Making Sense of the Mixed Model Analyses Available in R

by Peter Claussen

Gyling Data Management

Given a sequence of steps in the analysis of experimental data,

- Data Entry
- Model Fitting
- Diagnostics
- Model Building
- Summary Statistics
- Hypothesis Testing
- Presentation

how easily can an unfamiliar analysis method be inserted into a familiar routine?

This workflow, in R, will be illustrated with the following mixed effect data sets ...

1 Simulated Data for an RCBD with Two Missing Observations.

Table 5.8 from ***Planning, Construction, and Statistical Analysis of Comparative Experiments***, F. G. Giesbrecht and M. L. Gumpertz, *Wiley-Interscience*. (2004).

These data are an simulated randomized complete block design with $t = 6$ treatments and $r = 4$ replicates or whole blocks. Assuming blocks are random effects, we would write the expected means squares for an analysis of variance as

Source	d.f.	$E[\mathrm{MS}]$
Block	$r-1$	$\sigma^2_\epsilon + t\sigma^2_b$
Treatment	$t-1$	$\sigma^2_\epsilon + r\theta^2_\tau$
Residual	$(r-1)(t-1)$	σ^2_ϵ

Table 1. Expected mean squares in an analysis of variance of an RCB experiment. $\theta^2_\tau = (\Sigma^t_{j=1}\tau^2_j)\,/\,(t-1)$ for treatments effects τ_j.

This allows us to estimate properties of random effects in a mixed effects statitistical model from a linear analysis of variance.

The data were simulated with means $=\{4.5,\ 4.7,\ 4.9,\ 5.1,\ 5.3,\ 5.5\}$, $\sigma_b^2 = 1$ and $\sigma_\epsilon^2 = 1$. Observations are missing for treatment 1 in replicates 1 and 2.

Two missing observations means that treatment and replicate effects are confounded, and there is no unique decomposition of sums of squares.

Thus, replicate variance σ_b^2 cannot be estimated by setting Block MS to the expected value and solving algebraically, that is, $\sigma_b^2 = (\text{Block MS?} - \sigma_\epsilon^2)\,/\,t?$.

These data might be entered in R as

```
> Table5.7 <- data.frame(
          Block = as.factor(c(1,1,1,1,1,1,2,2,2,2,2,2,3,3,3,3,3,3,4,4,4,4,4,4)),
          Treatment = as.factor(c(4,5,6,3,1,2,6,4,2,5,3,1,2,3,1,6,5,4,3,6,2,5,4,1)),
          Observation = c(3.43,5.25,6.47,2.8,NA,2.66,8.43,6.09,6.41,5.69,7.04,NA,6.07,
                  6.19,4.93,5.95,4.99,3.26,5.22,7.35,4.48,6.34,6.71,3.13))
  #some packages choke on missing values
  Table5.7 <- subset(Table5.7, !is.na(Table5.7$Observation))
```

The data were simulated with means $=\{4.5,\ 4.7,\ 4.9,\ 5.1,\ 5.3,\ 5.5\}$, $\sigma_b^2 = 1$ and $\sigma_\epsilon^2 = 1$. Observations are missing for treatment 1 in replicates 1 and 2.

Two missing observations means that treatment and replicate effects are confounded, and there is no unique decomposition of sums of squares.

Thus, replicate variance σ_b^2 cannot be estimated by setting Block MS to the expected value and solving algebraically, that is, $\sigma_b^2 = (\text{Block MS?} - \sigma_\epsilon^2)\,/\,t?$.

These data might be entered in R as

```
> Table5.7 <- data.frame(
          Block = as.factor(c(1,1,1,1,1,1,2,2,2,2,2,2,3,3,3,3,3,3,4,4,4,4,4,4)),
          Treatment = as.factor(c(4,5,6,3,1,2,6,4,2,5,3,1,2,3,1,6,5,4,3,6,2,5,4,1)),
          Observation = c(3.43,5.25,6.47,2.8,NA,2.66,8.43,6.09,6.41,5.69,7.04,NA,6.07,
                  6.19,4.93,5.95,4.99,3.26,5.22,7.35,4.48,6.34,6.71,3.13))
  #some packages choke on missing values
  Table5.7 <- subset(Table5.7, !is.na(Table5.7$Observation))
```

2 Series of Similar Experiments

Exercise 16.8.1 from ***Principles and Procedures of Statistics, A Biometrical Approach***, R. G. D. Steel and J. H. Torrie, *McGraw-Hill*. (1960).

> *Twelve strains of soybeans were compared in a randomized complete block experiment with three blocks at each of three locations in North Carolina. Yields given here are in grams per plot.*

```
> Ex16.8.1 <- data.frame(
  Location=as.factor(c(rep("Clayton", 36), rep("Clinton", 36), rep("Plymouth", 36))),
  Strain=as.factor(rep(c(rep("Tracy", 3), rep("Centennial", 3), rep("N72-137", 3),
                         rep("N72-3058", 3), rep("N72-3148", 3), rep("R73-81", 3),
                         rep("D74-7741", 3), rep("N73-693", 3), rep("N73-877", 3),
                         rep("N73-882", 3), rep("N73-1102", 3), rep("R75-12", 3)),3)),
  Block=as.factor(rep(c(1, 2, 3),36)),
  Yield=c(1178, 1089, 960, 1187, 1180, 1235, 1451, 1177, 1723, 1318, 1012, 990,
          1345, 1335, 1303, 1175, 1064, 1158, 1111, 1111, 1099, 1388, 1214, 1222,
          1254, 1249, 1135, 1179, 1247, 1096, 1345, 1265, 1178, 1136, 1161, 1004,
          1583, 1841, 1464, 1713, 1684, 1378, 1369, 1608, 1647, 1547, 1647, 1603,
          1622, 1801, 1929, 1800, 1787, 1520, 1820, 1521, 1851, 1464, 1607, 1642,
          1775, 1513, 1570, 1673, 1507, 1390, 1894, 1547, 1751, 1422, 1393, 1342,
          1307, 1365, 1542, 1425, 1475, 1276, 1289, 1671, 1420, 1250, 1202, 1407,
          1546, 1489, 1724, 1344, 1197, 1319, 1280, 1260, 1605, 1583, 1503, 1303,
          1656, 1371, 1107, 1398, 1497, 1583, 1586, 1423,  1524, 911, 1202, 1012
   ))
```

3 Series of Similar Experiments

Exercise 16.8.1 from ***Principles and Procedures of Statistics, A Biometrical Approach***, R. G. D. Steel and J. H. Torrie, *McGraw-Hill*. (1960).

