirrigated:densitylow
## irrigationirrigated:densitymedium
```

```
## irrigationirrigated:fertilizerNP
## irrigationirrigated:fertilizerP
## densitylow:fertilizerNP
## densitymedium:fertilizerNP
## densitylow:fertilizerP
## densitymedium:fertilizerP
## irrigationirrigated:densitylow:fertilizerNP
## irrigationirrigated:densitymedium:fertilizerNP  0.500
## irrigationirrigated:densitylow:fertilizerP      0.500
## irrigationirrigated:densitymedium:fertilizerP   0.250
##                                                irrgtnrrgtd:dnstym:NP
## irrigationirrigated
## densitylow
## densitymedium
## fertilizerNP
## fertilizerP
## irrigationirrigated:densitylow
## irrigationirrigated:densitymedium
## irrigationirrigated:fertilizerNP
## irrigationirrigated:fertilizerP
## densitylow:fertilizerNP
## densitymedium:fertilizerNP
## densitylow:fertilizerP
## densitymedium:fertilizerP
## irrigationirrigated:densitylow:fertilizerNP
## irrigationirrigated:densitymedium:fertilizerNP
## irrigationirrigated:densitylow:fertilizerP      0.250
## irrigationirrigated:densitymedium:fertilizerP   0.500
##                                                irrgtnrrgtd:dnstyl:P
## irrigationirrigated
## densitylow
## densitymedium
## fertilizerNP
## fertilizerP
## irrigationirrigated:densitylow
## irrigationirrigated:densitymedium
## irrigationirrigated:fertilizerNP
## irrigationirrigated:fertilizerP
## densitylow:fertilizerNP
## densitymedium:fertilizerNP
## densitylow:fertilizerP
## densitymedium:fertilizerP
## irrigationirrigated:densitylow:fertilizerNP
## irrigationirrigated:densitymedium:fertilizerNP
## irrigationirrigated:densitylow:fertilizerP
## irrigationirrigated:densitymedium:fertilizerP   0.500
##
## Standardized Within-Group Residuals:
##         Min          Q1         Med          Q3         Max
## -2.12362051 -0.37841448 -0.03057736  0.41805005  1.90433187
##
## Number of Observations: 72
## Number of Groups:
##                              block              irrigation %in% block
```

```
##                                 4                                     8
## density %in% irrigation %in% block
##                                24
```

```r
# simplify the model
model <- lme(yield ~ (irrigation + density + fertilizer)^2,
             random = ~ 1 |  block/irrigation/density)
summary(model)
```

```
## Linear mixed-effects model fit by REML
##  Data: NULL
##        AIC      BIC    logLik
##   503.1256 540.2136 -233.5628
##
## Random effects:
##  Formula: ~1 | block
##          (Intercept)
## StdDev: 0.0005567465
##
##  Formula: ~1 | irrigation %in% block
##         (Intercept)
## StdDev:    1.982562
##
##  Formula: ~1 | density %in% irrigation %in% block
##         (Intercept) Residual
## StdDev:    7.041303 9.142696
##
## Fixed effects: yield ~ (irrigation + density + fertilizer)^2
##                                       Value Std.Error DF   t-value p-value
## (Intercept)                         82.47222  5.443438 40 15.150760  0.0000
## irrigationirrigated                 27.80556  7.069256  3  3.933307  0.0293
## densitylow                           0.87500  7.256234 12  0.120586  0.9060
## densitymedium                       13.45833  7.256234 12  1.854727  0.0884
## fertilizerNP                         3.58333  5.278538 40  0.678850  0.5011
## fertilizerP                          0.50000  5.278538 40  0.094723  0.9250
## irrigationirrigated:densitylow     -29.75000  8.800165 12 -3.380618  0.0055
## irrigationirrigated:densitymedium -19.66667  8.800165 12 -2.234807  0.0452
## irrigationirrigated:fertilizerNP    16.83333  5.278538 40  3.189014  0.0028
## irrigationirrigated:fertilizerP     13.50000  5.278538 40  2.557526  0.0144
## densitylow:fertilizerNP              7.12500  6.464862 40  1.102112  0.2770
## densitymedium:fertilizerNP          -4.87500  6.464862 40 -0.754076  0.4552
## densitylow:fertilizerP               4.75000  6.464862 40  0.734741  0.4668
## densitymedium:fertilizerP           -3.50000  6.464862 40 -0.541388  0.5912
##  Correlation:
##                                    (Intr) irrgtn dnstyl dnstym frtlNP
## irrigationirrigated                -0.649
## densitylow                         -0.667  0.377
## densitymedium                      -0.667  0.377  0.500
## fertilizerNP                       -0.485  0.187  0.273  0.273
## fertilizerP                        -0.485  0.187  0.273  0.273  0.500
## irrigationirrigated:densitylow      0.404 -0.622 -0.606 -0.303  0.000
## irrigationirrigated:densitymedium   0.404 -0.622 -0.303 -0.606  0.000
## irrigationirrigated:fertilizerNP    0.242 -0.373  0.000  0.000 -0.500
## irrigationirrigated:fertilizerP     0.242 -0.373  0.000  0.000 -0.250
## densitylow:fertilizerNP             0.297  0.000 -0.445 -0.223 -0.612
```

```
## densitymedium:fertilizerNP         0.297  0.000 -0.223 -0.445 -0.612
## densitylow:fertilizerP             0.297  0.000 -0.445 -0.223 -0.306
## densitymedium:fertilizerP          0.297  0.000 -0.223 -0.445 -0.306
##                                   frtlzP irrgtnrrgtd:dnstyl
## irrigationirrigated
## densitylow
## densitymedium
## fertilizerNP
## fertilizerP
## irrigationirrigated:densitylow     0.000
## irrigationirrigated:densitymedium  0.000  0.500
## irrigationirrigated:fertilizerNP  -0.250  0.000
## irrigationirrigated:fertilizerP   -0.500  0.000
## densitylow:fertilizerNP           -0.306  0.000
## densitymedium:fertilizerNP        -0.306  0.000
## densitylow:fertilizerP            -0.612  0.000
## densitymedium:fertilizerP         -0.612  0.000
##                                   irrgtnrrgtd:dnstym irr:NP irrg:P
## irrigationirrigated
## densitylow
## densitymedium
## fertilizerNP
## fertilizerP
## irrigationirrigated:densitylow
## irrigationirrigated:densitymedium
## irrigationirrigated:fertilizerNP   0.000
## irrigationirrigated:fertilizerP    0.000              0.500
## densitylow:fertilizerNP            0.000              0.000  0.000
## densitymedium:fertilizerNP         0.000              0.000  0.000
## densitylow:fertilizerP             0.000              0.000  0.000
## densitymedium:fertilizerP          0.000              0.000  0.000
##                                   dnstyl:NP dnstym:NP dnstyl:P
## irrigationirrigated
## densitylow
## densitymedium
## fertilizerNP
## fertilizerP
## irrigationirrigated:densitylow
## irrigationirrigated:densitymedium
## irrigationirrigated:fertilizerNP
## irrigationirrigated:fertilizerP
## densitylow:fertilizerNP
## densitymedium:fertilizerNP         0.500
## densitylow:fertilizerP             0.500     0.250
## densitymedium:fertilizerP          0.250     0.500     0.500
##
## Standardized Within-Group Residuals:
##         Min          Q1         Med          Q3         Max
## -2.03880106 -0.51395226  0.02840501  0.62000583  1.95642504
##
## Number of Observations: 72
## Number of Groups:
##                               block              irrigation %in% block
##                                   4                                   8
```

```
## density %in% irrigation %in% block
##                                 24
```

```
# remove the fertilizer by density interaction
model <- lme(yield ~ irrigation * density + irrigation * fertilizer,
             random = ~ 1 | block/irrigation/density)
summary(model)
```

```
## Linear mixed-effects model fit by REML
##  Data: NULL
##        AIC      BIC    logLik
##   519.9035 549.6834 -245.9517
##
## Random effects:
##  Formula: ~1 | block
##          (Intercept)
## StdDev: 0.0005551447
##
##  Formula: ~1 | irrigation %in% block
##         (Intercept)
## StdDev:    1.982613
##
##  Formula: ~1 | density %in% irrigation %in% block
##         (Intercept) Residual
## StdDev:    7.057132 9.105995
##
## Fixed effects: yield ~ irrigation * density + irrigation * fertilizer
##                                       Value Std.Error DF   t-value p-value
## (Intercept)                         82.08333  4.994999 44 16.433103  0.0000
## irrigationirrigated                 27.80556  7.063995  3  3.936236  0.0292
## densitylow                           4.83333  6.222653 12  0.776732  0.4524
## densitymedium                       10.66667  6.222653 12  1.714167  0.1122
## fertilizerNP                         4.33333  3.717507 44  1.165656  0.2500
## fertilizerP                          0.91667  3.717507 44  0.246581  0.8064
## irrigationirrigated:densitylow     -29.75000  8.800161 12 -3.380620  0.0055
## irrigationirrigated:densitymedium  -19.66667  8.800161 12 -2.234808  0.0452
## irrigationirrigated:fertilizerNP    16.83333  5.257349 44  3.201867  0.0025
## irrigationirrigated:fertilizerP     13.50000  5.257349 44  2.567834  0.0137
##  Correlation:
##                                    (Intr) irrgtn dnstyl dnstym frtlNP
## irrigationirrigated                -0.707
## densitylow                         -0.623  0.440
## densitymedium                      -0.623  0.440  0.500
## fertilizerNP                       -0.372  0.263  0.000  0.000
## fertilizerP                        -0.372  0.263  0.000  0.000  0.500
## irrigationirrigated:densitylow      0.440 -0.623 -0.707 -0.354  0.000
## irrigationirrigated:densitymedium   0.440 -0.623 -0.354 -0.707  0.000
## irrigationirrigated:fertilizerNP    0.263 -0.372  0.000  0.000 -0.707
## irrigationirrigated:fertilizerP     0.263 -0.372  0.000  0.000 -0.354
##                                    frtlzP irrgtnrrgtd:dnstyl
## irrigationirrigated
## densitylow
## densitymedium
## fertilizerNP
## fertilizerP
```

```
## irrigationirrigated:densitylow     0.000
## irrigationirrigated:densitymedium  0.000  0.500
## irrigationirrigated:fertilizerNP  -0.354  0.000
## irrigationirrigated:fertilizerP   -0.707  0.000
##                                   irrgtnrrgtd:dnstym irr:NP
## irrigationirrigated
## densitylow
## densitymedium
## fertilizerNP
## fertilizerP
## irrigationirrigated:densitylow
## irrigationirrigated:densitymedium
## irrigationirrigated:fertilizerNP   0.000
## irrigationirrigated:fertilizerP    0.000              0.500
##
## Standardized Within-Group Residuals:
##        Min         Q1        Med         Q3        Max
## -2.4377709 -0.5312543  0.0670152  0.5727941  2.0034253
##
## Number of Observations: 72
## Number of Groups:
##                               block              irrigation %in% block
##                                   4                                  8
## density %in% irrigation %in% block
##                                  24
```

```
# use anova to compare models

model.lme <- lme(yield ~ irrigation * density * fertilizer,
                 random = ~ 1| block/irrigation/density, method = "ML")

model.lme.2 <- update(model.lme,~. - irrigation:density:fertilizer)
anova(model.lme, model.lme.2)
```

```
##             Model df      AIC      BIC    logLik   Test  L.Ratio p-value
## model.lme       1 22 573.5108 623.5974 -264.7554
## model.lme.2     2 18 569.0046 609.9845 -266.5023 1 vs 2 3.493788  0.4788
```

```
# remove two way interaction
model.lme.3 <- update(model.lme.2, ~. - density:fertilizer)
anova(model.lme.3, model.lme.2)
```

```
##             Model df      AIC      BIC    logLik   Test  L.Ratio p-value
## model.lme.3     1 14 565.1933 597.0667 -268.5967
## model.lme.2     2 18 569.0046 609.9845 -266.5023 1 vs 2 4.188774  0.3811
```

```
model.lme.4 <- update(model.lme.3,~. - irrigation:fertilizer)
anova(model.lme.3, model.lme.4) # model3 is better
```

```
##             Model df      AIC      BIC    logLik   Test  L.Ratio p-value
## model.lme.3     1 14 565.1933 597.0667 -268.5967
## model.lme.4     2 12 572.3373 599.6573 -274.1687 1 vs 2 11.14397  0.0038
```

```
model.lme.5 <- update(model.lme.2, ~. - irrigation:density)
anova(model.lme.5, model.lme.2)
```

```
##             Model df      AIC      BIC    logLik   Test  L.Ratio p-value
```

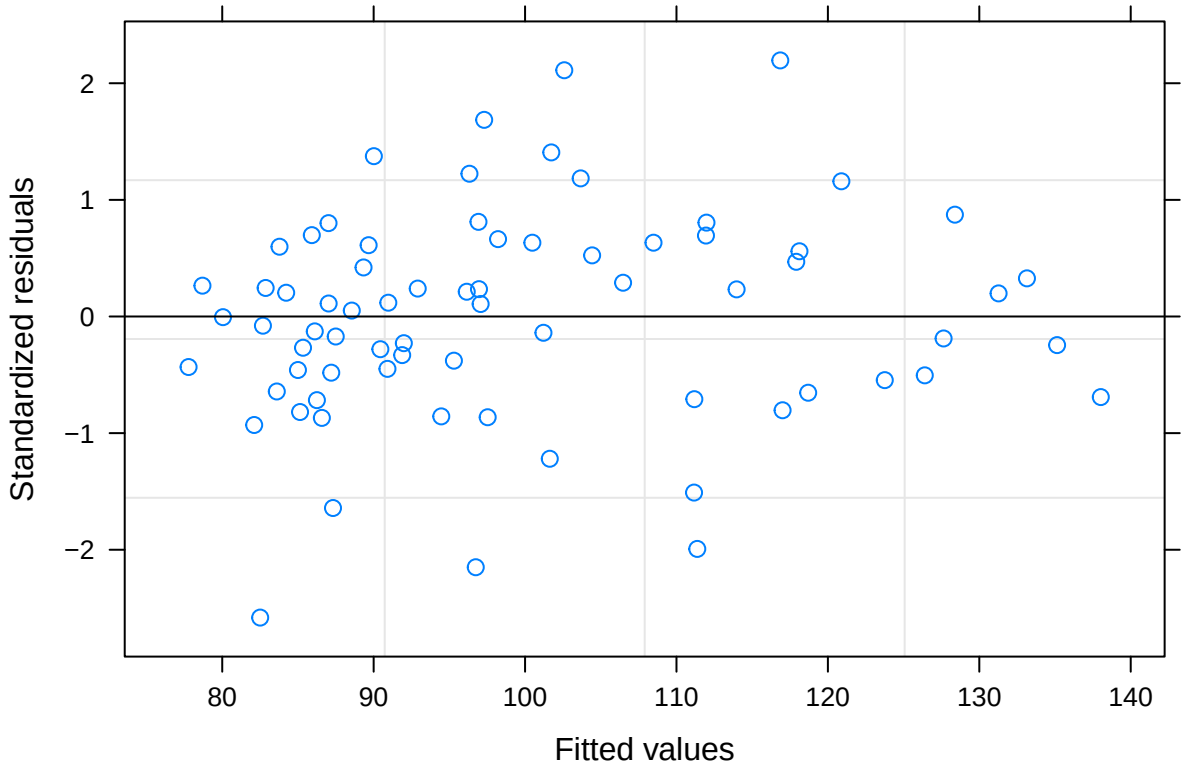
```
## model.lme.5     1 16 576.7134 613.1400 -272.3567
## model.lme.2     2 18 569.0046 609.9845 -266.5023 1 vs 2 11.70883  0.0029
```

