References

Abramowitz, M. and Stegun, I. A. (1964). *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, Dover, New York.

Bates, D. M. and Chambers, J. M. (1992). "Nonlinear models," in Chambers and Hastie (1992), Chapter 10, pp. 421–454.

Bates, D. M. and Pinheiro, J. C. (1998). Computational methods for multilevel models, *Technical Memorandum BL0112140-980226-01TM*, Bell Labs, Lucent Technologies, Murray Hill, NJ.

Bates, D. M. and Watts, D. G. (1980). Relative curvature measures of nonlinearity, *Journal of the Royal Statistical Society, Ser. B* **42**: 1–25.

Bates, D. M. and Watts, D. G. (1988). *Nonlinear Regression Analysis and Its Applications*, Wiley, New York.

Beal, S. and Sheiner, L. (1980). The NONMEM system, *American Statistician* **34**: 118–119.

Becker, R. A., Cleveland, W. S. and Shyu, M.-J. (1996). The visual design and control of trellis graphics displays, *Journal of Computational and Graphical Statistics* **5**(2): 123–156.

Bennett, J. E. and Wakefield, J. C. (1993). Markov chain Monte Carlo for nonlinear hierarchical models, *Technical Report TR-93-11*, Statistics Section, Imperial College, London.

Boeckmann, A. J., Sheiner, L. B. and Beal, S. L. (1994). *NONMEM Users Guide: Part V*, NONMEM Project Group, University of California, San Francisco.

Box, G. E. P., Hunter, W. G. and Hunter, J. S. (1978). *Statistics for Experimenters*, Wiley, New York.

Box, G. E. P., Jenkins, G. M. and Reinsel, G. C. (1994). *Time Series Analysis: Forecasting and Control*, 3rd ed., Holden-Day, San Francisco.

Brillinger, D. (1987). Comment on a paper by C. R. Rao, *Statistical Science* **2**: 448–450.

Bryk, A. and Raudenbush, S. (1992). *Hierarchical Linear Models for Social and Behavioral Research*, Sage, Newbury Park, CA.

Carroll, R. J. and Ruppert, D. (1988). *Transformation and Weighting in Regression*, Chapman & Hall, New York.

Chambers, J. M. (1977). *Computational Methods for Data Analysis*, Wiley, New York.

Chambers, J. M. and Hastie, T. J. (eds) (1992). *Statistical Models in S*, Chapman & Hall, New York.

Cleveland, W. S. (1994). *Visualizing Data*, Hobart Press, Summit, NJ.

Cleveland, W. S., Grosse, E. and Shyu, W. M. (1992). "Local regression models," in Chambers and Hastie (1992), Chapter 8, pp. 309–376.

Cochran, W. G. and Cox, G. M. (1957). *Experimental Designs*, 2nd ed., Wiley, New York.

Cox, D. R. and Hinkley, D. V. (1974). *Theoretical Statistics*, Chapman & Hall, London.

Cressie, N. A. C. (1993). *Statistics for Spatial Data*, Wiley, New York.

Cressie, N. A. C. and Hawkins, D. M. (1980). Robust estimation of the variogram, *Journal of the International Association of Mathematical Geology* **12**: 115–125.

Crowder, M. and Hand, D. (1990). *Analysis of Repeated Measures*, Chapman & Hall, London.

Davidian, M. and Gallant, A. R. (1992). Smooth nonparametric maximum likelihood estimation for population pharmacokinetics, with application to quinidine, *Journal of Pharmacokinetics and Biopharmaceutics* **20**: 529–556.

Davidian, M. and Giltinan, D. M. (1995). *Nonlinear Models for Repeated Measurement Data*, Chapman & Hall, London.

Davis, P. J. and Rabinowitz, P. (1984). *Methods of Numerical Integration*, 2nd ed., Academic Press, New York.

Dempster, A. P., Laird, N. M. and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm, *Journal of the Royal Statistical Society, Ser. B* **39**: 1–22.

Devore, J. L. (2000). *Probability and Statistics for Engineering and the Sciences*, 5th ed., Wadsworth, Belmont, CA.

Diggle, P. J., Liang, K.-Y. and Zeger, S. L. (1994). *Analysis of Longitudinal Data*, Oxford University Press, New York.

Dongarra, J. J., Bunch, J. R., Moler, C. B. and Stewart, G. W. (1979). *Linpack Users' Guide*, SIAM, Philadelphia.

Draper, N. R. and Smith, H. (1998). *Applied Regression Analysis*, 3rd ed., Wiley, New York.

Gallant, A. R. and Nychka, D. W. (1987). Seminonparametric maximum likelihood estimation, *Econometrica* **55**: 363–390.

Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **6**: 721–741.

Geweke, J. (1989). Bayesian inference in econometric models using Monte Carlo integration, *Econometrica* **57**: 1317–1339.

Gibaldi, M. and Perrier, D. (1982). *Pharmacokinetics*, Marcel Dekker, New York.

Goldstein, H. (1987). *Multilevel Models in Education and Social Research*, Oxford University Press, Oxford.

Goldstein, H. (1995). *Multilevel Statistical Models*, Halstead Press, New York.

Golub, G. H. (1973). Some modified matrix eigenvalue problems, *SIAM Review* **15**: 318–334.

Golub, G. H. and Welsch, J. H. (1969). Calculation of Gaussian quadrature rules, *Mathematical Computing* **23**: 221–230.

Grasela and Donn (1985). Neonatal population pharmacokinetics of phenobarbital derived from routine clinical data, *Developmental Pharmacology and Therapeutics* **8**: 374–0383.

Hand, D. and Crowder, M. (1996). *Practical Longitudinal Data Analysis*, Texts in Statistical Science, Chapman & Hall, London.

Harville, D. A. (1977). Maximum likelihood approaches to variance component estimation and to related problems, *Journal of the American Statistical Association* **72**: 320–340.

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications, *Biometrika* **57**: 97–109.

Jones, R. H. (1993). *Longitudinal Data with Serial Correlation: A State-space Approach*, Chapman & Hall, London.

Joyner and Boore (1981). Peak horizontal acceleration and velocity from strong-motion records including records from the 1979 Imperial Valley, California, earthquake, *Bulletin of the Seismological Society of America* **71**: 2011–2038.

Kennedy, William J., J. and Gentle, J. E. (1980). *Statistical Computing*, Marcel Dekker, New York.

Kung, F. H. (1986). Fitting logistic growth curve with predetermined carrying capacity, *ASA Proceedings of the Statistical Computing Section* pp. 340–343.

Kwan, K. C., Breault, G. O., Umbenhauer, E. R., McMahon, F. G. and Duggan, D. E. (1976). Kinetics of indomethicin absorption, elimination, and enterohepatic circulation in man, *Journal of Pharmacokinetics and Biopharmaceutics* **4**: 255–280.

Laird, N. M. and Ware, J. H. (1982). Random-effects models for longitudinal data, *Biometrics* **38**: 963–974.

Lehmann, E. L. (1986). *Testing Statistical Hypotheses*, Wiley, New York.

Leonard, T., Hsu, J. S. J. and Tsui, K. W. (1989). Bayesian marginal inference, *Journal of the American Statistical Association* **84**: 1051–1058.

Lindley, D. and Smith, A. (1972). Bayes estimates for the linear model, *Journal of the Royal Statistical Society, Ser. B* **34**: 1–41.

Lindstrom, M. J. and Bates, D. M. (1988). Newton–Raphson and EM algorithms for linear mixed-effects models for repeated-measures data (corr: 94v89 p1572), *Journal of the American Statistical Association* **83**: 1014–1022.

Lindstrom, M. J. and Bates, D. M. (1990). Nonlinear mixed-effects models for repeated measures data, *Biometrics* **46**: 673–687.

Littell, R. C., Milliken, G. A., Stroup, W. W. and Wolfinger, R. D. (1996). *SAS System for Mixed Models*, SAS Institute Inc., Cary, NC.

Longford, N. T. (1993). *Random Coefficient Models*, Oxford University Press, New York.

Ludbrook, J. (1994). Repeated measurements and multiple comparisons in cardiovascular research, *Cardiovascular Research* **28**: 303–311.

Mallet, A. (1986). A maximum likelihood estimation method for random coefficient regression models, *Biometrika* **73**(3): 645–656.

Mallet, A., Mentre, F., Steimer, J.-L. and Lokiek, F. (1988). Nonparametric maximum likelihood estimation for population pharmacokinetics, with applications to Cyclosporine, *Journal of Pharmacokinetics and Biopharmaceutics* **16**: 311–327.

Matheron, G. (1962). *Traite de Geostatistique Appliquee*, Vol. I of *Memoires du Bureau de Recherches Geologiques et Minieres*, Editions Technip, Paris.

Milliken, G. A. and Johnson, D. E. (1992). *Analysis of Messy Data. Volume 1: Designed Experiments*, Chapman & Hall, London.

Patterson, H. D. and Thompson, R. (1971). Recovery of interblock information when block sizes are unequal, *Biometrika* **58**: 545–554.

Pierson, R. A. and Ginther, O. J. (1987). Follicular population dynamics during the estrus cycle of the mare, *Animal Reproduction Science* **14**: 219–231.

Pinheiro, J. C. (1994). *Topics in Mixed-Effects Models*, Ph.D. thesis, University of Wisconsin, Madison, WI.

Pinheiro, J. C. and Bates, D. M. (1995). Approximations to the log-likelihood function in the nonlinear mixed-effects model, *Journal of Computational and Graphical Statistics* **4**(1): 12–35.

Potthoff, R. F. and Roy, S. N. (1964). A generalized multivariate analysis of variance model useful especially for growth curve problems, *Biometrika* **51**: 313–326.

Potvin, C., Lechowicz, M. J. and Tardif, S. (1990). The statistical analysis of ecophysiological response curves obtained from experiments involving repeated measures, *Ecology* **71**: 1389–1400.

Ramos, R. Q. and Pantula, S. G. (1995). Estimation of nonlinear random coefficient models, *Statistics & Probability Letters* **24**: 49–56.

Sakamoto, Y., Ishiguro, M. and Kitagawa, G. (1986). *Akaike Information Criterion Statistics*, Reidel, Dordrecht, Holland.

Schwarz, G. (1978). Estimating the dimension of a model, *Annals of Statistics* **6**: 461–464.

Searle, S. R., Casella, G. and McCulloch, C. E. (1992). *Variance Components*, Wiley, New York.

Seber, G. A. F. and Wild, C. J. (1989). *Nonlinear Regression*, Wiley, New York.

Self, S. G. and Liang, K. Y. (1987). Asymptotic properties of maximum likelihood estimators and likelihood ratio tests under nonstandard conditions, *Journal of the American Statistical Association* **82**: 605–610.

Sheiner, L. B. and Beal, S. L. (1980). Evaluation of methods for estimating population pharmacokinetic parameters. I. Michaelis–Menten model: Routine clinical pharmacokinetic data, *Journal of Pharmacokinetics and Biopharmaceutics* **8**(6): 553–571.

Snedecor, G. W. and Cochran, W. G. (1980). *Statistical Methods*, 7th ed., Iowa State University Press, Ames, IA.

Soo, Y.-W. and Bates, D. M. (1992). Loosely coupled nonlinear least squares, *Computational Statistics and Data Analysis* **14**: 249–259.

Stram, D. O. and Lee, J. W. (1994). Variance components testing in the longitudinal mixed-effects models, *Biometrics* **50**: 1171–1177.

Stroup, W. W. and Baenziger, P. S. (1994). Removing spatial variation from wheat yield trials: a comparison of methods, *Crop Science* **34**: 62–66.

Thisted, R. A. (1988). *Elements of Statistical Computing*, Chapman & Hall, London.

Tierney, L. and Kadane, J. B. (1986). Accurate approximations for posterior moments and densities, *Journal of the American Statistical Association* **81**(393): 82–86.

Venables, W. N. and Ripley, B. D. (1999). *Modern Applied Statistics with S-PLUS*, 3rd ed., Springer-Verlag, New York.

Verme, C. N., Ludden, T. M., Clementi, W. A. and Harris, S. C. (1992). Pharmacokinetics of quinidine in male patients: A population analysis, *Clinical Pharmacokinetics* **22**: 468–480.

Vonesh, E. F. and Carter, R. L. (1992). Mixed-effects nonlinear regression for unbalanced repeated measures, *Biometrics* **48**: 1–18.

Vonesh, E. F. and Chinchilli, V. M. (1997). *Linear and Nonlinear Models for the Analysis of Repeated Measures*, Marcel Dekker, New York.

Wakefield, J. (1996). The Bayesian analysis of population pharmacokinetic models, *Journal of the American Statistical Association* **91**: 62–75.

Wilkinson, G. N. and Rogers, C. E. (1973). Symbolic description of factorial models for analysis of variance, *Applied Statistics* **22**: 392–399.

Wolfinger, R. D. (1993). Laplace's approximation for nonlinear mixed models, *Biometrika* **80**: 791–795.

Wolfinger, R. D. and Tobias, R. D. (1998). Joint estimation of location, dispersion, and random effects in robust design, *Technometrics* **40**: 62–71.

Yates, F. (1935). Complex experiments, *Journal of the Royal Statistical Society (Supplement)* **2**: 181–247.

Appendix A
Data Used in Examples and Exercises

We have used several sets of data in our examples and exercises. In this appendix we list all the data sets that are available as the NLMEDATA library included with the nlme 3.1 distribution and we describe in greater detail the data sets referenced in the text.

The title of each section in this appendix gives the name of the corresponding groupedData object from the nlme library, followed by a short description of the data. The formula stored with the data and a short description of each of the columns is also given.

We have adopted certain conventions for the ordering and naming of columns in these descriptions. The first column provides the response, the second column is the primary covariate, if present, and the next column is the primary grouping factor. Other covariates and grouping factors, if present, follow. Usually we use lowercase for the names of the response and the primary covariate. One exception to this rule is the name `Time` for a covariate. We try to avoid using the name `time` because it conflicts with a standard S function.

Table A.1 lists the groupedData objects in the `NLMEDATA` library that is part of the nlme distribution.

TABLE A.1: Data sets included with the nlme library distribution. The data sets whose names are shown in bold are described in this appendix.

Alfalfa	Yields of three varieties of alfalfa
Assay	Laboratory data on a biochemical assay
BodyWeight	Rat weight over time for different diets
CO2	Carbon dioxide uptake by grass plants
Cephamadole	Pharmacokinetic data
ChickWeight	Growth of chicks on different diets
Dialyzer	Performance of high-flux hemodialyzers
DNase	Assay of DNase
Earthquake	Severity of earthquakes
ergoStool	Ergometrics experiment with stool types
Fatigue	Metal fatique data
Gasoline	Gasoline yields for different crude samples
Glucose	Glucose levels over time
Glucose2	Glucose levels over time after alcohol ingestion
Gun	Naval gun firing data from Hicks (1993)
IGF	Assay data on Insulin-like Growth Factor
Indometh	Pharmacokinetic data on indomethicin
Loblolly	Growth of Loblolly pines
Machines	Productivity of workers on machines
MathAchSchool	School demographic data for MathAchieve
MathAchieve	Mathematics achievement scores
Meat	Tenderness of meat
Milk	Milk production by diet
Muscle	Muscle response by conc of $CaCl_2$
Nitrendipene	Assay of nitrendipene
Oats	Yield under different fertilizers
Orange	Growth of orange trees
Orthodont	Orthodontic measurement over time
Ovary	Number of large ovarian follicles over time
Oxboys	Heights of boys in Oxford, England
Oxide	Oxide coating on a semiconductor
PBG	Change in blood pressure vs. dose of phenylbiguanide
PBIB	A partially balanced incomplete block design
Phenobarb	Neonatal pharmacokinetics of phenobarbitol
Pixel	X-ray pixel intensities over time
Quinidine	Pharmacokinetic study of quinidine
Rail	Travel times of ultrasonic waves in railway rails
RatPupWeight	Weights of rat pups by litter
Relaxin	Assays of relaxin
Remifentanil	Pharmacokinetics of remifentanil
Soybean	Soybean growth by variety

TABLE A.1: (continued)

Spruce	Spruce tree growth
Tetracycline1	Pharmacokinetics of tetracycline
Tetracycline2	Pharmacokinetics of tetracycline
Theoph	Pharmacokinetics of theophylline
Wafer	Current vs. voltage on semiconductor wafers
Wheat	Yields by growing conditions
Wheat2	Yields from a randomized complete block design

Other data sets may be included with later versions of the library, which will be made available at `http://nlme.stat.wisc.edu`.

A.1 Alfalfa—Split-Plot Experiment on Varieties of Alfalfa

These data are described in Snedecor and Cochran (1980, §16.15) as an example of a split-plot design. The treatment structure used in the experiment was a 3×4 full factorial, with three varieties of alfalfa and four dates of third cutting in 1943. The experimental units were arranged into six blocks, each subdivided into four plots. The varieties of alfalfa (*Cossac*, *Ladak*, and *Ranger*) were assigned randomly to the blocks and the dates of third cutting (*None*, *S1*—September 1, *S20*—September 20, and *O7*—October 7) were randomly assigned to the plots. All four dates were used on each block. The data are presented in Figure A.1.

The display formula for these data is

```
Yield ~ Date | Block / Variety
```

based on the columns named:

`Yield`: the plot yield (T/acre).

`Date`: the third cutting date—None, S1, S20, or O7.

`Block`: a factor identifying the block—1 through 6.

`Variety`: alfalfa variety—Cossac, Ladak, or Ranger.

A.2 Assay—Bioassay on Cell Culture Plate

These data, courtesy of Rich Wolfe and David Lansky from Searle, Inc., come from a bioassay run on a 96-well cell culture plate. The assay is performed using a split-block design. The 8 rows on the plate are labeled A–H

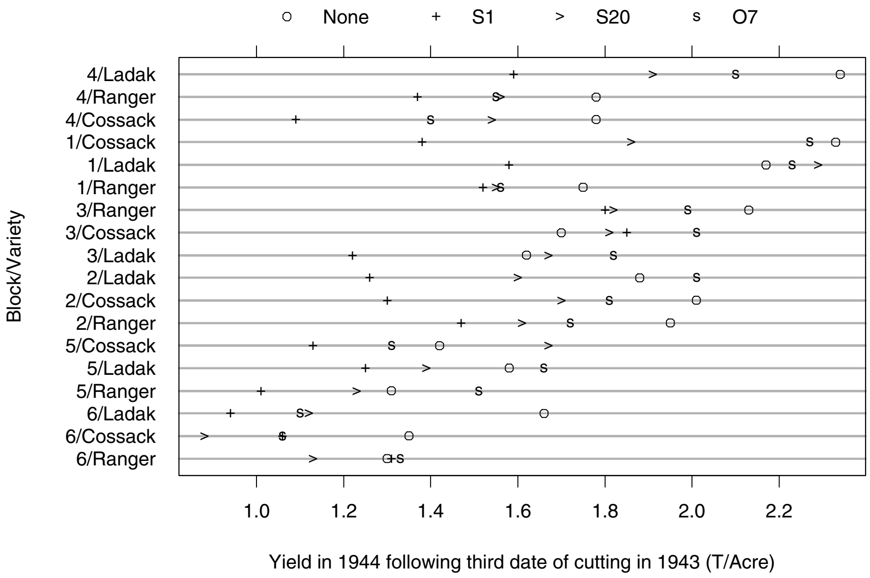

FIGURE A.1. Plot yields in a split-plot experiment on alfalfa varieties and dates of third cutting.

from top to bottom and the 12 columns on the plate are labeled 1–12 from left to right. Only the central 60 wells of the plate are used for the bioassay (the intersection of rows B–G and columns 2–11). There are two blocks in the design: Block 1 contains columns 2–6 and Block 2 contains columns 7–11. Within each block, six samples are assigned randomly to rows and five (serial) dilutions are assigned randomly to columns. The response variable is the logarithm of the optical density. The cells are treated with a compound that they metabolize to produce the stain. Only live cells can make the stain, so the optical density is a measure of the number of cells that are alive and healthy. The data are displayed in Figure 4.13 (p. 164).

Columns

The display formula for these data is

```
logDens ~ 1 | Block
```

based on the columns named:

`logDens`: log-optical density.

`Block`: a factor identifying the block where the wells are measured.

`sample`: a factor identifying the sample corresponding to the well, varying from “a” to “f.”

`dilut`: a factor indicating the dilution applied to the well, varying from 1 to 5.

A.3 BodyWeight—Body Weight Growth in Rats

Hand and Crowder (1996) describe data on the body weights of rats measured over 64 days. These data also appear in Table 2.4 of Crowder and Hand (1990). The body weights of the rats (in grams) are measured on day 1 and every seven days thereafter until day 64, with an extra measurement on day 44. The experiment started several weeks before "day 1." There are three groups of rats, each on a different diet. A plot of the data is presented in Figure 3.2 (p. 104).

Columns

The display formula for these data is

```
weight ~ Time | Rat
```

based on the columns named:

`weight`: body weight of the rat (grams).

`Time`: time at which the measurement is made (days).

`Rat`: a factor identifying the rat whose weight is measured.

`Diet`: a factor indicating the diet the rat receives.

A.4 Cefamandole—Pharmacokinetics of Cefamandole

Davidian and Giltinan (1995, §1.1, p. 2) describe data, shown in Figure A.2, obtained during a pilot study to investigate the pharmacokinetics of the drug cefamandole. Plasma concentrations of the drug were measured on six healthy volunteers at 14 time points following an intraveneous dose of 15 mg/kg body weight of cefamandole.

Columns

The display formula for these data is

```
conc ~ Time | Subject
```

based on the columns named:

`conc`: observed plasma concentration of cefamandole (mcg/ml).