> *Twelve strains of soybeans were compared in a randomized complete block experiment with three blocks at each of three locations in North Carolina. Yields given here are in grams per plot.*

```
> Ex16.8.1 <- data.frame(
  Location=as.factor(c(rep("Clayton", 36), rep("Clinton", 36), rep("Plymouth", 36))),
  Strain=as.factor(rep(c(rep("Tracy", 3), rep("Centennial", 3), rep("N72-137", 3),
                         rep("N72-3058", 3), rep("N72-3148", 3), rep("R73-81", 3),
                         rep("D74-7741", 3), rep("N73-693", 3), rep("N73-877", 3),
                         rep("N73-882", 3), rep("N73-1102", 3), rep("R75-12", 3)),3)),
  Block=as.factor(rep(c(1, 2, 3),36)),
  Yield=c(1178, 1089, 960, 1187, 1180, 1235, 1451, 1177, 1723, 1318, 1012, 990,
          1345, 1335, 1303, 1175, 1064, 1158, 1111, 1111, 1099, 1388, 1214, 1222,
          1254, 1249, 1135, 1179, 1247, 1096, 1345, 1265, 1178, 1136, 1161, 1004,
          1583, 1841, 1464, 1713, 1684, 1378, 1369, 1608, 1647, 1547, 1647, 1603,
          1622, 1801, 1929, 1800, 1787, 1520, 1820, 1521, 1851, 1464, 1607, 1642,
          1775, 1513, 1570, 1673, 1507, 1390, 1894, 1547, 1751, 1422, 1393, 1342,
          1307, 1365, 1542, 1425, 1475, 1276, 1289, 1671, 1420, 1250, 1202, 1407,
          1546, 1489, 1724, 1344, 1197, 1319, 1280, 1260, 1605, 1583, 1503, 1303,
          1656, 1371, 1107, 1398, 1497, 1583, 1586, 1423,  1524, 911, 1202, 1012
   ))
```

When we examine the analysis of variance table, we see that block (in location) variance is smaller than residual error.

```
> summary(aov(Yield ~ Location:Block + Location*Strain,data=Ex16.8.1))
```

Assuming random locations, the expected mean squares for Location:Block is $\sigma^2_\epsilon + t\sigma^2_{b(l)}$, implying $\sigma^2_{b(l)} = (11543 - 19708)/12 = -680$. Variances can't be negative, so we can use this data set to compare how analysis methods solve for non-estimable variances.

When we examine the analysis of variance table, we see that block (in location) variance is smaller than residual error.

```
> summary(aov(Yield ~ Location:Block + Location*Strain,data=Ex16.8.1))
```

```
summary(aov(Yield ~ Location:Block + Location*Strain,data=Ex16.8.1))
                Df  Sum Sq Mean Sq F value  Pr(>F)
Location         2 3113626 1556813  78.994 < 2e-16 ***
Strain          11  925090   84099   4.267 8.7e-05 ***
Location:Block   6   69256   11543   0.586   0.740
Location:Strain 22  532900   24223   1.229   0.256
Residuals       66 1300723   19708
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Assuming random locations, the expected mean squares for Location:Block is $\sigma^2_\epsilon + t\sigma^2_{b(l)}$, implying $\sigma^2_{b(l)} = (11543 - 19708)/12 = -680$. Variances can't be negative, so we can use this data set to compare how analysis methods solve for non-estimable variances.

4 Multi-environment Breeding Trial

Sample RCBD Data from

META-R (Multi Environment Trail Analysis with R for Windows) Version 5.0, G. Alvarado, M. López, M. Vargas, Á. Pacheco, F. Rodríguez, J. Burgueño, J. Crossa, International Maize and Wheat Improvement Center, `http://hdl.handle.net/11529/10201`, (2015).

and originally described in

META: A suite of SAS programs to analyze multienvironment breeding trials, M. Vargas, E. Combs, G. Alvarado, G. Atlin, K. Mathews, and J. Crossa, *Agronomy Journal*. 105(1):11 (2013).

This is a sufficiently large data set to test computational performance of LMM packages.

```
> setwd("~/Work/git/ASA_CSSA_SSSA/literate")
  rcbd.dat <- read.csv("sample RCBD data.csv", header=TRUE)
  head(rcbd.dat)
```

Note that for the remainder of this presentation, we will be working the the variable names used in the original data.

This is a sufficiently large data set to test computational performance of LMM packages.

```
> setwd("~/Work/git/ASA_CSSA_SSSA/literate")
  rcbd.dat <- read.csv("sample RCBD data.csv", header=TRUE)
  head(rcbd.dat)
```

```
  Site Country     Loca Plot Repe BLK Entry  YLD AD SD  PH  EH rEPH rEPP nP
1    3  Mexico Cotaxtla    1    1   1    21 7.00 54 55 280 150 0.54  0.9 58
2    3  Mexico Cotaxtla    2    1   1    22 8.39 51 52 270 140 0.52    1 58
3    3  Mexico Cotaxtla    3    1   1    32 6.85 52 53 265 140 0.53  0.9 54
4    3  Mexico Cotaxtla    4    1   1    11 8.09 53 54 275 140 0.51    1 53
5    3  Mexico Cotaxtla    5    1   2     4 6.86 51 52 260 125 0.48  0.9 58
6    3  Mexico Cotaxtla    6    1   2    29 6.45 51 52 275 130 0.47  0.9 57
```

Note that for the remainder of this presentation, we will be working the the variable names used in the original data.

We illustrate a typical workflow using `lm` applied to the first example data set. The first step is to fit a model.

```
> Table5.7.lm <- lm(Observation ~ Block + Treatment, data = Table5.7)
```

We illustrate a typical workflow using `lm` applied to the first example data set. The first step is to fit a model.

```
> Table5.7.lm <- lm(Observation ~ Block + Treatment, data = Table5.7)
```

The most commonly used data structure in R is the list, and many functions return complex data objects as lists. However, R objects can have a class attribute as well.

```
> class(Table5.7.lm)
```

R also provides for generic functions; functions that have a simple interface that can be associated with multiple different implementations, each for a different R class.

```
> plot
```

```
> methods("plot")
```

The most commonly used data structure in R is the list, and many functions return complex data objects as lists. However, R objects can have a class attribute as well.

```
> class(Table5.7.lm)
```

```
[1] "lm"
```

R also provides for generic functions; functions that have a simple interface that can be associated with multiple different implementations, each for a different R class.

```
> plot
```

```
> methods("plot")
```

The most commonly used data structure in R is the list, and many functions return complex data objects as lists. However, R objects can have a class attribute as well.

```
> class(Table5.7.lm)
```

```
[1] "lm"
```

R also provides for generic functions; functions that have a simple interface that can be associated with multiple different implementations, each for a different R class.

```
> plot
```

```
plot
function (x, y, ...)
UseMethod("plot")
<bytecode: 0x7fc1bb9da580>
<environment: namespace:graphics>
```

```
> methods("plot")
```

The most commonly used data structure in R is the list, and many functions return complex data objects as lists. However, R objects can have a class attribute as well.

```
> class(Table5.7.lm)
```

```
[1] "lm"
```

R also provides for generic functions; functions that have a simple interface that can be associated with multiple different implementations, each for a different R class.

```
> plot
```

```
plot
function (x, y, ...)
UseMethod("plot")
<bytecode: 0x7fc1bb9da580>
<environment: namespace:graphics>
```

```
> methods("plot")
```

```
methods("plot")
 [1] plot.HoltWinters*   plot.TukeyHSD*      plot.acf*
 [4] plot.data.frame*    plot.decomposed.ts* plot.default
 [7] plot.dendrogram*    plot.density*       plot.ecdf
[10] plot.factor*        plot.formula*       plot.function
[13] plot.hclust*        plot.histogram*     plot.isoreg*
[16] plot.lm*            plot.medpolish*     plot.mlm*
[19] plot.ppr*           plot.prcomp*        plot.princomp*
[22] plot.profile.nls*   plot.raster*        plot.spec*
[25] plot.stepfun        plot.stl*           plot.table*
[28] plot.ts             plot.tskernel*
see '?methods' for accessing help and source code
```

If we structure our workflow using generic functions, we can easily swap complex analysis results like ...

```
> str(Table5.7.lm)
```

If we structure our workflow using generic functions, we can easily swap complex analysis results like ...