```
summary(model.lme.3)
```

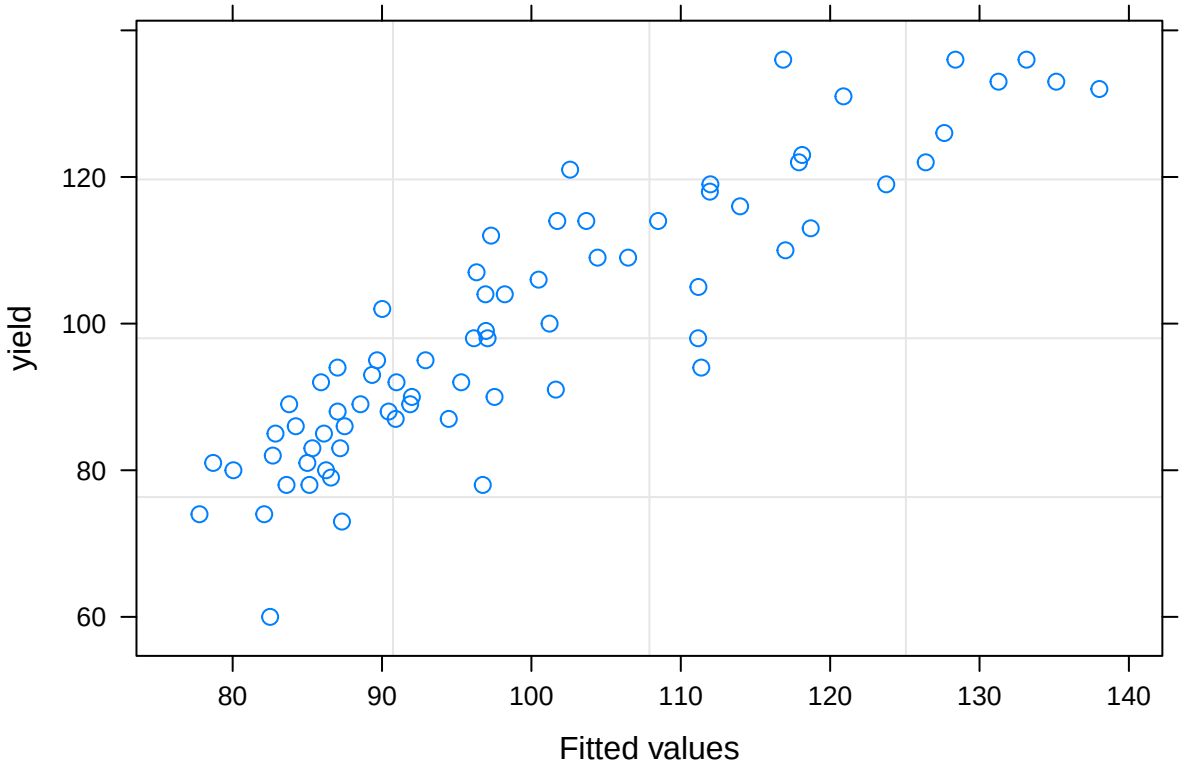
```
## Linear mixed-effects model fit by maximum likelihood
##  Data: NULL
##        AIC      BIC    logLik
##   565.1933 597.0667 -268.5967
##
## Random effects:
##  Formula: ~1 | block
##          (Intercept)
## StdDev: 0.0005335774
##
##  Formula: ~1 | irrigation %in% block
##         (Intercept)
## StdDev:    1.716893
##
##  Formula: ~1 | density %in% irrigation %in% block
##         (Intercept) Residual
## StdDev:    5.722412 8.718327
##
## Fixed effects: yield ~ irrigation + density + fertilizer + irrigation:density +      irrigation:ferti
##                                     Value Std.Error DF   t-value p-value
## (Intercept)                      82.08333  4.756285 44 17.257868  0.0000
## irrigationirrigated              27.80556  6.726402  3  4.133793  0.0257
## densitylow                        4.83333  5.807346 12  0.832279  0.4215
## densitymedium                    10.66667  5.807346 12  1.836754  0.0911
## fertilizerNP                      4.33333  3.835553 44  1.129781  0.2647
## fertilizerP                       0.91667  3.835553 44  0.238992  0.8122
## irrigationirrigated:densitylow  -29.75000  8.212827 12 -3.622382  0.0035
## irrigationirrigated:densitymedium -19.66667  8.212827 12 -2.394628  0.0338
## irrigationirrigated:fertilizerNP  16.83333  5.424290 44  3.103325  0.0033
## irrigationirrigated:fertilizerP   13.50000  5.424290 44  2.488805  0.0167
##  Correlation:
##                                  (Intr) irrgtn dnstyl dnstym frtlNP
## irrigationirrigated              -0.707
## densitylow                       -0.610  0.432
## densitymedium                    -0.610  0.432  0.500
## fertilizerNP                     -0.403  0.285  0.000  0.000
## fertilizerP                      -0.403  0.285  0.000  0.000  0.500
## irrigationirrigated:densitylow    0.432 -0.610 -0.707 -0.354  0.000
## irrigationirrigated:densitymedium  0.432 -0.610 -0.354 -0.707  0.000
## irrigationirrigated:fertilizerNP  0.285 -0.403  0.000  0.000 -0.707
## irrigationirrigated:fertilizerP   0.285 -0.403  0.000  0.000 -0.354
##                                  frtlzP irrgtnrrgtd:dnstyl
## irrigationirrigated
## densitylow
## densitymedium
## fertilizerNP
## fertilizerP
## irrigationirrigated:densitylow    0.000
## irrigationirrigated:densitymedium  0.000  0.500
## irrigationirrigated:fertilizerNP  -0.354  0.000
```

```
## irrigationirrigated:fertilizerP   -0.707  0.000
##                                   irrgtnrrgtd:dnstym irr:NP
## irrigationirrigated
## densitylow
## densitymedium
## fertilizerNP
## fertilizerP
## irrigationirrigated:densitylow
## irrigationirrigated:densitymedium
## irrigationirrigated:fertilizerNP   0.000
## irrigationirrigated:fertilizerP    0.000              0.500
##
## Standardized Within-Group Residuals:
##         Min          Q1         Med          Q3         Max
## -2.58166957 -0.51480864  0.07893418  0.60157089  2.19570827
##
## Number of Observations: 72
## Number of Groups:
##                                block              irrigation %in% block
##                                    4                                  8
## density %in% irrigation %in% block
##                                   24
```

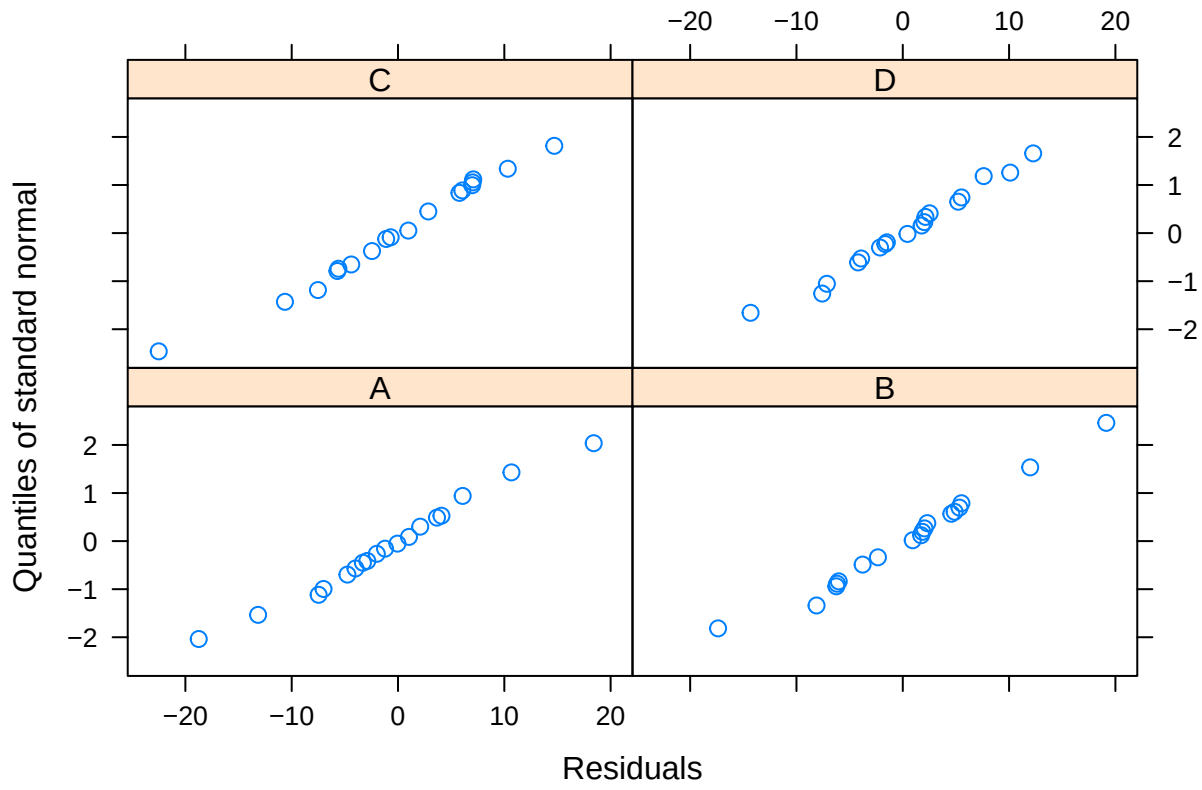
```
plot(model.lme.3)
```




```
plot(model.lme.3, yield ~ fitted(.))
```




```
qqnorm(model.lme.3, ~ resid(.) | block) # close to normal distribution
```




```
# do the analysis by lmer
library(lme4)
```

```
## Loading required package: Matrix
##
## Attaching package: 'lme4'
## The following object is masked from 'package:nlme':
##
##     lmList
```

```
b <- block
bi <- block:irrigation

bid <- block:irrigation:density

model1 <- lmer(yield ~ irrigation * density * fertilizer
               + (1|b) + (1|bi) + (1|bid), REML = FALSE)
print(model1, cor = F) # switch off the matrix of correlations for the fixed effects
```

```
## Linear mixed model fit by maximum likelihood  ['lmerMod']
## Formula: yield ~ irrigation * density * fertilizer + (1 | b) + (1 | bi) +
##     (1 | bid)
##       AIC       BIC    logLik  deviance  df.resid
##  573.5108  623.5974 -264.7554  529.5108        50
## Random effects:
##  Groups   Name        Std.Dev.
##  bid      (Intercept) 6.041
##  bi       (Intercept) 1.717
##  b        (Intercept) 0.000
##  Residual             8.048
## Number of obs: 72, groups:  bid, 24; bi, 8; b, 4
## Fixed Effects:
##                                     (Intercept)
##                                           80.50
##                              irrigationirrigated
##                                           31.75
##                                       densitylow
##                                            5.50
##                                    densitymedium
##                                           14.75
##                                     fertilizerNP
##                                            5.50
##                                     fertilizerP
##                                            4.50
##                  irrigationirrigated:densitylow
##                                          -39.00
##                irrigationirrigated:densitymedium
##                                          -22.25
##                 irrigationirrigated:fertilizerNP
##                                           13.00
##                  irrigationirrigated:fertilizerP
##                                            5.50
##                         densitylow:fertilizerNP
##                                            3.25
##                       densitymedium:fertilizerNP
##                                           -6.75
##                          densitylow:fertilizerP
```

```
##                                          -5.25
##                       densitymedium:fertilizerP
##                                          -5.50
##    irrigationirrigated:densitylow:fertilizerNP
##                                           7.75
## irrigationirrigated:densitymedium:fertilizerNP
##                                           3.75
##     irrigationirrigated:densitylow:fertilizerP
##                                          20.00
##  irrigationirrigated:densitymedium:fertilizerP
##                                           4.00
```

```
detach(yields)
```

Mixed-effects models with temporal pseudoreplication

When the random effect(for example, week) is continuous, we should use '$random = \sim week|plant$ instead of $random = \sim 1|week$, while the latter used for categorical random effects.

```
results <- read.table("fertilizer.txt", header = TRUE)
attach(results)
head(results)
```

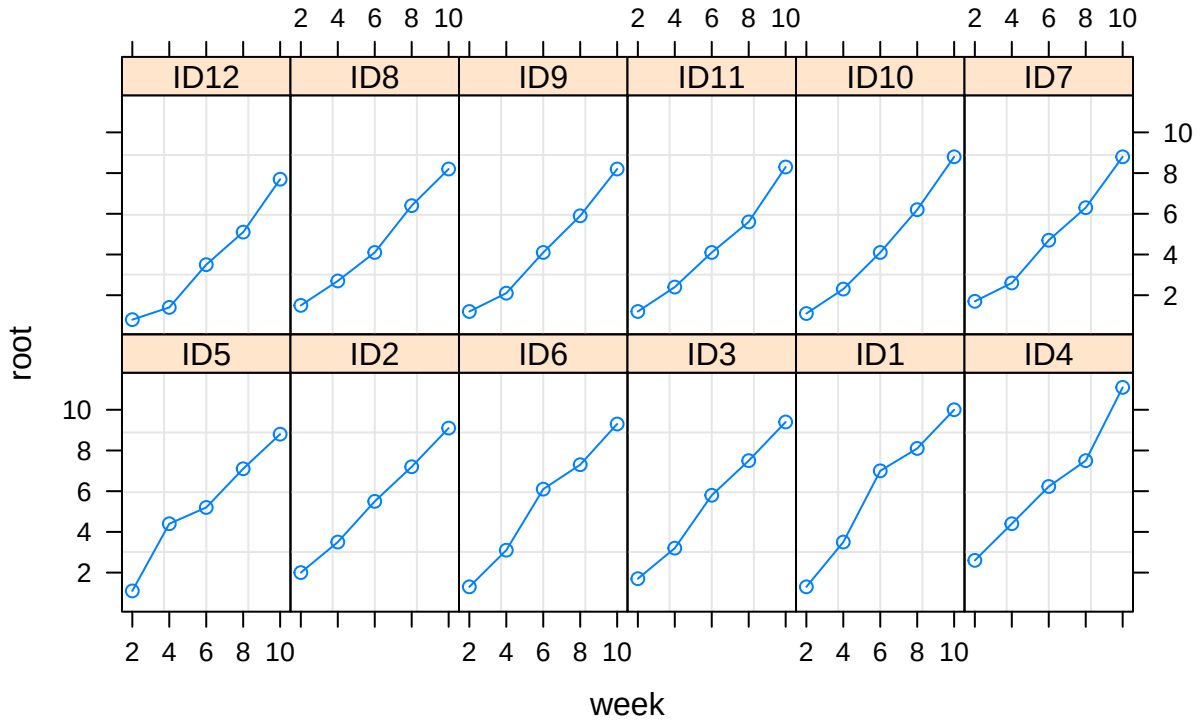
```
##   root week plant fertilizer
## 1  1.3    2   ID1      added
## 2  3.5    4   ID1      added
## 3  7.0    6   ID1      added
## 4  8.1    8   ID1      added
## 5 10.0   10   ID1      added
## 6  2.0    2   ID2      added
```

```
# root is y


library(nlme)
library(lattice)

# trellis plotting,
# convert dataframe into groupedData object,
# specify the nesting structure
# indicate the fixed effect by defining "fertilizer" as outer to this nesting(fixed effect)

# week is random effects
results <- groupedData(root ~ week | plant, outer = ~ fertilizer, results)
str(results)
```

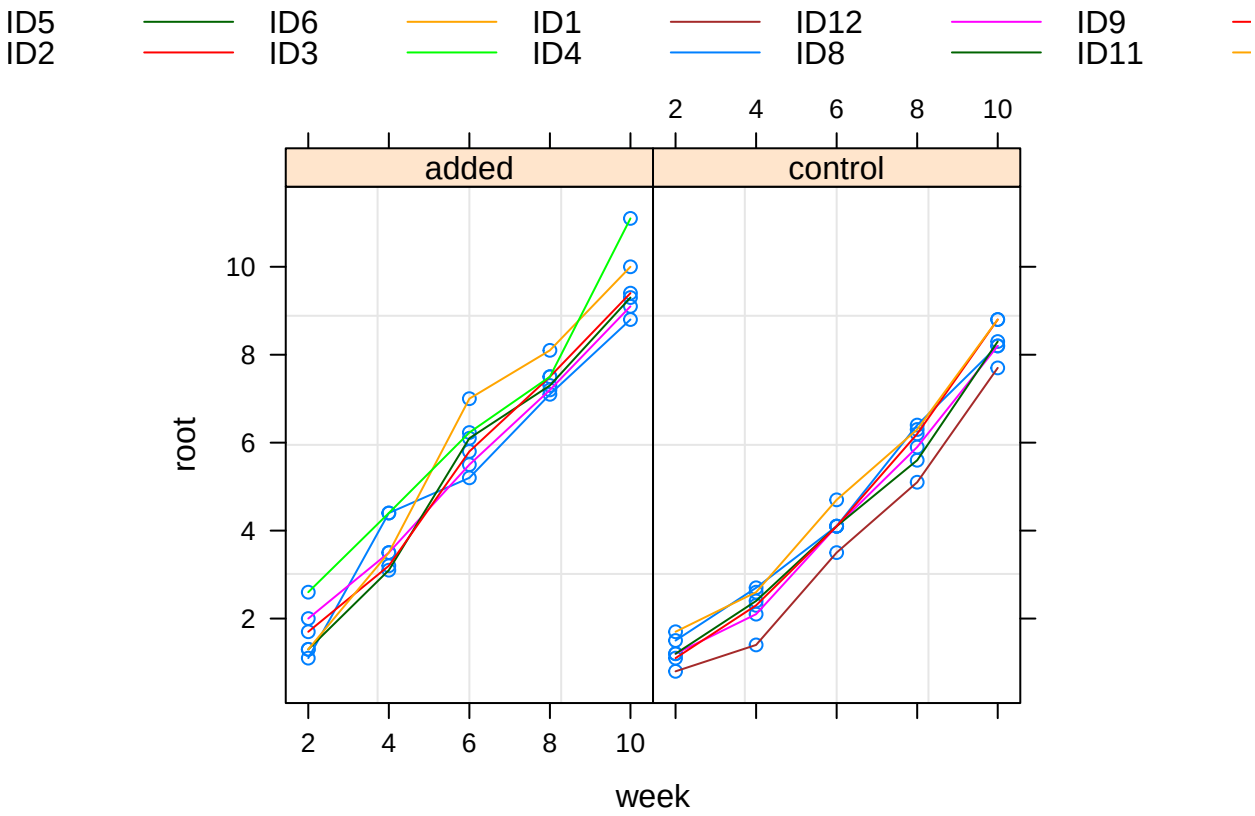
```
## Classes 'nfnGroupedData', 'nfGroupedData', 'groupedData' and 'data.frame':   60 obs. of  4 variables
##  $ root      : num  1.3 3.5 7 8.1 10 2 3.5 5.5 7.2 9.1 ...
##  $ week      : int  2 4 6 8 10 2 4 6 8 10 ...
##  $ plant     : Ord.factor w/ 12 levels "ID5"<"ID2"<"ID6"<..: 5 5 5 5 5 2 2 2 2 2 ...
##  $ fertilizer: Factor w/ 2 levels "added","control": 1 1 1 1 1 1 1 1 1 1 ...
##  - attr(*, "formula")=Class 'formula'  language root ~ week | plant
##   .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##  - attr(*, "outer")=Class 'formula'  language ~fertilizer
##   .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
```