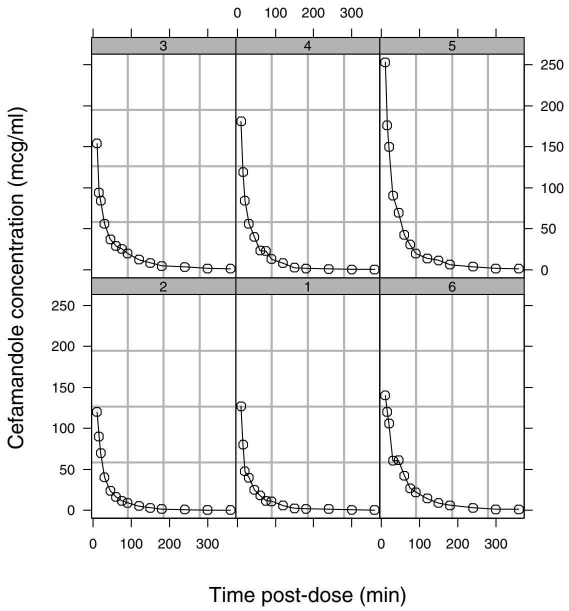

FIGURE A.2. Plasma concentration of cefamandole versus time post-injection for six healthy volunteers.

`Time`: time at which the sample was drawn (minutes post-injection).

`Subject`: a factor giving the subject from which the sample was drawn.

Models

Davidian and Giltinan (1995) use the biexponential model SSbiexp (§C.4, p. 514) with these data.

A.5 CO2—Carbon Dioxide Uptake

Potvin et al. (1990) describe an experiment on the cold tolerance of a C_4 grass species, *Echinochloa crus-galli*. The CO_2 uptake of six plants from Québec and six plants from Mississippi was measured at several levels of ambient CO_2 concentration. Half the plants of each type were chilled overnight before the experiment was conducted. The data are shown in Figure 8.15 (p. 369).

Columns

The display formula for these data is

```
uptake ~ conc | Plant
```

based on the columns named:

`uptake`: carbon dioxide uptake rate (μmol/m^2 sec).

`conc`: ambient concentration of carbon dioxide (mL/L).

`Plant`: a factor giving a unique identifier for each plant.

`Type`: origin of the plant, Québec or Mississippi.

`Treatment`: treatment, chilled or nonchilled.

Models

Potvin et al. (1990) suggest using a modified form of the asymptotic regression model `SSasymp` (§C.1, p. 511), which we have coded as `SSasympOff` (§C.2, p. 512).

A.6 Dialyzer—High-Flux Hemodialyzer

Vonesh and Carter (1992) describe data measured on high-flux hemodialyzers to assess their *in vivo* ultrafiltration characteristics. The ultrafiltration rates (in mL/hr) of 20 high-flux dialyzers were measured at seven different transmembrane pressures (in dmHg). The *in vitro* evaluation of the dialyzers used bovine blood at flow rates of either 200 dl/min or 300 dl/min. The data, shown in Figure 5.1 (p. 215), are also analyzed in Littell et al. (1996, §8.2).

Columns

The display formula for these data is

```
rate ~ pressure | Subject
```

based on the columns named:

`rate`: hemodialyzer ultrafiltration rate (mL/hr).

`pressure`: transmembrane pressure (dmHg).

`Subject`: a factor giving a unique identifier for each subject.

`QB`: bovine blood flow rate (dL/min)—200 or 300.

`index`: index of observation within subject—1 through 7.

A.7 DNase—Assay Data for the Protein DNase

Davidian and Giltinan (1995, §5.2.4, p. 134) describe data, shown in Figure 3.8 (p. 115), obtained during the development of an ELISA assay for the recombinant protein DNase in rat serum.

Columns

The display formula for these data is

```
density ~ conc | Run
```

based on the columns named:

`density`: the measured optical density in the assay. Duplicate optical density measurements were obtained.

`conc`: the known concentration of the protein.

`Run`: a factor giving the run from which the data were obtained.

Models

Davidian and Giltinan (1995) use the four-parameter logistic model, SSfpl (§C.6, p. 517) with these data, modeling the optical density as a logistic function of the logarithm of the concentration.

A.8 Earthquake—Earthquake Intensity

These data, shown in Figure A.3, are measurements recorded at available seismometer locations for 23 large earthquakes in western North America between 1940 and 1980. They were originally given in Joyner and Boore (1981); are mentioned in Brillinger (1987); and are analyzed in §11.4 of Davidian and Giltinan (1995).

Columns

The display formula for these data is

```
accel ~ distance | Quake
```

based on the columns named:

`accel`: maximum horizontal acceleration observed (g).

`distance`: the distance from the seismological measuring station to the epicenter of the earthquake (km).

`Quake`: a factor indicating the earthquake on which the measurements were made.

`Richter`: the intensity of the earthquake on the Richter scale.

`soil`: soil condition at the measuring station—either soil or rock.

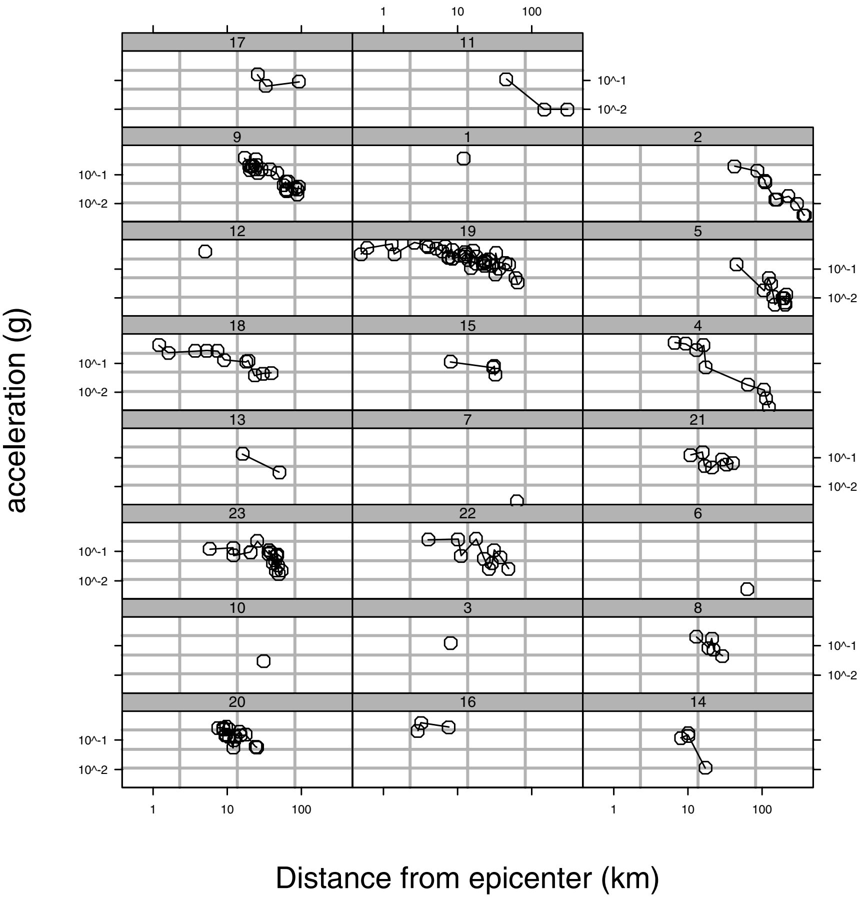

FIGURE A.3. Lateral acceleration versus distance from the epicenter for 23 large earthquakes in western North America. Both the acceleration and the distance are on a logarithmic scale. Earthquakes of greatest intensity as measured on the Richter scale are in the uppermost panels.

A.9 ergoStool—Ergometrics Experiment with Stool Types

Devore (2000, Exercise 11.9, p. 447) cites data from an article in *Ergometrics* (1993, pp. 519-535) on "The Effects of a Pneumatic Stool and a One-Legged Stool on Lower Limb Joint Load and Muscular Activity." These data are shown in Figure 1.5 (p. 13).

The display formula for these data is

```
effort ~ Type | Subject
```

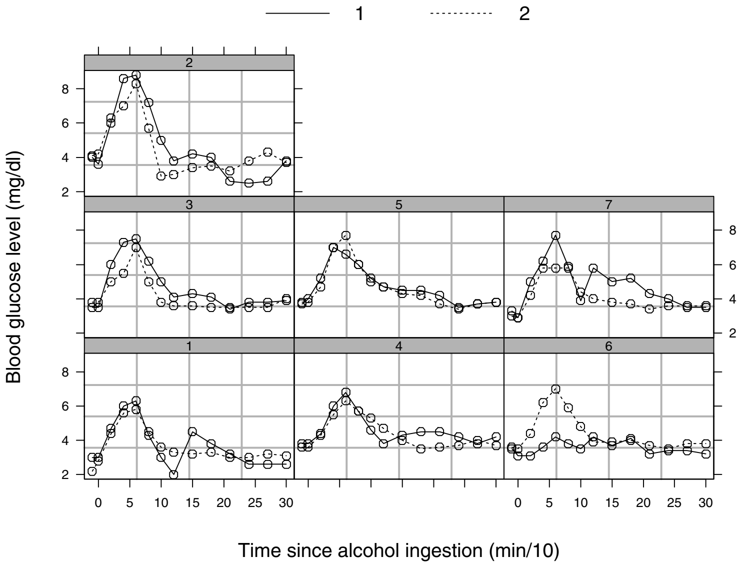

FIGURE A.4. Blood glucose levels of seven subjects measured over a period of 5 hours on two different occasions. In both dates the subjects took alcohol at time 0, but on the second occasion a dietary additive was used.

based on the columns named:

`effort`: effort to arise from a stool

`Type`: a factor giving the stool type

`Subject`: a factor giving a unique identifier for the subject in the experiment

A.10 Glucose2—Glucose Levels Following Alcohol Ingestion

Hand and Crowder (1996, Table A.14, pp. 180–181) describe data on the blood glucose levels measured at 14 time points over 5 hours for 7 volunteers who took alcohol at time 0. The same experiment was repeated on a second date with the same subjects but with a dietary additive used for all subjects. A plot of the data is presented in Figure A.4.

Columns

The display formula for these data is

```
glucose ~ Time | Subject/Date
```

based on the columns named:

`weight`: blood glucose level (in mg/dl).

`Time`: time since alcohol ingestion (in min/10).

`Subject`: a factor identifying the subject whose glucose level is measured.

`Date`: a factor indicating the occasion in which the experiment was conducted.

A.11 IGF—Radioimmunoassay of IGF-I Protein

Davidian and Giltinan (1995, §3.2.1, p. 65) describe data, shown in Figure 4.6 (p. 144), obtained during quality control radioimmunoassays for ten different lots of radioactive tracer used to calibrate the Insulin-like Growth Factor (IGF-I) protein concentration measurements.

Columns

The display formula for these data is

```
conc ~ age | Lot
```

based on the columns named:

`conc`: the estimated concentration of IGF-I protein, in ng/ml.

`age`: the age (in days) of the radioactive tracer.

`Lot`: a factor giving the radioactive tracer lot.

A.12 Indometh—Indomethicin Kinetics

Kwan et al. (1976) present data on the plasma concentrations of indomethicin following intravenous injection. There are six different subjects in the experiment. The sampling times, ranging from 15 minutes post-injection to 8 hours post-injection, are the same for each subject. The data, presented in Figure 6.3 (p. 277), are analyzed in Davidian and Giltinan (1995, §2.1)

The display formula for these data is

```
conc ~ time | Subject
```

based on the columns named:

`conc`: observed plasma concentration of indomethicin (mcg/ml).

`time`: time at which the sample was drawn (hours post-injection).

`Subject`: a factor indicating the subject from whom the sample is drawn.

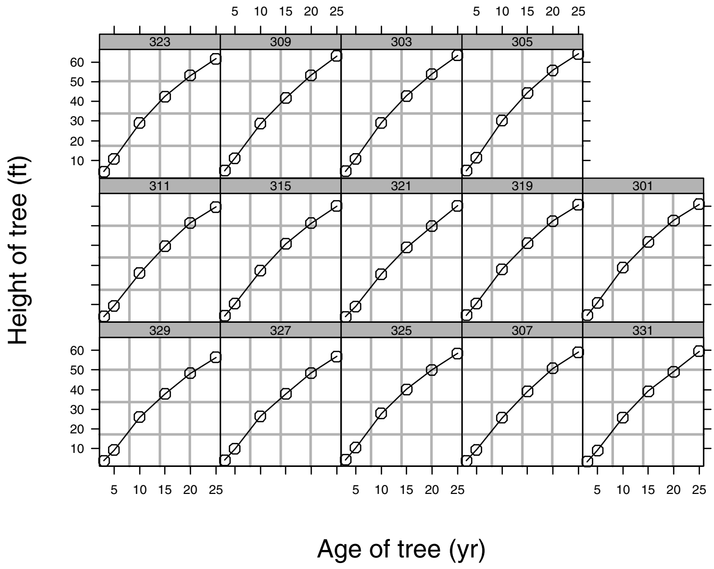

FIGURE A.5. Height of Loblolly pine trees over time

Models

Davidian and Giltinan (1995) use the biexponential model SSbiexp (§C.4, p. 514) with these data.

A.13 Loblolly—Growth of Loblolly Pine Trees

Kung (1986) presents data, shown in Figure A.5, on the growth of Loblolly pine trees.

The display formula for these data is

```
height ~ age | Seed
```

based on the columns named:

`height`: height of the tree (ft).

`age`: age of the tree (yr).

`Seed`: a factor indicating the seed source for the tree.

A.14 Machines—Productivity Scores for Machines and Workers

Data on an experiment to compare three brands of machines used in an industrial process are presented in Milliken and Johnson (1992, §23.1, p. 285). Six workers were chosen randomly among the employees of a factory to operate each machine three times. The response is an overall productivity score taking into account the number and quality of components produced. These data, shown in Figure 1.9 (p. 22), are analyzed in Milliken and Johnson (1992) with an ANOVA model.

The display formula for these data is

```
score ~ Machine | Worker
```

based on the columns named:

`score`: productivity score.

`Machine`: a factor identifying the machine brand—A, B, or C.

`Worker`: a factor giving the unique identifier for each worker.

A.15 Oats—Split-plot Experiment on Varieties of Oats

These data have been introduced by Yates (1935) as an example of a split-plot design. The treatment structure used in the experiment was a 3×4 full factorial, with three varieties of oats and four concentrations of nitrogen. The experimental units were arranged into six blocks, each with three whole-plots subdivided into four subplots. The varieties of oats were assigned randomly to the whole-plots and the concentrations of nitrogen to the subplots. All four concentrations of nitrogen were used on each whole-plot.

The data, presented in Figure 1.20 (p. 47), are analyzed in Venables and Ripley (1999, §6.11).

The display formula for these data is

```
yield ~ nitro | Block
```

based on the columns named:

`yield`: the subplot yield (bushels/acre).

`nitro`: nitrogen concentration (cwt/acre)—0.0, 0.2, 0.4, or 0.6.

`Block`: a factor identifying the block—I through VI.

`Variety`: oats variety—Golden Rain, Marvellous, or Victory.

A.16 Orange—Growth of Orange Trees

Draper and Smith (1998, Exercise 24.N, p. 559) present data on the growth of a group of orange trees. These data are plotted in Figure 8.1 (p. 339).

The display formula for these data is

```
circumference ~ age | Tree
```

based on the columns named:

`circumference`: circumference of the tree (mm)

`age`: time in days past the arbitrary origin of December 31, 1968.

`Tree`: a factor identifying the tree on which the measurement is made.

Models

The logistic growth model, `SSlogis` (§C.7, p. 519) provides a reasonable fit to these data.

A.17 Orthodont—Orthodontic Growth Data

Investigators at the University of North Carolina Dental School followed the growth of 27 children (16 males, 11 females) from age 8 until age 14. Every two years they measured the distance between the pituitary and the pterygomaxillary fissure, two points that are easily identified on x-ray exposures of the side of the head. These data are reported in Potthoff and Roy (1964) and plotted in Figure 1.11 (p. 31).

The display formula for these data is

```
distance ~ age | Subject
```

based on the columns named:

`distance`: the distance from the center of the pituitary to the pterygomaxillary fissure (mm).

`age`: the age of the subject when the measurement is made (years).

`Subject`: a factor identifying the subject on whom the measurement was made.

`Sex`: a factor indicating if the subject is male or female.

Models:

Based on the relationship shown in Figure 1.11 we begin with a simple linear relationship between distance and age

A.18 Ovary—Counts of Ovarian Follicles

Pierson and Ginther (1987) report on a study of the number of large ovarian follicles detected in different mares at several times in their estrus cycles. These data are shown in Figure 5.10 (p. 240).

The display formula for these data is

```
follicles ~ Time | Mare
```

based on the columns named:

`follicles`: the number of ovarian follicles greater than 10 mm in diameter.

`Time`: time in the estrus cycle. The data were recorded daily from 3 days before ovulation until 3 days after the next ovulation. The measurement times for each mare are scaled so that the ovulations for each mare occur at times 0 and 1.

`Mare`: a factor indicating the mare on which the measurement is made.

A.19 Oxboys—Heights of Boys in Oxford

These data are described in Goldstein (1987) as data on the height of a selection of boys from Oxford, England versus a standardized age. We display the data in Figure 3.1 (p. 99).

The display formula for these data is

```
height ~ age | Subject
```

based on the columns named:

`height`: height of the boy (cm)

`age`: standardized age (dimensionless)

`Subject`: a factor giving a unique identifier for each boy in the experiment

`Occasion`: an ordered factor—the result of converting `age` from a continuous variable to a count so these slightly unbalanced data can be analyzed as balanced.

A.20 Oxide—Variability in Semiconductor Manufacturing

These data are described in Littell et al. (1996, §4.4, p. 155) as coming "from a passive data collection study in the semiconductor industry where the objective is to estimate the variance components to determine the assignable causes of the observed variability." The observed response is the thickness of the oxide layer on silicon wafers, measured at three different sites of each of three wafers selected from each of eight lots sampled from the population of lots. We display the data in Figure 4.14 (p. 168).

The display formula for these data is

```
Thickness ~ 1 | Lot/Wafer
```

based on the columns named:

`Thickness`: thickness of the oxide layer.

`Lot`: a factor giving a unique identifier for each lot.

`Wafer`: a factor giving a unique identifier for each wafer within a lot.

A.21 PBG—Effect of Phenylbiguanide on Blood Pressure

Data on an experiment to examine the effect of a antagonist MDL 72222 on the change in blood pressure experienced with increasing dosage of phenylbiguanide are described in Ludbrook (1994) and analyzed in Venables and Ripley (1999, §8.8). Each of five rabbits was exposed to increasing doses of phenylbiguanide after having either a placebo or the HD_5-antagonist MDL 72222 administered. The data are shown in Figure 3.4 (p. 107).

The display formula for these data is

```
deltaBP ~ dose | Rabbit
```

based on the columns named:

`deltaBP`: change in blood pressure (mmHg).

`dose`: dose of phenylbiguanide (μg).

`Rabbit`: a factor identifying the test animal.

`Treatment`: a factor identifying whether the observation was made after administration of `placebo` or the HD_5-antagonist `MDL 72222`.

Models

The form of the response suggests a logistic model `SSlogis` (§C.7, p. 519) for the change in blood pressure as function of the logarithm of the concentration of PBG.

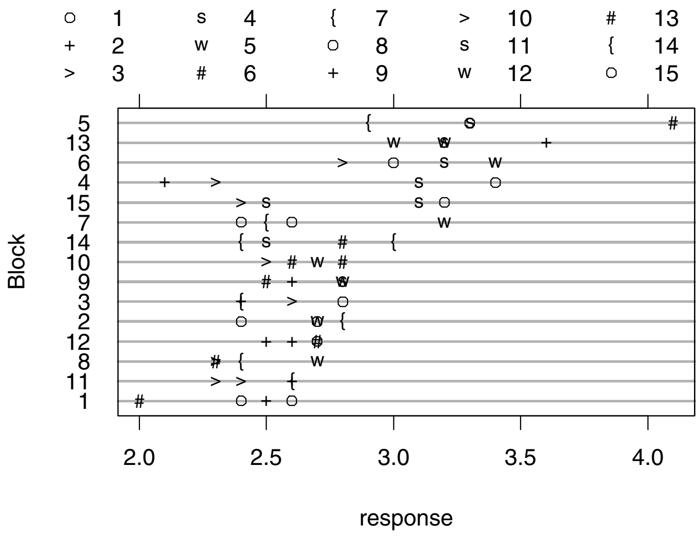

FIGURE A.6. Data on the response in an experiment conducted using fifteen treatments in fifteen blocks of size four. The responses are shown by block with different characters indicating different treatments.

A.22 PBIB—A Partially Balanced Incomplete Block Design

Data from a partially balanced incomplete block design in which there were fifteen treatments used in fifteen blocks of size four. The blocking is incomplete in that only a subset of the treatments can be used in each block. It is partially balanced in that every pair of treatments occurs together in a block the same number of times.

These data were described in Cochran and Cox (1957, p. 456). They are also used as data set 1.5.1 in Littell et al. (1996, §1.5.1). The data are shown in Figure A.6.

The display formula for these data is

```
response ~ Treatment | Block
```

based on the columns named:

`response`: the continuous response in the experiment

`Treatment`: the treatment factor

`Block`: the block

A.23 Phenobarb—Phenobarbitol Kinetics

Data from a pharmacokinetics study of phenobarbital in neonatal infants. During the first few days of life the infants receive multiple doses of phenobarbital for prevention of seizures. At irregular intervals blood samples are drawn and serum phenobarbital concentrations are determined. The data, displayed in Figure 6.15 (p. 296), were originally given in Grasela and Donn (1985) and are analyzed in Boeckmann et al. (1994) and in Davidian and Giltinan (1995, §6.6).

The display formula for these data is

```
conc ~ time | Subject
```

based on the columns named:

`conc`: phenobarbital concentration in the serum (μg/L).

`time`: time when the sample is drawn or drug administered (hr).

`Subject`: a factor identifying the infant.

`Wt`: birth weight of the infant (kg).