```
> str(Table5.7.lm)
```

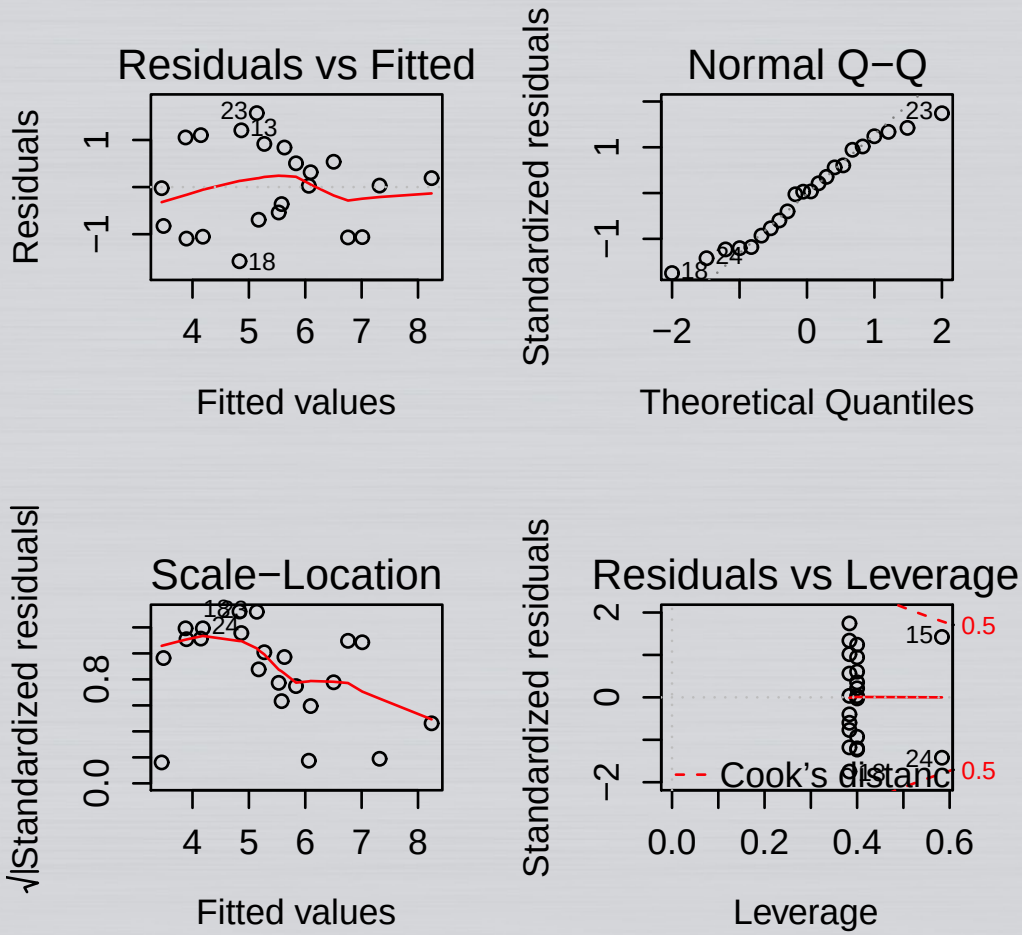
```
str(Table5.7.lm)
List of 13
 $ coefficients : Named num [1:9] 2.496 2.61 1.381 1.687 0.989 ...
  ..- attr(*, "names")= chr [1:9] "(Intercept)" "Block2" "Block3" "Block4" ...
 $ residuals    : Named num [1:22] -0.023 1.102 0.839 -1.093 -0.826 ...
  ..- attr(*, "names")= chr [1:22] "1" "2" "3" "4" ...
 $ effects      : Named num [1:22] -25.347 3.378 0.664 2.339 1.109 ...
  ..- attr(*, "names")= chr [1:22] "(Intercept)" "Block2" "Block3" "Block4" ...
 $ rank         : int 9
 $ fitted.values: Named num [1:22] 3.45 4.15 5.63 3.89 3.49 ...
  ..- attr(*, "names")= chr [1:22] "1" "2" "3" "4" ...
 $ assign       : int [1:9] 0 1 1 1 2 2 2 2 2
 $ qr           :List of 5
  ..$ qr   : num [1:22, 1:9] -4.69 0.213 0.213 0.213 0.213 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:22] "1" "2" "3" "4" ...
  .. .. ..$ : chr [1:9] "(Intercept)" "Block2" "Block3" "Block4" ...
  .. ..- attr(*, "assign")= int [1:9] 0 1 1 1 2 2 2 2 2
  .. ..- attr(*, "contrasts")=List of 2
  .. .. ..$ Block    : chr "contr.treatment"
  .. .. ..$ Treatment: chr "contr.treatment"
  ..$ qraux: num [1:9] 1.21 1.1 1.13 1.22 1.49 ...
  ..$ pivot: int [1:9] 1 2 3 4 5 6 7 8 9
  ..$ tol  : num 1e-07
  ..$ rank : int 9
  ..- attr(*, "class")= chr "qr"
 $ df.residual  : int 13
 $ contrasts    :List of 2
  ..$ Block    : chr "contr.treatment"
```

... into routine steps, like plotting diagnostics.

```
> par(mfrow = c(2, 2))
  plot(Table5.7.lm)
  par(mfrow = c(1,1));v()
```

... into routine steps, like plotting diagnostics.

```
> par(mfrow = c(2, 2))
  plot(Table5.7.lm)
  par(mfrow = c(1,1));v()
```


Residuals vs Fitted
Residuals
Fitted values
Normal Q−Q
Standardized residuals
Theoretical Quantiles
Scale−Location
√|Standardized residuals|
Fitted values
Residuals vs Leverage
Standardized residuals
Leverage
Cook's distanc
0.5
0.5

Other generic functions that will fit into our simple workflow include `update`, which can be used to build new models from existing results ...

```
> Table5.7.red.lm <- update(Table5.7.lm, . ~ . - Block)
```

Other generic functions that will fit into our simple workflow include `update`, which can be used to build new models from existing results ...

```
> Table5.7.red.lm <- update(Table5.7.lm, . ~ . - Block)
```

... print a "pretty" report ...

```
> summary(Table5.7.lm)
```

... print a "pretty" report ...

```
> summary(Table5.7.lm)
```

```
summary(Table5.7.lm)

Call:
lm(formula = Observation ~ Block + Treatment, data = Table5.7)

Residuals:
     Min       1Q   Median       3Q      Max
-1.57367 -0.79233  0.02958  0.76387  1.56967

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.4960     1.0255   2.434  0.03011 *
Block2        2.6100     0.7252   3.599  0.00324 **
Block3        1.3807     0.7099   1.945  0.07374 .
Block4        1.6873     0.7099   2.377  0.03350 *
Treatment2    0.9895     1.0255   0.965  0.35222
Treatment3    1.3970     1.0255   1.362  0.19626
Treatment4    0.9570     1.0255   0.933  0.36774
Treatment5    1.6520     1.0255   1.611  0.13121
Treatment6    3.1345     1.0255   3.056  0.00918 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.147 on 13 degrees of freedom
Multiple R-squared:  0.6684,    Adjusted R-squared:  0.4644
F-statistic: 3.276 on 8 and 13 DF,  p-value: 0.0282
```

.. and compute an analysis of variance, which can be used to test terms in a model ...

```
> anova(Table5.7.lm)
```

... or to compare different models.

```
> anova(Table5.7.red.lm, Table5.7.lm)
```

.. and compute an analysis of variance, which can be used to test terms in a model ...