```
##  - attr(*, "FUN")=function (x)
##  - attr(*, "order.groups")= logi TRUE
```

```
plot(results)
```




```
plot(results, outer = TRUE)
```



```r
# linear mixed effects using REML
model <- lme(root ~ fertilizer, random = ~week|plant)
summary(model)
```

```
## Linear mixed-effects model fit by REML
##  Data: NULL
##        AIC      BIC    logLik
##   171.0236 183.3863 -79.51181
##
## Random effects:
##  Formula: ~week | plant
##  Structure: General positive-definite, Log-Cholesky parametrization
##             StdDev    Corr
## (Intercept) 2.8639832 (Intr)
## week        0.9369412 -0.999
## Residual    0.4966308
##
## Fixed effects: root ~ fertilizer
##                       Value Std.Error DF   t-value p-value
## (Intercept)        2.799710 0.1438367 48 19.464500   0e+00
## fertilizercontrol -1.039383 0.2034158 10 -5.109644   5e-04
##  Correlation:
##                   (Intr)
## fertilizercontrol -0.707
##
## Standardized Within-Group Residuals:
##        Min         Q1        Med         Q3        Max
## -1.9928118 -0.6586834 -0.1004301  0.6949714  2.0225381
##
## Number of Observations: 60
## Number of Groups: 12
```

```r
# one-way anova on non-pseudoreplicated data
model2 <- aov(root ~ fertilizer, subset =(week == 10))
summary(model2)
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## fertilizer   1  4.941   4.941   11.49 0.0069 **
## Residuals   10  4.302   0.430
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
summary.lm(model2)
```

```
##
## Call:
## aov(formula = root ~ fertilizer, subset = (week == 10))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.8167 -0.3667 -0.1333  0.4042  1.4833
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)         9.6167     0.2678  35.915 6.65e-12 ***
```

```
## fertilizercontrol  -1.2833     0.3787  -3.389   0.0069 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6559 on 10 degrees of freedom
## Multiple R-squared:  0.5346, Adjusted R-squared:  0.488
## F-statistic: 11.49 on 1 and 10 DF,  p-value: 0.006897
```

```
# the above two models are slightly different
# lm/aov estimates linear models by maximum likelihood estimates of the parameters based on arithmetic
# lme use BLUP estimates

detach(results)
```

Times series analysis in mixed-effects models

- Use `ACF` to check the autocorrelation structure of residuals.
- Model autocorrelation structure with standard `corStruct` classes, see Chapter 26.6 for more details(page 863).

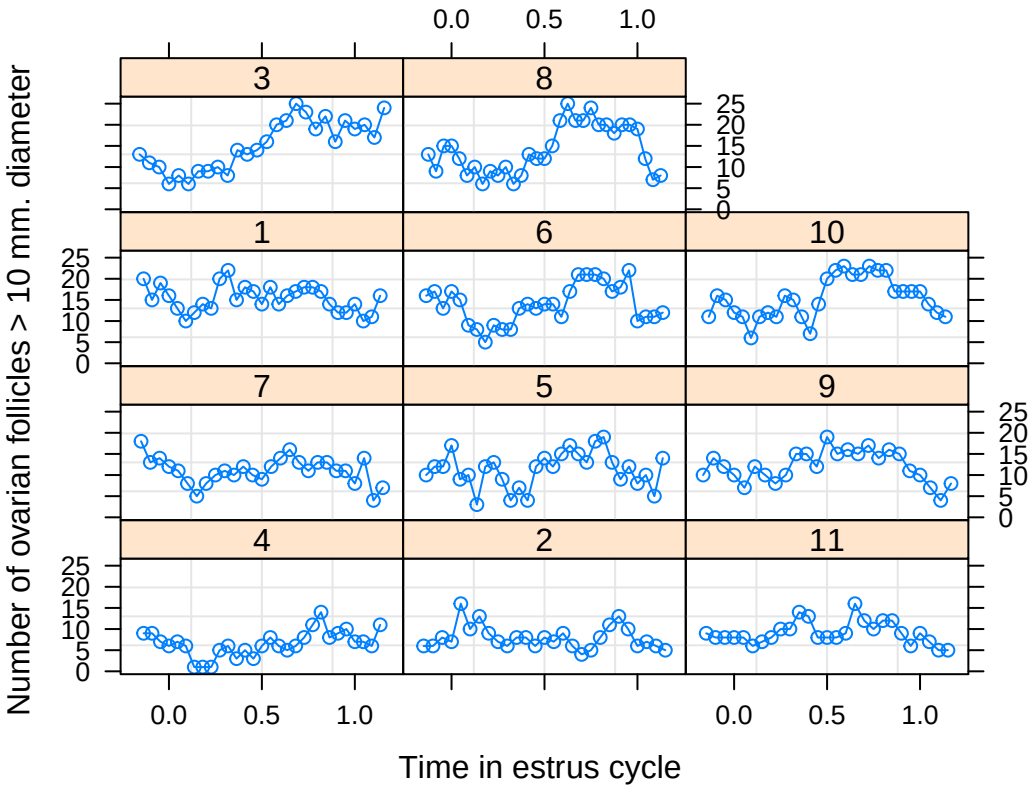
```
data(Ovary)
attach(Ovary)
names(Ovary)
```

```
## [1] "Mare"      "Time"      "follicles"
```

```
str(Ovary)
```

```
## Classes 'nfnGroupedData', 'nfGroupedData', 'groupedData' and 'data.frame':   308 obs. of  3 variables
##  $ Mare     : Ord.factor w/ 11 levels "4"<"2"<"11"<"7"<..: 7 7 7 7 7 7 7 7 7 7 ...
##  $ Time     : num  -0.1364 -0.0909 -0.0455 0 0.0455 ...
##  $ follicles: num  20 15 19 16 13 10 12 14 13 20 ...
##  - attr(*, "formula")=Class 'formula'  language follicles ~ Time | Mare
##   .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##  - attr(*, "labels")=List of 2
##   ..$ x: chr "Time in estrus cycle"
##   ..$ y: chr "Number of ovarian follicles > 10 mm. diameter"
```

```
# follicles is y

plot(Ovary)
```

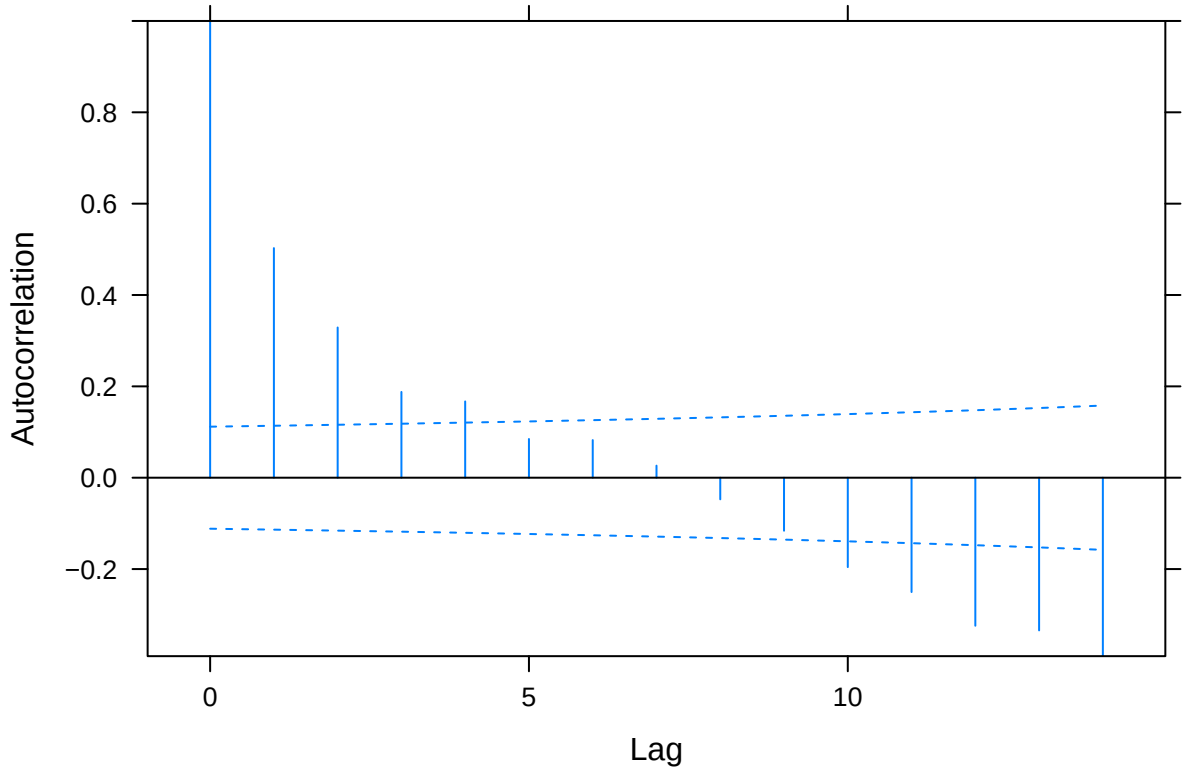



```
# fit model based on prior knowledge and estimate the residuals correlation structure
model <- lme(follicles ~ sin(2 * pi * Time) + cos(2 * pi * Time),
             data = Ovary, random = ~ 1| Mare)
summary(model)
```

```
## Linear mixed-effects model fit by REML
##  Data: Ovary
##        AIC      BIC    logLik
##   1669.36 1687.962 -829.6802
##
## Random effects:
##  Formula: ~1 | Mare
##         (Intercept) Residual
## StdDev:    3.041344 3.400466
##
## Fixed effects: follicles ~ sin(2 * pi * Time) + cos(2 * pi * Time)
##                         Value Std.Error  DF    t-value p-value
## (Intercept)        12.182244 0.9390009 295  12.973623  0.0000
## sin(2 * pi * Time) -3.339612 0.2894013 295 -11.539727  0.0000
## cos(2 * pi * Time) -0.862422 0.2715987 295  -3.175353  0.0017
##  Correlation:
##                    (Intr) s(*p*T
## sin(2 * pi * Time)  0.00
## cos(2 * pi * Time) -0.06   0.00
##
## Standardized Within-Group Residuals:
##        Min         Q1        Med         Q3        Max
## -2.4500138 -0.6721813 -0.1349236  0.5922957  3.5506618
##
```

```
## Number of Observations: 308
## Number of Groups: 11
```

```
plot(ACF(model), alpha = 0.05) # alpha = 0.05 shows the 95% critical lines
```



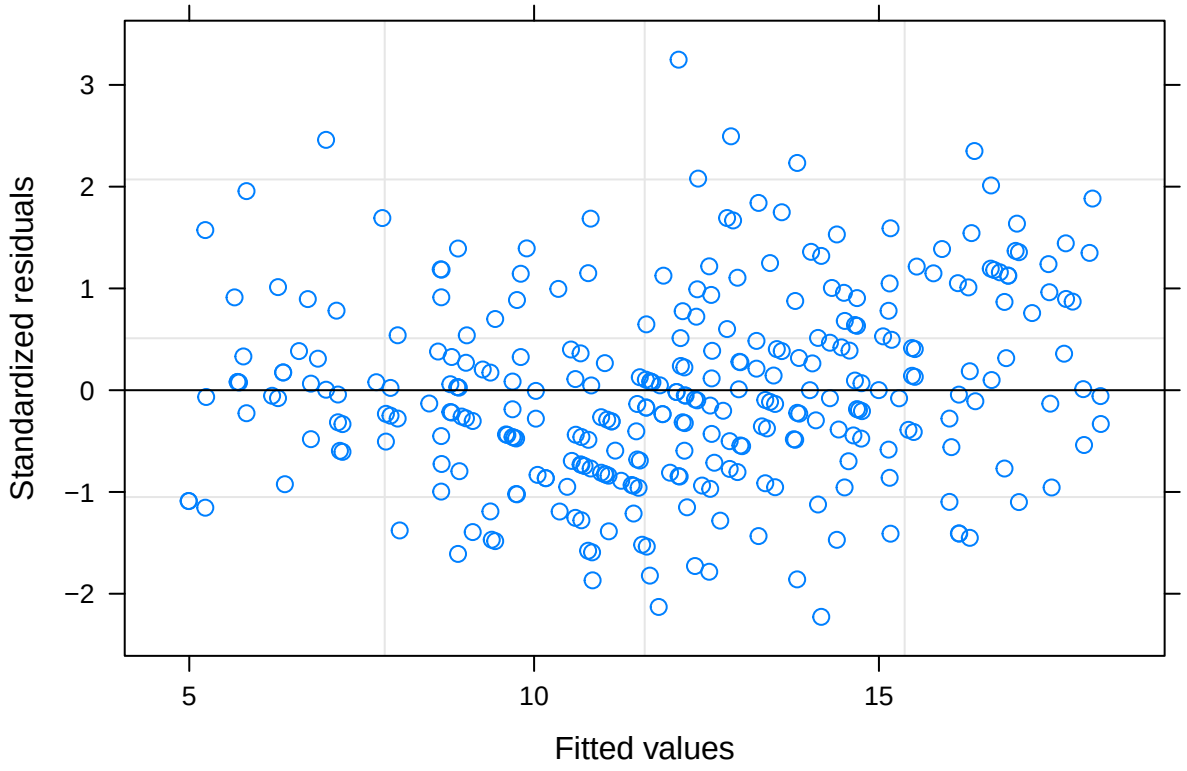

```
# assume the first two lags have non zero correlations
# moving average
model2 <- update(model, correlation = corARMA(q = 2))
anova(model, model2) # better
```

```
##        Model df      AIC      BIC    logLik   Test L.Ratio p-value
## model      1  5 1669.360 1687.962 -829.6802
## model2     2  7 1574.895 1600.937 -780.4476 1 vs 2 98.4652  <.0001
```

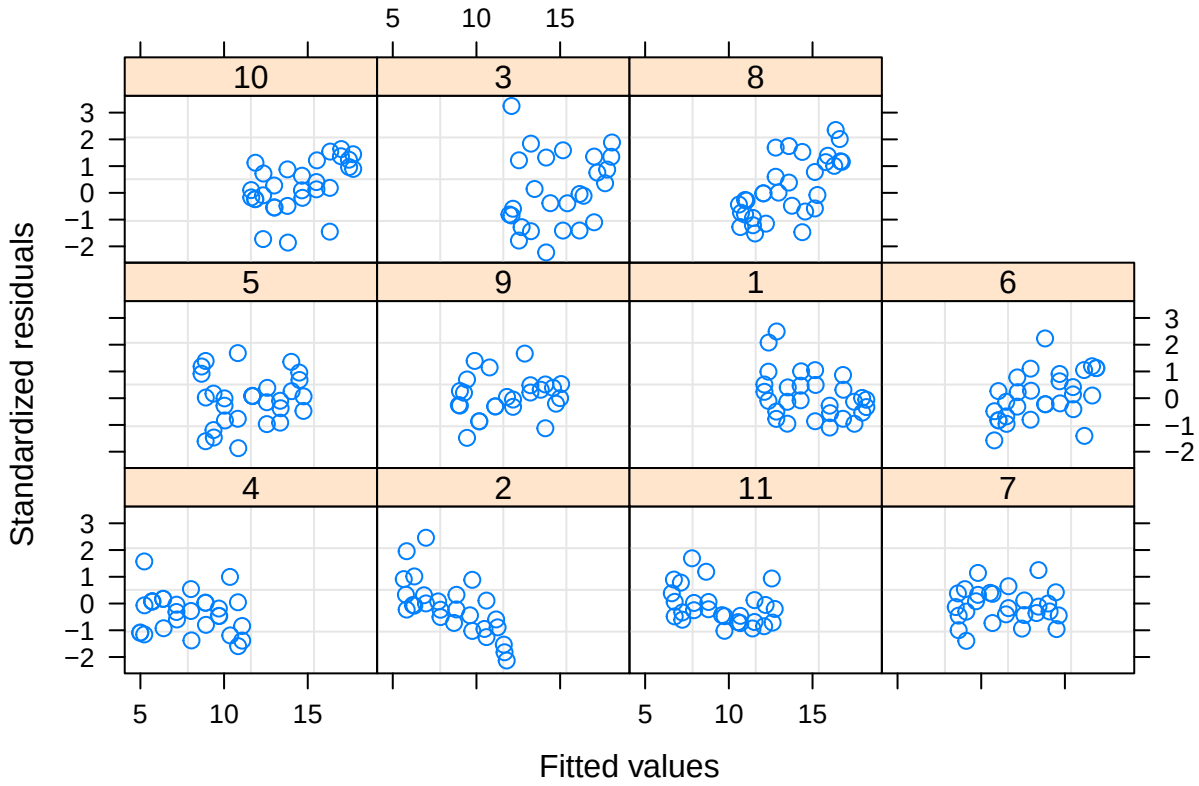
```
# fit first order autoregressive model
model3 <- update(model2, correlation = corAR1())
anova(model2, model3) # better than model2
```