`Apgar`: the 5-minute Apgar score for the infant. This is an indication of health of the newborn infant. The scale is $1 - 10$.

`ApgarInd`: a factor indicating whether the 5-minute Apgar score is < 5 or ≥ 5.

`dose`: dose of drug administered (μg/kg).

Models

A one-compartments open model with intravenous administration and first-order elimination, described in §6.4, is used for these data

A.24 Pixel—Pixel Intensity in Lymphnodes

These data are from an experiment conducted by Deborah Darien, Department of Medical Sciences, School of Veterinary Medicine, University of Wisconsin, Madison. The mean pixel intensity of the right and left lymphnodes in the axillary region obtained from CT scans of 10 dogs were recorded over a period of 14 days after intravenous application of a contrast. The data are shown in Figure 1.17 (p. 42).

The display formula for these data is

```
pixel ~ day | Dog
```

based on the columns named:

`pixel`: mean pixel intensity of lymphnode in the CT scan.

`day`: number of days since contrast administration.

`Dog`: a factor giving the unique identifier for each dog.

`Side`: a factor indicating the side on which the measurement was made.

A.25 Quinidine—Quinidine Kinetics

Verme, Ludden, Clementi and Harris (1992) analyze routine clinical data on patients receiving the drug quinidine as a treatment for cardiac arrythmia (atrial fibrillation of ventricular arrythmias). All patients were receiving oral quinidine doses. At irregular intervals blood samples were drawn and serum concentrations of quinidine were determined. These data, shown in Figure A.7, are analyzed in several publications, including Davidian and Giltinan (1995, §9.3).

The display formula for these data is

```
conc ~ time | Subject
```

based on the columns named:

`conc`: serum quinidine concentration (mg/L).

`time`: time (hr) at which the drug was administered or the blood sample drawn. This is measured from the time the patient entered the study.

`Subject`: a factor identifying the patient on whom the data were collected.

`dose`: dose of drug administered (mg). Although there were two different forms of quinidine administered, the doses were adjusted for differences in salt content by conversion to milligrams of quinidine base.

`interval`: when the drug has been given at regular intervals for a sufficiently long period of time to assume steady state behavior, the interval is recorded.

`Age`: age of the subject on entry to the study (yr).

`Height`: height of the subject on entry to the study (in.).

`Weight`: body weight of the subject (kg).

`Race`: a factor identifying the race—Caucasian, Black, or Latin.

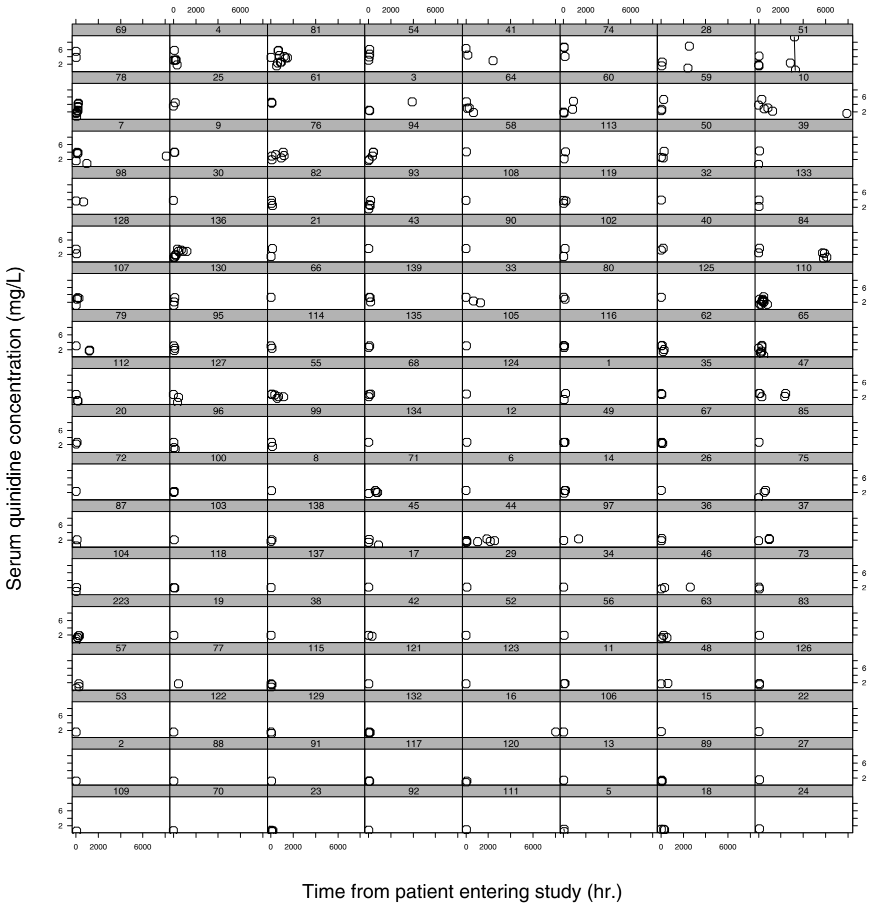

FIGURE A.7. Serum concentrations of quinidine in 136 hospitalized patients under varying dosage regimens versus time since entering the study.

`Smoke`: a factor giving smoking status at the time of the measurement—no or yes.

`Ethanol`: a factor giving ethanol (alcohol) abuse status at the time of the measurement—none, current, or former.

`Heart`: a factor indicating congestive heart failure for the subject—none/mild, moderate, or severe.

`Creatinine`: a factor in eight levels coding the creatinine clearance and other measurements. Creatinine clearance is divided into those greater than 50 mg/min and those less than 50 mg/min.

`glyco`: alpha-1 acid glycoprotein concentration (mg/dL). Often measured at the same time as the quinidine concentration.

Models

A model for these data is described in §8.2.2.

A.26 Rail—Evaluation of Stress in Rails

Devore (2000, Example 10.10, p. 427) cites data from an article in *Materials Evaluation* on "a study of travel time for a certain type of wave that results from longitudinal stress of rails used for railroad track." The data are displayed in Figure 1.1 (p. 4).

The display formula for these data is

```
travel ~ 1 | Rail
```

based on the columns named:

`travel`: travel time for ultrasonic head-waves in the rail (nanoseconds). The value given is the original travel time minus 36,100 nanoseconds.

`Rail`: a factor giving the number of the rail on which the measurement was made.

A.27 Soybean—Soybean Leaf Weight over Time

These data, shown in Figure 6.10 (p. 288), are described in Davidian and Giltinan (1995, §1.1.3, p. 7) as "Data from an experiment to compare growth patterns of two genotypes of soybeans: Plant Introduction #416937 (P), an experimental strain, and Forrest (F), a commercial variety."

The display formula for these data is

```
weight ~ Time | Plot
```

based on the columns named:

`weight`: average leaf weight per plant (g).

`Time`: time the sample was taken (days after planting).

`Plot`: a factor giving a unique identifier for each plot.

`Variety`: a factor indicating the variety; Forrest (F) or Plant Introduction #416937 (P)

`Year`: the year the plot was planted.

Models

The form of the response suggests a logistic model, `SSlogis` (§C.7, p. 519).

A.28 Spruce—Growth of Spruce Trees

Diggle et al. (1994, Example 1.3, page 5) describe data on the growth of spruce trees that have been exposed to an ozone-rich atmosphere or to a normal atmosphere. These data are plotted in Figures A.8–A.10. The display formula for these data is

```
logSize ~ days | Tree
```

based on the columns named:

`logSize`: the logarithm of an estimate of the volume of the tree trunk

`days`: number of days since the beginning of the experiment

`Tree`: a factor giving a unique identifier for each tree

`Plot`: a factor identifying the plot in which the tree was grown. The levels of this factor are Ozone1, Ozone2, Normal1, and Normal2.

`Treatment` a factor indicating whether the tree was grown in an ozone-rich atmosphere or a normal atmosphere.

A.29 Theoph—Theophylline Kinetics

Boeckmann et al. (1994) report data from a study by Dr. Robert Upton of the kinetics of the anti-asthmatic drug theophylline. Twelve subjects were given oral doses of theophylline then serum concentrations were measured at 11 time points over the next 25 hours. Davidian and Giltinan (1995) also analyze these data, shown in Figure 8.6 (p. 352).

The display formula for these data is

```
conc ~ Time | Subject
```

based on the columns named:

`conc`: theophylline concentration in the sample (mg/L).

`Time`: time since drug administration when the sample was drawn (hr).

`Subject`: a factor identifying the subject.

`Wt`: weight of the subject (kg).

`Dose`: dose administered to the subject (mg/kg).

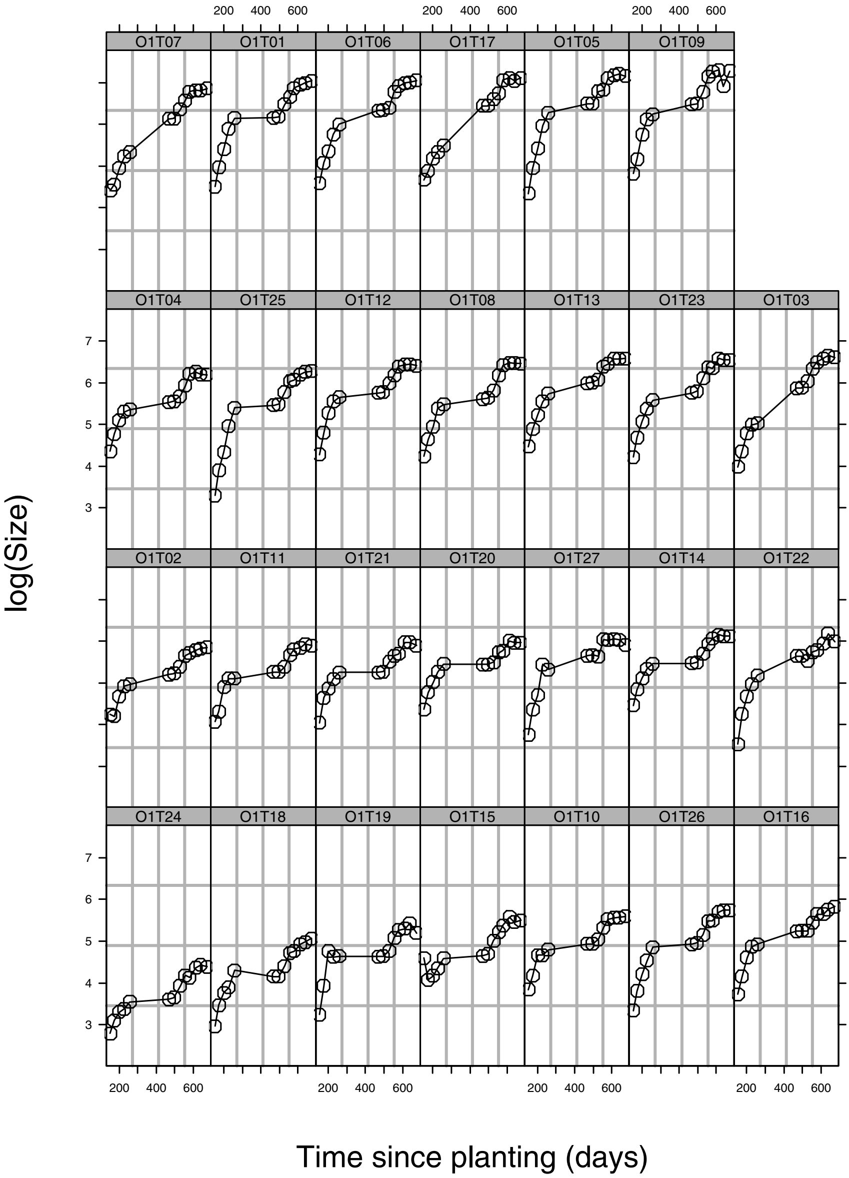

FIGURE A.8. Growth measures in the logarithm of an estimate of the volume of the spruce tree trunk versus time. These 27 trees were in the first plot that was exposed to an ozone-rich atmosphere throughout the experiment

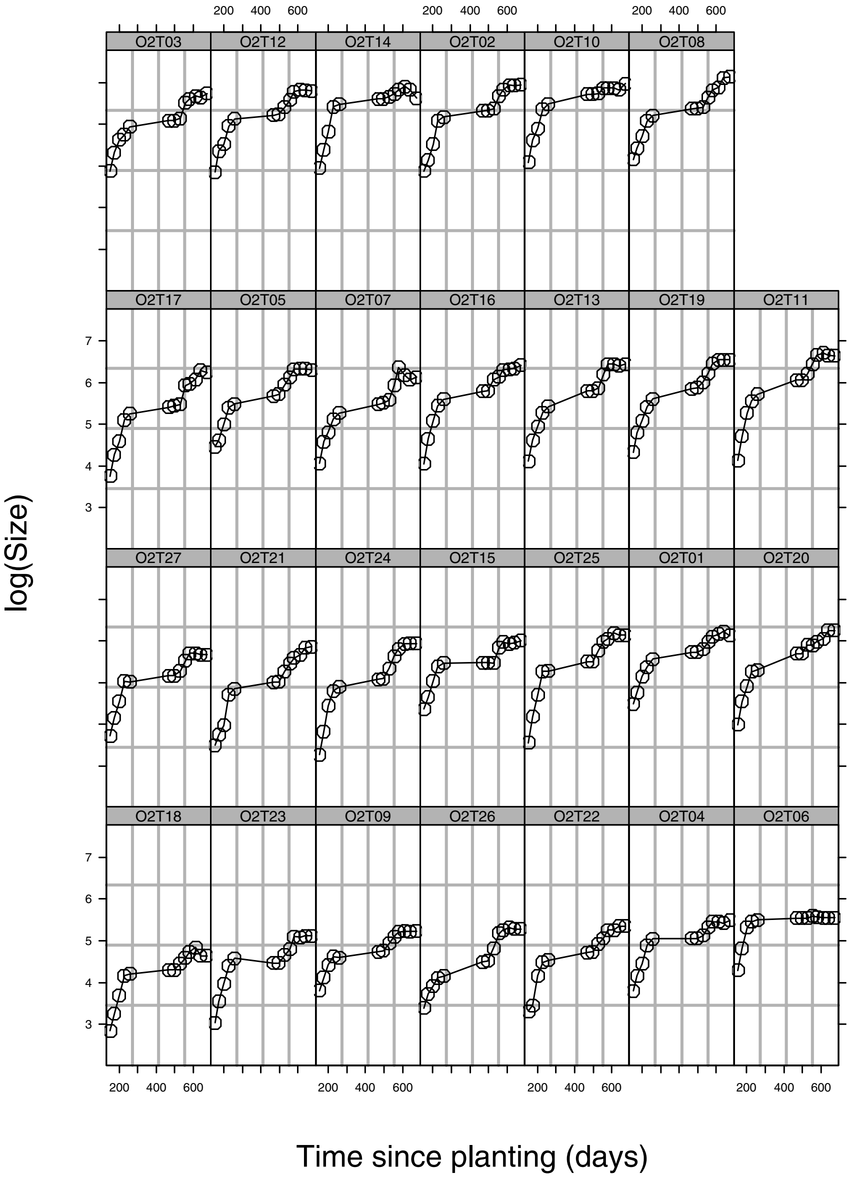

FIGURE A.9. Growth measures in the logarithm of an estimate of the volume of the spruce tree trunk versus time. These 27 trees were in the second plot that was exposed to an ozone-rich atmosphere throughout the experiment

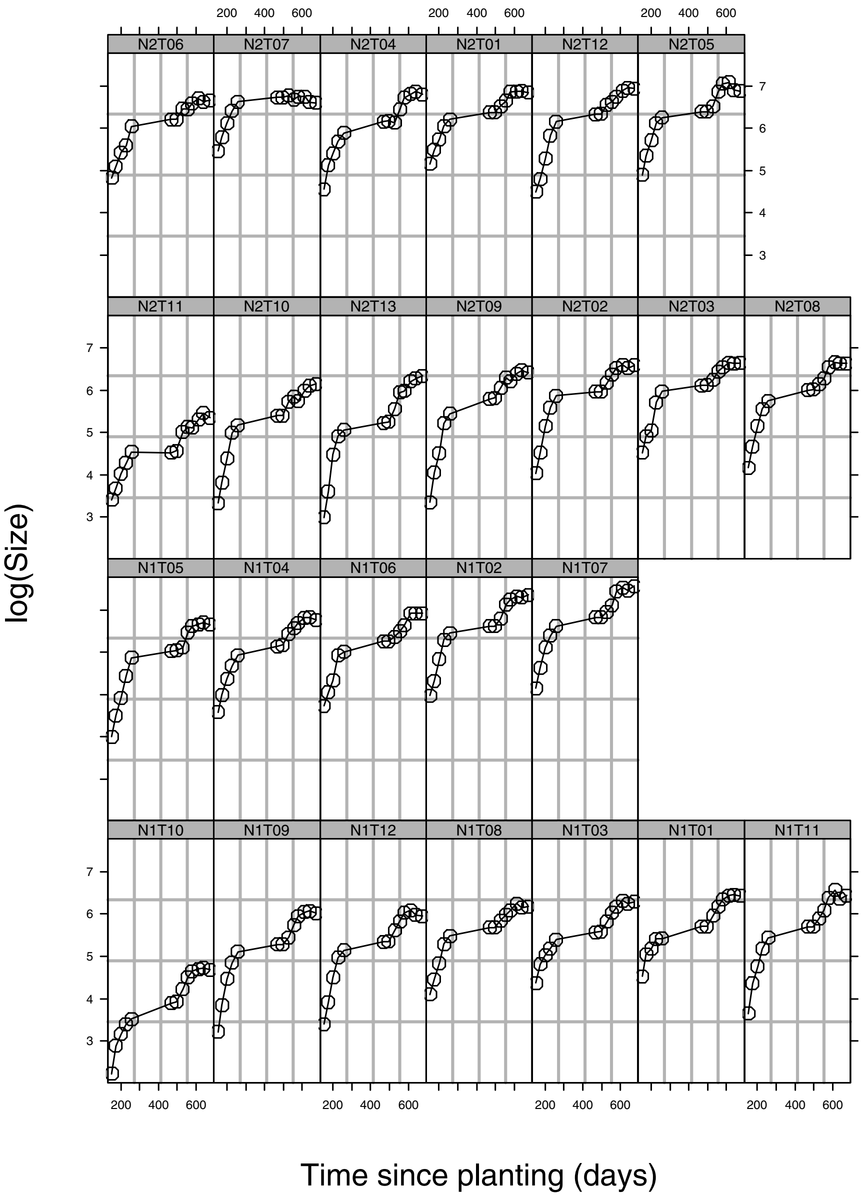

FIGURE A.10. Growth measures in the logarithm of an estimate of the volume of the spruce tree trunk versus time. These 25 trees were in the first and second plots that were exposed to an normal atmosphere throughout the experiment

Models:

Both Boeckmann et al. (1994) and Davidian and Giltinan (1995) use a two-compartment open pharmacokinetic model, which we code as SSfol (§C.5, p. 516), for these data.

A.30 Wafer—Modeling of Analog MOS Circuits

In an experiment conducted at the Microelectronics Division of Lucent Technologies to study the variability in the manufacturing of analog MOS circuits, the intensities of the current at five ascending voltages were collected on n-channel devices. Measurements were made on eight sites of each of ten wafers. Figure 3.11 (p. 118) shows the response curves for each site, by wafer.

The display formula for these data is

```
current ~ voltage | Wafer/Site
```

based on the columns named:

`current`: the intensity of current (mA).

`voltage`: the voltage applied to the device (V).

`Wafer`: a factor giving a unique identifier for each wafer.

`Site`: a factor giving an identifier for each site within a wafer.

A.31 Wheat2—Wheat Yield Trials

Stroup and Baenziger (1994) report data on an agronomic yield trial to compare 56 different varieties of wheat. The experimental units were organized according to a randomized complete block design with four blocks. All 56 varieties of wheat were used in each block. The latitude and longitude of each experimental unit in the trial were also recorded. The data, shown in Figure 5.22 (p. 261), are also analyzed in Littell et al. (1996, §9.6.2).

Columns

The display formula for these data is

```
yield ~ variety | Block
```

based on the columns named:

`yield`: wheat yield.

`variety`: a factor giving the unique identifier for each wheat variety.

`Block`: a factor giving a unique identifier for each block in the experiment.

`latitude`: latitude of the experimental unit.

`longitude`: longitude of the experimental unit.

Appendix B
S Functions and Classes

There are over 300 different functions and classes defined in the nlme library. In this appendix we reproduce the on-line documentation for those functions and classes that are most frequently used in the examples in the text. The documentation for all the functions and classes in the library is available with the library.

`ACF` *Autocorrelation Function*

```
ACF(object, maxLag, ...)
```

Arguments

`object`	Any object from which an autocorrelation function can be obtained. Generally an object resulting from a model fit, from which residuals can be extracted.
`maxLag`	Maximum lag for which the autocorrelation should be calculated.
`...`	Some methods for this generic require additional arguments.

Description

This function is generic; method functions can be written to handle specific classes of objects. Classes that already have methods for this function include gls and lme.

Value

Will depend on the method function used; see the appropriate documentation.

See Also

`ACF.gls`, `ACF.lme`

`ACF.lme` *Autocorrelation Function for `lme` Residuals*

```
ACF(object, maxLag, resType)
```

Arguments

`object`	An object inheriting from class `lme`, representing a fitted linear mixed-effects model.
`maxLag`	An optional integer giving the maximum lag for which the autocorrelation should be calculated. Defaults to maximum lag in the within-group residuals.
`resType`	An optional character string specifying the type of residuals to be used. If `"response"`, the “raw” residuals (observed – fitted) are used; else, if `"pearson"`, the standardized residuals (raw residuals divided by the corresponding standard errors) are used; else, if `"normalized"`, the normalized residuals (standardized residuals premultiplied by the inverse square-root factor of the estimated error correlation matrix) are used. Partial matching of arguments is used, so only the first character needs to be provided. Defaults to `"pearson"`.