```
> anova(Table5.7.lm)
```

```
Analysis of Variance Table

Response: Observation
           Df Sum Sq Mean Sq F value  Pr(>F)
Block       3 17.322  5.7740  4.3921 0.02422 *
Treatment   5 17.132  3.4264  2.6063 0.07619 .
Residuals  13 17.090  1.3146
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

... or to compare different models.

```
> anova(Table5.7.red.lm, Table5.7.lm)
```

.. and compute an analysis of variance, which can be used to test terms in a model ...

```
> anova(Table5.7.lm)
```

```
Analysis of Variance Table

Response: Observation
           Df Sum Sq Mean Sq F value  Pr(>F)
Block       3 17.322  5.7740  4.3921 0.02422 *
Treatment   5 17.132  3.4264  2.6063 0.07619 .
Residuals  13 17.090  1.3146
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

... or to compare different models.

```
> anova(Table5.7.red.lm, Table5.7.lm)
```

```
Analysis of Variance Table

Model 1: Observation ~ Treatment
Model 2: Observation ~ Block + Treatment
  Res.Df    RSS Df Sum of Sq      F  Pr(>F)
1     16 34.665
2     13 17.090  3    17.575 4.4561 0.02316 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Other useful packages are build upon generic functions. The `multcomp` package provides an interface for multiple tests and allows for user specified contrasts. This package assumes both `coef` and `vcov` are available for a given model.

```
> library(multcomp)
  summary(glht(Table5.7.lm,linfct=mcp(Treatment="Dunnett")))
```

Simultaneous Inference in General Parametric Models, T. Hothorn, F. Bretz and P. Westfall, *Biometrical Journal*. 50(3):346–363 (2008).

Other useful packages are build upon generic functions. The `multcomp` package provides an interface for multiple tests and allows for user specified contrasts. This package assumes both `coef` and `vcov` are available for a given model.

```
> library(multcomp)
  summary(glht(Table5.7.lm,linfct=mcp(Treatment="Dunnett")))

         Simultaneous Tests for General Linear Hypotheses

Multiple Comparisons of Means: Dunnett Contrasts

Fit: lm(formula = Observation ~ Block + Treatment, data = Table5.7)

Linear Hypotheses:
           Estimate Std. Error t value Pr(>|t|)
2 - 1 == 0   0.9895     1.0255   0.965   0.7220
3 - 1 == 0   1.3970     1.0255   1.362   0.4639
4 - 1 == 0   0.9570     1.0255   0.933   0.7432
5 - 1 == 0   1.6520     1.0255   1.611   0.3323
6 - 1 == 0   3.1345     1.0255   3.056   0.0299 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
(Adjusted p values reported -- single-step method)
```

Simultaneous Inference in General Parametric Models, T. Hothorn, F. Bretz and P. Westfall, *Biometrical Journal*. 50(3):346–363 (2008).

`glht` performs a set of multiple comparisons, and `cld` is used to produce a compact letter display

```
> cld(glht(Table5.7.lm,linfct=mcp(Treatment="Tukey")),decreasing=TRUE)
```

Like `lm`, `glht` returns a complex object, and explicitly invoked functions like `summary` or implicitly invoked `print` functions make these objects legible.

```
> str(glht(Table5.7.lm,linfct=mcp(Treatment="Tukey")))
```

`glht` performs a set of multiple comparisons, and `cld` is used to produce a compact letter display

```
> cld(glht(Table5.7.lm,linfct=mcp(Treatment="Tukey")),decreasing=TRUE)
```

```
  1   2   3   4   5   6
"a" "a" "a" "a" "a" "a"
```

Like `lm`, `glht` returns a complex object, and explicitly invoked functions like `summary` or implicitly invoked `print` functions make these objects legible.

```
> str(glht(Table5.7.lm,linfct=mcp(Treatment="Tukey")))
```

`glht` performs a set of multiple comparisons, and `cld` is used to produce a compact letter display

```
> cld(glht(Table5.7.lm,linfct=mcp(Treatment="Tukey")),decreasing=TRUE)
```

```
  1   2   3   4   5   6
"a" "a" "a" "a" "a" "a"
```

Like `lm`, `glht` returns a complex object, and explicitly invoked functions like `summary` or implicitly invoked `print` functions make these objects legible.

```
> str(glht(Table5.7.lm,linfct=mcp(Treatment="Tukey")))
```

```
str(glht(Table5.7.lm,linfct=mcp(Treatment="Tukey")))
List of 9
 $ model      :List of 13
  ..$ coefficients : Named num [1:9] 2.496 2.61 1.381 1.687 0.989 ...
  .. ..- attr(*, "names")= chr [1:9] "(Intercept)" "Block2" "Block3" "Block4" ...
  ..$ residuals    : Named num [1:22] -0.023 1.102 0.839 -1.093 -0.826 ...
  .. ..- attr(*, "names")= chr [1:22] "1" "2" "3" "4" ...
  ..$ effects      : Named num [1:22] -25.347 3.378 0.664 2.339 1.109 ...
  .. ..- attr(*, "names")= chr [1:22] "(Intercept)" "Block2" "Block3" "Block4" ...
  ..$ rank         : int 9
  ..$ fitted.values: Named num [1:22] 3.45 4.15 5.63 3.89 3.49 ...
  .. ..- attr(*, "names")= chr [1:22] "1" "2" "3" "4" ...
  ..$ assign       : int [1:9] 0 1 1 1 2 2 2 2 2
  ..$ qr           :List of 5
  .. ..$ qr   : num [1:22, 1:9] -4.69 0.213 0.213 0.213 0.213 ...
  .. .. ..- attr(*, "dimnames")=List of 2
  .. .. .. ..$ : chr [1:22] "1" "2" "3" "4" ...
  .. .. .. ..$ : chr [1:9] "(Intercept)" "Block2" "Block3" "Block4" ...
  .. .. ..- attr(*, "assign")= int [1:9] 0 1 1 1 2 2 2 2 2
  .. .. ..- attr(*, "contrasts")=List of 2
  .. .. .. ..$ Block    : chr "contr.treatment"
  .. .. .. ..$ Treatment: chr "contr.treatment"
  .. ..$ qraux: num [1:9] 1.21 1.1 1.13 1.22 1.49 ...
  .. ..$ pivot: int [1:9] 1 2 3 4 5 6 7 8 9
  .. ..$ tol  : num 1e-07
  .. ..$ rank : int 9
  .. ..- attr(*, "class")= chr "qr"
  ..$ df.residual  : int 13
  ..$ contrasts    :List of 2
```

The `lsmeans` package provides an interface producing means tables and test for main effects.

```
> library(lsmeans)
  lsmeans(Table5.7.lm, cld ~ Treatment)
```

Least-Squares Means: The R Package lsmeans, R. Lenth, *Journal of Statistical Software*. 69(1):1-33 (2016).

The `lsmeans` package provides an interface producing means tables and test for main effects.

```
> library(lsmeans)
  lsmeans(Table5.7.lm, cld ~ Treatment)
```

```
Loading required package: estimability
> lsmeans(Table5.7.lm, cld ~ Treatment)
 Treatment lsmean        SE df lower.CL upper.CL .group
 1         3.9155 0.8503251 13 2.078484 5.752516  1
 4         4.8725 0.5732890 13 3.633984 6.111016  1
 2         4.9050 0.5732890 13 3.666484 6.143516  1
 3         5.3125 0.5732890 13 4.073984 6.551016  1
 5         5.5675 0.5732890 13 4.328984 6.806016  1
 6         7.0500 0.5732890 13 5.811484 8.288516  1

Results are averaged over the levels of: Block
Confidence level used: 0.95
P value adjustment: tukey method for comparing a family of 6 estimates
significance level used: alpha = 0.05
```

Least-Squares Means: The R Package lsmeans, R. Lenth, *Journal of Statistical Software*. 69(1):1-33 (2016).