```
##        Model df      AIC      BIC    logLik   Test L.Ratio p-value
## model2     1  7 1574.895 1600.937 -780.4476
## model3     2  6 1562.447 1584.769 -775.2233 1 vs 2 10.4484  0.0012
```

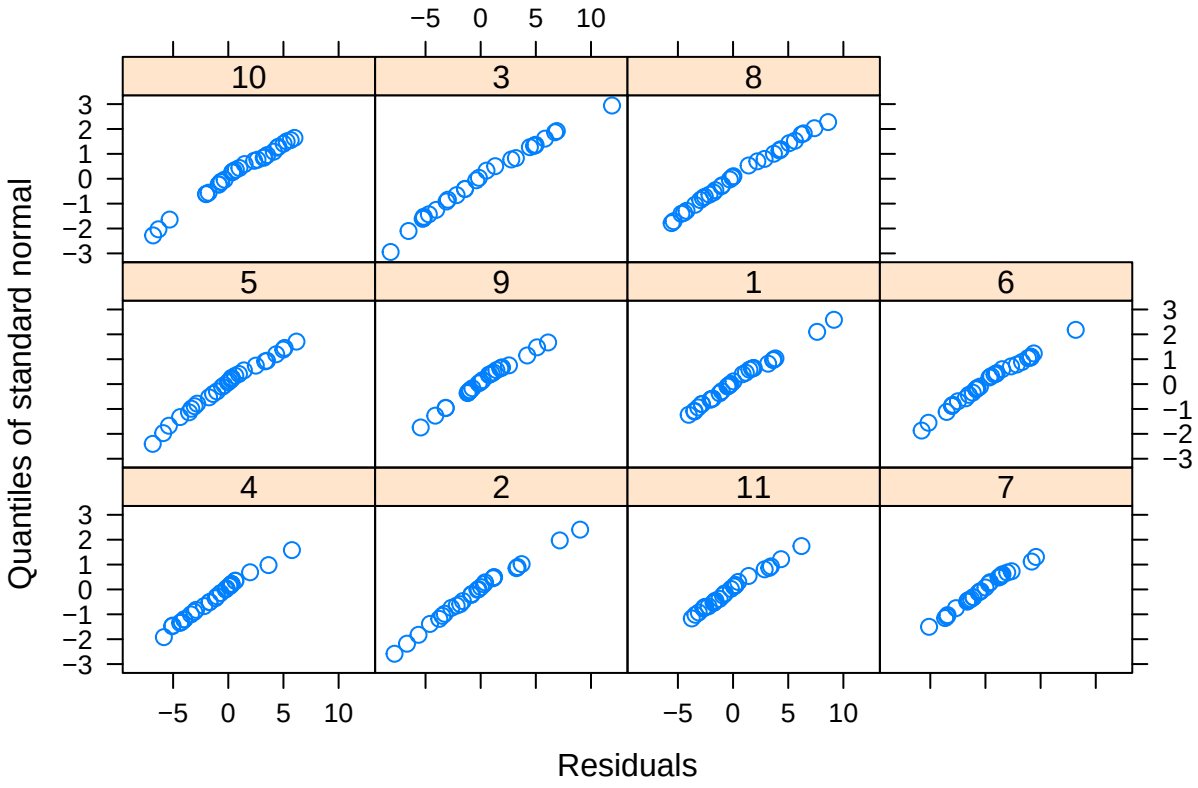
```
# error checking for model3
plot(model3)
```




```
# plot by mare
plot(model3, resid(., type = "p") ~ fitted(.) | Mare) # OK
```



```
# check normal error assumption
qqnorm(model3, ~ resid(.) | Mare) # OK
```




```
detach(Ovary)
```

Random effects in designed experiments

Re-analyze the data in Chapter 11.4, instead of using `aov` and `Error`, we specify linear mixed effects model here.

```
dd <- read.table("rats.txt", header = TRUE)
attach(dd)
head(dd)
```

```
##   Glycogen Treatment Rat Liver
## 1      131         1   1     1
## 2      130         1   1     1
## 3      131         1   1     2
## 4      125         1   1     2
## 5      136         1   1     3
## 6      142         1   1     3
```

```
# Glycogen is y
str(dd)
```

```
## 'data.frame':    36 obs. of  4 variables:
##  $ Glycogen : int  131 130 131 125 136 142 150 148 140 143 ...
##  $ Treatment: int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Rat      : int  1 1 1 1 1 1 2 2 2 2 ...
##  $ Liver    : int  1 1 2 2 3 3 1 1 2 2 ...
```

```
# convert covariates into factors
Treatment <- factor(Treatment)
Liver <- factor(Liver)
Rat <- factor(Rat)

# unique factor levels for each rat and each liver bit
rat <- Treatment:Rat
rat
```

```
##  [1] 1:1 1:1 1:1 1:1 1:1 1:1 1:2 1:2 1:2 1:2 1:2 1:2 2:1 2:1 2:1 2:1 2:1
## [18] 2:1 2:2 2:2 2:2 2:2 2:2 2:2 2:2 3:1 3:1 3:1 3:1 3:1 3:1 3:2 3:2 3:2 3:2
## [35] 3:2 3:2
## Levels: 1:1 1:2 2:1 2:2 3:1 3:2
```

```
str(rat)
```

```
##  Factor w/ 6 levels "1:1","1:2","2:1",..: 1 1 1 1 1 1 2 2 2 2 ...
```

```
liver <- Treatment:Rat:Liver
liver
```

```
##  [1] 1:1:1 1:1:1 1:1:2 1:1:2 1:1:3 1:1:3 1:2:1 1:2:1 1:2:2 1:2:2 1:2:3
## [12] 1:2:3 2:1:1 2:1:1 2:1:2 2:1:2 2:1:3 2:1:3 2:2:1 2:2:1 2:2:2 2:2:2
## [23] 2:2:3 2:2:3 3:1:1 3:1:1 3:1:2 3:1:2 3:1:3 3:1:3 3:2:1 3:2:1 3:2:2
## [34] 3:2:2 3:2:3 3:2:3
## 18 Levels: 1:1:1 1:1:2 1:1:3 1:2:1 1:2:2 1:2:3 2:1:1 2:1:2 2:1:3 ... 3:2:3
```

```
# fit the model
library(lme4)
model <- lmer(Glycogen ~ Treatment + (1 | rat) + (1 | liver))
summary(model)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: Glycogen ~ Treatment + (1 | rat) + (1 | liver)
##
## REML criterion at convergence: 219.6
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -1.48212 -0.47263  0.03062  0.42934  1.82935
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  liver    (Intercept) 14.17    3.764
##  rat      (Intercept) 36.06    6.005
##  Residual             21.17    4.601
## Number of obs: 36, groups:  liver, 18; rat, 6
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)  140.500      4.707  29.848
## Treatment2    10.500      6.657   1.577
## Treatment3    -5.333      6.657  -0.801
##
## Correlation of Fixed Effects:
##            (Intr) Trtmn2
## Treatment2 -0.707
```

```
## Treatment3 -0.707  0.500
```

```
# express variance components in percentages
vars <- c(14.167, 36.065, 21.167)
100 * vars/sum(vars)
```

```
## [1] 19.84201 50.51191 29.64607
```

```
# 19.8% is between liver bits within rats
# 29.6% between readings within liver bits within rats
detach(dd)
```

Regression in mixed effects models

1. Use `lmList` to fit lots of linear regression models
2. Use `lme` to fit one mixed-effects model

```
yields <- read.table("farms.txt", header = TRUE)
attach(yields)

names(yields)
```

```
## [1] "N"    "size" "farm"
```

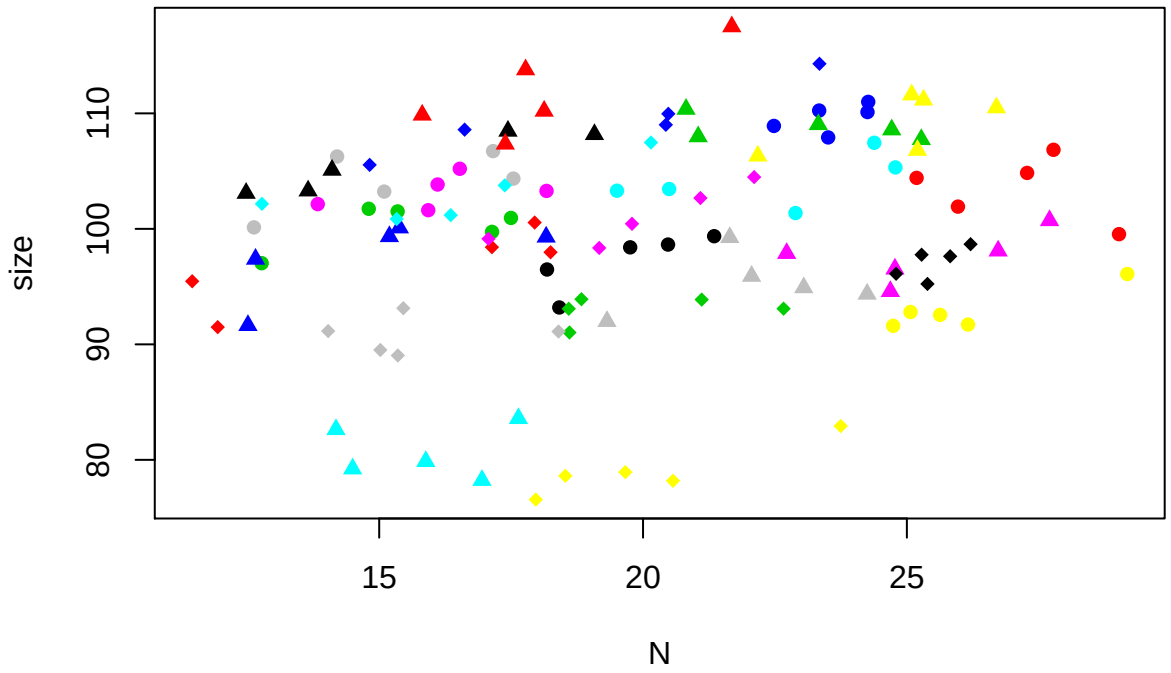
```
str(yields)
```

```
## 'data.frame':    120 obs. of  3 variables:
##  $ N   : num  18.2 20.5 21.3 18.4 19.8 ...
##  $ size: num  96.5 98.6 99.4 93.2 98.4 ...
##  $ farm: int  1 1 1 1 1 2 2 2 2 2 ...
```

```
# size is y
table(farm)
```

```
## farm
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
##  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
```

```
plot(N, size, pch = rep(16:18, each = 40), col = farm)
```




```
# fit  a set of linear models
linear.models <- lmList(size ~ N | farm, data = yields)

coef(linear.models) # intercepts and slopes vary a lot
```

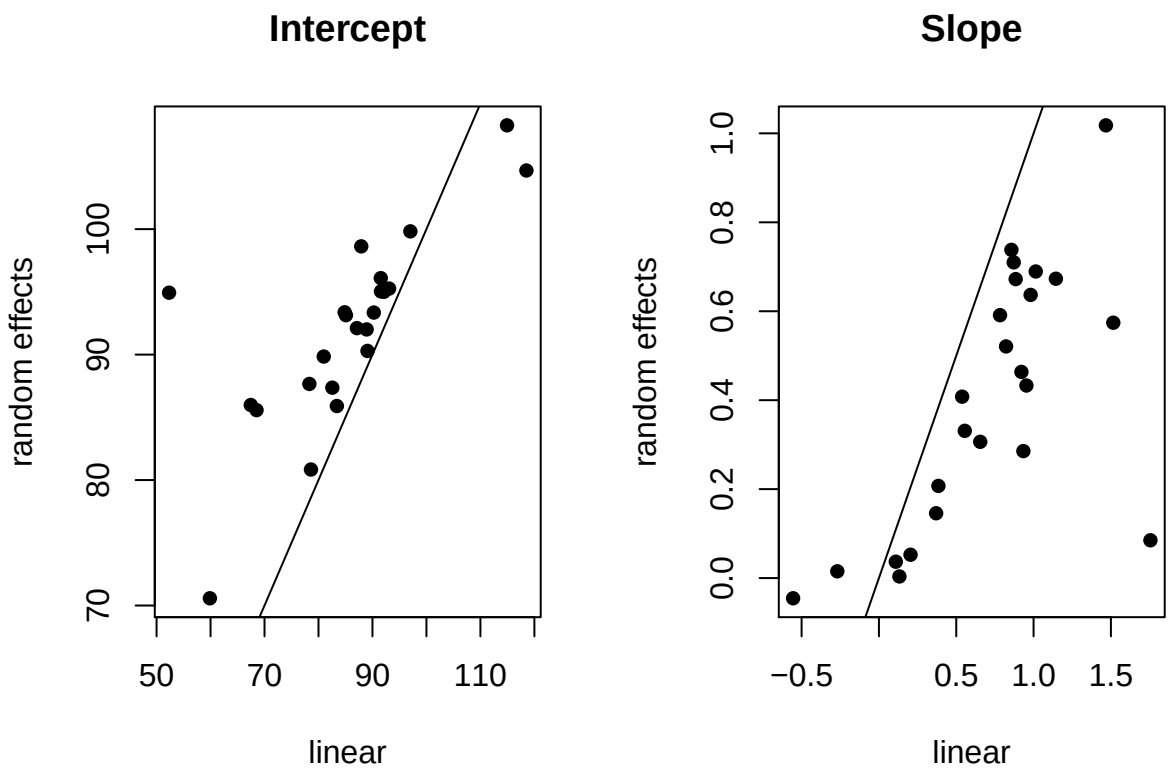
```
##    (Intercept)          N
## 1     67.46260  1.5153805
## 2    118.52443 -0.5550273
## 3     91.58055  0.5551292
## 4     87.92259  0.9212662
## 5     92.12023  0.5380276
## 6     97.01996  0.3845431
## 7     68.52117  0.9339957
## 8     91.54383  0.8220482
## 9     92.04667  0.8842662
## 10    85.08964  1.4676459
## 11   114.93449 -0.2689370
## 12    82.56263  1.0138488
## 13    78.60940  0.1324811
## 14    80.97221  0.6551149
## 15    84.85382  0.9809902
## 16    87.12280  0.3699154
## 17    52.31711  1.7555136
## 18    83.40400  0.8715070
## 19    88.91675  0.2043755
## 20    93.08216  0.8567066
## 21    90.24868  0.7830692
## 22    78.30970  1.1441291
## 23    59.88093  0.9536750
## 24    89.07963  0.1091016
```

```
# fit mixed effects model specified entirely in terms of random effects
library(nlme)
```

```
random.model <- lme(size ~ 1, random = ~ N | farm)
coef(random.model) # less extreme
```

```
##    (Intercept)            N
## 1     85.98140  0.574205215
## 2    104.67367 -0.045401680
## 3     95.03442  0.331080803
## 4     98.62680  0.463579414
## 5     95.00270  0.407906089
## 6     99.82294  0.207203520
## 7     85.57345  0.285520416
## 8     96.09462  0.520896262
## 9     95.22186  0.672262751
## 10    93.14158  1.017995381
## 11   108.27201  0.015213518
## 12    87.36387  0.689406279
## 13    80.83933  0.003617362
## 14    89.84309  0.306402228
## 15    93.37051  0.636778490
## 16    92.10914  0.145772152
## 17    94.93395  0.084935395
## 18    85.90160  0.709943195
## 19    92.00628  0.052486034
## 20    95.26296  0.738029262
## 21    93.35069  0.591151820
## 22    87.66161  0.673119132
## 23    70.57826  0.432994136
## 24    90.29151  0.036747243
```

```
# plot the intercepts and slopes from the two models
mm <- coef(random.model)
ll <- coef(linear.models)

par(mfrow=c(1,2))
plot(ll[, 1], mm[, 1], pch = 16, xlab = "linear",
     ylab = "random effects", main = "Intercept")
abline(0, 1)
plot(ll[, 2], mm[, 2], pch = 16, xlab = "linear",
     ylab = "random effects", main = "Slope")
abline(0,1)
```




```
par(mfrow = c(1, 1))


# fit mixed model with both fixed effects and random effects
# use method = "ML" for model comparisons

farm <- factor(farm)
mixed.model1 <- lme(size ~ N * farm, random = ~ 1 | farm, method = "ML")
mixed.model2 <- lme(size ~ N + farm, random = ~ 1 | farm, method = "ML")
mixed.model3 <- lme(size ~ N, random = ~ 1 | farm, method = "ML")
mixed.model4 <- lme(size ~ 1, random = ~ 1 | farm, method = "ML")
anova(mixed.model1, mixed.model2, mixed.model3, mixed.model4)
```

```
##              Model df      AIC      BIC    logLik   Test   L.Ratio p-value
## mixed.model1     1 50 542.9035 682.2781 -221.4518
## mixed.model2     2 27 524.2971 599.5594 -235.1486 1 vs 2  27.39359  0.2396
## mixed.model3     3  4 614.3769 625.5269 -303.1885 2 vs 3 136.07981  <.0001
## mixed.model4     4  3 658.0058 666.3683 -326.0029 3 vs 4  45.62892  <.0001
```