Description

This method function calculates the empirical autocorrelation function (Box et al., 1994) for the within-group residuals from an `lme` fit. The autocorrelation values are calculated using pairs of residuals within the innermost group level. The autocorrelation function is useful for investigating serial correlation models for equally spaced data.

Value

A data frame with columns `lag` and `ACF` representing, respectively, the lag between residuals within a pair and the corresponding empirical autocorrelation. The returned value inherits from class `ACF`.

See Also

`ACF.gls`, `plot.ACF`

Examples

```
fm1 <- lme(follicles ~ sin(2*pi*Time) + cos(2*pi*Time), Ovary,
           random = ~ sin(2*pi*Time) | Mare)
ACF(fm1, maxLag = 11)
```

`anova.lme` *Compare Likelihoods of Fitted Objects*

```
anova(object, ..., test, type, adjustSigma, Terms, L,
      verbose)
```

Arguments

`object`	A fitted model object inheriting from class `lme`, representing a mixed-effects model.
`...`	Other optional fitted model objects inheriting from classes `gls`, `gnls`, `lm`, `lme`, `lmList`, `nlme`, `nlsList`, or `nls`.
`test`	An optional logical value controlling whether likelihood ratio tests should be used to compare the fitted models represented by `object` and the objects in `...`. Defaults to `TRUE`.
`type`	An optional character string specifying the type of sum of squares to be used in F-tests for the terms in the model. If `"sequential"`, the sequential sum of squares obtained by including the terms in the order they appear in the model is used; else, if `"marginal"`, the marginal sum of squares obtained by deleting a term from the model at a time is used. This argument is only used when a single fitted object is passed to the function. Partial matching of arguments is used, so only the first character needs to be provided. Defaults to `"sequential"`.
`adjustSigma`	An optional logical value. If `TRUE` and the estimation method used to obtain `object` was maximum likelihood, the residual standard error is multiplied by $\sqrt{n_{\text{obs}}/(n_{\text{obs}} - n_{\text{par}})}$, converting it to a REML-like estimate. This argument is only used when a single fitted object is passed to the function. Default is `TRUE`.

`Terms`	An optional integer or character vector specifying which terms in the model should be jointly tested to be zero using a Wald F-test. If given as a character vector, its elements must correspond to term names; else, if given as an integer vector, its elements must correspond to the order in which terms are included in the model. This argument is only used when a single fitted object is passed to the function. Default is `NULL`.
`L`	An optional numeric vector or array specifying linear combinations of the coefficients in the model that should be tested to be zero. If given as an array, its rows define the linear combinations to be tested. If names are assigned to the vector elements (array columns), they must correspond to names of the coefficients and will be used to map the linear combination(s) to the coefficients; else, if no names are available, the vector elements (array columns) are assumed in the same order as the coefficients appear in the model. This argument is only used when a single fitted object is passed to the function. Default is `NULL`.
`verbose`	An optional logical value. If `TRUE`, the calling sequences for each fitted model object are printed with the rest of the output, being omitted if `verbose = FALSE`. Defaults to `FALSE`.

Description

When only one fitted model object is present, a data frame with the sums of squares, numerator degrees of freedom, denominator degrees of freedom, F-values, and p-values for Wald tests for the terms in the model (when `Terms` and `L` are `NULL`), a combination of model terms (when `Terms` in not `NULL`), or linear combinations of the model coefficients (when `L` is not `NULL`). Otherwise, when multiple fitted objects are being compared, a data frame with the degrees of freedom, the (restricted) log-likelihood, the Akaike Information Criterion (AIC), and the Bayesian Information Criterion (BIC) of each object is returned. If `test=TRUE`, whenever two consecutive objects have different number of degrees of freedom, a likelihood ratio statistic, with the associated p-value is included in the returned data frame.

Value

A data frame inheriting from class anova.lme.

Note

Likelihood comparisons are not meaningful for objects fit using restricted maximum likelihood and with different fixed effects.

See Also

`gls`, `gnls`, `nlme`, `lme`, `AIC`, `BIC`, `print.anova.lme`

Examples

```
fm1 <- lme(distance ~ age, Orthodont, random = ~ age | Subject)
anova(fm1)
fm2 <- update(fm1, random = pdDiag(~age))
anova(fm1, fm2)
```

`coef.lme` *Extract* `lme` *Coefficients*

```
coef(object, augFrame, level, data, which, FUN,
     omitGroupingFactor)
```

Arguments

`object`	An object inheriting from class lme, representing a fitted linear mixed-effects model.
`augFrame`	An optional logical value. If `TRUE`, the returned data frame is augmented with variables defined in `data`; else, if `FALSE`, only the coefficients are returned. Defaults to `FALSE`.
`level`	An optional positive integer giving the level of grouping to be used in extracting the coefficients from an object with multiple nested grouping levels. Defaults to the highest or innermost level of grouping.
`data`	An optional data frame with the variables to be used for augmenting the returned data frame when `augFrame = TRUE`. Defaults to the data frame used to fit `object`.
`which`	An optional positive integer or character vector specifying which columns of `data` should be used in the augmentation of the returned data frame. Defaults to all columns in `data`.

`FUN` An optional summary function or a list of summary functions to be applied to group-varying variables, when collapsing `data` by groups. Group-invariant variables are always summarized by the unique value that they assume within that group. If `FUN` is a single function it will be applied to each noninvariant variable by group to produce the summary for that variable. If `FUN` is a list of functions, the names in the list should designate classes of variables in the frame such as `ordered`, `factor`, or `numeric`. The indicated function will be applied to any group-varying variables of that class. The default functions to be used are `mean` for numeric factors, and `Mode` for both `factor` and `ordered`. The `Mode` function, defined internally in `gsummary`, returns the modal or most popular value of the variable. It is different from the `mode` function that returns the S-language mode of the variable.

`omitGroupingFactor`
An optional logical value. When `TRUE` the grouping factor itself will be omitted from the groupwise summary of `data`, but the levels of the grouping factor will continue to be used as the row names for the returned data frame. Defaults to `FALSE`.

Description

The estimated coefficients at level i are obtained by adding together the fixed-effects estimates and the corresponding random-effects estimates at grouping levels less or equal to i. The resulting estimates are returned as a data frame, with rows corresponding to groups and columns to coefficients. Optionally, the returned data frame may be augmented with covariates summarized over groups.

Value

A data frame inheriting from class coef.lme with the estimated coefficients at level `level` and, optionally, other covariates summarized over groups. The returned object also inherits from classes ranef.lme and data.frame.

See Also

`lme`, `fixef.lme`, `ranef.lme`, `plot.ranef.lme`, `gsummary`

Examples

```
fm1 <- lme(distance ~ age, Orthodont, random = ~ age | Subject)
```

```
coef(fm1)
coef(fm1, augFrame = TRUE)
```

`coef.lmList` *Extract lmList Coefficients*

```
coef(object, augFrame, data, which, FUN,
     omitGroupingFactor)
```

Arguments

`object`	An object inheriting from class `lmList`, representing a list of `lm` objects with a common model.
`augFrame`	An optional logical value. If `TRUE`, the returned data frame is augmented with variables defined in the data frame used to produce `object`; else, if `FALSE`, only the coefficients are returned. Defaults to `FALSE`.
`data`	An optional data frame with the variables to be used for augmenting the returned data frame when `augFrame = TRUE`. Defaults to the data frame used to fit `object`.
`which`	An optional positive integer or character vector specifying which columns of the data frame used to produce `object` should be used in the augmentation of the returned data frame. Defaults to all variables in the data.
`FUN`	An optional summary function or a list of summary functions to be applied to group-varying variables, when collapsing the data by groups. Group-invariant variables are always summarized by the unique value that they assume within that group. If `FUN` is a single function it will be applied to each noninvariant variable by group to produce the summary for that variable. If `FUN` is a list of functions, the names in the list should designate classes of variables in the frame such as `ordered`, `factor`, or `numeric`. The indicated function will be applied to any group-varying variables of that class. The default functions to be used are `mean` for numeric factors, and `Mode` for both `factor` and `ordered`. The `Mode` function, defined internally in `gsummary`, returns the modal or most popular value of the variable. It is different from the `mode` function that returns the S-language mode of the variable.

`omitGroupingFactor`
: An optional logical value. When `TRUE` the grouping factor itself will be omitted from the groupwise summary of `data` but the levels of the grouping factor will continue to be used as the row names for the returned data frame. Defaults to `FALSE`.

Description

The coefficients of each `lm` object in the `object` list are extracted and organized into a data frame, with rows corresponding to the `lm` components and columns corresponding to the coefficients. Optionally, the returned data frame may be augmented with covariates summarized over the groups associated with the `lm` components.

Value

A data frame inheriting from class coef.lmList with the estimated coefficients for each `lm` component of `object` and, optionally, other covariates summarized over the groups corresponding to the `lm` components. The returned object also inherits from classes ranef.lmList and data.frame.

See Also

`lmList`, `fixed.effects.lmList`, `ranef.lmList`,
`plot.ranef.lmList`, `gsummary`

Examples

```
fm1 <- lmList(distance ~ age|Subject, data = Orthodont)
coef(fm1)
coef(fm1, augFrame = TRUE)
```

`fitted.lme` *Extract `lme` Fitted Values*

```
fitted(object, level, asList)
```

Arguments

`object`	An object inheriting from class lme, representing a fitted linear mixed-effects model.
`level`	An optional integer vector giving the level(s) of grouping to be used in extracting the fitted values from `object`. Level values increase from outermost to innermost grouping, with level zero corresponding to

	the population fitted values. Defaults to the highest or innermost level of grouping.
`asList`	An optional logical value. If `TRUE` and a single value is given in `level`, the returned object is a list with the fitted values split by groups; else the returned value is either a vector or a data frame, according to the length of `level`. Defaults to `FALSE`.

Description

The fitted values at level i are obtained by adding together the population-fitted values (based only on the fixed-effects estimates) and the estimated contributions of the random effects to the fitted values at grouping levels less or equal to i. The resulting values estimate the best linear unbiased predictions (BLUPs) at level i.

Value

If a single level of grouping is specified in `level`, the returned value is either a list with the fitted values split by groups (`asList = TRUE`) or a vector with the fitted values (`asList = FALSE`); else, when multiple grouping levels are specified in `level`, the returned object is a data frame with columns given by the fitted values at different levels and the grouping factors.

See Also

`lme`, `residuals.lme`

Examples

```
fm1 <- lme(distance ~ age + Sex, data = Orthodont, random = ~ 1)
fitted(fm1, level = 0:1)
```

`fixef` *Extract Fixed Effects*

```
fixef(object, ...)
fixed.effects(object, ...)
```

Arguments

`object`	Any fitted model object from which fixed-effects estimates can be extracted.
`...`	Some methods for this generic function require additional arguments.

Description

This function is generic; method functions can be written to handle specific classes of objects. Classes that already have methods for this function include `lmList` and `lme`.

Value

Will depend on the method function used; see the appropriate documentation.

See Also

`fixef.lmList`, `fixef.lme`

`gapply` *Apply a Function by Groups*

```
gapply(object, which, FUN, form, level, groups, ...)
```

Arguments

`object`	An object to which the function will be applied, usually a `groupedData` object or a `data.frame`. Must inherit from class `data.frame`.
`which`	An optional character or positive integer vector specifying which columns of `object` should be used with `FUN`. Defaults to all columns in `object`.
`FUN`	Function to apply to the distinct sets of rows of the data frame `object` defined by the values of `groups`.
`form`	An optional one-sided formula that defines the groups. When this formula is given the right-hand side is evaluated in `object`, converted to a factor if necessary, and the unique levels are used to define the groups. Defaults to `formula(object)`.
`level`	An optional positive integer giving the level of grouping to be used in an object with multiple nested grouping levels. Defaults to the highest or innermost level of grouping.
`groups`	An optional factor that will be used to split the rows into groups. Defaults to `getGroups(object, form, level)`.
`...`	Optional additional arguments to the summary function `FUN`. Often it is helpful to specify `na.rm = TRUE`.

Description

Applies the function to the distinct sets of rows of the data frame defined by `groups`.

Value

Returns a data frame with as many rows as there are levels in the `groups` argument.

See Also

`gsummary`

Examples

```
## Find number of nonmissing "conc" observations for each Subject
gapply( Quinidine, FUN = function(x) sum(!is.na(x$conc)) )
```

`getGroups` *Extract Grouping Factors from an Object*

```
getGroups(object, form, level, data)
```

Arguments

`object`	Any object.
`form`	An optional formula with a conditioning expression on its right hand side (i.e., an expression involving the | operator). Defaults to `formula(object)`.
`level`	A positive integer vector with the level(s) of grouping to be used when multiple nested levels of grouping are present. This argument is optional for most methods of this generic function and defaults to all levels of nesting.
`data`	A data frame in which to interpret the variables named in `form`. Optional for most methods.

Description

This function is generic; method functions can be written to handle specific classes of objects. Classes that already have methods for this function include corStruct, data.frame, gls, lme, lmList, and varFunc.

Value

Will depend on the method function used; see the appropriate documentation.

See Also

`getGroupsFormula`, `getGroups.data.frame`, `getGroups.gls`, `getGroups.lmList`, `getGroups.lme`

`gls` *Fit Linear Model Using Generalized Least Squares*

```
gls(model, data, correlation, weights, subset, method,
    na.action, control, verbose)
```

Arguments

`model`	A two-sided linear formula object describing the model, with the response on the left of a `~` operator and the terms, separated by `+` operators, on the right.
`data`	An optional data frame containing the variables named in `model`, `correlation`, `weights`, and `subset`. By default the variables are taken from the environment from which `gls` is called.
`correlation`	An optional `corStruct` object describing the within-group correlation structure. See the documentation of `corClasses` for a description of the available corStruct classes. If a grouping variable is to be used, it must be specified in the `form` argument to the `corStruct` constructor. Defaults to `NULL`, corresponding to uncorrelated errors.
`weights`	An optional `varFunc` object or one-sided formula describing the within-group heteroscedasticity structure. If given as a formula, it is used as the argument to `varFixed`, corresponding to fixed variance weights. See the documentation on `varClasses` for a description of the available varFunc classes. Defaults to `NULL`, corresponding to homoscesdatic errors.
`subset`	An optional expression indicating which subset of the rows of `data` should be used in the fit. This can be a logical vector, or a numeric vector indicating which observation numbers are to be included, or a character vector of the row names to be included. All observations are included by default.
`method`	A character string. If `"REML"` the model is fit by maximizing the restricted log-likelihood. If `"ML"` the log-likelihood is maximized. Defaults to `"REML"`.

`na.action`	A function that indicates what should happen when the data contain `NA`s. The default action (`na.fail`) causes `gls` to print an error message and terminate if there are any incomplete observations.
`control`	A list of control values for the estimation algorithm to replace the default values returned by the function `glsControl`. Defaults to an empty list.
`verbose`	An optional logical value. If `TRUE` information on the evolution of the iterative algorithm is printed. Default is `FALSE`.

Description

This function fits a linear model using generalized least squares. The errors are allowed to be correlated and/or have unequal variances.

Value

An object of class `gls` representing the linear model fit. Generic functions such as `print`, `plot`, and `summary` have methods to show the results of the fit. See `glsObject` for the components of the fit. The functions `resid`, `coef`, and `fitted` can be used to extract some of its components.

References

The different correlation structures available for the `correlation` argument are described in Box et al. (1994), Littell et al. (1996), and Venables and Ripley (1999). The use of variance functions for linear and nonlinear models is presented in detail in Carroll and Ruppert (1988) and Davidian and Giltinan (1995).

See Also

`glsControl`, `glsObject`, `varFunc`, `corClasses`, `varClasses`

Examples

```
# AR(1) errors within each Mare
fm1 <- gls(follicles ~ sin(2*pi*Time) + cos(2*pi*Time), Ovary,
          correlation = corAR1(form = ~ 1 | Mare))
# variance increases as a power of the absolute fitted values
fm2 <- gls(follicles ~ sin(2*pi*Time) + cos(2*pi*Time), Ovary,
          weights = varPower())
```

gnls — *Fit Nonlinear Model Using Generalized Least Squares*

```
gnls(model, data, params, start, correlation, weights,
     subset, na.action, naPattern, control, verbose)
```

Arguments

`model`	A two-sided formula object describing the model, with the response on the left of a `~` operator and a nonlinear expression involving parameters and covariates on the right. If `data` is given, all names used in the formula should be defined as parameters or variables in the data frame.
`data`	An optional data frame containing the variables used in `model`, `correlation`, `weights`, `subset`, and `naPattern`. By default the variables are taken from the environment from which `gnls` is called.
`params`	An optional two-sided linear formula of the form `p1+···+pn~x1+···+xm`, or list of two-sided formulas of the form `p1~x1+···+xm`, with possibly different models for each parameter. The `p1,...,pn` represent parameters included on the right-hand side of `model` and `x1+···+xm` define a linear model for the parameters (when the left-hand side of the formula contains several parameters, they are all assumed to follow the same linear model described by the right-hand side expression). A `1` on the right-hand side of the formula(s) indicates a single fixed effect for the corresponding parameter(s). By default, the parameters are obtained from the names of `start`.
`start`	An optional named list, or numeric vector, with the initial values for the parameters in `model`. It can be omitted when a `selfStarting` function is used in `model`, in which case the starting estimates will be obtained from a single call to the `nls` function.
`correlation`	An optional `corStruct` object describing the within-group correlation structure. See the documentation of `corClasses` for a description of the available `corStruct` classes. If a grouping variable is to be used, it must be specified in the `form` argument to the `corStruct` constructor. Defaults to `NULL`, corresponding to uncorrelated errors.

`weights`	An optional `varFunc` object or one-sided formula describing the within-group heteroscedastic structure. If given as a formula, it is used as the argument to `varFixed`, corresponding to fixed variance weights. See the documentation on `varClasses` for a description of the available `varFunc` classes. Defaults to `NULL`, corresponding to homoscesdatic errors.
`subset`	An optional expression indicating which subset of the rows of `data` should be used in the fit. This can be a logical vector, or a numeric vector indicating which observation numbers are to be included, or a character vector of the row names to be included. All observations are included by default.
`na.action`	A function that indicates what should happen when the data contain `NA`s. The default action (`na.fail`) causes `gnls` to print an error message and terminate if there are any incomplete observations.
`naPattern`	An expression or formula object, specifying which returned values are to be regarded as missing.
`control`	A list of control values for the estimation algorithm to replace the default values returned by the function `gnlsControl`. Defaults to an empty list.
`verbose`	An optional logical value. If `TRUE` information on the evolution of the iterative algorithm is printed. Default is `FALSE`.

Description

This function fits a nonlinear model using generalized least squares. The errors are allowed to be correlated and/or have unequal variances.

Value

An object of class `gnls`, also inheriting from class `gls`, representing the nonlinear model fit. Generic functions such as `print`, `plot` and `summary` have methods to show the results of the fit. See `gnlsObject` for the components of the fit. The functions `resid`, `coef`, and `fitted` can be used to extract some of its components.

References

The different correlation structures available for the `correlation` argument are described in Box et al. (1994), Littell et al. (1996), and Venables and Ripley (1999). The use of variance functions for linear and nonlinear models is presented in detail in Carroll and Ruppert (1988) and Davidian and Giltinan (1995).

See Also

`gnlsControl`, `gnlsObject`, `varFunc`, `corClasses`, `varClasses`

Examples

```
# variance increases with a power of the absolute fitted values
fm1 <- gnls(weight ~ SSlogis(Time, Asym, xmid, scal), Soybean,
            weights = varPower())
# errors follow an auto-regressive process of order 1
fm2 <- gnls(weight ~ SSlogis(Time, Asym, xmid, scal), Soybean,
            correlation = corAR1())
```

`groupedData` *Construct a* `groupedData` *Object*

```
groupedData(formula, data, order.groups, FUN, outer, inner,
            labels, units)
```

Arguments

`formula`	A formula of the form `resp ~ cov	group` where `resp` is the response, `cov` is the primary covariate, and `group` is the grouping factor. The expression `1` can be used for the primary covariate when there is no other suitable candidate. Multiple nested grouping factors can be listed separated by the `/` symbol as in `fact1/fact2`. In an expression like this the `fact2` factor is nested within the `fact1` factor.
`data`	A data frame in which the expressions in `formula` can be evaluated. The resulting `groupedData` object will consist of the same data values in the same order, but with additional attributes.
`order.groups`	An optional logical value, or list of logical values, indicating if the grouping factors should be converted to ordered factors according to the function `FUN` applied to the response from each group. If multiple levels of grouping are present, this argument can be either a single logical value (which will be repeated for all grouping levels) or a list of logical values. If no names are assigned to the list elements, they are

	assumed in the same order as the group levels (outermost to innermost grouping). Ordering within a level of grouping is done within the levels of the grouping factors which are outer to it. Changing the grouping factor to an ordered factor does not affect the ordering of the rows in the data frame, but it does affect the order of the panels in a trellis display of the data or models fitted to the data. Defaults to `TRUE`.
`FUN`	An optional summary function that will be applied to the values of the response for each level of the grouping factor, when `order.groups = TRUE`, to determine the ordering. Defaults to the `max` function.
`outer`	An optional one-sided formula, or list of one-sided formulas, indicating covariates that are outer to the grouping factor(s). If multiple levels of grouping are present, this argument can be either a single one-sided formula, or a list of one-sided formulas. If no names are assigned to the list elements, they are assumed in the same order as the group levels (outermost to innermost grouping). An outer covariate is invariant within the sets of rows defined by the grouping factor. Ordering of the groups is done in such a way as to preserve adjacency of groups with the same value of the outer variables. When plotting a `groupedData` object, the argument `outer = TRUE` causes the panels to be determined by the `outer` formula. The points within the panels are associated by level of the grouping factor. Defaults to `NULL`, meaning that no outer covariates are present.
`inner`	An optional one-sided formula, or list of one-sided formulas, indicating covariates that are inner to the grouping factor(s). If multiple levels of grouping are present, this argument can be either a single one-sided formula, or a list of one-sided formulas. If no names are assigned to the list elements, they are assumed in the same order as the group levels (outermost to innermost grouping). An inner covariate can change within the sets of rows defined by the grouping factor. An inner formula can be used to associate points in a plot of a `groupedData` object. Defaults to `NULL`, meaning that no inner covariates are present.
`labels`	An optional list of character strings giving labels for the response and the primary covariate. The label

	for the primary covariate is named x and that for the response is named y. Either label can be omitted.
units	An optional list of character strings giving the units for the response and the primary covariate. The units string for the primary covariate is named x and that for the response is named y. Either units string can be omitted.