The ggplot2 package is a large and extensive library for plotting high-quality graphics. We can illustrate the use of several of the previously described functions by writing a set of functions that produce a bar plot from a generic statistical model. The first step is to write a function the generate a means table:

```
> library(ggplot2)
  make.plot.table <- function(model, form="cld ~ Treatment", effect="Treatment") {
     model.tbl <- lsmeans(model,formula(form),model="kenward-roger")
     mcp.list <- list(effect="Tukey")
     names(mcp.list) <- effect
     attr(mcp.list, "interaction_average") <- TRUE
     attr(mcp.list, "covariate_average") <- TRUE
     class(mcp.list) <- "mcp"
     letters <- cld(glht(model,linfct=mcp.list,interaction_average = TRUE,
                                                covariate_average = TRUE),
                            decreasing=TRUE)$mcletters$Letters
     model.tbl$letters <- letters[model.tbl[,effect]]
     names(model.tbl) <- c("Treatment","Mean","Error","df","Lower","Upper","Group","Let-
  ters")
     return(model.tbl)
  }
```

ggplot2: Elegant Graphics for Data Analysis, H. Wickham, *Springer-Verlag New York*. (2009).

The ggplot2 package is a large and extensive library for plotting high-quality graphics. We can illustrate the use of several of the previously described functions by writing a set of functions that produce a bar plot from a generic statistical model. The first step is to write a function the generate a means table:

```
> library(ggplot2)
  make.plot.table <- function(model, form="cld ~ Treatment", effect="Treatment") {
     model.tbl <- lsmeans(model,formula(form),model="kenward-roger")
     mcp.list <- list(effect="Tukey")
     names(mcp.list) <- effect
     attr(mcp.list, "interaction_average") <- TRUE
     attr(mcp.list, "covariate_average") <- TRUE
     class(mcp.list) <- "mcp"
     letters <- cld(glht(model,linfct=mcp.list,interaction_average = TRUE,
                                                covariate_average = TRUE),
                            decreasing=TRUE)$mcletters$Letters
     model.tbl$letters <- letters[model.tbl[,effect]]
     names(model.tbl) <- c("Treatment","Mean","Error","df","Lower","Upper","Group","Let-
  ters")
     return(model.tbl)
  }
```

ggplot2: Elegant Graphics for Data Analysis, H. Wickham, *Springer-Verlag New York*. (2009).

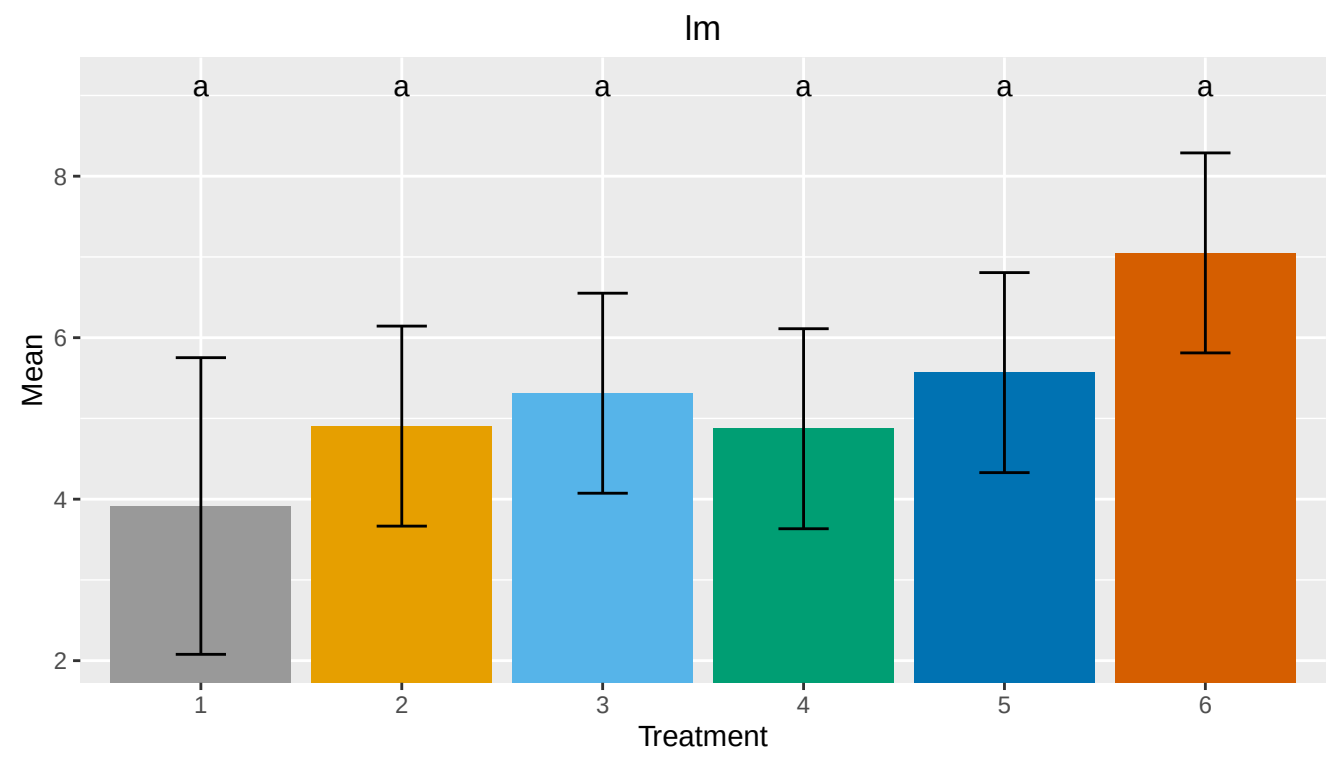
The second function takes an appropriately formated data table and generates a bar plot. This simplifies plotting results tables that cannot be easily generated using lsmeans.

```
> plot.lsmeans.tbl <- function(model.tbl,title="Means Plot") {
    cbPalette <- c("#999999", "#E69F00", "#56B4E9", "#009E73", "#0072B2", "#D55E00",
                   "#F0E442", "#CC79A7", "#734f80", "#2b5a74", "#004f39", "#787221",
                   "#003959", "#6aaf00", "#663cd3", "#000000")
    dodge <- position_dodge(width = 0.9)
    upper.lim <- max(model.tbl$Upper)
    upper.lim <-  upper.lim + 0.1*upper.lim
    limits <- aes(ymax = model.tbl$Upper, ymin = model.tbl$Lower)
    return(ggplot(data = model.tbl, aes(x = Treatment, y = Mean, fill = Treatment)) +
      geom_bar(stat = "identity", position = dodge) +
      coord_cartesian(ylim = c(min(model.tbl$Lower),upper.lim)) +
      geom_errorbar(limits, position = dodge, width = 0.25) +
      theme(legend.position = "none") + ggtitle(title) +
      scale_fill_manual(values=cbPalette) +
      geom_text(aes(x=model.tbl$Treatment,y=upper.lim,label=Letters))
    )
  }
```