```
# model2 is selected


# do analysis of variance
model <- lm(size ~ N * factor(farm))
summary(model)
```

```
##
## Call:
## lm(formula = size ~ N * factor(farm))
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -3.6077 -1.2947  0.0479  1.0732  4.1297
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       67.46260   14.43749   4.673 1.35e-05 ***
## N                  1.51538    0.73395   2.065   0.0426 *
## factor(farm)2     51.06183   22.86930   2.233   0.0287 *
## factor(farm)3     24.11794   16.54029   1.458   0.1492
## factor(farm)4     20.45999   34.59610   0.591   0.5561
## factor(farm)5     24.65762   17.29578   1.426   0.1583
## factor(farm)6     29.55736   17.74007   1.666   0.1000
## factor(farm)7      1.05856   20.53771   0.052   0.9590
## factor(farm)8     24.08122   16.23722   1.483   0.1424
## factor(farm)9     24.58407   15.45967   1.590   0.1162
## factor(farm)10    17.62703   16.68467   1.056   0.2943
## factor(farm)11    47.47189   18.24214   2.602   0.0112 *
## factor(farm)12    15.10002   15.77085   0.957   0.3415
## factor(farm)13    11.14680   17.82896   0.625   0.5338
## factor(farm)14    13.50961   19.36739   0.698   0.4877
## factor(farm)15    17.39122   20.74850   0.838   0.4047
## factor(farm)16    19.66019   18.72739   1.050   0.2973
## factor(farm)17   -15.14550   49.01250  -0.309   0.7582
## factor(farm)18    15.94140   15.15371   1.052   0.2963
## factor(farm)19    21.45414   17.99214   1.192   0.2370
## factor(farm)20    25.61956   15.50019   1.653   0.1027
## factor(farm)21    22.78608   15.65699   1.455   0.1499
## factor(farm)22    10.84710   17.69820   0.613   0.5419
## factor(farm)23    -7.58167   16.89435  -0.449   0.6549
## factor(farm)24    21.61703   17.28697   1.250   0.2152
## N:factor(farm)2   -2.07041    0.98369  -2.105   0.0388 *
## N:factor(farm)3   -0.96025    0.89786  -1.069   0.2884
## N:factor(farm)4   -0.59411    1.52204  -0.390   0.6974
## N:factor(farm)5   -0.97735    0.84718  -1.154   0.2525
## N:factor(farm)6   -1.13084    0.97207  -1.163   0.2485
## N:factor(farm)7   -0.58138    0.92164  -0.631   0.5302
## N:factor(farm)8   -0.69333    0.87773  -0.790   0.4322
## N:factor(farm)9   -0.63111    0.81550  -0.774   0.4415
## N:factor(farm)10  -0.04773    0.86512  -0.055   0.9562
## N:factor(farm)11  -1.78432    0.87838  -2.031   0.0459 *
## N:factor(farm)12  -0.50153    0.84820  -0.591   0.5562
## N:factor(farm)13  -1.38290    0.98604  -1.402   0.1651
## N:factor(farm)14  -0.86027    0.89294  -0.963   0.3386
## N:factor(farm)15  -0.53439    0.94640  -0.565   0.5741
## N:factor(farm)16  -1.14547    0.91070  -1.258   0.2125
## N:factor(farm)17   0.24013    1.97779   0.121   0.9037
## N:factor(farm)18  -0.64387    0.79080  -0.814   0.4182
## N:factor(farm)19  -1.31100    0.90886  -1.442   0.1535
## N:factor(farm)20  -0.65867    0.78956  -0.834   0.4069
## N:factor(farm)21  -0.73231    0.81990  -0.893   0.3747
## N:factor(farm)22  -0.37125    0.89597  -0.414   0.6798
## N:factor(farm)23  -0.56171    0.85286  -0.659   0.5122
## N:factor(farm)24  -1.40628    0.95103  -1.479   0.1436
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.978 on 72 degrees of freedom
## Multiple R-squared:  0.9678, Adjusted R-squared:  0.9468
## F-statistic: 46.07 on 47 and 72 DF,  p-value: < 2.2e-16
```

```
model2 <- lm(size ~ N + factor(farm))
anova(model, model2) # model2 selected
```

```
## Analysis of Variance Table
##
## Model 1: size ~ N * factor(farm)
## Model 2: size ~ N + factor(farm)
##   Res.Df    RSS  Df Sum of Sq      F Pr(>F)
## 1     72 281.60
## 2     95 353.81 -23   -72.212 0.8028  0.717
```

```
model3 <- lm(size ~ N)
anova(model2, model3)
```

```
## Analysis of Variance Table
##
## Model 1: size ~ N + factor(farm)
## Model 2: size ~ N
##   Res.Df    RSS  Df Sum of Sq      F    Pr(>F)
## 1     95  353.8
## 2    118 8454.9 -23   -8101.1 94.574 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
detach(yields)
```

Generalized linear mixed models

- `glmer` from `lme4` package is used.
- Average the random effects and only work on the fixed effects, if it's poisson error, use code like:

d2<-aggregate(data,list(farm,field),mean)

model<-lm(log(count)~factor(farm)+factor(field),data=d2)

summary(model)

Chapter 20 Non-linear Regression

When the relationship between y and x cannot be linearized by transformation of the response variable or(and) explanatory variables, non-linear regression will be useful.

`nls` stands for non-linear least squares.

Frequently used non-linear functions:

Name	Equation
Asymptotic functions	
Michaelise-Menten	$y = \frac{ax}{1+bx}$

Name	Equation
2-parameter asymptotic exponential	$y = a(1 - e^{-bx})$
3-parameter asymptotic exponential	$y = a - be^{-cx}$
S-shaped functions	
2-parameter logistic	$y = \frac{e^{a+bx}}{1+e^{a+bx}}$
3-parameter logistic	$y = \frac{a}{1+be^{-cx}}$
4-parameter logistic	$y = a + \frac{b-a}{1+e^{(c-x)/d}}$
Weibull	$y = a - be^{-cx^d}$
Gompertz	$y = ae^{-be^{-cx}}$
Humped curves	
Ricker curve	$y = axe^{-bx}$
First-order compartment	$y = ke^{-e^a x} - e^{-e^b x}$
Bell-shaped	$y = ae^{-\|bx\|^2}$
Biexponential	$y = ae^{bx} - ce^{-dx}$

```
deer <- read.table("jaws.txt", header = TRUE)
attach(deer)
head(deer)
```

```
##         age      bone
## 1  0.000000   0.00000
## 2  5.112000  20.22000
## 3  1.320000  11.11130
## 4 35.240000 140.65000
## 5  1.632931  26.15218
## 6  2.297635  10.00100
```

```
# bone is y

plot(age, bone, pch = 21, col = "purple", bg = "green")

# fit 3 parameter asymptotic exponential
model <- nls(bone ~ a - b * exp( - c * age), start = list(a = 120, b = 110, c = 0.064))
summary(model)
```

```
##
## Formula: bone ~ a - b * exp(-c * age)
##
## Parameters:
##   Estimate Std. Error t value Pr(>|t|)
## a 115.2528     2.9139   39.55  < 2e-16 ***
## b 118.6875     7.8925   15.04  < 2e-16 ***
## c   0.1235     0.0171    7.22 2.44e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.21 on 51 degrees of freedom
##
## Number of iterations to convergence: 5
## Achieved convergence tolerance: 2.391e-06
```

```
# 2 parameter
model2 <- nls(bone ~ a * (1 - exp(- c * age)), start = list(a = 120, c = 0.064))
```

```r
anova(model, model2) # minimal adequate
```

```
## Analysis of Variance Table
##
## Model 1: bone ~ a - b * exp(-c * age)
## Model 2: bone ~ a * (1 - exp(-c * age))
##   Res.Df Res.Sum Sq Df  Sum Sq F value Pr(>F)
## 1     51     8897.3
## 2     52     8929.1 -1 -31.843  0.1825  0.671
```

```r
# add fitted lines
av <- seq(0, 50, 0.1)

bv <- predict(model2, list(age = av))
lines(av, bv, col = "red")

summary(model2)
```

```
##
## Formula: bone ~ a * (1 - exp(-c * age))
##
## Parameters:
##    Estimate Std. Error t value Pr(>|t|)
## a 115.58056    2.84365  40.645  < 2e-16 ***
## c   0.11882    0.01233   9.635 3.69e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.1 on 52 degrees of freedom
##
## Number of iterations to convergence: 5
## Achieved convergence tolerance: 1.369e-06
```

```r
str(summary(model2))
```

```
## List of 11
##  $ formula     :Class 'formula'  language bone ~ a * (1 - exp(-c * age))
##   .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##  $ residuals   : num [1:54] 0 -32.4 -5.67 26.83 5.77 ...
##  $ sigma       : num 13.1
##  $ df          : int [1:2] 2 52
##  $ cov.unscaled: num [1:2, 1:2] 4.71e-02 -1.39e-04 -1.39e-04 8.86e-07
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "a" "c"
##   .. ..$ : chr [1:2] "a" "c"
##  $ call        : language nls(formula = bone ~ a * (1 - exp(-c * age)), start = list(a = 120,      c
##  $ convInfo    :List of 5
##   ..$ isConv     : logi TRUE
##   ..$ finIter    : int 5
##   ..$ finTol     : num 1.37e-06
##   ..$ stopCode   : int 0
##   ..$ stopMessage: chr "converged"
##  $ control     :List of 5
##   ..$ maxiter  : num 50
##   ..$ tol      : num 1e-05
```

```
##   ..$ minFactor: num 0.000977
##   ..$ printEval: logi FALSE
##   ..$ warnOnly : logi FALSE
##  $ na.action   : NULL
##  $ coefficients: num [1:2, 1:4] 115.5806 0.1188 2.8436 0.0123 40.6452 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "a" "c"
##   .. ..$ : chr [1:4] "Estimate" "Std. Error" "t value" "Pr(>|t|)"
##  $ parameters  : num [1:2, 1:4] 115.5806 0.1188 2.8436 0.0123 40.6452 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "a" "c"
##   .. ..$ : chr [1:4] "Estimate" "Std. Error" "t value" "Pr(>|t|)"
##  - attr(*, "class")= chr "summary.nls"
```

```
sum.model2 <- summary(model2)
sum.model2$sigma
```

```
## [1] 13.10398
```

```
sum.model2$df[2]
```

```
## [1] 52
```

```
# sum of squares of error for model2
sse <- as.vector((sum.model2$sigma)^2 * sum.model2$df[2])
sse
```

```
## [1] 8929.143
```

```
# total variation
null <- lm(bone ~ 1)
str(summary.aov(null)) # one list
```

```
## List of 1
##  $ :Classes 'anova' and 'data.frame':    1 obs. of  5 variables:
##   ..$ Df     : num 53
##   ..$ Sum Sq : num 59008
##   ..$ Mean Sq: num 1113
##   ..$ F value: num NA
##   ..$ Pr(>F) : num NA
##  - attr(*, "class")= chr [1:2] "summary.aov" "listof"
```

```
sst <- as.vector(unlist(summary.aov(null)[[1]][2]))
sst
```

```
## [1] 59007.99
```

```
# percentage of variation explained by the model
100*(sst - sse)/sst
```

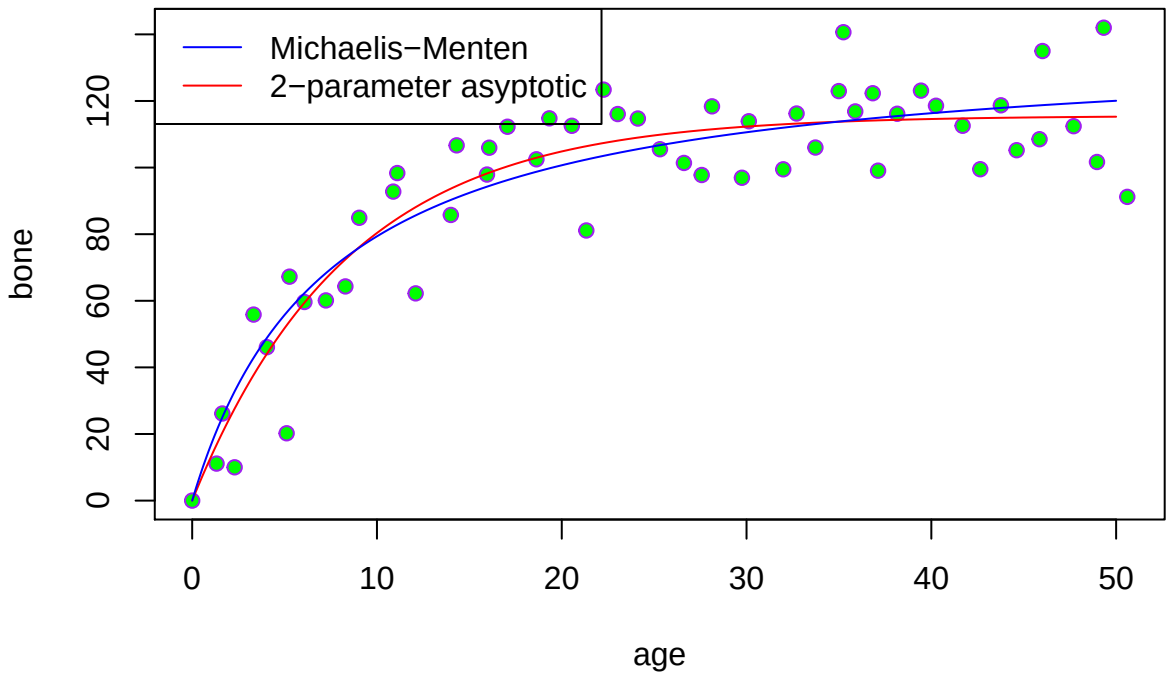
```
## [1] 84.86791
```

```
# compare Michaelis-Menten and asyptotic exponential
(model3 <- nls(bone ~ a * age/(1 + b * age), start = list(a = 8, b = 0.08)))
```

```
## Nonlinear regression model
##   model: bone ~ a * age/(1 + b * age)
##    data: parent.frame()
##      a      b
## 18.725  0.136
```

```
##  residual sum-of-squares: 9854
##
## Number of iterations to convergence: 7
## Achieved convergence tolerance: 1.553e-06
```

```
# add fitted lines to the plot
yv <- predict(model3, list(age = av))
lines(av, yv, col = "blue")
legend("topleft", legend = c("Michaelis-Menten", "2-parameter asyptotic"),
       col = c("blue", "red"), lty = 1)
```




```
detach(deer)
```

Generalized additive models

When we don't have any theory or any mechanistic model to suggest a particular functional form to describe the relationship, GAM will be useful.