Description

An object of the groupedData class is constructed from the formula and data by attaching the formula as an attribute of the data, along with any of outer, inner, labels, and units that are given. If order.groups is TRUE the grouping factor is converted to an ordered factor with the ordering determined by FUN. Depending on the number of grouping levels and the type of primary covariate, the returned object will be of one of three classes: nfnGroupedData—numeric covariate, single level of nesting; nffGroupedData—factor covariate, single level of nesting; and nmGroupedData—multiple levels of nesting. Several modeling and plotting functions can use the formula stored with a groupedData object to construct default plots and models.

Value

An object of one of the classes nfnGroupedData, nffGroupedData, or nmGroupedData, also inheriting from classes groupedData and data.frame.

See Also

formula, gapply, gsummary, lme

Examples

```
Orth.new <-  # create a new copy of the groupedData object
  groupedData( distance ~ age | Subject,
       data = as.data.frame( Orthodont ),
       FUN = mean,
       outer = ~ Sex,
       labels = list(x = "Age",
        y = "Distance from pituitary to pterygomaxillary fissure"),
       units = list( x = "(yr)", y = "(mm)") )
plot( Orth.new )         # trellis plot by Subject
formula( Orth.new )      # extractor for the formula
gsummary( Orth.new )     # apply summary by Subject
fm1 <- lme( Orth.new )   # fixed and groups formulae extracted
                         # from object
```

gsummary *Summarize by Groups*

```
gsummary(object, FUN, omitGroupingFactor, form, level,
         groups, invariantsOnly, ...)
```

Arguments

`object`	An object to be summarized, usually a `groupedData` object or a `data.frame`.
`FUN`	An optional summary function or a list of summary functions to be applied to each variable in the frame. The function or functions are applied only to variables in `object` that vary within the groups defined by `groups`. Invariant variables are always summarized by group using the unique value that they assume within that group. If `FUN` is a single function it will be applied to each noninvariant variable by group to produce the summary for that variable. If `FUN` is a list of functions, the names in the list should designate classes of variables in the frame such as `ordered`, `factor`, or `numeric`. The indicated function will be applied to any non-invariant variables of that class. The default functions to be used are `mean` for numeric factors, and `Mode` for both `factor` and `ordered`. The `Mode` function, defined internally in `gsummary`, returns the modal or most popular value of the variable. It is different from the `mode` function that returns the S-language mode of the variable.
`omitGroupingFactor`	An optional logical value. When `TRUE` the grouping factor itself will be omitted from the groupwise summary, but the levels of the grouping factor will continue to be used as the row names for the data frame that is produced by the summary. Defaults to `FALSE`.
`form`	An optional one-sided formula that defines the groups. When this formula is given, the right-hand side is evaluated in `object`, converted to a factor if necessary, and the unique levels are used to define the groups. Defaults to `formula(object)`.
`level`	An optional positive integer giving the level of grouping to be used in an object with multiple nested

grouping levels. Defaults to the highest or innermost level of grouping.

`groups` An optional factor that will be used to split the rows into groups. Defaults to `getGroups(object, form, level)`.

`invariantsOnly`
An optional logical value. When `TRUE` only those covariates that are invariant within each group will be summarized. The summary value for the group is always the unique value taken on by that covariate within the group. The columns in the summary are of the same class as the corresponding columns in `object`. By definition, the grouping factor itself must be an invariant. When combined with `omitGroupingFactor = TRUE`, this option can be used to discover is there are invariant covariates in the data frame. Defaults to `FALSE`.

`...` Optional additional arguments to the summary functions that are invoked on the variables by group. Often it is helpful to specify `na.rm = TRUE`.

Description

Provide a summary of the variables in a data frame by groups of rows. This is most useful with a `groupedData` object to examine the variables by group.

Value

A `data.frame` with one row for each level of the grouping factor. The number of columns is at most the number of columns in `object`.

See Also

`summary`, `groupedData`, `getGroups`

Examples

```
gsummary( Orthodont )  # default summary by Subject
## gsummary with invariantsOnly = TRUE and
## omitGroupingFactor = TRUE determines whether there
## are covariates like Sex that are invariant within
## the repeated observations on the same Subject.
gsummary( Orthodont, inv = TRUE, omit = TRUE )
```

`intervals` *Confidence Intervals on Coefficients*

```
intervals(object, level, ...)
```

Arguments

`object`	A fitted model object from which parameter estimates can be extracted.
`level`	An optional numeric value for the interval confidence level. Defaults to 0.95.
`...`	Some methods for the generic may require additional arguments.

Description

Confidence intervals on the parameters associated with the model represented by `object` are obtained. This function is generic; method functions can be written to handle specific classes of objects. Classes which already have methods for this function include: gls, lme, and lmList.

Value

Will depend on the method function used; see the appropriate documentation.

See Also

`intervals.gls`, `intervals.lme`, `intervals.lmList`

`intervals.lme` *Confidence Intervals on `lme` Parameters*

```
intervals(object, level, which)
```

Arguments

`object`	An object inheriting from class lme, representing a fitted linear mixed-effects model.
`level`	An optional numeric value with the confidence level for the intervals. Defaults to 0.95.

`which`	An optional character string specifying the subset of parameters for which to construct the confidence intervals. Possible values are `"all"` for all parameters, `"var-cov"` for the variance–covariance parameters only, and `"fixed"` for the fixed effects only. Defaults to `"all"`.

Description

Approximate confidence intervals for the parameters in the linear mixed-effects model represented by `object` are obtained, using a normal approximation to the distribution of the (restricted) maximum likelihood estimators (the estimators are assumed to have a normal distribution centered at the true parameter values and with covariance matrix equal to the negative inverse Hessian matrix of the (restricted) log-likelihood evaluated at the estimated parameters). Confidence intervals are obtained in an unconstrained scale first, using the normal approximation, and, if necessary, transformed to the constrained scale. The `pdNatural` parametrization is used for general positive-definite matrices.

Value

A list with components given by data frames with rows corresponding to parameters and columns `lower`, `est.`, and `upper` representing, respectively, lower confidence limits, the estimated values, and upper confidence limits for the parameters. Possible components are:

`fixed`	Fixed effects, only present when `which` is not equal to `"var-cov"`.
`reStruct`	Random-effects variance–covariance parameters, only present when `which` is not equal to `"fixed"`.
`corStruct`	Within-group correlation parameters, only present when `which` is not equal to `"fixed"` and a correlation structure is used in `object`.
`varFunc`	Within-group variance function parameters, only present when `which` is not equal to `"fixed"` and a variance function structure is used in `object`.
`sigma`	Within-group standard deviation.

See Also

`lme`, `print.intervals.lme`, `pdNatural`

Examples

```
fm1 <- lme(distance ~ age, Orthodont, random = ~ age | Subject)
intervals(fm1)
```

`intervals.lmList` *Confidence Intervals on `lmList` Coefficients*

```
intervals(object, level, pool)
```

Arguments

`object`	An object inheriting from class lmList, representing a list of `lm` objects with a common model.
`level`	An optional numeric value with the confidence level for the intervals. Defaults to 0.95.
`pool`	An optional logical value indicating whether a pooled estimate of the residual standard error should be used. Default is `attr(object, "pool")`.

Description

Confidence intervals on the linear model coefficients are obtained for each `lm` component of `object` and organized into a three-dimensional array. The first dimension corresponding to the names of the `object` components. The second dimension is given by `lower`, `est.`, and `upper` corresponding, respectively, to the lower confidence limit, estimated coefficient, and upper confidence limit. The third dimension is given by the coefficients names.

Value

A three-dimensional array with the confidence intervals and estimates for the coefficients of each `lm` component of `object`.

See Also

`lmList`, `plot.intervals.lmList`

Examples

```
fm1 <- lmList(distance ~ age | Subject, Orthodont)
intervals(fm1)
```

`lme` *Linear Mixed-Effects Models*

```
lme(fixed, data, random, correlation, weights, subset,
    method, na.action, control)
```

Arguments

`fixed`	A two-sided linear formula object describing the fixed-effects part of the model, with the response on the left of a `~` operator and the terms, separated by `+` operators, on the right, an `lmList` object, or a `groupedData` object. The method functions `lme.lmList` and `lme.groupedData` are documented separately.
`data`	An optional data frame containing the variables named in `fixed`, `random`, `correlation`, `weights`, and `subset`. By default the variables are taken from the environment from which `lme` is called.
`random`	Optionally, any of the following: (i) a one-sided formula of the form `~x1+···+xn	g1/···/gm`, with `x1+···+xn` specifying the model for the random effects and `g1/···/gm` the grouping structure (`m` may be equal to 1, in which case no `/` is required). The random-effects formula will be repeated for all levels of grouping, in the case of multiple levels of grouping; (ii) a list of one-sided formulas of the form `~x1+···+xn	g`, with possibly different random-effects models for each grouping level. The order of nesting will be assumed the same as the order of the elements in the list; (iii) a one-sided formula of the form `~x1+···+xn`, or a `pdMat` object with a formula (i.e., a non-`NULL` value for `formula(object)`), or a list of such formulas or `pdMat` objects. In this case, the grouping structure formula will be derived from the data used to fit the linear mixed-effects model, which should inherit from class groupedData; (iv) a named list of formulas or `pdMat` objects as in (iii), with the grouping factors as names. The order of nesting will be assumed the same as the order of the order of the elements in the list; (v) an `reStruct` object. See the documentation on `pdClasses` for a description of the available pdMat classes. Defaults to a formula consisting of the right-hand side of `fixed`.

`correlation`	An optional `corStruct` object describing the within-group correlation structure. See the documentation of `corClasses` for a description of the available `corStruct` classes. Defaults to `NULL`, corresponding to no within-group correlations.
`weights`	An optional `varFunc` object or one-sided formula describing the within-group heteroscedasticity structure. If given as a formula, it is used as the argument to `varFixed`, corresponding to fixed variance weights. See the documentation on `varClasses` for a description of the available `varFunc` classes. Defaults to `NULL`, corresponding to homocesdatic within-group errors.
`subset`	An optional expression indicating the subset of the rows of `data` that should be used in the fit. This can be a logical vector, or a numeric vector indicating which observation numbers are to be included, or a character vector of the row names to be included. All observations are included by default.
`method`	A character string. If `"REML"` the model is fit by maximizing the restricted log-likelihood. If `"ML"` the log-likelihood is maximized. Defaults to `"REML"`.
`na.action`	A function that indicates what should happen when the data contain `NA`s. The default action (`na.fail`) causes `lme` to print an error message and terminate if there are any incomplete observations.
`control`	A list of control values for the estimation algorithm to replace the default values returned by the function `lmeControl`. Defaults to an empty list.

Description

This generic function fits a linear mixed-effects model in the formulation described in Laird and Ware (1982), but allowing for nested random effects. The within-group errors are allowed to be correlated and/or have unequal variances.

Value

An object of class `lme` representing the linear mixed-effects model fit. Generic functions such as `print`, `plot` and `summary` have methods to show the results of the fit. See `lmeObject` for the components of the fit. The functions `resid`, `coef`, `fitted`, `fixef`, and `ranef` can be used to extract some of its components.

See Also

`lmeControl`, `lme.lmList`, `lme.groupedData`, `lmeObject`, `lmList`, `reStruct`, `reStruct`, `varFunc`, `pdClasses`, `corClasses`, `varClasses`

Examples

```
fm1 <- lme(distance ~ age, data = Orthodont) # random is ~ age
fm2 <- lme(distance ~ age + Sex, data = Orthodont, random = ~ 1)
```

---

lmeControl *Control Values for `lme` Fit*

```
lmeControl(maxIter, msMaxIter, tolerance, niterEM, msTol,
           msScale, msVerbose, returnObject, gradHess,
           apVar, .relStep, natural)
```

Arguments

`maxIter`	Maximum number of iterations for the `lme` optimization algorithm. Default is 50.
`msMaxIter`	Maximum number of iterations for the `ms` optimization step inside the `lme` optimization. Default is 50.
`tolerance`	Tolerance for the convergence criterion in the `lme` algorithm. Default is 1e-6.
`niterEM`	Number of iterations for the EM algorithm used to refine the initial estimates of the random-effects variance–covariance coefficients. Default is 25.
`msTol`	Tolerance for the convergence criterion in `ms`, passed as the `rel.tolerance` argument to the function (see documentation on `ms`). Default is 1e-7.
`msScale`	Scale function passed as the `scale` argument to the `ms` function (see documentation on that function). Default is `lmeScale`.
`msVerbose`	A logical value passed as the `trace` argument to `ms` (see documentation on that function). Default is `FALSE`.
`returnObject`	A logical value indicating whether the fitted object should be returned when the maximum number of iterations is reached without convergence of the algorithm. Default is `FALSE`.

`gradHess`	A logical value indicating whether numerical gradient vectors and Hessian matrices of the log-likelihood function should be used in the `ms` optimization. This option is only available when the correlation structure (`corStruct`) and the variance function structure (`varFunc`) have no "varying" parameters and the `pdMat` classes used in the random effects structure are `pdSymm` (general positive-definite), `pdDiag` (diagonal), `pdIdent` (multiple of the identity), or `pdCompSymm` (compound symmetry). Default is `TRUE`.
`apVar`	A logical value indicating whether the approximate covariance matrix of the variance–covariance parameters should be calculated. Default is `TRUE`.
`.relStep`	Relative step for numerical derivatives calculations. Default is `.Machine$double.eps^(1/3)`.
`natural`	A logical value indicating whether the `pdNatural` parameterization should be used for general positive-definite matrices (`pdSymm`) in `reStruct`, when the approximate covariance matrix of the estimators is calculated. Default is `TRUE`.

Description

The values supplied in the function call replace the defaults and a list with all possible arguments is returned. The returned list is used as the `control` argument to the `lme` function.

Value

A list with components for each of the possible arguments.

See Also

`lme`, `ms`, `lmeScale`

Examples

```
# decrease the maximum number iterations in the ms call and
# request that information on the evolution of the ms iterations
# be printed
lmeControl(msMaxIter = 20, msVerbose = TRUE)
```

`lmList` *List of `lm` Objects with a Common Model*

```
lmList(object, data, level, na.action, pool)
```

Arguments

`object`	Either a linear formula object of the form `y ~ x1+···+xn	g` or a `groupedData` object. In the formula, `y` represents the response, `x1,...,xn` the covariates, and `g` the grouping factor specifying the partitioning of the data according to which different `lm` fits should be performed. The grouping factor `g` may be omitted from the formula, in which case the grouping structure will be obtained from `data`, which must inherit from class `groupedData`. The method function `lmList.groupedData` is documented separately.
`data`	A data frame in which to interpret the variables named in `object`.
`level`	An optional integer specifying the level of grouping to be used when multiple nested levels of grouping are present.
`na.action`	A function that indicates what should happen when the data contain `NA`s. The default action (`na.fail`) causes `lmList` to print an error message and terminate if there are any incomplete observations.
`pool`	An optional logical value that is preserved as an attribute of the returned value. This will be used as the default for `pool` in calculations of standard deviations or standard errors for summaries.

Description

`Data` is partitioned according to the levels of the grouping factor `g` and individual `lm` fits are obtained for each `data` partition, using the model defined in `object`.

Value

A list of `lm` objects with as many components as the number of groups defined by the grouping factor. Generic functions such as `coef`, `fixef`, `lme`, `pairs`, `plot`, `predict`, `ranef`, `summary`, and `update` have methods that can be applied to an `lmList` object.

See Also

`lm`, `lme.lmList`.

Examples

```
fm1 <- lmList(distance ~ age | Subject, Orthodont)
```

`logLik` *Extract Log-Likelihood*

```
logLik(object, ...)
```

Arguments

`object`	Any object from which a log-likelihood, or a contribution to a log-likelihood, can be extracted.
`...`	Some methods for this generic function require additional arguments.

Description

This function is generic; method functions can be written to handle specific classes of objects. Classes which already have methods for this function include: corStruct, gls, lm, lme, lmList, lmeStruct, reStruct, and varFunc.

Value

Will depend on the method function used; see the appropriate documentation.

`nlme` *Nonlinear Mixed-Effects Models*

```
nlme(model, data, fixed, random, groups, start,
     correlation, weights, subset, method, na.action,
     naPattern, control, verbose)
```

Arguments

`model`	A nonlinear model formula, with the response on the left of a `~` operator and an expression involving parameters and covariates on the right, or an `nlsList` object. If `data` is given, all names used in the formula should be defined as parameters or variables in the data frame. The method function `nlme.nlsList` is documented separately.
`data`	An optional data frame containing the variables named in `model`, `fixed`, `random`, `correlation`, `weights`, `subset`, and `naPattern`. By default the variables are taken from the environment from which `nlme` is called.
`fixed`	A two-sided linear formula of the form `f1+···+fn ~ x1+···+xm`, or a list of two-sided formulas of the form `f1 ~ x1+···+xm`, with possibly different models for different parameters. The names of the parameters, `f1,...,fn`, are included on the right-hand side of `model` and the `x1+···+xm` expressions define linear models for these parameters (when the left-hand side of the formula contains several parameters, they all are assumed to follow the same linear model, described by the right-hand side expression). A `1` on the right-hand side of the formula(s) indicates a single fixed effects for the corresponding parameter(s).
`random`	Optionally, any of the following: (i) a two-sided formula of the form `r1+···+rn ~ x1+···+xm|g1/···/gQ`, with `r1,...,rn` naming parameters included on the right-hand side of `model`, `x1+···+xm` specifying the random-effects model for these parameters and `g1/···/gQ` the grouping structure (`Q` may be equal to 1, in which case no `/` is required). The random-effects formula will be repeated for all levels of grouping, in the case of multiple levels of grouping; (ii) a two-sided formula of the form `r1+···+rn ~ x1+···+xm`, a list of two-sided formulas of the form `r1~x1+···+xm`, with possibly different random-effects models for different parameters, a `pdMat` object with a two-sided formula, or list of two-sided formulas (i.e., a non-`NULL` value for `formula(random)`), or a list of `pdMat` objects with two-sided formulas, or lists of two-sided formulas. In this case, the grouping structure formula will be given in `groups`, or derived from the

	data used to fit the nonlinear mixed-effects model, which should inherit from class `groupedData`; (iii) a named list of formulas, lists of formulas, or `pdMat` objects as in (ii), with the grouping factors as names. The order of nesting will be assumed the same as the order of the order of the elements in the list; (iv) an `reStruct` object. See the documentation on `pdClasses` for a description of the available `pdMat` classes. Defaults to `fixed`, resulting in all fixed effects having also random effects.
`groups`	An optional one-sided formula of the form `~g1` (single level of nesting) or `~g1/`$\cdots$`/gQ` (multiple levels of nesting), specifying the partitions of the data over which the random effects vary. `g1,...,gQ` must evaluate to factors in `data`. The order of nesting, when multiple levels are present, is taken from left to right (i.e., `g1` is the first level, `g2` the second, etc.).
`start`	An optional numeric vector, or list of initial estimates for the fixed effects and random effects. If declared as a numeric vector, it is converted internally to a list with a single component `fixed`, given by the vector. The `fixed` component is required, unless the model function inherits from class `selfStart`, in which case initial values will be derived from a call to `nlsList`. An optional `random` component is used to specify initial values for the random effects and should consist of a matrix, or a list of matrices with length equal to the number of grouping levels. Each matrix should have as many rows as the number of groups at the corresponding level and as many columns as the number of random effects in that level.
`correlation`	An optional `corStruct` object describing the within-group correlation structure. See the documentation of `corClasses` for a description of the available `corStruct` classes. Defaults to `NULL`, corresponding to no within-group correlations.
`weights`	An optional `varFunc` object or one-sided formula describing the within-group heteroscedasticity structure. If given as a formula, it is used as the argument to `varFixed`, corresponding to fixed variance weights. See the documentation on `varClasses` for a description of the available `varFunc` classes. Defaults to `NULL`, corresponding to homoscesdatic within-group errors.

`subset`	An optional expression indicating the subset of the rows of `data` that should be used in the fit. This can be a logical vector, or a numeric vector indicating which observation numbers are to be included, or a character vector of the row names to be included. All observations are included by default.
`method`	A character string. If `"REML"` the model is fit by maximizing the restricted log-likelihood. If `"ML"` the log-likelihood is maximized. Defaults to `"ML"`.
`na.action`	A function that indicates what should happen when the data contain `NA`s. The default action (`na.fail`) causes `nlme` to print an error message and terminate if there are any incomplete observations.
`naPattern`	An expression or formula object, specifying which returned values are to be regarded as missing.
`control`	A list of control values for the estimation algorithm to replace the default values returned by the function `nlmeControl`. Defaults to an empty list.
`verbose`	An optional logical value. If `TRUE` information on the evolution of the iterative algorithm is printed. Default is `FALSE`.

Description

This generic function fits a nonlinear mixed-effects model in the formulation described in Lindstrom and Bates (1990), but allowing for nested random effects. The within-group errors are allowed to be correlated and/or have unequal variances.

Value

An object of class `nlme` representing the nonlinear mixed-effects model fit. Generic functions such as `print`, `plot` and `summary` have methods to show the results of the fit. See `nlmeObject` for the components of the fit. The functions `resid`, `coef`, `fitted`, `fixef`, and `ranef` can be used to extract some of its components.