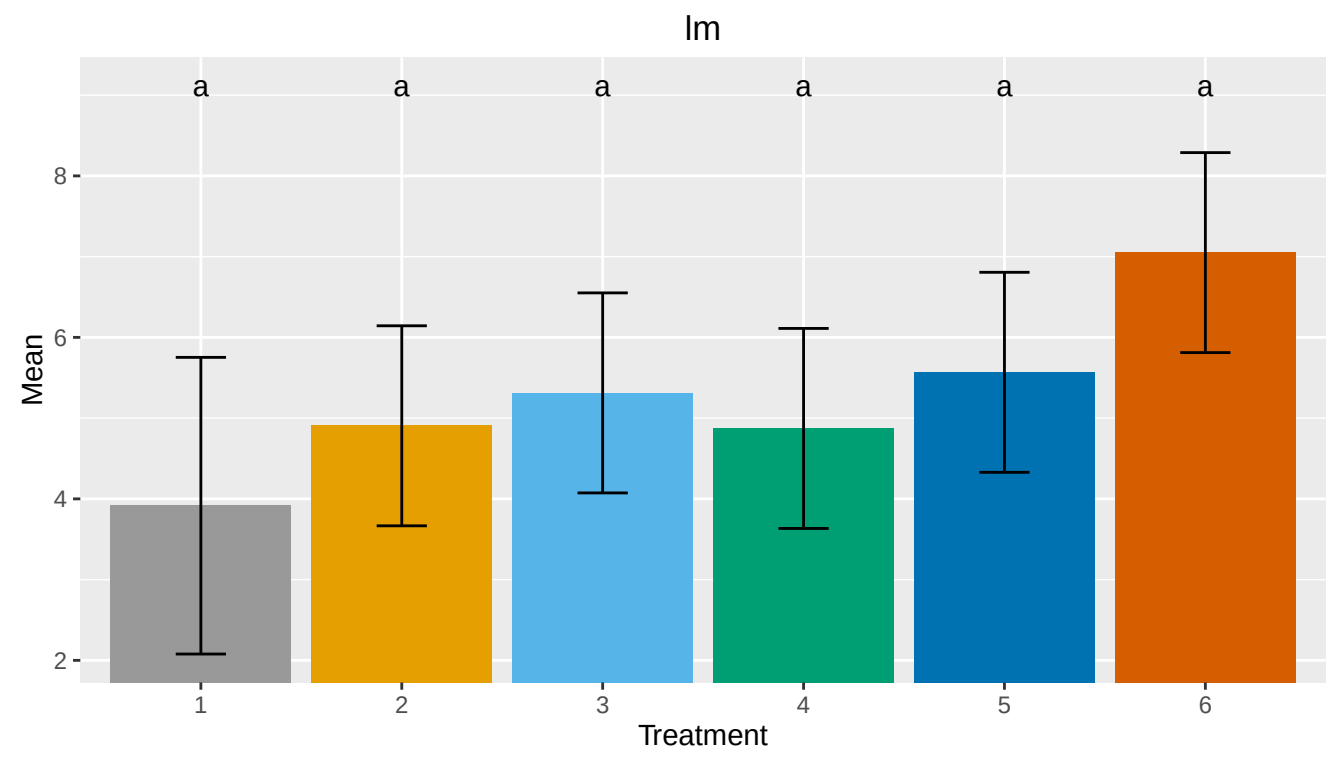
The second function takes an appropriately formated data table and generates a bar plot. This simplifies plotting results tables that cannot be easily generated using lsmeans.

```
> plot.lsmeans.tbl <- function(model.tbl,title="Means Plot") {
    cbPalette <- c("#999999", "#E69F00", "#56B4E9", "#009E73", "#0072B2", "#D55E00",
                   "#F0E442", "#CC79A7", "#734f80", "#2b5a74", "#004f39", "#787221",
                   "#003959", "#6aaf00", "#663cd3", "#000000")
    dodge <- position_dodge(width = 0.9)
    upper.lim <- max(model.tbl$Upper)
    upper.lim <-  upper.lim + 0.1*upper.lim
    limits <- aes(ymax = model.tbl$Upper, ymin = model.tbl$Lower)
    return(ggplot(data = model.tbl, aes(x = Treatment, y = Mean, fill = Treatment)) +
      geom_bar(stat = "identity", position = dodge) +
      coord_cartesian(ylim = c(min(model.tbl$Lower),upper.lim)) +
      geom_errorbar(limits, position = dodge, width = 0.25) +
      theme(legend.position = "none") + ggtitle(title) +
      scale_fill_manual(values=cbPalette) +
      geom_text(aes(x=model.tbl$Treatment,y=upper.lim,label=Letters))
    )
  }
```

```
> plot.lsmeans.tbl(make.plot.table(Table5.7.lm))
```



```
> plot.lsmeans.tbl(make.plot.table(Table5.7.lm))
```



Now that we've established a workflow, we can consider how to integrate mixed models. First we define some models mathematically.

Model 1

We can write a linear mixed model for a randomized complete block design as

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{Z}_b\mathbf{u}_b + \mathbf{1} \otimes \mathbf{e}$$

where $\boldsymbol{\beta} = [\tau_1, \tau_2, ..., \tau_t]$ are the fixed treatment effects and $\boldsymbol{X}$ is the design matrix mapping treatments to observations $\boldsymbol{y}$. Random block effects are denoted $\mathbf{u}_b$ with a corresponding design matrix $\boldsymbol{Z}_b$. Further, we require

$$\begin{aligned} \mathbf{u}_b &\sim \mathcal{N}(0, \sigma_b^2) \\ \mathbf{e} &\sim \mathcal{N}(0, \sigma_\epsilon^2) \end{aligned}$$

In most cases, these requirements are included in the computations to solve LMM, as opposed to computed after the fact, from the expected mean squares from linear models.

Model 2

For a series of RCB experiments at different locations, we extend Model 1 by

$$\boldsymbol{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{Z}_b\mathbf{u}_b + \boldsymbol{Z}_l\mathbf{u}_l + \boldsymbol{Z}_{\tau\times l}\mathbf{u}_{\tau\times l} + \mathbf{1}\otimes\mathbf{e}$$

where $\boldsymbol{Z}_l\mathbf{u}_l$ represent the random location effects and $\boldsymbol{Z}_{\tau\times l}\mathbf{u}_{\tau\times l}$ represents treatment by location interaction. We add additional constraints

$$\begin{aligned}\mathbf{u}_l &\sim \mathcal{N}(0,\sigma_l^2)\\ \mathbf{u}_{\tau\times l} &\sim \mathcal{N}(0,\sigma_{\tau\times l}^2)\end{aligned}$$

and note that $\boldsymbol{Z}_b\mathbf{u}_b$ includes blocks within trials. If we were to model trials as fixed (or cannot compute crossed random effects), this would be the only random term in the model.

Model 3

From ***A Bayesian Approach for Assessing the Stability of Genotypes***, J. M. Cotes, J. Crossa, A. Sanches, and P. L. Cornelius, *Crop Science*. 46:2654–2665 (2006)

Starting with Model 2, we allow each treatment j to vary independently across locations, and allow each location i to have a different residual error. Then

$$\boldsymbol{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{Z}_b\mathbf{u}_b + \boldsymbol{Z}_l\mathbf{u}_l + \sum_{j=1}^{t} \boldsymbol{Z}_{\tau_j \times l}\mathbf{u}_{\tau_j \times l} + \mathbf{1}_{n_i} \otimes \mathbf{e}_i$$

and

$$\begin{aligned} \mathbf{u}_{\tau_j \times l} &\sim \mathcal{N}(0, \sigma^2_{\tau_j \times l}) \\ \mathbf{e}_i &\sim \mathcal{N}(0, \sigma^2_{e_i}) \end{aligned}$$

This greatly increased computational complexity, but relaxes requirements for homogeneity of variances in AOV.