```
rm(x, y)
```

```
## Warning in rm(x, y): object 'x' not found
```

```
## Warning in rm(x, y): object 'y' not found
```

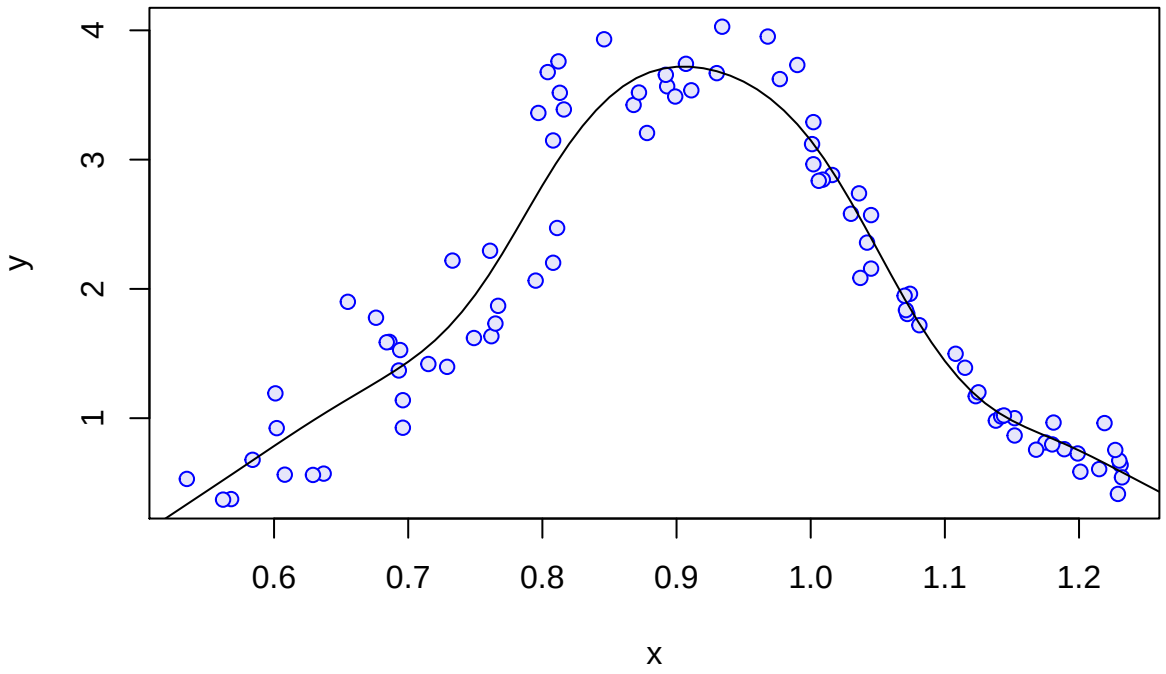
```
humped <- read.table("hump.txt", header = TRUE)
attach(humped)
names(humped)
```

```
## [1] "y" "x"
```

```
plot(x, y, pch = 21, col = "blue", bg = "lavender")

library(mgcv)
```

```
## This is mgcv 1.8-16. For overview type 'help("mgcv-package")'.
```

```
model <- gam(y ~ s(x))

xv <- seq(0.5, 1.3, 0.01)
yv <- predict(model, list(x = xv))
lines(xv, yv)
```




```
summary(model)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## y ~ s(x)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.95737    0.03446    56.8   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##        edf Ref.df     F p-value
## s(x) 7.452  8.403 116.7  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.919   Deviance explained = 92.6%
## GCV = 0.1156  Scale est. = 0.1045    n = 88
```

```r
detach(humped)
```

Grouped data for non-linear estimation

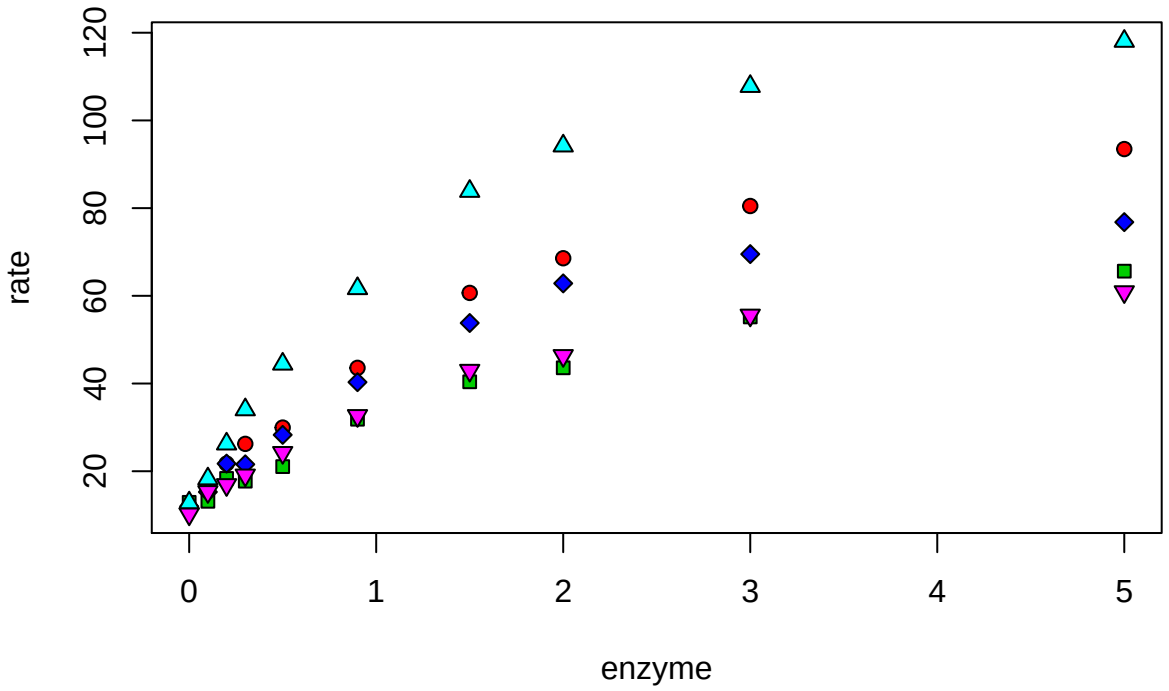
- `nlsList` fits the same functional form of a group of subjects by the "|"
- `nlme` fits the nonlinear mixed effects model

More details on https://dnett.github.io/S510/29GLMMannotated.PDF.

```r
reaction <- read.table("reaction.txt", header  = TRUE)
attach(reaction)
head(reaction)
```

```
##   strain enzyme     rate
## 1      A    0.0 11.91119
## 2      A    0.1 16.46677
## 3      A    0.2 21.73446
## 4      A    0.3 26.23806
## 5      A    0.5 29.95274
## 6      A    0.9 43.57491
```

```r
# rate is y


plot(enzyme, rate, pch = 20 + as.numeric(strain), bg = 1+as.numeric(strain))
```

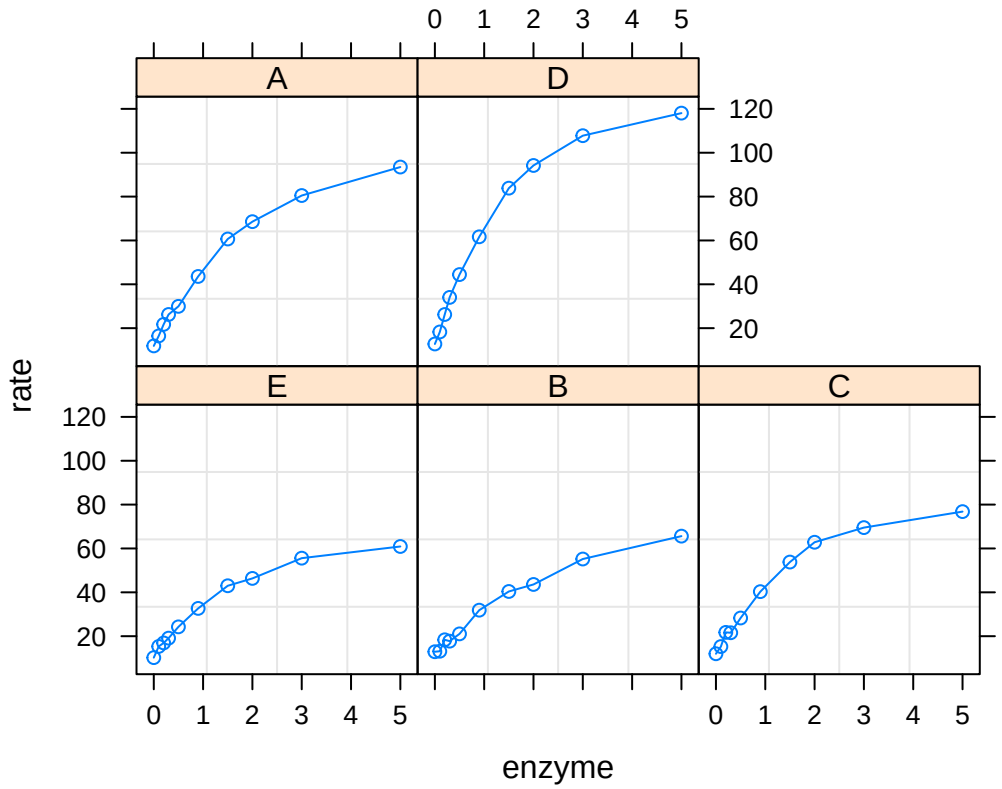



```r
library(nlme)

# fit the same model but with different parameters for each strain
model <- nlsList(rate ~ c + a * enzyme/(1 + b * enzyme)|strain,
                 data = reaction, start = c(a = 20, b = 0.25, c = 10))
```

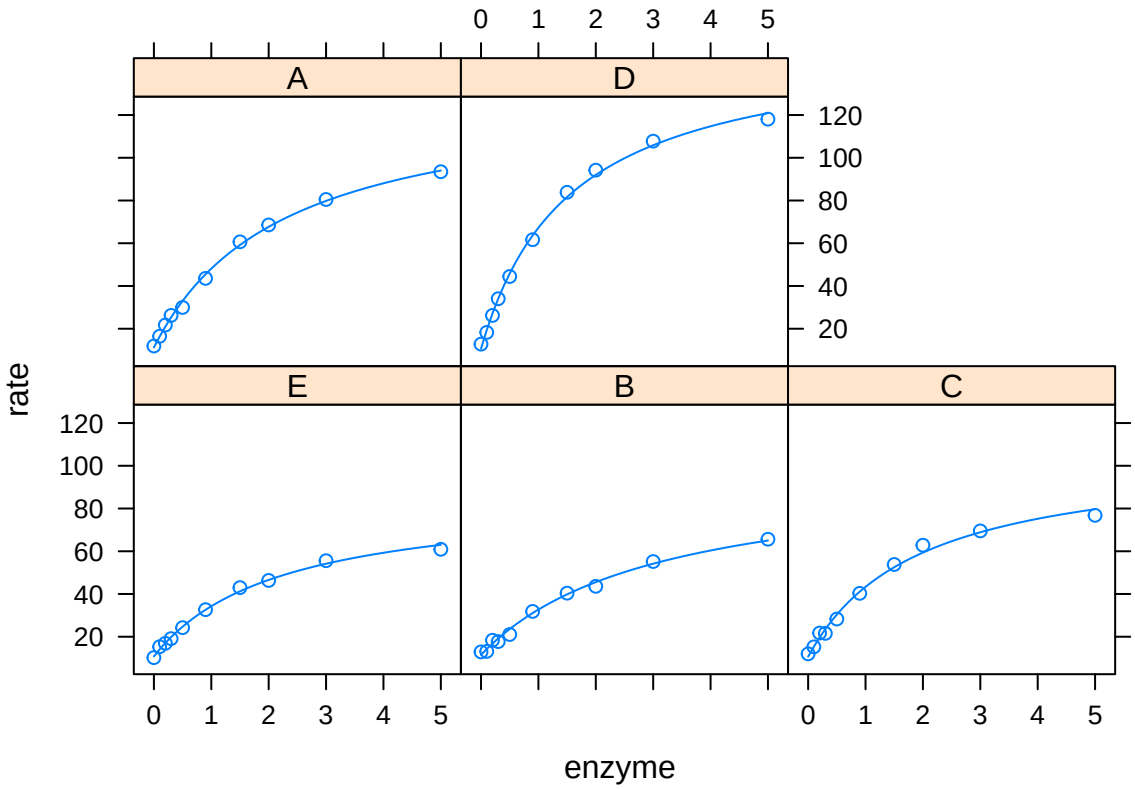
```r
summary(model)
```

```
## Call:
##   Model: rate ~ c + a * enzyme/(1 + b * enzyme) | strain
##    Data: reaction
##
## Coefficients:
##    a
##   Estimate Std. Error   t value     Pr(>|t|)
## A 51.79746   4.093791 12.652687 1.943004e-06
## B 26.05893   3.063474  8.506335 2.800345e-05
## C 51.86774   5.086678 10.196781 7.842354e-05
## D 94.46245   5.813975 16.247482 2.973297e-06
## E 37.50984   4.840749  7.748768 6.462816e-06
##    b
##    Estimate Std. Error   t value     Pr(>|t|)
## A 0.4238572 0.04971637  8.525506 2.728564e-05
## B 0.2802433 0.05761532  4.864041 9.173723e-04
## C 0.5584897 0.07412453  7.534479 5.150212e-04
## D 0.6560539 0.05207361 12.598587 1.634553e-05
## E 0.5253479 0.09354863  5.615774 5.412404e-05
##    c
##   Estimate Std. Error   t value     Pr(>|t|)
## A 11.46498   1.194155  9.600916 1.244487e-05
## B 11.73312   1.120451 10.471780 7.049414e-06
## C 10.53219   1.254928  8.392664 2.671650e-04
## D 10.40964   1.294447  8.041767 2.909373e-04
## E 10.30139   1.240664  8.303123 4.059886e-06
##
## Residual standard error: 1.81625 on 35 degrees of freedom
```

```r
# plot
reaction <- groupedData(rate ~ enzyme | strain, data = reaction)
plot(reaction)
```




```
# fit non-linear mixed effects model
model2 <- nlme(rate ~ c + a * enzyme/(1 + b * enzyme), fixed = a + b + c ~ 1,
      random = a + b + c ~ 1 | strain, data = reaction, start = c(a = 20, b = 0.25, c = 10))

plot(augPred(model2))
```




```
# augPred returns a data frame with four columns representing, respectively,
# the values of the primary covariate, the groups (if object does not have a
# grouping structure, all elements will be 1), the predicted or observed values,
# and the type of value in the third column
model2
```

```
## Nonlinear mixed-effects model fit by maximum likelihood
##   Model: rate ~ c + a * enzyme/(1 + b * enzyme)
##   Data: reaction
##   Log-likelihood: -116.7403
##   Fixed: a + b + c ~ 1
##         a          b          c
## 51.5988154  0.4766508 10.9853653
##
## Random effects:
##  Formula: list(a ~ 1, b ~ 1, c ~ 1)
##  Level: strain
##  Structure: General positive-definite, Log-Cholesky parametrization
##          StdDev     Corr
## a        22.9152883 a      b
## b         0.1132360  0.876
## c         0.4229321 -0.537 -0.875
## Residual  1.7105942
##
## Number of Observations: 50
## Number of Groups: 5
```

```
summary(model2)
```

```
## Nonlinear mixed-effects model fit by maximum likelihood
```

```
##   Model: rate ~ c + a * enzyme/(1 + b * enzyme)
##  Data: reaction
##        AIC      BIC    logLik
##   253.4806 272.6008 -116.7403
##
## Random effects:
##  Formula: list(a ~ 1, b ~ 1, c ~ 1)
##  Level: strain
##  Structure: General positive-definite, Log-Cholesky parametrization
##          StdDev     Corr
## a        22.9152883 a      b
## b         0.1132360  0.876
## c         0.4229321 -0.537 -0.875
## Residual  1.7105942
##
## Fixed effects: a + b + c ~ 1
##      Value Std.Error DF   t-value p-value
## a 51.59882 10.741427 43  4.803721       0
## b  0.47665  0.058785 43  8.108329       0
## c 10.98537  0.556441 43 19.742190       0
##  Correlation:
##   a      b
## b  0.843
## c -0.314 -0.543
##
## Standardized Within-Group Residuals:
##         Min          Q1         Med          Q3         Max
## -1.79185237 -0.65630853  0.05689238  0.74268320  2.02723094
##
## Number of Observations: 50
## Number of Groups: 5
```

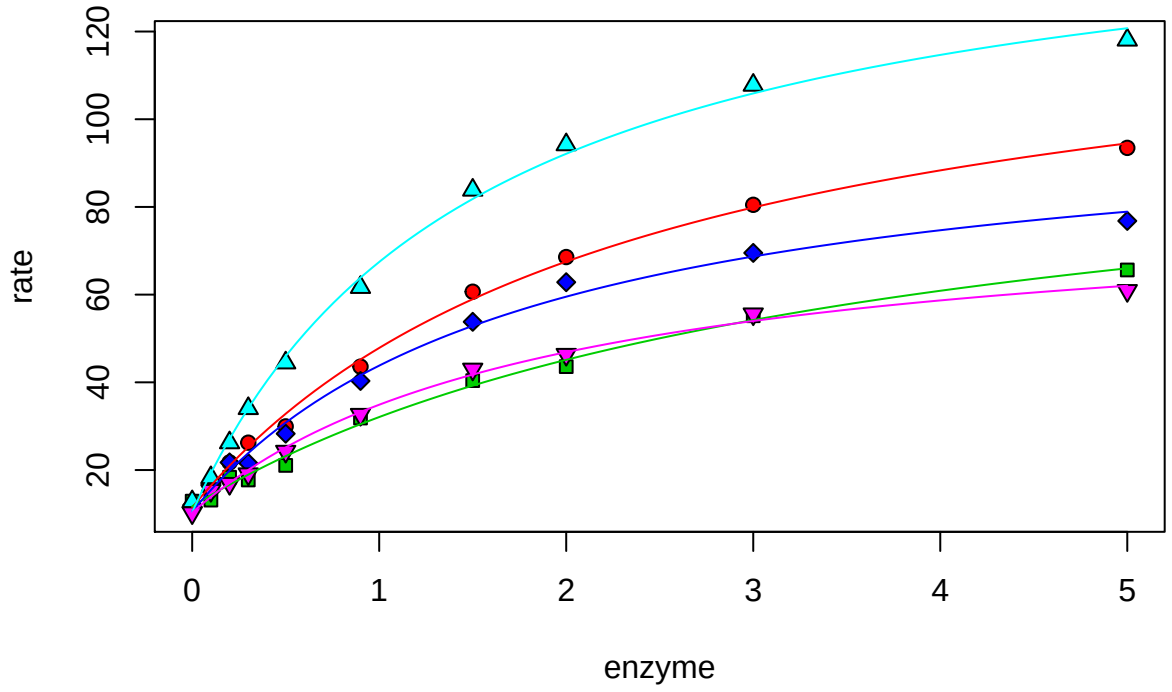
```
coef(model2)
```

```
##          a         b        c
## E 34.09020 0.4533418 10.81737
## B 28.01289 0.3238706 11.54802
## C 49.63862 0.5193743 10.67202
## A 53.20494 0.4426268 11.23602
## D 93.04742 0.6440403 10.65340
```

```
v <- vcov(model2)
v
```

```
##            a           b          c
## a 108.455558  0.500400977 -1.76227393
## b   0.500401  0.003248372 -0.01668663
## c  -1.762274 -0.016686634  0.29104908
```

```
plot(enzyme, rate, pch = 20 + as.numeric(strain), bg = 1 + as.numeric(strain))

xv <- seq(min(enzyme), max(enzyme), length = 100)
for(i in 1:5){
yv <- coef(model)[i, 3] + coef(model)[i, 1] * xv/(1 + coef(model)[i, 2] * xv)
lines(xv, yv, col = (i + 1)) }
```