See Also

`nlmeControl`, `nlme.nlsList`, `nlmeObject`, `nlsList`, `reStruct`, `varFunc`, `pdClasses`, `corClasses`, `varClasses`

Examples

```
## all parameters as fixed and random effects
```

```
fm1 <- nlme(weight ~ SSlogis(Time, Asym, xmid, scal),
            data = Soybean, fixed = Asym + xmid + scal ~ 1,
            start = c(18, 52, 7.5))
## only Asym and xmid as random, with a diagonal covariance
fm2 <- nlme(weight ~ SSlogis(Time, Asym, xmid, scal),
            data = Soybean, fixed = Asym + xmid + scal ~ 1,
            random = pdDiag(Asym + xmid ~ 1),
            start = c(18, 52, 7.5))
```

nlmeControl *Control Values for* `nlme` *Fit*

```
nlmeControl(maxIter, pnlsMaxIter, msMaxIter, minScale,
            tolerance, niterEM, pnlsTol, msTol, msScale,
            returnObject, msVerbose, gradHess, apVar,
            .relStep, natural)
```

Arguments

`maxIter`	Maximum number of iterations for the `nlme` optimization algorithm. Default is 50.
`pnlsMaxIter`	Maximum number of iterations for the PNLS optimization step inside the `nlme` optimization. Default is 7.
`msMaxIter`	Maximum number of iterations for the `ms` optimization step inside the `nlme` optimization. Default is 50.
`minScale`	Minimum factor by which to shrink the default step size in an attempt to decrease the sum of squares in the `PNLS` step. Default 0.001.
`tolerance`	Tolerance for the convergence criterion in the `nlme` algorithm. Default is 1e-6.
`niterEM`	Number of iterations for the EM algorithm used to refine the initial estimates of the random-effects variance–covariance coefficients. Default is 25.
`pnlsTol`	Tolerance for the convergence criterion in `PNLS` step. Default is 1e-3.
`msTol`	Tolerance for the convergence criterion in `ms`, passed as the `rel.tolerance` argument to the function (see documentation on `ms`). Default is 1e-7.
`msScale`	Scale function passed as the `scale` argument to the `ms` function (see documentation on that function). Default is `lmeScale`.

`returnObject`	A logical value indicating whether the fitted object should be returned when the maximum number of iterations is reached without convergence of the algorithm. Default is `FALSE`.
`msVerbose`	A logical value passed as the `trace` argument to `ms` (see documentation on that function). Default is `FALSE`.
`gradHess`	A logical value indicating whether numerical gradient vectors and Hessian matrices of the log-likelihood function should be used in the `ms` optimization. This option is only available when the correlation structure (`corStruct`) and the variance function structure (`varFunc`) have no "varying" parameters and the `pdMat` classes used in the random effects structure are `pdSymm` (general positive-definite), `pdDiag` (diagonal), `pdIdent` (multiple of the identity), or `pdCompSymm` (compound symmetry). Default is `TRUE`.
`apVar`	A logical value indicating whether the approximate covariance matrix of the variance–covariance parameters should be calculated. Default is `TRUE`.
`.relStep`	Relative step for numerical derivatives calculations. Default is `.Machine$double.eps^(1/3)`.
`natural`	A logical value indicating whether the `pdNatural` parameterization should be used for general positive-definite matrices (`pdSymm`) in `reStruct`, when the approximate covariance matrix of the estimators is calculated. Default is `TRUE`.

Description

The values supplied in the function call replace the defaults and a list with all possible arguments is returned. The returned list is used as the `control` argument to the `nlme` function.

Value

A list with components for each of the possible arguments.

See Also

`nlme`, `ms`, `nlmeStruct`

Examples

```
# decrease the maximum number iterations in the ms call and
# request that information on the evolution of the ms iterations
# be printed
nlmeControl(msMaxIter = 20, msVerbose = TRUE)
```

nlsList — *List of `nls` Objects with a Common Model*

```
nlsList(model, data, start, control, level, na.action,
        pool)
```

Arguments

`model`	Either a nonlinear model formula, with the response on the left of a ~ operator and an expression involving parameters, covariates, and a grouping factor separated by the `|` operator on the right, or a `selfStart` function. The method function `nlsList.selfStart` is documented separately.
`data`	A data frame in which to interpret the variables named in `model`.
`start`	An optional named list with initial values for the parameters to be estimated in `model`. It is passed as the `start` argument to each `nls` call and is required when the nonlinear function in `model` does not inherit from class selfStart.
`control`	A list of control values passed as the `control` argument to `nls`. Defaults to an empty list.
`level`	An optional integer specifying the level of grouping to be used when multiple nested levels of grouping are present.
`na.action`	A function that indicates what should happen when the data contain `NA`s. The default action (`na.fail`) causes `nlsList` to print an error message and terminate if there are any incomplete observations.
`pool`	An optional logical value that is preserved as an attribute of the returned value. This will be used as the default for `pool` in calculations of standard deviations or standard errors for summaries.

Description

`Data` is partitioned according to the levels of the grouping factor defined in `model` and individual `nls` fits are obtained for each `data` partition, using the model defined in `model`.

Value

A list of `nls` objects with as many components as the number of groups defined by the grouping factor. Generic functions such as `coef`, `fixef`, `lme`, `pairs`, `plot`, `predict`, `ranef`, `summary`, and `update` have methods that can be applied to an `nlsList` object.

See Also

`nls`, `nlme.nlsList`.

Examples

```
fm1 <- nlsList(uptake ~ SSasympOff(conc, Asym, lrc, c0),
   data = CO2, start = c(Asym = 30, lrc = -4.5, c0 = 52))
fm1
```

`pairs.lme` *Pairs Plot of an* `lme` *Object*

```
pairs(object, form, label, id, idLabels, grid, ...)
```

Arguments

`object`	An object inheriting from class `lme`, representing a fitted linear mixed-effects model.
`form`	An optional one-sided formula specifying the desired type of plot. Any variable present in the original data frame used to obtain `object` can be referenced. In addition, `object` itself can be referenced in the formula using the symbol `"."`. Conditional expressions on the right of a `|` operator can be used to define separate panels in a trellis display. The expression on the right-hand side of `form`, and to the left of the `|` operator, must evaluate to a data frame with at least two columns. Default is `~ coef(.)` , corresponding to a pairs plot of the coefficients evaluated at the innermost level of nesting.
`id`	An optional numeric value, or one-sided formula. If given as a value, it is used as a significance level for an outlier test based on the Mahalanobis distances of the estimated random effects. Groups with random effects distances greater than the $1-value$ percentile of the appropriate chi-square distribution are identified in the plot using `idLabels`. If given as a one-sided formula, its right-hand side must evaluate to a

	logical, integer, or character vector which is used to identify points in the plot. If missing, no points are identified.
`idLabels`	An optional vector, or one-sided formula. If given as a vector, it is converted to character and used to label the points identified according to `id`. If given as a one-sided formula, its right-hand side must evaluate to a vector which is converted to character and used to label the identified points. Default is the innermost grouping factor.
`grid`	An optional logical value indicating whether a grid should be added to plot. Default is `FALSE`.
`...`	Optional arguments passed to the trellis plot function.

Description

Diagnostic plots for the linear mixed-effects fit are obtained. The `form` argument gives considerable flexibility in the type of plot specification. A conditioning expression (on the right side of a `|` operator) always implies that different panels are used for each level of the conditioning factor, according to a trellis display. The expression on the right-hand side of the formula, before a `|` operator, must evaluate to a data frame with at least two columns. If the data frame has two columns, a scatter plot of the two variables is displayed (the trellis function `xyplot` is used). Otherwise, if more than two columns are present, a scatter plot matrix with pairwise scatter plots of the columns in the data frame is displayed (the trellis function `splom` is used).

Value

A diagnostic trellis plot.

See Also

`lme`, `xyplot`, `splom`

Examples

```
fm1 <- lme(distance ~ age, Orthodont, random = ~ age | Subject)
# scatter plot of coefficients by gender, identifying
# unusual subjects
pairs(fm1, ~coef(., augFrame = T) | Sex, id = 0.1, adj = -0.5)
# scatter plot of estimated random effects
pairs(fm1, ~ranef(.))
```

`plot.lme` *Plot an lme Object*

```
plot(object, form, abline, id, idLabels, idResType, grid,
     ...)
```

Arguments

`object`	An object inheriting from class `lme`, representing a fitted linear mixed-effects model.
`form`	An optional formula specifying the desired type of plot. Any variable present in the original data frame used to obtain `object` can be referenced. In addition, `object` itself can be referenced in the formula using the symbol `"."`. Conditional expressions on the right of a `|` operator can be used to define separate panels in a trellis display. Default is `resid(., type = "p") ~ fitted(.)` , corresponding to a plot of the standardized residuals versus fitted values, both evaluated at the innermost level of nesting.
`abline`	An optional numeric value, or numeric vector of length two. If given as a single value, a horizontal line will be added to the plot at that coordinate; else, if given as a vector, its values are used as the intercept and slope for a line added to the plot. If missing, no lines are added to the plot.
`id`	An optional numeric value, or one-sided formula. If given as a value, it is used as a significance level for a two-sided outlier test for the standardized, or normalized residuals. Observations with absolute standardized (normalized) residuals greater than the $1 - \mathit{value}/2$ quantile of the standard normal distribution are identified in the plot using `idLabels`. If given as a one-sided formula, its right-hand side must evaluate to a logical, integer, or character vector which is used to identify observations in the plot. If missing, no observations are identified.
`idLabels`	An optional vector, or one-sided formula. If given as a vector, it is converted to character and used to label the observations identified according to `id`. If given as a one-sided formula, its right-hand side must evaluate to a vector that is converted to character and

used to label the identified observations. Default is the innermost grouping factor.

`idResType` An optional character string specifying the type of residuals to be used in identifying outliers, when `id` is a numeric value. If `"pearson"`, the standardized residuals (raw residuals divided by the corresponding standard errors) are used; else, if `"normalized"`, the normalized residuals (standardized residuals premultiplied by the inverse square-root factor of the estimated error correlation matrix) are used. Partial matching of arguments is used, so only the first character needs to be provided. Defaults to `"pearson"`.

Description

Diagnostic plots for the linear mixed-effects fit are obtained. The `form` argument gives considerable flexibility in the type of plot specification. A conditioning expression (on the right side of a `|` operator) always implies that different panels are used for each level of the conditioning factor, according to a trellis display. If `form` is a one-sided formula, histograms of the variable on the right-hand side of the formula, before a `|` operator, are displayed (the trellis function `histogram` is used). If `form` is two-sided and both its left- and right-hand side variables are numeric, scatter plots are displayed (the trellis function `xyplot` is used). Finally, if `form` is two-sided and its left-hand side variable is a factor, boxplots of the right-hand side variable by the levels of the left-hand side variable are displayed (the trellis function `bwplot` is used).

Value

A diagnostic trellis plot.

See Also

`lme`, `xyplot`, `bwplot`, `histogram`

Examples

```
fm1 <- lme(distance ~ age, Orthodont, random = ~ age | Subject)
# standardized residuals versus fitted values by gender
plot(fm1, resid(., type = "p") ~ fitted(.) | Sex, abline = 0)
# box-plots of residuals by Subject
plot(fm1, Subject ~ resid(.))
# observed versus fitted values by Subject
plot(fm1, distance ~ fitted(.) | Subject, abline = c(0,1))
```

`plot.nfnGroupedData` *Plot an* `nfnGroupedData` *Object*

```
plot(x, outer, inner, innerGroups, xlab, ylab, strip,
     aspect, panel, key, grid, ...)
```

Arguments

`x`	An object inheriting from class `nfnGroupedData`, representing a `groupedData` object with a numeric primary covariate and a single grouping level.
`outer`	An optional logical value or one-sided formula, indicating covariates that are outer to the grouping factor, which are used to determine the panels of the trellis plot. If equal to `TRUE`, `attr(object, "outer")` is used to indicate the outer covariates. An outer covariate is invariant within the sets of rows defined by the grouping factor. Ordering of the groups is done in such a way as to preserve adjacency of groups with the same value of the outer variables. Defaults to `NULL`, meaning that no outer covariates are to be used.
`inner`	An optional logical value or one-sided formula, indicating a covariate that is inner to the grouping factor, which is used to associate points within each panel of the trellis plot. If equal to `TRUE`, `attr(object, "inner")` is used to indicate the inner covariate. An inner covariate can change within the sets of rows defined by the grouping factor. Defaults to `NULL`, meaning that no inner covariate is present.
`innerGroups`	An optional one-sided formula specifying a factor to be used for grouping the levels of the `inner` covariate. Different colors, or line types, are used for each level of the `innerGroups` factor. Default is `NULL`, meaning that no `innerGroups` covariate is present.
`xlab, ylab`	Optional character strings with the labels for the plot. Default is the corresponding elements of `attr( object, "labels")` and `attr(object, "units")` pasted together.
`strip`	An optional function passed as the `strip` argument to the `xyplot` function. Default is `strip.default( ..., style = 1)` (see `trellis.args`).

`aspect`	An optional character string indicating the aspect ratio for the plot passed as the `aspect` argument to the `xyplot` function. Default is `"xy"` (see `trellis.args`).
`panel`	An optional function used to generate the individual panels in the trellis display, passed as the `panel` argument to the `xyplot` function.
`key`	An optional logical function or function. If `TRUE` and `innerGroups` is non-`NULL`, a legend for the different `innerGroups` levels is included at the top of the plot. If given as a function, it is passed as the `key` argument to the `xyplot` function. Default is `TRUE` if `innerGroups` is non-`NULL` and `FALSE` otherwise.
`grid`	An optional logical value indicating whether a grid should be added to plot. Default is `TRUE`.
`...`	Optional arguments passed to the `xyplot` function.

Description

A trellis plot of the response versus the primary covariate is generated. If outer variables are specified, the combination of their levels are used to determine the panels of the trellis display. Otherwise, the levels of the grouping variable determine the panels. A scatter plot of the response versus the primary covariate is displayed in each panel, with observations corresponding to same inner group joined by line segments. The trellis function `xyplot` is used.

Value

A trellis plot of the response versus the primary covariate.

See Also

`groupedData`, `xyplot`

Examples

```
# different panels per Subject
plot(Orthodont)
# different panels per gender
plot(Orthodont, outer = TRUE)
```

plot.nmGroupedData *Plot an nmGroupedData Object*

```
plot(x, collapseLevel, displayLevel, outer, inner,
     preserve, FUN, subset, grid, ...)
```

Arguments

`x`	An object inheriting from class `nmGroupedData`, representing a `groupedData` object with multiple grouping factors.
`collapseLevel`	An optional positive integer or character string indicating the grouping level to use when collapsing the data. Level values increase from outermost to innermost grouping. Default is the highest or innermost level of grouping.
`displayLevel`	An optional positive integer or character string indicating the grouping level to use for determining the panels in the trellis display, when `outer` is missing. Default is `collapseLevel`.
`outer`	An optional logical value or one-sided formula, indicating covariates that are outer to the `displayLevel` grouping factor, which are used to determine the panels of the trellis plot. If equal to `TRUE`, the `displayLevel` element `attr(object, "outer")` is used to indicate the outer covariates. An outer covariate is invariant within the sets of rows defined by the grouping factor. Ordering of the groups is done in such a way as to preserve adjacency of groups with the same value of the outer variables. Defaults to `NULL`, meaning that no outer covariates are to be used.
`inner`	An optional logical value or one-sided formula, indicating a covariate that is inner to the `displayLevel` grouping factor, which is used to associate points within each panel of the trellis plot. If equal to `TRUE`, `attr(object, "outer")` is used to indicate the inner covariate. An inner covariate can change within the sets of rows defined by the grouping factor. Defaults to `NULL`, meaning that no inner covariate is present.
`preserve`	An optional one-sided formula indicating a covariate whose levels should be preserved when collapsing

	the data according to the `collapseLevel` grouping factor. The collapsing factor is obtained by pasting together the levels of the `collapseLevel` grouping factor and the values of the covariate to be preserved. Default is `NULL`, meaning that no covariates need to be preserved.
`FUN`	An optional summary function or a list of summary functions to be used for collapsing the data. The function or functions are applied only to variables in `object` that vary within the groups defined by `collapseLevel`. Invariant variables are always summarized by group using the unique value that they assume within that group. If `FUN` is a single function it will be applied to each noninvariant variable by group to produce the summary for that variable. If `FUN` is a list of functions, the names in the list should designate classes of variables in the data such as `ordered`, `factor`, or `numeric`. The indicated function will be applied to any noninvariant variables of that class. The default functions to be used are `mean` for numeric factors, and `Mode` for both `factor` and `ordered`. The `Mode` function, defined internally in `gsummary`, returns the modal or most popular value of the variable. It is different from the `mode` function that returns the S-language mode of the variable.
`subset`	An optional named list. Names can be either positive integers representing grouping levels, or names of grouping factors. Each element in the list is a vector indicating the levels of the corresponding grouping factor to be used for plotting the data. Default is `NULL`, meaning that all levels are used.
`grid`	An optional logical value indicating whether a grid should be added to plot. Default is `TRUE`.
`...`	Optional arguments passed to the trellis plot function.

Description

The `groupedData` object is summarized by the values of the `display-Level` grouping factor (or the combination of its values and the values of the covariate indicated in `preserve`, if any is present). The collapsed data is used to produce a new `groupedData` object, with grouping factor given by the `displayLevel` factor, which is plotted using the appropriate `plot` method for `groupedData` objects with single level of grouping.

Value

A trellis display of the data collapsed over the values of the `collapseLevel` grouping factor and grouped according to the `displayLevel` grouping factor.

See Also

`groupedData`, `collapse.groupedData`, `plot.nfnGroupedData`, `plot.nffGroupedData`

Examples

```
# no collapsing, panels by Dog
plot(Pixel, display = "Dog", inner = ~Side)
# collapsing by Dog, preserving day
plot(Pixel, collapse = "Dog", preserve = ~day)
```

`plot.Variogram` *Plot a `Variogram` Object*

```
plot(object, smooth, showModel, sigma, span, xlab, ylab,
     type, ylim, ...)
```

Arguments

`object`	An object inheriting from class Variogram, consisting of a data frame with two columns named `variog` and `dist`, representing the semivariogram values and the corresponding distances.
`smooth`	An optional logical value controlling whether a `loess` smoother should be added to the plot. Defaults to `TRUE`, when `showModel` is `FALSE`.
`showModel`	An optional logical value controlling whether the semivariogram corresponding to an `"modelVariog"` attribute of `object`, if any is present, should be added to the plot. Defaults to `TRUE`, when the `"modelVariog"` attribute is present.
`sigma`	An optional numeric value used as the height of a horizontal line displayed in the plot. Can be used to represent the process standard deviation. Default is `NULL`, implying that no horizontal line is drawn.
`span`	An optional numeric value with the smoothing parameter for the `loess` fit. Default is 0.6.

`xlab,ylab`	Optional character strings with the x- and y-axis labels. Default respectively to `"Distance"` and `"Semi-variogram"`.
`type`	An optional character indicating the type of plot. Defaults to `"p"`.
`ylim`	An optional numeric vector with the limits for the y-axis. Defaults to `c(0, max(object$variog))`.
`...`	Optional arguments passed to the trellis `xyplot` function.

Description

An `xyplot` of the semivariogram versus the distances is produced. If `smooth = TRUE`, a `loess` smoother is added to the plot. If `showModel = TRUE` and `object` includes an `"modelVariog"` attribute, the corresponding semivariogram is added to the plot.

Value

An `xyplot` trellis plot.

See Also

`Variogram`, `xyplot`, `loess`

Examples

```
fm1 <- lme(follicles ~ sin(2*pi*Time) + cos(2*pi*Time), Ovary)
plot(Variogram(fm1, form = ~ Time | Mare, maxDist = 0.7))
```

`predict.lme` *Predictions from an `lme` Object*

```
predict(object, newdata, level, asList, na.action)
```

Arguments

`object`	An object inheriting from class `lme`, representing a fitted linear mixed-effects model.
`newdata`	An optional data frame to be used for obtaining the predictions. All variables used in the fixed- and random-effects models, as well as the grouping factors, must be present in the data frame. If missing, the fitted values are returned.

`level`	An optional integer vector giving the level(s) of grouping to be used in obtaining the predictions. Level values increase from outermost to innermost grouping, with level zero corresponding to the population predictions. Defaults to the highest or innermost level of grouping.
`asList`	An optional logical value. If `TRUE` and a single value is given in `level`, the returned object is a list with the predictions split by groups; else the returned value is either a vector or a data frame, according to the length of `level`.
`na.action`	A function that indicates what should happen when `newdata` contains `NA`s. The default action (`na.fail`) causes the function to print an error message and terminate if there are any incomplete observations.

Description

The predictions at level i are obtained by adding together the population predictions (based only on the fixed-effects estimates) and the estimated contributions of the random effects to the predictions at grouping levels less or equal to i. The resulting values estimate the best linear unbiased predictions (BLUPs) at level i. If group values not included in the original grouping factors are present in `newdata`, the corresponding predictions will be set to `NA` for levels greater or equal to the level at which the unknown groups occur.

Value

If a single level of grouping is specified in `level`, the returned value is either a list with the predictions split by groups (`asList = TRUE`) or a vector with the predictions (`asList = FALSE`); else, when multiple grouping levels are specified in `level`, the returned object is a data frame with columns given by the predictions at different levels and the grouping factors.