Function	Package	Version, Date	Maintainer
lme	nlme	3.1-128, 2016-05-04	R-core
glmmPQL	MASS	7.3-45, 2015-11-10	Brian Ripley
lmer	lme4	1.1-12, 2016-04-16	Ben Bolker
blmer	blme	1.0-4, 2015-06-13	Vincent Dorie
glmmadmb	glmmADMB	0.8.3.3, 2016-01-19	Ben Bolker
glmmLasso	glmmLasso	1.4.4, 2016-05-28	Andreas Groll
lmm	minque	1.1, 2014-09-06	Jixiang Wu
MCMCglmm	MCMCglmm	2.23, 2016-10-10	Jarrod Hadfield
inla	INLA	0.0-1468872408, 2016-07-18	Havard Rue
brm	brms	1.1.0, 2016-10-11	Paul-Christian Buerkner

Table 2. Summary of LMM packages compared in this presentation. Primary maintainer listed for brevity, complete attribution can be obtained using `packageDescription("lme")`, etc.

	Mixed Model Formula
lm	`obs ~ Block + Treatment`
lme	`obs ~ Treatment, random = ~ 1 | Block`
glmmPQL	`obs ~ Treatment, random = ~ 1 | Block, family=gaussian`
lmer	`obs ~ Treatment + (1 | Block)`
blmer	`obs ~ Treatment + (1 | Block), fixef.prior = normal`
glmmadmb	`obs ~ Treatment, random = ~ 1 | Block`
glmmLasso	`obs ~ Treatment, rnd = list(Block = ~1)`
lmm	`obs ~ Treatment | Block`
MCMCglmm	`obs ~ Treatment, random = ~ Block`
inla	`obs ~ Treatment + f(Block, model="iid")`
brm	`obs ~ Treatment + (1 | Block)`

Table 3. Syntax for Model 1.

A mixed effects model with fixed treatments and random blocks, for the first example data set. Packages tend to be consistent with regard to specifying fixed effects, but many use idiosyncratic syntax for random effects.

Which of the workflow steps described previously be applied to objects obtained by using the preceding functions?

	plot	summary	anova(.)	anova(.,.)	update
lm	yes	yes	yes	yes	yes
lme	yes	yes	yes	yes	yes
glmmPQL	yes	yes	no	no	yes
lmer	yes	yes	yes	yes	yes
blmer	yes	yes	yes	yes	yes
glmmadmb	no	yes	no	yes	yes
glmmLasso	no	yes	no	no	no
lmm	no	yes	no	no	no
MCMCglmm	yes	yes	no	no	no
inla	yes	yes	no	no	no
brm	yes	yes	no	no	yes

Table 4. Generic functions provided for objects of different classes of LMM analysis objects.

Seems simple, but there are caveats and exceptions.

	plot	summary	anova(.)	anova(.,.)	update
lm	yes	yes	yes	yes	yes
lme	yes **a**	yes	yes	yes	yes
glmmPQL	yes **a**	yes	no	no	yes
lmer	yes **b**	yes	yes	yes	yes
blmer	yes **b**	yes	yes	yes	yes
glmmadmb	no	yes	no	yes	yes
glmmLasso	no	yes	no	no	no
lmm	no	yes	no	no	no
MCMCglmm	yes	yes	no	no	no
inla	yes	yes	no	no	no
brm	yes	yes	no	no	yes

Table 5. Generic functions provided for objects of different classes of LMM analysis objects.

a. Residual plot only, `qqnorm` is avaible for QQ plots

b. Residual plot only, `qqnorm` is not available

`qqnorm` can also be called for any object that produces an appropriate vector, e.g. `qqnorm(resid(Table5.7.lmer))`.

	plot	summary	anova(.)	anova(.,.)	update
lm	yes	yes	yes	yes	yes
lme	yes a	yes	yes	yes **c**	yes
glmmPQL	yes a	yes	no	no	yes
lmer	yes b	yes	yes **e**	yes **d**	yes
blmer	yes b	yes	yes **e**	yes **d**	yes
glmmadmb	no	yes	no	yes	yes
glmmLasso	no	yes	no	no	no
lmm	no	yes	no	no	no
MCMCglmm	yes	yes	no	no	no
inla	yes	yes	no	no	no
brm	yes	yes	no	no	yes

Table 6. Generic functions provided for objects of different classes of LMM analysis objects.

c. `fitted objects with different fixed effects. REML comparisons are not meaningful`