```
detach(reaction)
rm(xv, yv)
```

Non-linear time series models (temporal pseudo-replication)

```
nl.ts <- read.table("nonlinear.txt", header = TRUE)
attach(nl.ts)
head(nl.ts)
```

```
##   time dish isolate     diam
## 1    0    1       A 1.387028
## 2    1    1       A 4.811886
## 3    2    1       A 6.081095
## 4    3    1       A 6.641911
## 5    4    1       A 8.088018
## 6    5    1       A 8.444012
```

```
# group by dish
growth <- groupedData(diam ~ time | dish, data = nl.ts)

model <- nlme(diam ~ a + b * time/(1 + c * time),
      fixed = a + b + c ~ 1,
      random = a + b + c ~ 1,
      data = growth,
      correlation = corAR1(),
      start = c(a = 0.5, b = 5, c = 0.5))
summary(model)
```

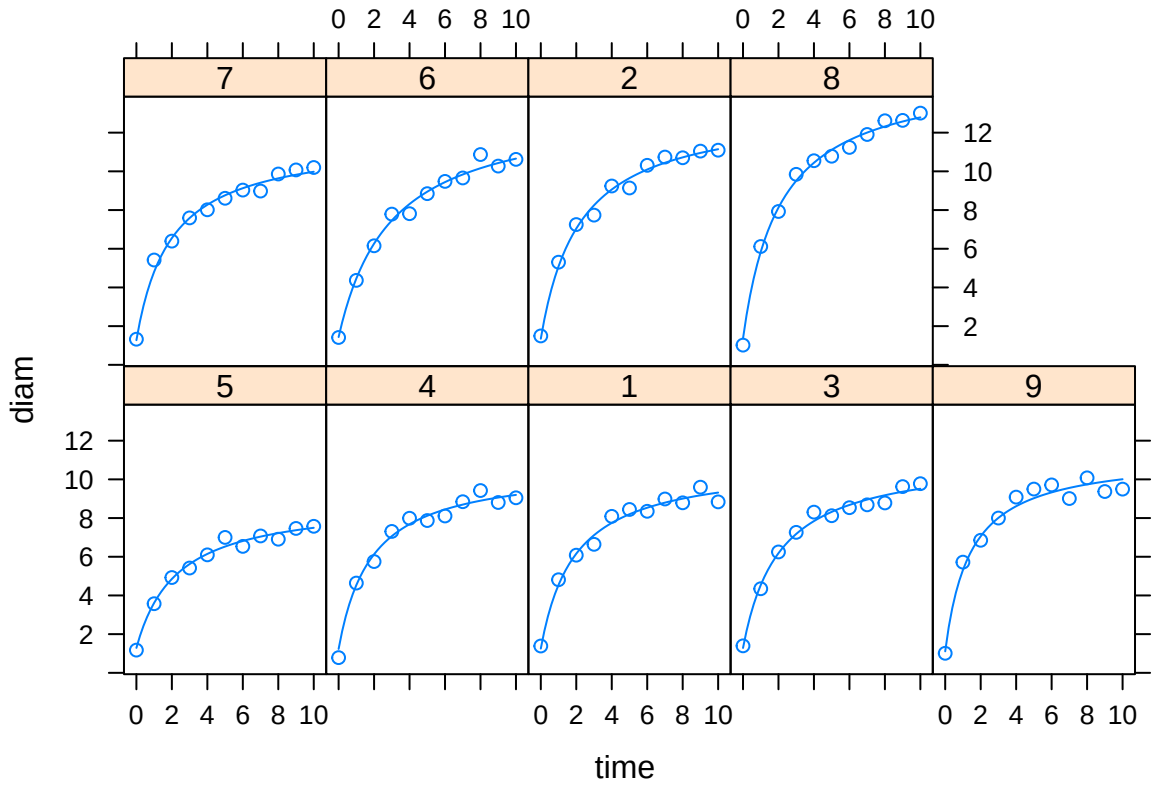
```
## Nonlinear mixed-effects model fit by maximum likelihood
##   Model: diam ~ a + b * time/(1 + c * time)
##  Data: growth
```

```
##        AIC      BIC    logLik
##   129.7694 158.3157 -53.88469
##
## Random effects:
##  Formula: list(a ~ 1, b ~ 1, c ~ 1)
##  Level: dish
##  Structure: General positive-definite, Log-Cholesky parametrization
##          StdDev    Corr
## a        0.1014470 a      b
## b        1.2060359 -0.557
## c        0.1095789 -0.958  0.772
## Residual 0.3150067
##
## Correlation Structure: AR(1)
##  Formula: ~1 | dish
##  Parameter estimate(s):
##        Phi
## -0.0334497
## Fixed effects: a + b + c ~ 1
##      Value Std.Error DF  t-value p-value
## a 1.288262 0.1086390 88 11.85819       0
## b 5.215251 0.4741948 88 10.99812       0
## c 0.498222 0.0450643 88 11.05579       0
##  Correlation:
##   a      b
## b -0.506
## c -0.542  0.823
##
## Standardized Within-Group Residuals:
##         Min          Q1         Med          Q3         Max
## -1.74222923 -0.64713581 -0.03349732  0.70298876  2.24686690
##
## Number of Observations: 99
## Number of Groups: 9
```

```
coef(model)
```

```
##          a        b         c
## 5 1.288830 3.348754 0.4393773
## 4 1.235631 5.075217 0.5373945
## 1 1.252726 5.009538 0.5212435
## 3 1.285847 4.843221 0.4885947
## 9 1.111140 7.171294 0.7061046
## 7 1.272571 5.361569 0.5158168
## 6 1.435781 4.055246 0.3397513
## 2 1.348523 5.440496 0.4553725
## 8 1.363309 6.631921 0.4803384
```

```
plot(augPred(model))
```




```
detach(nl.ts)
```

Self-starting functions

Function	Description
SSasymp	asymptotic regression model;
SSasympOff	asymptotic regression model with an offset;
SSasympOrig	asymptotic regression model through the origin;
SSbiexp	biexponential model;
SSfol	first-order compartment model;
SSfpl	four-parameter logistic model;
SSgompertz	Gompertz growth model;
SSlogis	logistic model;
SSmicmen	Michaelis–Menten model;
SSweibull	Weibull growth curve model.

```
# self starting Michaelis-menten model
data <- read.table("mm.txt", header = TRUE)
attach(data)
names(data)
```

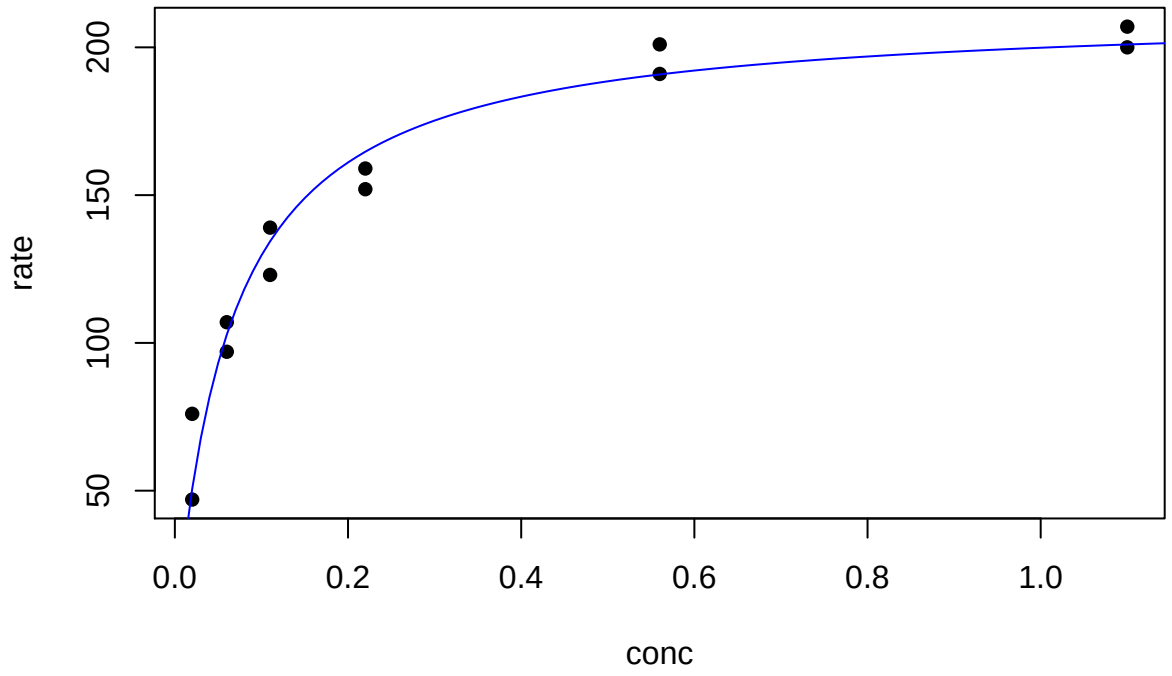
```
## [1] "conc" "rate"
```

```
plot(rate ~ conc, pch = 16)

model <- nls(rate ~ SSmicmen(conc, a, b))
```

```
# a is the max value of rate, b is the value of conc at which half of the max response attained
summary(model)
```

```
##
## Formula: rate ~ SSmicmen(conc, a, b)
##
## Parameters:
##    Estimate Std. Error t value Pr(>|t|)
## a 2.127e+02  6.947e+00  30.615 3.24e-11 ***
## b 6.412e-02  8.281e-03   7.743 1.57e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.93 on 10 degrees of freedom
##
## Number of iterations to convergence: 0
## Achieved convergence tolerance: 1.937e-06
```

```
xv <- seq(0, 1.2, 0.01)
yv <- predict(model, list(conc = xv))
lines(xv, yv, col = "blue")
```

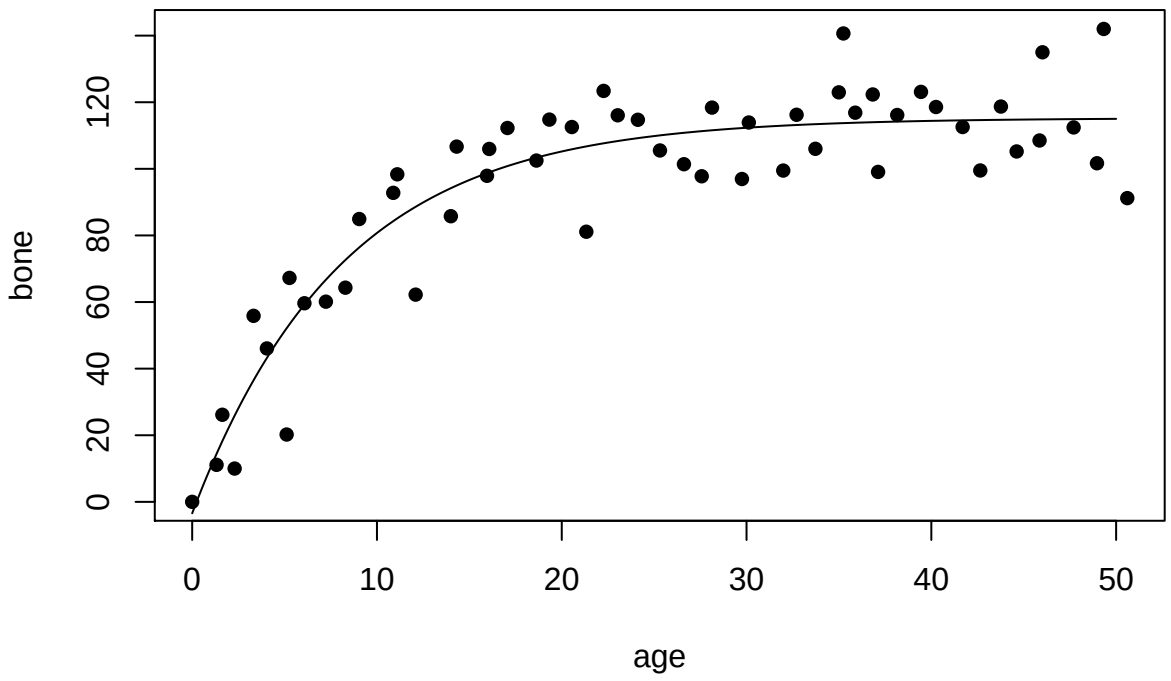



```
detach(data)


# self starting asymptotic exponential model
deer <- read.table("jaws.txt", header = TRUE)
attach(deer)
names(deer)
```

```
## [1] "age"  "bone"
```

```
model <- nls(bone ~ SSasymp(age, a, b, c))
plot(age, bone, pch = 16)
```

```
xv <- seq(0, 50, 0.2)
yv <- predict(model, list(age =xv))
lines(xv, yv)
```



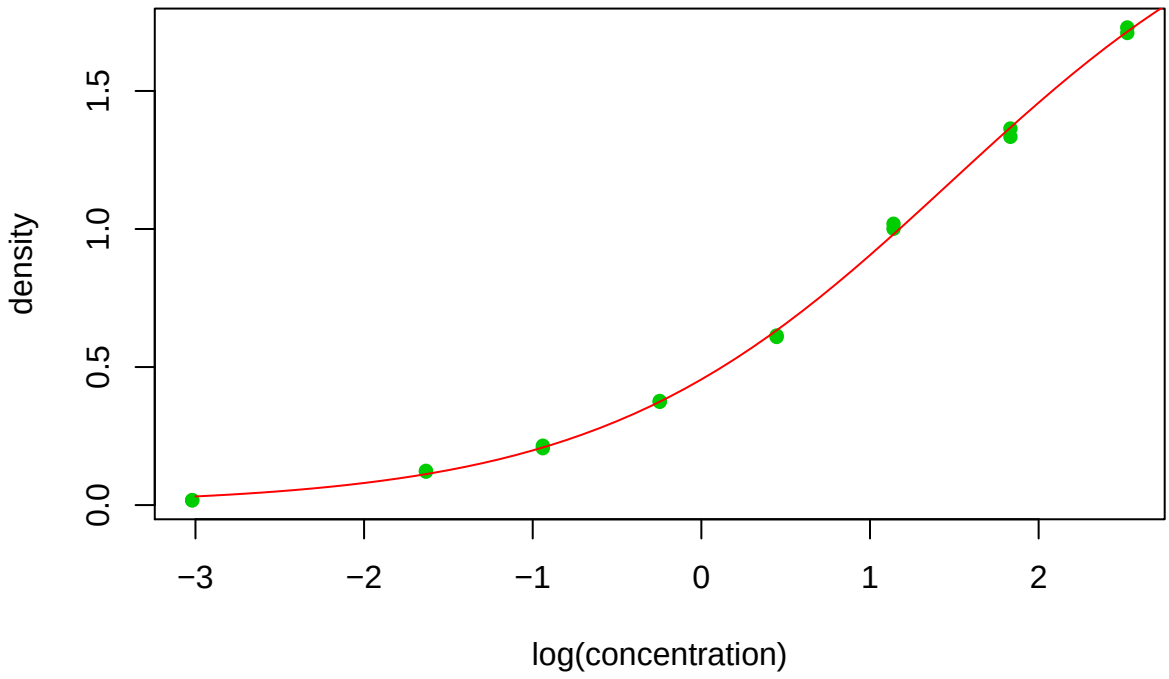

```
summary(model)
```

```
##
## Formula: bone ~ SSasymp(age, a, b, c)
##
## Parameters:
##   Estimate Std. Error t value Pr(>|t|)
## a 115.2527     2.9139  39.553   <2e-16 ***
## b  -3.4348     8.1961  -0.419    0.677
## c  -2.0915     0.1385 -15.101   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.21 on 51 degrees of freedom
##
## Number of iterations to convergence: 0
## Achieved convergence tolerance: 2.45e-07
```

```
detach(deer)



# self starting logistic
sslogistic <- read.table("sslogistic.txt", header = TRUE)
attach(sslogistic)

names(sslogistic)
```

```
## [1] "density"       "concentration"
```

```
plot(density ~ log(concentration), pch = 16, col = "green3")

model <- nls(density ~ SSlogis(log(concentration), a, b, c))

xv <- seq(-3, 3, 0.1)
yv <- predict(model, list(concentration = exp(xv)))
lines(xv, yv, col = "red")
```