See Also

`lme`, `fitted.lme`

Examples

```
fm1 <- lme(distance ~ age, Orthodont, random = ~ age | Subject)
newOrth <- data.frame(Sex = c("Male","Male","Female","Female",
                              "Male","Male"),
                      age = c(15, 20, 10, 12, 2, 4),
                      Subject = c("M01","M01","F30","F30","M04",
                                   "M04"))
    predict(fm1, newOrth, level = 0:1)
```

qqnorm.lme — *Normal Plot of Residuals or Random Effects from an `lme` Object*

```
    qqnorm(object, form, abline, id, idLabels, grid, ...)
```

Arguments

`object`	An object inheriting from class `lme`, representing a fitted linear mixed-effects model.
`form`	An optional one-sided formula specifying the desired type of plot. Any variable present in the original data frame used to obtain `object` can be referenced. In addition, `object` itself can be referenced in the formula using the symbol `"."`. Conditional expressions on the right of a `|` operator can be used to define separate panels in a trellis display. The expression on the right-hand side of `form` and to the left of a `|` operator must evaluate to a residuals vector, or a random effects matrix. Default is `~ resid(., type = "p")`, corresponding to a normal plot of the standardized residuals evaluated at the innermost level of nesting.
`abline`	An optional numeric value, or numeric vector of length two. If given as a single value, a horizontal line will be added to the plot at that coordinate; else, if given as a vector, its values are used as the intercept and slope for a line added to the plot. If missing, no lines are added to the plot.
`id`	An optional numeric value, or one-sided formula. If given as a value, it is used as a significance level for a two-sided outlier test for the standardized residuals (random effects). Observations with absolute standardized residuals (random effects) greater than the $1 - value/2$ quantile of the standard normal distribution are identified in the plot using `idLabels`. If given as a one-sided formula, its right-hand side must evaluate to a logical, integer, or character vector which is used to identify observations in the plot. If missing, no observations are identified.

`idLabels`	An optional vector, or one-sided formula. If given as a vector, it is converted to character and used to label the observations identified according to `id`. If given as a one-sided formula, its right-hand side must evaluate to a vector that is converted to character and used to label the identified observations. Default is the innermost grouping factor.
`grid`	An optional logical value indicating whether a grid should be added to plot. Default is `FALSE`.
`...`	Optional arguments passed to the trellis plot function.

Description

Diagnostic plots for assessing the normality of residuals and random effects in the linear mixed-effects fit are obtained. The `form` argument gives considerable flexibility in the type of plot specification. A conditioning expression (on the right side of a `|` operator) always implies that different panels are used for each level of the conditioning factor, according to a trellis display.

Value

A diagnostic trellis plot for assessing normality of residuals or random effects.

See Also

`lme`, `plot.lme`

Examples

```
fm1 <- lme(distance ~ age, Orthodont, random = ~ age | Subject)
# normal plot of standardized residuals by gender
qqnorm(fm1, ~ resid(., type = "p") | Sex, abline = c(0, 1))
# normal plots of random effects
qqnorm(fm1, ~ranef(.))
```

`ranef` *Extract Random Effects*

```
ranef(object, ...)
```

Arguments

`object`	Any fitted model object from which random effects estimates can be extracted.
`...`	Some methods for this generic function require additional arguments.

Description

This function is generic; method functions can be written to handle specific classes of objects. Classes that already have methods for this function include lmList and lme.

Value

Will depend on the method function used; see the appropriate documentation.

See Also

`ranef.lmList`, `ranef.lme`

`ranef.lme` *Extract `lme` Random Effects*

```
ranef(object, augFrame, level, data, which, FUN, standard,
      omitGroupingFactor)
```

Arguments

`object`	An object inheriting from class lme, representing a fitted linear mixed-effects model.
`augFrame`	An optional logical value. If `TRUE`, the returned data frame is augmented with variables defined in `data`; else, if `FALSE`, only the coefficients are returned. Defaults to `FALSE`.
`level`	An optional vector of positive integers giving the levels of grouping to be used in extracting the random effects from an object with multiple nested grouping levels. Defaults to all levels of grouping.
`data`	An optional data frame with the variables to be used for augmenting the returned data frame when `augFrame = TRUE`. Defaults to the data frame used to fit `object`.

`which`	An optional positive integer vector specifying which columns of `data` should be used in the augmentation of the returned data frame. Defaults to all columns in `data`.
`FUN`	An optional summary function or a list of summary functions to be applied to group-varying variables, when collapsing `data` by groups. Group-invariant variables are always summarized by the unique value that they assume within that group. If `FUN` is a single function it will be applied to each noninvariant variable by group to produce the summary for that variable. If `FUN` is a list of functions, the names in the list should designate classes of variables in the frame such as `ordered`, `factor`, or `numeric`. The indicated function will be applied to any group-varying variables of that class. The default functions to be used are `mean` for numeric factors, and `Mode` for both `factor` and `ordered`. The `Mode` function, defined internally in `gsummary`, returns the modal or most popular value of the variable. It is different from the `mode` function that returns the S-language mode of the variable.
`standard`	An optional logical value indicating whether the estimated random effects should be "standardized"(i.e., divided by the corresponding estimated standard error). Defaults to `FALSE`.
`omitGroupingFactor`	An optional logical value. When `TRUE`, the grouping factor itself will be omitted from the groupwise summary of `data`, but the levels of the grouping factor will continue to be used as the row names for the returned data frame. Defaults to `FALSE`.

Description

The estimated random effects at level i are represented as a data frame with rows given by the different groups at that level and columns given by the random effects. If a single level of grouping is specified, the returned object is a data frame; else, the returned object is a list of such data frames. Optionally, the returned data frame(s) may be augmented with covariates summarized over groups.

Value

A data frame, or list of data frames, with the estimated random effects at the grouping level(s) specified in `level` and, optionally, other covariates summarized over groups. The returned object inherits from classes ranef.lme and data.frame.

See Also

`lme`, `fixed.effects.lme`, `coef.lme`, `plot.ranef.lme`, `gsummary`

Examples

```
fm1 <- lme(distance ~ age, Orthodont, random = ~ age | Subject)
ranef(fm1)
ranef(fm1, augFrame = TRUE)
```

`ranef.lmList` *Extract `lmList` Random Effects*

```
ranef(object, augFrame, data, which, FUN, standard,
      omitGroupingFactor)
```

Arguments

`object`	An object inheriting from class lmList, representing a list of `lm` objects with a common model.
`augFrame`	An optional logical value. If `TRUE`, the returned data frame is augmented with variables defined in the data frame used to produce `object`; else, if `FALSE`, only the random effects are returned. Defaults to `FALSE`.
`data`	An optional data frame with the variables to be used for augmenting the returned data frame when `augFrame = TRUE`. Defaults to the data frame used to fit `object`.
`which`	An optional positive integer or character vector specifying which columns of the data frame used to produce `object` should be used in the augmentation of the returned data frame. Defaults to all variables in the data.
`FUN`	An optional summary function or a list of summary functions to be applied to group-varying variables, when collapsing the data by groups. Group-invariant variables are always summarized by the unique value that they assume within that group. If `FUN` is a single function it will be applied to each noninvariant variable by group to produce the summary for that variable. If `FUN` is a list of functions, the names in

the list should designate classes of variables in the frame such as `ordered`, `factor`, or `numeric`. The indicated function will be applied to any group-varying variables of that class. The default functions to be used are `mean` for numeric factors, and `Mode` for both `factor` and `ordered`. The `Mode` function, defined internally in `gsummary`, returns the modal or most popular value of the variable. It is different from the `mode` function that returns the S-language mode of the variable.

`standard` An optional logical value indicating whether the estimated random effects should be "standardized" (i.e., divided by the corresponding estimated standard error). Defaults to `FALSE`.

`omitGroupingFactor` An optional logical value. When `TRUE`, the grouping factor itself will be omitted from the groupwise summary of `data`, but the levels of the grouping factor will continue to be used as the row names for the returned data frame. Defaults to `FALSE`.

Description

A data frame containing the differences between the coefficients of the individual `lm` fits and the average coefficients.

Value

A data frame with the differences between the individual `lm` coefficients in `object` and their average. Optionally, the returned data frame may be augmented with covariates summarized over groups or the differences may be standardized.

See Also

`lmList`, `fixef.lmList`

Examples

```
fm1 <- lmList(distance ~ age | Subject, Orthodont)
ranef(fm1)
ranef(fm1, standard = TRUE)
ranef(fm1, augFrame = TRUE)
```

residuals.lme *Extract lme Residuals*

```
residuals(object, level, type, asList)
```

Arguments

object	An object inheriting from class lme, representing a fitted linear mixed-effects model.
level	An optional integer vector giving the level(s) of grouping to be used in extracting the residuals from `object`. Level values increase from outermost to innermost grouping, with level zero corresponding to the population residuals. Defaults to the highest or innermost level of grouping.
type	An optional character string specifying the type of residuals to be used. If `"response"`, the "raw" residuals (observed – fitted) are used; else, if `"pearson"`, the standardized residuals (raw residuals divided by the corresponding standard errors) are used; else, if `"normalized"`, the normalized residuals (standardized residuals premultiplied by the inverse square-root factor of the estimated error correlation matrix) are used. Partial matching of arguments is used, so only the first character needs to be provided. Defaults to `"pearson"`.
asList	An optional logical value. If `TRUE` and a single value is given in `level`, the returned object is a list with the residuals split by groups; else the returned value is either a vector or a data frame, according to the length of `level`. Defaults to `FALSE`.

Description

The residuals at level i are obtained by subtracting the fitted levels at that level from the response vector (and dividing by the estimated within-group standard error, if `type="pearson"`). The fitted values at level i are obtained by adding together the population-fitted values (based only on the fixed-effects estimates) and the estimated contributions of the random effects to the fitted values at grouping levels less or equal to i.

Value

If a single level of grouping is specified in `level`, the returned value is either a list with the residuals split by groups (`asList = TRUE`) or a vector with the residuals (`asList = FALSE`); else, when multiple grouping levels are specified in `level`, the returned object is a data frame with columns given by the residuals at different levels and the grouping factors.

See Also

`lme`, `fitted.lme`

Examples

```
fm1 <- lme(distance ~ age + Sex, data = Orthodont, random = ~ 1)
residuals(fm1, level = 0:1)
```

`selfStart` *Construct Self-Starting Nonlinear Models*

```
selfStart(model, initial, parameters, template)
```

Description

This function is generic; methods functions can be written to handle specific classes of objects. Available methods include `selfStart.default` and `selfStart.formula`. See the documentation on the appropriate method function.

Value

A function object of the selfStart class.

See Also

`selfStart.default`, `selfStart.formula`

`selfStart.default` *Construct Self-Starting Nonlinear Models*

```
selfStart(model, initial, parameters, template)
```

Arguments

`model`	A function object defining a nonlinear model.
`initial`	A function object, taking three arguments: `mCall`, `data`, and `LHS`, representing, respectively, a matched call to the function `model`, a data frame in which to interpret the variables in `mCall`, and the expression from the left-hand side of the model formula in the call to `nls`. This function should return initial values for the parameters in `model`.
`parameters, template`	These arguments are included for consistency with the generic function, but are not used in the `default` method. See the documentation on `selfStart.formula`.

Description

A method for the generic function `selfStart` for formula objects.

Value

A function object of class selfStart, corresponding to a self-starting nonlinear model function. An `initial` attribute (defined by the `initial` argument) is added to the function to calculate starting estimates for the parameters in the model automatically.

See Also

`selfStart.formula`

Examples

```
# ‘first.order.log.model’ is a function object defining a first
#  order compartment model
# ‘first.order.log.initial’ is a function object which calculates
# initial values for the parameters in ‘first.order.log.model’
# self-starting first order compartment model
SSfol <- selfStart(first.order.log.model,
                   first.order.log.initial)
```

`selfStart.formula` *Construct Self-Starting Nonlinear Models*

```
selfStart(model, initial, parameters, template)
```

Arguments

`model`	A nonlinear formula object of the form `~expression`.
`initial`	A function object, taking three arguments: `mCall`, `data`, and `LHS`, representing, respectively, a matched call to the function `model`, a data frame in which to interpret the variables in `mCall`, and the expression from the left-hand side of the model formula in the call to `nls`. This function should return initial values for the parameters in `model`.
`parameters`	A character vector specifying the terms on the right-hand side of `model` for which initial estimates should be calculated. Passed as the `namevec` argument to the `deriv` function.
`template`	An optional prototype for the calling sequence of the returned object, passed as the `function.arg` argument to the `deriv` function. By default, a template is generated with the covariates in `model` coming first and the parameters in `model` coming last in the calling sequence.

Description

A method for the generic function `selfStart` for formula objects.

Value

A function object of class `selfStart`, obtained by applying `deriv` to the right-hand side of the `model` formula. An `initial` attribute (defined by the `initial` argument) is added to the function to calculate starting estimates for the parameters in the model automatically.

See Also

`selfStart.default`, `deriv`

Examples

```
## self-starting logistic model
SSlogis <- selfStart(~ Asym/(1 + exp((xmid - x)/scal)),
  function(mCall, data, LHS)
  {
    xy <- sortedXyData(mCall[["x"]], LHS, data)
    if(nrow(xy) < 4) {
      stop("Too few distinct x values to fit a logistic")
    }
    z <- xy[["y"]]
    if (min(z) <= 0) { z <- z + 0.05 * max(z) } # avoid zeroes
    z <- z/(1.05 * max(z))          # scale to within unit height
    xy[["z"]] <- log(z/(1 - z))     # logit transformation
    aux <- coef(lm(x ~ z, xy))
    parameters(xy) <- list(xmid = aux[1], scal = aux[2])
    pars <- as.vector(coef(nls(y ~ 1/(1 + exp((xmid - x)/scal)),
                             data = xy, algorithm = "plinear")))
    value <- c(pars[3], pars[1], pars[2])
    names(value) <- mCall[c("Asym", "xmid", "scal")]
    value
  }, c("Asym", "xmid", "scal"))
```

`Variogram` *Calculate Semivariogram*

```
Variogram(object, distance, ...)
```

Description

This function is generic; method functions can be written to handle specific classes of objects. Classes that already have methods for this function include default, gls and lme. See the appropriate method documentation for a description of the arguments.

Value

Will depend on the method function used; see the appropriate documentation.

See Also

`Variogram.default`,`Variogram.gls`, `Variogram.lme`, `plot.Variogram`

`Variogram.lme` *Calculate Semivariogram for Residuals from an `lme` Object*

```
Variogram(object, distance, form, resType, data,
          na.action, maxDist, length.out, collapse, nint,
          breaks, robust, metric)
```

Arguments

`object`	An object inheriting from class `lme`, representing a fitted linear mixed-effects model.
`distance`	An optional numeric vector with the distances between residual pairs. If a grouping variable is present, only the distances between residual pairs within the same group should be given. If missing, the distances are calculated based on the values of the arguments `form`, `data`, and `metric`, unless `object` includes a `corSpatial` element, in which case the associated covariate (obtained with the `getCovariate` method) is used.
`form`	An optional one-sided formula specifying the covariate(s) to be used for calculating the distances between residual pairs and, optionally, a grouping factor for partitioning the residuals (which must appear to the right of a `|` operator in `form`). Default is `~1`, implying that the observation order within the groups is used to obtain the distances.
`resType`	An optional character string specifying the type of residuals to be used. If `"response"`, the “raw” residuals (observed – fitted) are used; else, if `"pearson"`, the standardized residuals (raw residuals divided by the corresponding standard errors) are used; else, if `"normalized"`, the normalized residuals (standardized residuals premultiplied by the inverse squareroot factor of the estimated error correlation matrix) are used. Partial matching of arguments is used, so only the first character needs to be provided. Defaults to `"pearson"`.
`data`	An optional data frame in which to interpret the variables in `form`. By default, the same data used to fit `object` is used.
`na.action`	A function that indicates what should happen when the data contain `NA`s. The default action (`na.fail`)

	causes an error message to be printed and the function to terminate, if there are any incomplete observations.
`maxDist`	An optional numeric value for the maximum distance used for calculating the semivariogram between two residuals. By default all residual pairs are included.
`length.out`	An optional integer value. When `object` includes a `corSpatial` element, its semivariogram values are calculated and this argument is used as the `length.out` argument to the corresponding `Variogram` method. Defaults to `50`.
`collapse`	An optional character string specifying the type of collapsing to be applied to the individual semivariogram values. If equal to `"quantiles"`, the semivariogram values are split according to quantiles of the distance distribution, with equal number of observations per group, with possibly varying distance interval lengths. Else, if `"fixed"`, the semivariogram values are divided according to distance intervals of equal lengths, with possibly different number of observations per interval. Else, if `"none"`, no collapsing is used and the individual semivariogram values are returned. Defaults to `"quantiles"`.
`nint`	An optional integer with the number of intervals to be used when collapsing the semivariogram values. Defaults to `20`.
`robust`	An optional logical value specifying if a robust semivariogram estimator should be used when collapsing the individual values. If `TRUE` the robust estimator is used. Defaults to `FALSE`.
`breaks`	An optional numeric vector with the breakpoints for the distance intervals to be used in collapsing the semivariogram values. If not missing, the option specified in `collapse` is ignored.
`metric`	An optional character string specifying the distance metric to be used. The currently available options are `"euclidean"` for the root sum-of-squares of distances; `"maximum"` for the maximum difference; and `"manhattan"` for the sum of the absolute differences. Partial matching of arguments is used, so only the first three characters need to be provided. Defaults to `"euclidean"`.

Description

This method function calculates the semivariogram for the within-group residuals from an `lme` fit. The semivariogram values are calculated for pairs of residuals within the same group. If `collapse` is different from `"none"`, the individual semivariogram values are collapsed using either a robust estimator (`robust = TRUE`) defined in Cressie (1993), or the average of the values within the same distance interval. The semivariogram is useful for modeling the error term correlation structure.

Value

A data frame with columns `variog` and `dist` representing, respectively, the semivariogram values and the corresponding distances. If the semivariogram values are collapsed, an extra column, `n.pairs`, with the number of residual pairs used in each semivariogram calculation, is included in the returned data frame. If `object` includes a `corSpatial` element, a data frame with its corresponding semivariogram is included in the returned value, as an attribute `"modelVariog"`. The returned value inherits from class Variogram.

See Also

`lme`, `Variogram.default`, `Variogram.gls`, `plot.Variogram`

Examples

```
fm1 <- lme(weight ~ Time * Diet, BodyWeight, ~ Time | Rat)
Variogram(fm1, form = ~ Time | Rat, nint = 10, robust = TRUE)
```

Appendix C
A Collection of Self-Starting Nonlinear Regression Models

We have mentioned several self-starting nonlinear regression models in the text. In this appendix we describe each of the self-starting models included with the `nlme` library. For each model we give the model formula, a description of the parameters, and the strategy used to obtain starting estimates.

C.1 `SSasymp`—The Asymptotic Regression Model

The asymptotic regression model is used to model a response y that approaches a horizontal asymptote as $x \to \infty$. We write it as

$$y(x) = \phi_1 + (\phi_2 - \phi_1) \exp[-\exp(\phi_3) x], \tag{C.1}$$

so that ϕ_1 is the asymptote as $x \to \infty$ and ϕ_2 is $y(0)$. These parameters are shown in Figure C.1. The parameter ϕ_3 is the logarithm of the rate constant. We use the logarithm to enforce positivity of the rate constant so the model does approach an asymptote. The corresponding half-life $t_{0.5} = \log 2/\exp(\phi_3)$ is illustrated in Figure C.1.

C.1.1 Starting Estimates for `SSasymp`

Starting values for the asymptotic regression model are obtained by:

1. Using `NLSstRtAsymptote` to get an estimate $\phi_1^{(0)}$ of the asymptote.
2. Regressing $\log(|y - \phi_1^{(0)}|)$ on t. The estimated slope is $-\exp(\phi_3^{(0)})$.

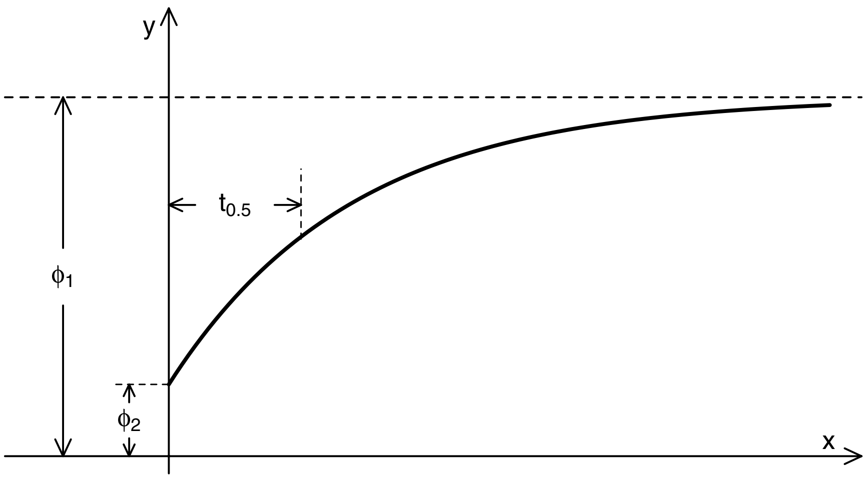

FIGURE C.1. The asymptotic regression model showing the parameters ϕ_1, the asymptotic response as $x \to \infty$, ϕ_2, the response at $x = 0$, and $t_{0.5}$, the half-life.

3. Using an algorithm for partially linear models (Bates and Chambers, 1992, §10.2.5) to refine estimates of ϕ_1, ϕ_2, and ϕ_3 in

$$y(x) = \phi_1 + (\phi_2 - \phi_1)\exp[\exp(\phi_3)x].$$

   Because ϕ_1 and ϕ_2 occur linearly in the model expression, the least squares fit iterates over a single parameter.

These estimates are the final nonlinear regression estimates.

C.2 `SSasympOff`—Asymptotic Regression with an Offset

This is an alternative form of the asymptotic regression model that provides a more stable parameterization for the `CO2` data. It is written

$$y(x) = \phi_1\{1 - \exp[-\exp(\phi_2) \times (x - \phi_3)]\}. \qquad \text{(C.2)}$$

As in `SSasymp`, ϕ_1 is the asymptote as $x \to \infty$. In this formulation ϕ_2 is the logarithm of the rate constant, corresponding to a half-life of $t_{0.5} = \log 2/\exp(\phi_2)$, and ϕ_3 is the value of x at which $y = 0$. The parameters ϕ_1, $t_{0.5}$, and ϕ_3 are shown in Figure C.2.