```
summary(model)
```

```
##
## Formula: density ~ SSlogis(log(concentration), a, b, c)
##
## Parameters:
##   Estimate Std. Error t value Pr(>|t|)
## a  2.34518    0.07815   30.01 2.17e-13 ***
## b  1.48309    0.08135   18.23 1.22e-10 ***
## c  1.04146    0.03227   32.27 8.51e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01919 on 13 degrees of freedom
##
## Number of iterations to convergence: 0
## Achieved convergence tolerance: 3.281e-06
```

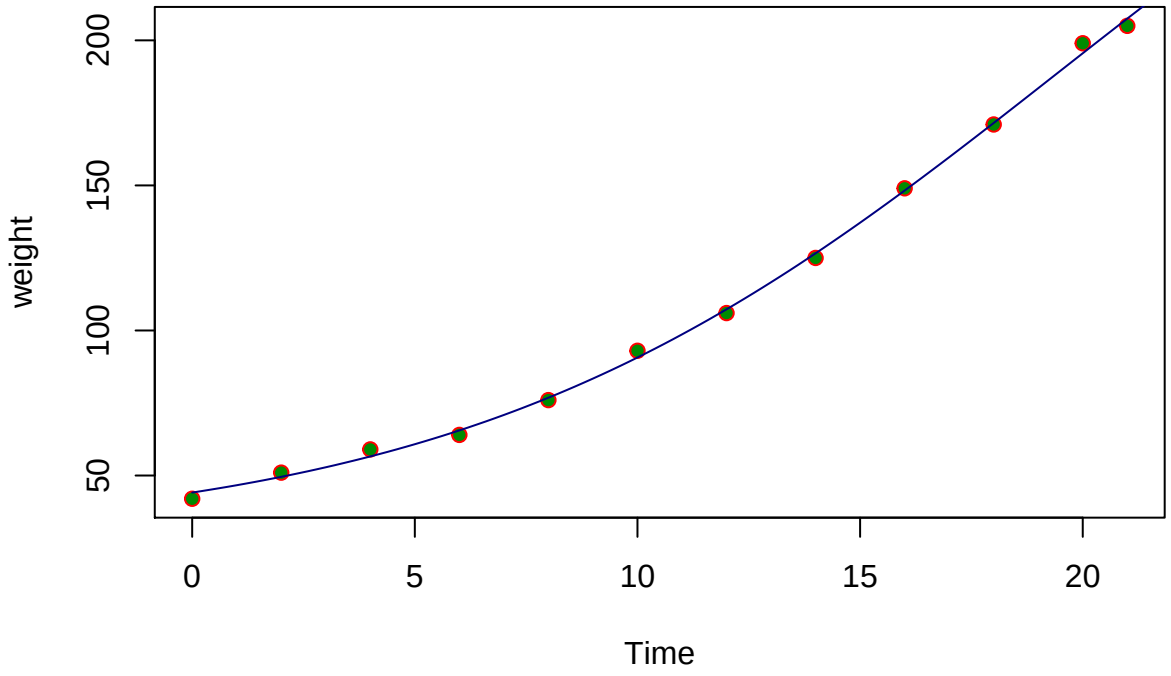
```
detach(sslogistic)
```

```
# four-parameter logistic
```

```
data <- read.table("chicks.txt", header = TRUE)
attach(data)
names(data)
```

```
## [1] "weight" "Time"
```

```
model <- nls(weight ~ SSfpl(Time, a, b, c, d))
xv <- seq(0, 22, 0.2)
yv <- predict(model, list(Time = xv))
plot(weight ~ Time, pch = 21, col = "red", bg = "green4")
lines(xv, yv, col = "navy")
```




```
summary(model)
```

```
##
## Formula: weight ~ SSfpl(Time, a, b, c, d)
##
## Parameters:
##   Estimate Std. Error t value Pr(>|t|)
## a   27.453      6.601   4.159 0.003169 **
## b  348.971     57.899   6.027 0.000314 ***
## c   19.391      2.194   8.836 2.12e-05 ***
## d    6.673      1.002   6.662 0.000159 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.351 on 8 degrees of freedom
##
## Number of iterations to convergence: 0
## Achieved convergence tolerance: 2.476e-07
```

```
detach(data)
```

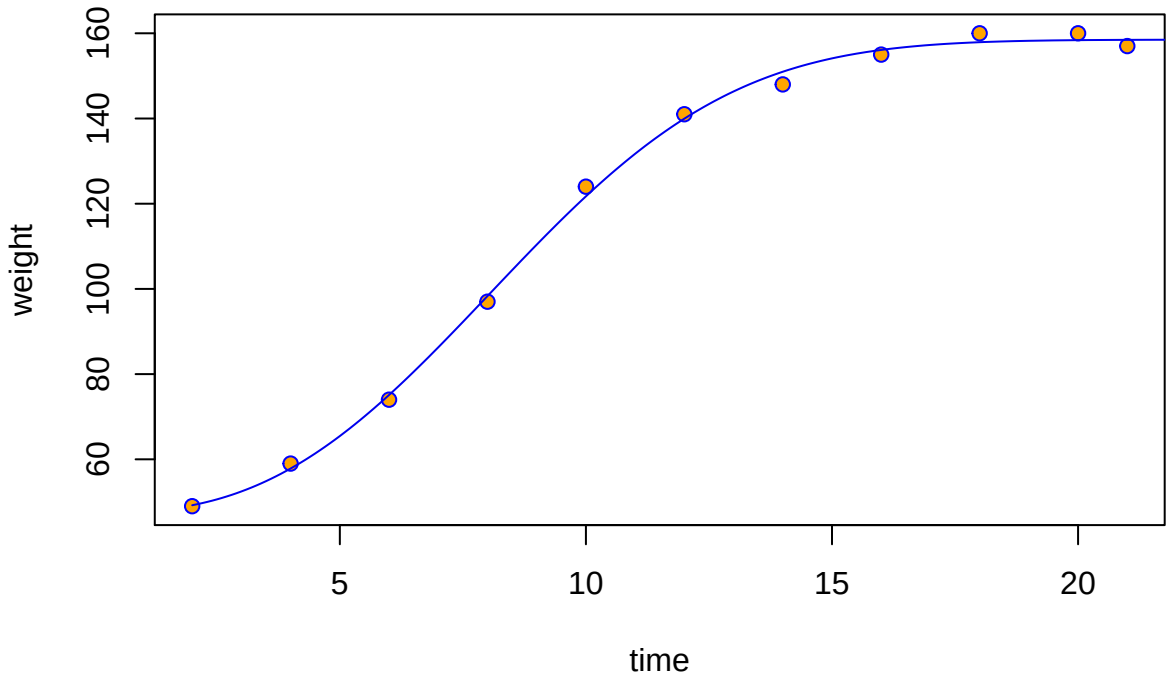
```r
# self start weibull growth function
weights <- read.table("weibull.growth.txt", header = TRUE)
attach(weights)
names(weights)
```

```
## [1] "weight" "time"
```

```r
model <- nls(weight ~ SSweibull(time, Asym, Drop, lrc, pwr))
summary(model)
```

```
##
## Formula: weight ~ SSweibull(time, Asym, Drop, lrc, pwr)
##
## Parameters:
##      Estimate Std. Error t value Pr(>|t|)
## Asym 158.5012     1.1769  134.67 3.28e-13 ***
## Drop 110.9971     2.6330   42.16 1.10e-09 ***
## lrc   -5.9934     0.3733  -16.05 8.83e-07 ***
## pwr    2.6461     0.1613   16.41 7.62e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.061 on 7 degrees of freedom
##
## Number of iterations to convergence: 0
## Achieved convergence tolerance: 5.692e-06
```

```r
xt <- seq(2, 22, 0.1)
yw <- predict(model, list(time = xt))
plot(time, weight, pch = 21, col = "blue", bg = "orange")
lines(xt, yw, col = "blue2")
```



```
detach(weights)



# self starting first order compartment function
foldat <- read.table("fol.txt", header = TRUE)
attach(foldat)
names(foldat)
```

```
## [1] "Wt"   "Dose" "Time" "conc"
```

```
model <- nls(conc ~ SSfol(Dose, Time, a, b, c))
summary(model)
```

```
##
## Formula: conc ~ SSfol(Dose, Time, a, b, c)
##
## Parameters:
##   Estimate Std. Error t value Pr(>|t|)
## a  -2.9196     0.1709 -17.085 1.40e-07 ***
## b   0.5752     0.1728   3.328   0.0104 *
## c  -3.9159     0.1273 -30.768 1.35e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.732 on 8 degrees of freedom
##
## Number of iterations to convergence: 8
## Achieved convergence tolerance: 4.907e-06
```

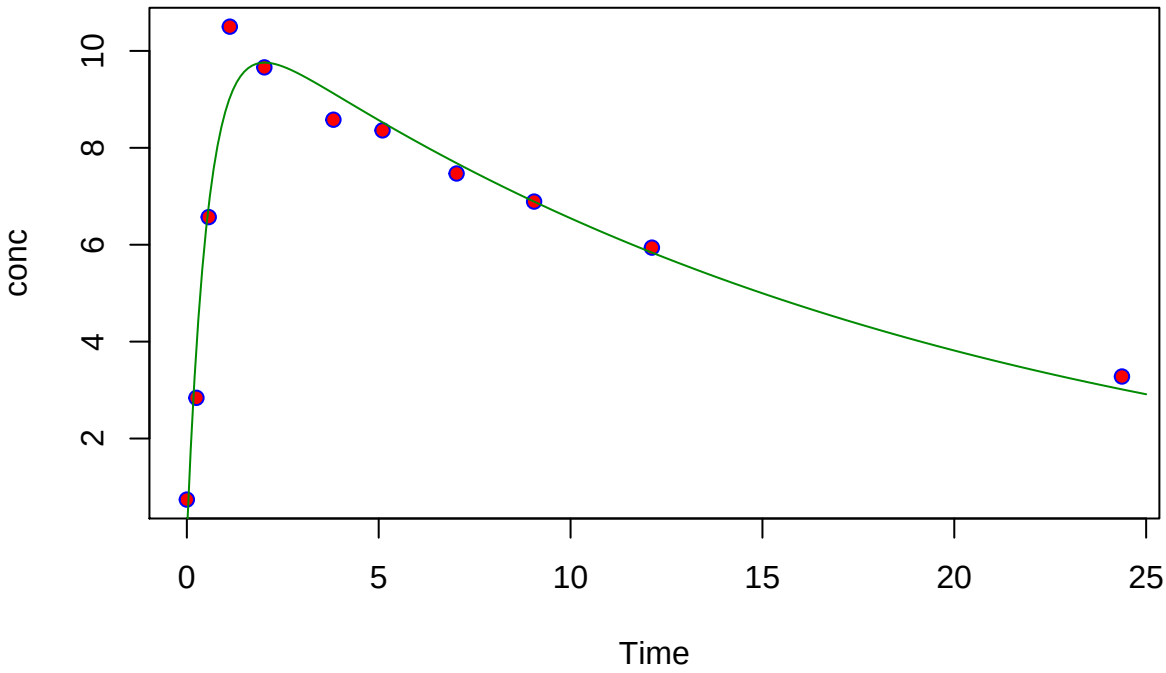
```
xv <- seq(0, 25, 0.1)
yv <- predict(model, list(Time = xv))
```

```
## Warning in .expr4 * (.expr8 - .expr12): longer object length is not a
## multiple of shorter object length
```

```
## Warning in .expr4 * (.expr8 * (.expr5 * input)): longer object length is
## not a multiple of shorter object length
```

```
## Warning in .expr4 * (.expr12 * (.expr9 * input)): longer object length is
## not a multiple of shorter object length
```

```
plot(conc ~ Time, pch = 21, col = "blue", bg = "red")
lines(xv, yv, col = "green4")
```




```
detach(foldat)
```

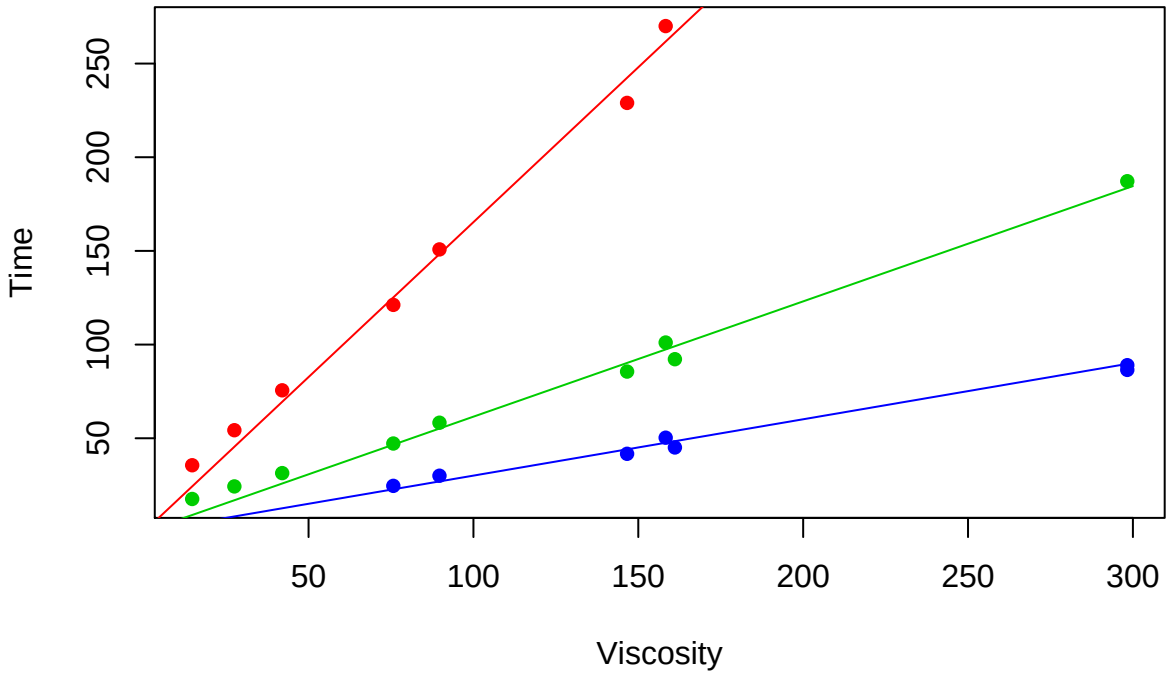
Bootstrapping a family of non-linear regression

```
library(MASS)
data(stormer)
attach(stormer)

model <- nls(Time ~ b * Viscosity/(Wt - c), start = list(b = 29, c = 2))
summary(model)
```

```
##
## Formula: Time ~ b * Viscosity/(Wt - c)
##
## Parameters:
##   Estimate Std. Error t value Pr(>|t|)
## b  29.4013     0.9155  32.114  < 2e-16 ***
## c   2.2182     0.6655   3.333  0.00316 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.268 on 21 degrees of freedom
##
## Number of iterations to convergence: 2
## Achieved convergence tolerance: 8.965e-06
```

```
plot(Viscosity, Time, pch = 16, col = 1 + as.numeric(factor(Wt)))
xv <- 0:300
yv <- predict(model, list(Wt = 20, Viscosity = xv))
lines(xv, yv, col = 2)
yv <- predict(model, list(Wt = 50, Viscosity = xv))
lines(xv, yv, col = 3)
```

```
yv <- predict(model, list(Wt = 100, Viscosity = xv))
lines(xv, yv, col = 4)
```




```
# homemade function to do bootstrap
bv <- numeric(1000)
cv <- numeric(1000)
for(i in 1:1000){
ss <- sample(1:23, replace = TRUE)
y <- Time[ss]
x1 <- Viscosity[ss]
x2 <- Wt[ss]
model <- nls(y ~ b * x1/(x2 - c), start = list(b = 29, c = 2))
bv[i] <- coef(model)[1]
cv[i] <- coef(model)[2]
}

quantile(bv, c(0.025, 0.975))
```

```
##     2.5%    97.5%
## 27.88269 30.60248
```

```
quantile(cv, c(0.025, 0.975))
```

```
##      2.5%     97.5%
## 0.9518806 3.6812025
```

```
# bootstrap by boot package
library(boot)
```

```
##
## Attaching package: 'boot'
```

```
## The following object is masked from 'package:lattice':
##
```

```
##     melanoma
```

```
rs <- resid(model)
fit <- fitted(model)

storm <- data.frame(fit, Viscosity, Wt)

statistic <- function(rs, i){
storm$y <- storm$fit + rs[i]
coef(nls(y ~ b * Viscosity/(Wt - c), data = storm, start = coef(model)))
}
# note: some iterations may have singular gradient and thus error message, re-run the program in this c
boot.model <- boot(rs, statistic, R = 1000)
boot.model
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = rs, statistic = statistic, R = 1000)
##
##
## Bootstrap Statistics :
##      original     bias    std. error
## t1*  174.4152   67.90103    60.67498
## t2* -284.5209 -134.28487   121.27837
```

```
boot.ci(boot.model,index=1)
```

```
## Warning in boot.ci(boot.model, index = 1): bootstrap variances needed for
## studentized intervals
```

```
## Warning in norm.inter(t, adj.alpha): extreme order statistics used as
## endpoints
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot.model, index = 1)
##
## Intervals :
## Level      Normal              Basic
## 95%   (-12.4, 225.4 )   (-43.4, 184.6 )
##
## Level     Percentile            BCa
## 95%   (164.2, 392.3 )   (143.9, 190.1 )
## Calculations and Intervals on Original Scale
## Warning : BCa Intervals used Extreme Quantiles
## Some BCa intervals may be unstable
```

```
boot.ci(boot.model,index=2)
```

```
## Warning in boot.ci(boot.model, index = 2): bootstrap variances needed for
## studentized intervals
```

```
## Warning in boot.ci(boot.model, index = 2): extreme order statistics used as
## endpoints
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot.model, index = 2)
##
## Intervals :
## Level      Normal              Basic
## 95%   (-387.9,   87.5 )   (-305.6,  155.1 )
##
## Level     Percentile            BCa
## 95%   (-724.2, -263.4 )   (-315.2, -225.2 )
## Calculations and Intervals on Original Scale
## Warning : BCa Intervals used Extreme Quantiles
## Some BCa intervals may be unstable
```

```
detach(stormer)
```