C.2.1 Starting Estimates for `SSasympOff`

First we fit `SSasymp` then we transform the parameters to the formulation used in `SSasympOff`. If ***o****mega* is the vector of parameters from `SSasymp` and

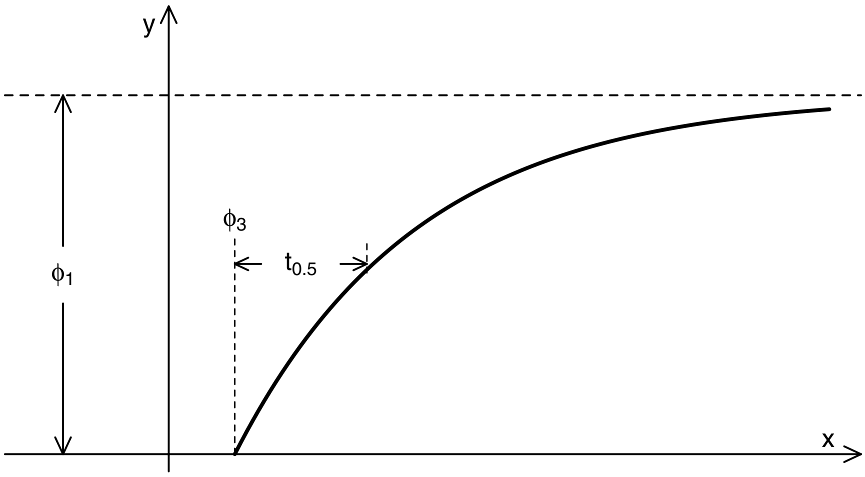

FIGURE C.2. The asymptotic regression model with an offset showing the parameters ϕ_1, the asymptote as $x \to \infty$, $t_{0.5}$, the half-life, and ϕ_3, the value of x for which $y = 0$.

$\boldsymbol{\phi}$ is the vector of parameters for `SSasympOff`, the correspondence is

$$\begin{aligned}
\phi_1 &= \omega_1, \\
\phi_2 &= \omega_3, \\
\phi_3 &= \exp(-\omega_3)\log[-(\omega_2 - \omega_1)/\omega_1].
\end{aligned}$$

These estimates are the final nonlinear regression estimates.

C.3 `SSasympOrig`—Asymptotic Regression Through the Origin

This form of the asymptotic regression model is constrained to pass through the origin. It is called the BOD model in Bates and Watts (1988) where it is used to model Biochemical Oxygen Demand curves. The model is written

$$y(x) = \phi_1[1 - \exp(-\exp(\phi_2)x]. \tag{C.3}$$

As in `SSasympOff`, ϕ_1 is the asymptote as $x \to \infty$ and ϕ_2 is the logarithm of the rate constant, corresponding to a half-life of $t_{0.5} = \log 2/\exp(\phi_2)$. The parameters ϕ_1 and $t_{0.5}$ are shown in Figure C.3.

C.3.1 Starting Estimates for `SSasympOrig`

Starting values for this regression model are obtained by:

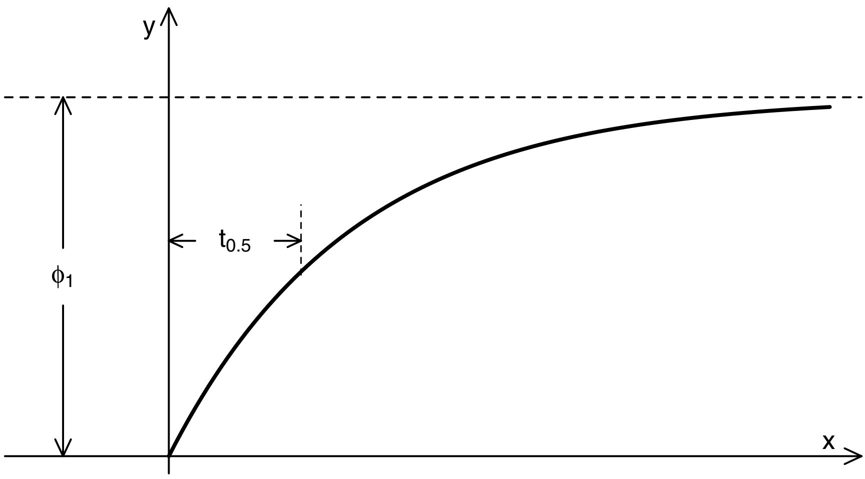

FIGURE C.3. The asymptotic regression model through the origin showing the parameters ϕ_1, the asymptote as $x \to \infty$ and $t_{0.5}$, the half-life.

1. Using `NLSstRtAsymptote` to get an estimate $\phi_1^{(0)}$ of the asymptote.
2. Obtaining an initial estimate of ϕ_2 as
$$\phi_2^{(0)} = \log \operatorname{abs} \sum_{i=1}^{n} \left[\log(1 - y_i/\phi_1^{(0)})/x_i \right] /n.$$
3. Using an algorithm for partially linear models to refine the estimates of ϕ_1 and ϕ_2. Because ϕ_1 occurs linearly in the model expression, the least squares fit iterates over a single parameter.

These estimates are the final nonlinear regression estimates.

C.4 `SSbiexp`—Biexponential Model

The biexponential model is a linear combination of two negative exponential terms
$$y(x) = \phi_1 \exp\left[-\exp(\phi_2)x\right] + \phi_3 \exp\left[-\exp(\phi_4)x\right]. \tag{C.4}$$
The parameters ϕ_1 and ϕ_3 are the coefficients of the linear combination, and the parameters ϕ_2 and ϕ_4 are the logarithms of the rate constants. The two sets of parameters (ϕ_1, ϕ_2) and (ϕ_3, ϕ_4) are *exchangeable*, meaning that the values of the pairs can be exchanged without changing the value of $y(x)$. We create an identifiable parameterization by requiring that $\phi_2 > \phi_4$.

A representative biexponential model, along with its constituent exponential curves, is shown in Figure C.4.

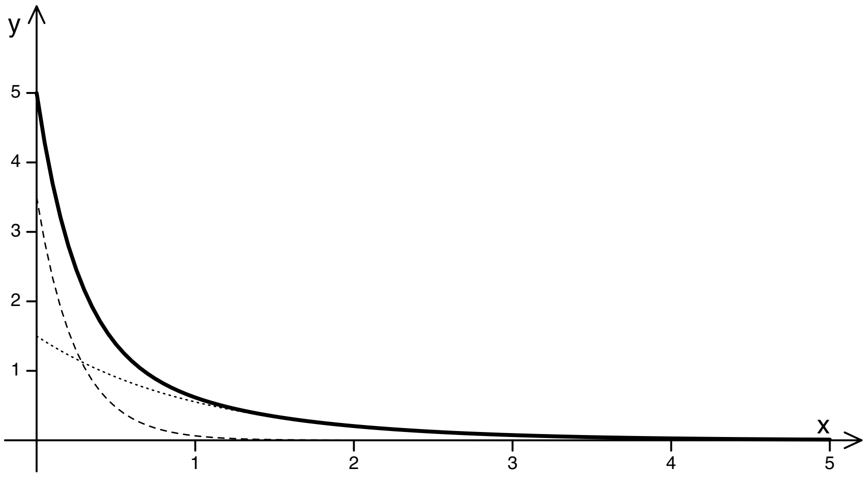

FIGURE C.4. A biexponential model showing the linear combination of the exponentials (solid line) and its constituent exponential curves (dashed line and dotted line). The dashed line is $3.5\exp(-4x)$ and the dotted line is $1.5\exp(-x)$.

C.4.1 Starting Estimates for `SSbiexp`

The starting estimates for the biexponential model are determined by *curve peeling*, which involves:

1. Choosing half the data with the largest x values and fitting the simple linear regression model
$$\log \operatorname{abs}(y) = a + bx.$$
2. Setting $\phi_3^{(0)} = \exp a$ and $\phi_4^{(0)} = \log \operatorname{abs}(b)$ and calculating the residuals $r_i = y_i - \phi_3^{(0)} \exp[-\exp(\phi_4^{(0)})x_i]$ for the half of the data with the smallest x values. Fit the simple linear regression model
$$\log \operatorname{abs}(r) = a + bx.$$
3. Setting $\phi_2^{(0)} = \log \operatorname{abs}(b)$ and using an algorithm for partially linear models to refine the estimates of ϕ_1, ϕ_2, ϕ_3, and ϕ_4. Because the model is linear in ϕ_1 and ϕ_3, the only starting estimates used in this step are those for ϕ_2 and ϕ_4 and the iterations are with respect to these two parameters.

The estimates obtained this way are the final nonlinear regression estimates.

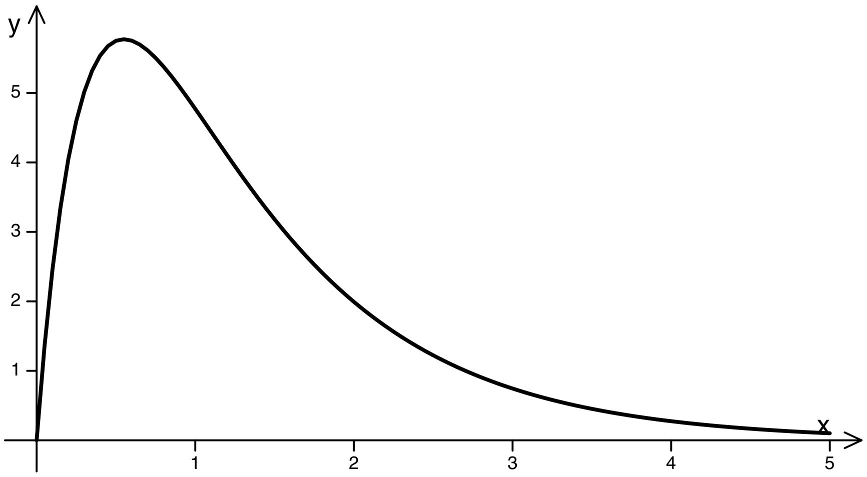

FIGURE C.5. A sample response curve from a first-order open-compartment model. The parameters correspond to an elimination rate constant of 1, an absorption rate constant of 3, and a clearance of 0.1. The dose is 1.

C.5 `SSfol`—First-Order Compartment Model

This model is derived from a compartment model in pharmacokinetics describing the concentration of a drug in the serum following a single oral dose. The model is based on first-order kinetics for the absorption of the drug from the digestive system and for the elimination of the drug from the circulatory system. Because the drug is eliminated from the circulatory system, the system of compartments is called an open system, and the model is a first-order open compartment model. It is written

$$y(x) = \frac{D\exp(\phi_1)\exp(\phi_2)}{\exp(\phi_3)\left[\exp(\phi_2)-\exp(\phi_1)\right]}\left\{\exp\left[-\exp(\phi_1)x\right]-\exp\left[-\exp(\phi_2)x\right]\right\}, \tag{C.5}$$

where D is the dose, ϕ_1 is the logarithm of the elimination rate constant, ϕ_2 is the logarithm of the absorption rate constant, and ϕ_3 is the logarithm of the clearance.

A sample response curve from a first-order open compartment model is shown in Figure C.5

C.5.1 Starting Estimates for `SSfol`

The starting estimates for the `SSfol` model are also determined by curve peeling. The steps are:

1. Determine the position of the maximum response. Fit the simple linear regression model

   $$\log(y) = a + bx$$

   to the data with x values greater than or equal to the position of the maximum response. Set $\phi_1^{(0)} = \log \operatorname{abs}(b)$ and $\phi_2^{(0)} = \phi_1^{(0)} + 1$.

2. Use an algorithm for partially linear models to fit the nonlinear regression model

   $$y(x) = k\{\exp[-\exp(\phi_1)x] - \exp[-\exp(\phi_2)x]\}$$

   refining the estimates of ϕ_1 and ϕ_2.

3. Use the current estimates of ϕ_1 and ϕ_2 and an algorithm for partially linear models to fit

   $$y(x) = kD\frac{\exp[-\exp(\phi_1)x] - \exp[\exp(\phi_2)x)]}{\exp(\phi_1) - \exp(\phi_2)}.$$

   Set $\phi_3 = \phi_1 + \phi_2 - \log k$.

These estimates are the final nonlinear regression estimates.

C.6 `SSfpl`—Four-Parameter Logistic Model

The four-parameter logistic model relates a response y to an input x via a sigmoidal or "S-shaped" function. We write it as

$$y(x) = \phi_1 + \frac{\phi_2 - \phi_1}{1 + \exp\left[\left(\phi_3 - x\right)/\phi_4\right]}. \tag{C.6}$$

We require that $\phi_4 > 0$ so the parameters are:

- ϕ_1 the horizontal asymptote as $x \to \infty$
- ϕ_2 the horizontal asymptote as $x \to -\infty$
- ϕ_3 the x value at the inflection point. At this value of x the response is midway between the asymptotes.
- ϕ_4 a scale parameter on the x-axis. When $x = \phi_3 + \phi_4$ the response is $\phi_1 + (\phi_2 - \phi_1)/(1 + e^{-1})$ or roughly three-quarters of the distance from ϕ_1 to ϕ_2.

These parameters are shown in Figure C.6

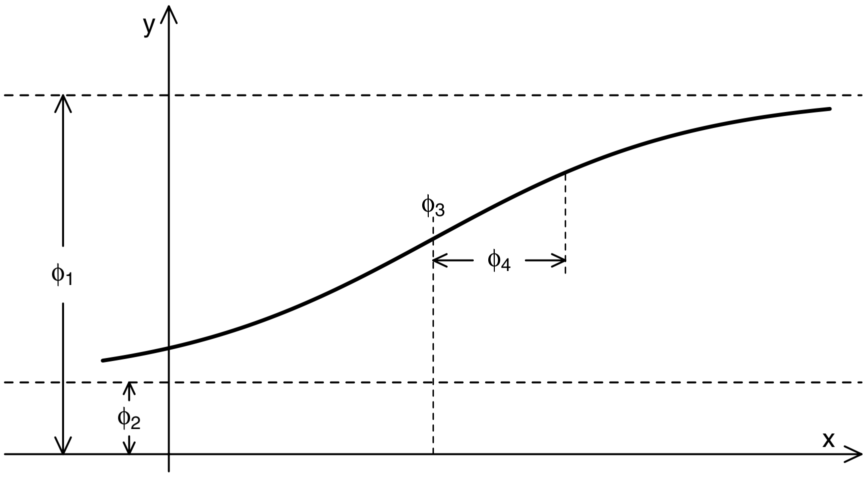

FIGURE C.6. The four-parameter logistic model. The parameters are the horizontal asymptote ϕ_1 as $x \to -\infty$, the horizontal asymptote ϕ_2 as $x \to \infty$, the x value at the inflection point (ϕ_3), and a scale parameter ϕ_4.

C.6.1 Starting Estimates for `SSfpl`

The steps in determining starting estimates for the `SSfpl` model are:

1. Use `NLSstClosestX` to determine $\phi_3^{(0)}$ as the x value corresponding a response at the midpoint of the range of the responses.

2. Use an algorithm for partially linear models to fit A, B, and ℓ while holding ϕ_3 fixed in the nonlinear regression model

   $$y(x) = A + \frac{B}{1 + \exp[(\phi_3 - x)/\exp\ell]}.$$

   The purpose of this fit is to refine the estimate of ℓ, the logarithm of the scale parameter ϕ_4. We start ℓ at zero.

3. Use the refined estimate of ℓ and an algorithm for partially linear models to fit

   $$y(x) = A + \frac{B}{1 + \exp[(\phi_3 - x)/\exp\ell]}$$

   with respect to A, B, ϕ_3 and ℓ. The estimates are then $\phi_1 = A$, $\phi_2 = A + B$, $\phi_4 = \exp\ell$ and ϕ_3.

These estimates are the final nonlinear regression estimates.

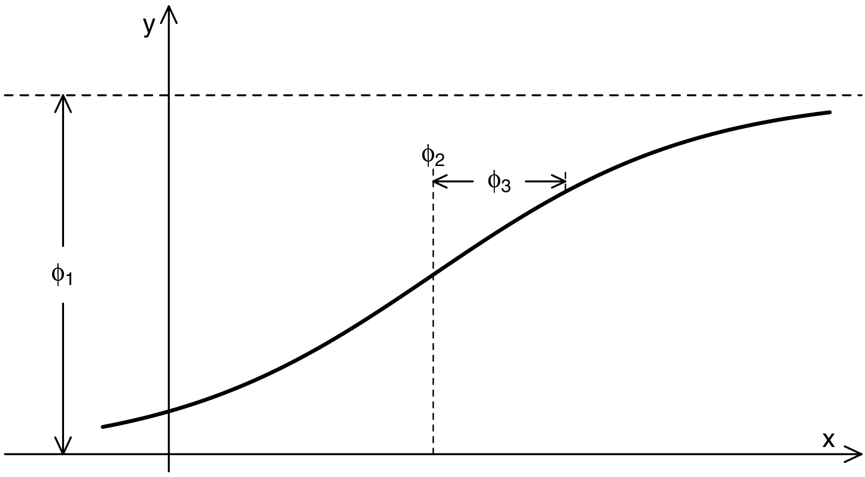

FIGURE C.7. The simple logistic model showing the parameters ϕ_1, the horizontal asymptote as $x \to \infty$, ϕ_2, the value of x for which $y = \phi_1/2$, and ϕ_3, a scale parameter on the x-axis. If $\phi_3 < 0$ the curve will be monotone decreasing instead of monotone increasing and ϕ_1 will be the horizontal asymptote as $x \to -\infty$.

C.7 `SSlogis`—Simple Logistic Model

The simple logistic model is a special case of the four-parameter logistic model in which one of the horizontal asymptotes is zero. We write it as

$$y(x) = \frac{\phi_1}{1 + \exp\left[(\phi_2 - x)/\phi_3\right]}. \tag{C.7}$$

For this model we do not require that the scale parameter ϕ_3 be positive. If $\phi_3 > 0$ then ϕ_1 is the horizontal asymptote as $x \to \infty$ and 0 is the horizontal asymptote as $x \to -\infty$. If $\phi_3 < 0$, these roles are reversed. The parameter ϕ_2 is the x value at which the response is $\phi_1/2$. It is the inflection point of the curve. The scale parameter ϕ_3 represents the distance on the x-axis between this inflection point and the point where the response is $\phi_1/\left(1 + e^{-1}\right) \approx 0.73\phi_1$. These parameters are shown in Figure C.7.

C.7.1 *Starting Estimates for `SSlogis`*

The starting estimates are determined by:

1. Scaling and, if necessary, shifting the responses y so the transformed responses y' are strictly within the interval $(0, 1)$.

2. Taking the logistic transformation

$$z = \log[y'/(1 - y')]$$

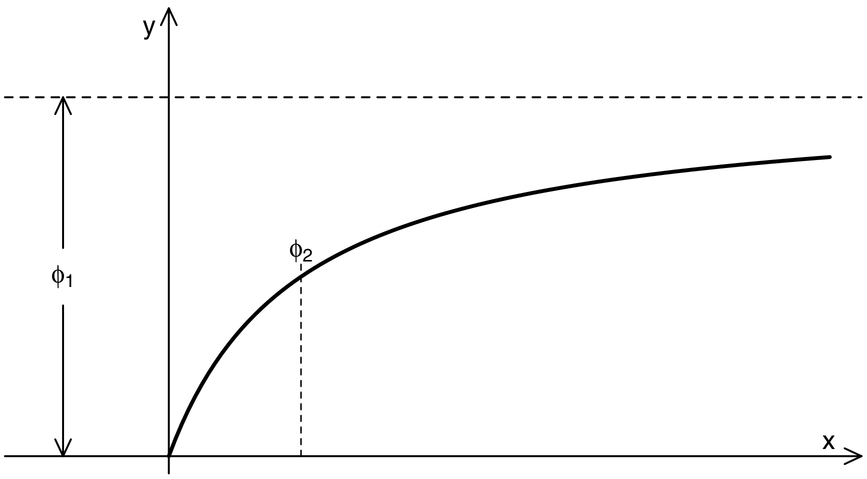

FIGURE C.8. The Michaelis–Menten model used in enzyme kinetics. The parameters are ϕ_1, the horizontal asymptote as $x \to \infty$ and ϕ_2, the value of x at which the response is $\phi_1/2$.

and fitting the simple linear regression model

$$x = a + bz.$$

3. Use $\phi_2^{(0)} = a$ and $\phi_3^{(0)} = b$ and an algorithm for partially linear models to fit

$$y = \frac{\phi_1}{1 + \exp[(\phi_2 - x)/\phi_3]}.$$

The resulting estimates are the final nonlinear regression estimates.

C.8 `SSmicmen`—Michaelis–Menten Model

The Michaelis–Menten model is used in enzyme kinetics to relate the initial rate of an enzymatic reaction to the concentration of the substrate. It is written

$$y(x) = \frac{\phi_1 x}{\phi_2 + x}, \tag{C.8}$$

where ϕ_1 is the horizontal asymptote as $x \to \infty$ and ϕ_2, the Michaelis parameter, is the value of x at which the response is $\phi_1/2$.

These parameters are shown in Figure C.8

C.8.1 Starting Estimates for `SSmicmen`

The starting estimates are obtained by:

1. Fitting a simple linear regression model

$$\frac{1}{y} = a + b\frac{1}{x}$$

   for the inverse response as a function of the inverse of x.

2. Setting $\phi_2^{(0)} = \text{abs}(b/a)$ and using an algorithm for partially linear models to fit

$$y = \frac{\phi_1 x}{\phi_2 + x}.$$

The resulting estimates are the final nonlinear regression estimates.

Index