d. `refitting model(s) with ML (instead of REML)`

e. no p-values for F tests

There are two different philosophies in practice with AOV on mixed models between `lme` and `lmer`.

```
> library(nlme);library(lme4)
  Table5.7.lme <- lme(Observation ~ Treatment, random=~ 1 | Block, data = Table5.7)
  anova(Table5.7.lme)
> Table5.7.lmer <- lmer(Observation ~ Treatment + ( 1 | Block), data = Table5.7)
  anova(Table5.7.lmer)
> anova(lm(Observation ~ Block + Treatment,data=Table5.7))
```

```
> library(nlme);library(lme4)
  Table5.7.lme <- lme(Observation ~ Treatment, random=~ 1 | Block, data = Table5.7)
  anova(Table5.7.lme)
```

```
            numDF denDF   F-value p-value
(Intercept)     1    13 103.23928  <.0001
Treatment       5    13   2.58997  0.0775
```

```
> Table5.7.lmer <- lmer(Observation ~ Treatment + ( 1 | Block), data = Table5.7)
  anova(Table5.7.lmer)
```

```
> anova(lm(Observation ~ Block + Treatment,data=Table5.7))
```

```
> library(nlme);library(lme4)
  Table5.7.lme <- lme(Observation ~ Treatment, random=~ 1 | Block, data = Table5.7)
  anova(Table5.7.lme)
```

```
            numDF denDF   F-value p-value
(Intercept)     1    13 103.23928  <.0001
Treatment       5    13   2.58997  0.0775
```

```
> Table5.7.lmer <- lmer(Observation ~ Treatment + ( 1 | Block), data = Table5.7)
  anova(Table5.7.lmer)
```

```
Analysis of Variance Table
          Df Sum Sq Mean Sq F value
Treatment  5 17.074  3.4148    2.59
```

```
> anova(lm(Observation ~ Block + Treatment,data=Table5.7))
```

```
> library(nlme);library(lme4)
  Table5.7.lme <- lme(Observation ~ Treatment, random=~ 1 | Block, data = Table5.7)
  anova(Table5.7.lme)
```

```
            numDF denDF   F-value p-value
(Intercept)     1    13 103.23928  <.0001
Treatment       5    13   2.58997  0.0775
```

```
> Table5.7.lmer <- lmer(Observation ~ Treatment + ( 1 | Block), data = Table5.7)
  anova(Table5.7.lmer)
```

```
Analysis of Variance Table
          Df Sum Sq Mean Sq F value
Treatment  5 17.074  3.4148    2.59
```

```
> anova(lm(Observation ~ Block + Treatment,data=Table5.7))
```

```
anova(lm(Observation ~ Block + Treatment,data=Table5.7))
Analysis of Variance Table

Response: Observation
          Df Sum Sq Mean Sq F value  Pr(>F)
Block      3 17.322  5.7740  4.3921 0.02422 *
Treatment  5 17.132  3.4264  2.6063 0.07619 .
Residuals 13 17.090  1.3146
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
> Table5.7.red.lme <- update(Table5.7.lme, . ~ . - Treatment)
  anova(Table5.7.red.lme,Table5.7.lme)
```

```
> Table5.7.red.lmer <- update(Table5.7.lmer, . ~ . - Treatment)
  anova(Table5.7.red.lmer,Table5.7.lmer)
```

```
> Table5.7.red.lme <- update(Table5.7.lme, . ~ . - Treatment)
  anova(Table5.7.red.lme,Table5.7.lme)
```

```
                Model df      AIC      BIC    logLik   Test  L.Ratio p-value
Table5.7.red.lme     1  3 85.68929 88.82286 -39.84465
Table5.7.lme         2  8 78.02787 84.20858 -31.01393 1 vs 2 17.66143  0.0034
Warning message:
In anova.lme(Table5.7.red.lme, Table5.7.lme) :
  fitted objects with different fixed effects. REML comparisons are not meaningful.
```

```
> Table5.7.red.lmer <- update(Table5.7.lmer, . ~ . - Treatment)
  anova(Table5.7.red.lmer,Table5.7.lmer)
```

```
> Table5.7.red.lme <- update(Table5.7.lme, . ~ . - Treatment)
  anova(Table5.7.red.lme,Table5.7.lme)
```

```
                 Model df      AIC      BIC    logLik   Test  L.Ratio p-value
Table5.7.red.lme     1  3 85.68929 88.82286 -39.84465
Table5.7.lme         2  8 78.02787 84.20858 -31.01393 1 vs 2 17.66143  0.0034
Warning message:
In anova.lme(Table5.7.red.lme, Table5.7.lme) :
  fitted objects with different fixed effects. REML comparisons are not meaningful.
```

```
> Table5.7.red.lmer <- update(Table5.7.lmer, . ~ . - Treatment)
  anova(Table5.7.red.lmer,Table5.7.lmer)
```

```
refitting model(s) with ML (instead of REML)
Data: Table5.7
Models:
Table5.7.red.lmer: Observation ~ (1 | Block)
Table5.7.lmer: Observation ~ Treatment + (1 | Block)
                 Df    AIC    BIC  logLik deviance  Chisq Chi Df Pr(>Chisq)
Table5.7.red.lmer  3 86.063 89.336 -40.032   80.063
Table5.7.lmer      8 83.702 92.430 -33.851   67.702 12.361      5    0.03016

Table5.7.red.lmer
Table5.7.lmer     *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

	plot	summary	anova(.)	anova(.,.)	update
lm	yes	yes	yes	yes	yes
lme	yes a	yes	yes	yes **c**	yes
glmmPQL	yes a	yes	no	no	yes
lmer	yes b	yes	yes **e**	yes **d**	yes
blmer	yes b	yes	yes **e**	yes **d**	yes
glmmadmb	no	yes	no	yes	yes
glmmLasso	no	yes	no	no	no
lmm	no	yes	no	no	no
MCMCglmm	yes	yes	no	no	no
inla	yes	yes	no	no	no
brm	yes	yes	no	no	yes

Table 7. Generic functions provided for objects of different classes of LMM analysis objects.

c. `fitted objects with different fixed effects. REML comparisons are not meaningful`

d. `refitting model(s) with ML (instead of REML)`

e. no p-values for F tests

`blmer` is a Bayesian extension to `lmer`, so tends to provide the same interface.

Maximum a posteriori estimation for linear and generalized linear mixed-effects models in a Bayesian setting, Vincent Dorie, *https://github.com/vdorie/blme*. (2016).

	plot	summary	anova(.)	anova(.,.)	update
lm	yes	yes	yes	yes	yes
lme	yes a	yes	yes c	yes	yes
glmmPQL	yes a	yes **f**	no **g**	no **g**	yes
lmer	yes b	yes	yes d,e	yes	yes
blmer	yes b	yes	yes d,e	yes	yes
glmmadmb	no	yes	no	yes	yes
glmmLasso	no	yes	no	no	no
lmm	no	yes	no	no	no
MCMCglmm	yes	yes	no	no	no
inla	yes	yes	no	no	no
brm	yes	yes	no	no	yes

Table 8. Generic functions provided for objects of different classes of LMM analysis objects.

f. `logLik.glmmPQL` returns `NA`, so `summary` is missing `logLik`, `AIC`, and `BIC` values

g. `'anova' is not available for PQL fits`

The analysis provided by `glmmPQL` is based on repeated calls to `nlme`, so many `nlme` functions apply. However, `glmmPQL` also overrides some functions to return null values, e.g.

```
logLik.glmmPQL <- function (object, ...) {
    ...
    val <- as.numeric(NA)
    ...
    val
}
```

The method is described briefly in

Modern Applied Statistics with S, . Venables, N. and B. Ripley, *Springer, New York*. Fourth Edition (2002). Briefly, `glmmPQL` uses penalized quasi-likelihood to find an optimal solution to a system of mixed-linear models; PQL estimates may not be appropriate in all cases where log-likelihood measures are used.

	plot	summary	anova(.)	anova(.,.)	update
lm	yes	yes	yes	yes	yes
lme	yes a	yes	yes c	yes	yes
glmmPQL	yes a	yes f	no g	no	yes
lmer	yes b	yes	yes d,e	yes	yes
blmer	yes b	yes	yes d,e	yes	yes
glmmadmb	no	yes **h**	no **i**	yes	yes
glmmLasso	no	yes	no	no	no
lmm	no	yes	no	no	no
MCMCglmm	yes	yes	no	no	no
inla	yes	yes	no	no	no
brm	yes	yes	no	no	yes

Table 9. Generic functions provided for objects of different classes of LMM analysis objects.

h. `AIC` reported, but not `BIC` or `logLik`.

i. `Error in anova.glmmadmb(Table5.7.admb) : Two or more model fits required`

	plot	summary	anova(.)	anova(.,.)	update
lm	yes	yes	yes	yes	yes
lme	yes a	yes	yes c	yes	yes
glmmPQL	yes a	yes f	no g	no	yes
lmer	yes b	yes	yes d,e	yes	yes
blmer	yes b	yes	yes d,e	yes	yes
glmmadmb	no	yes h	no i	yes **j**	yes
glmmLasso	no	yes	no	no	no
lmm	no	yes	no	no	no
MCMCglmm	yes	yes	no	no	no
inla	yes	yes	no	no	no
brm	yes	yes	no	no	yes

Table 10. Generic functions provided for objects of different classes of LMM analysis objects.

j.

```
Analysis of Deviance Table
Model 1: obs ~ 1
Model 2: obs ~ trt
  NoPar  LogLik Df Deviance Pr(>Chi)
1     3 -40.032
2     8 -33.851  5   12.361  0.03016 *
```

glmmADMB uses an alternative algorithm to numerically integrate model likelihood, implemented in the open source package described in

AD Model Builder: using automatic differentiation for statistical inference of highly parameterized complex nonlinear models., D. Fournier, H. Skaug, J. Ancheta, J. Ianelli, A. Magnusson, M. Maunder, A. Nielsen and J. Sibert, *Optim. Methods Softw*. 27:233-249 (2012).

glmmADMB not available from CRAN (the central R repository), but can be installed by

```
install.packages("glmmADMB", repos=c("http://glmmadmb.r-forge.r-project.org/repos", getOption("repos")),type="source")
```

	plot	summary	anova(.)	anova(.,.)	update
lm	yes	yes	yes	yes	yes
lme	yes a	yes	yes c	yes	yes
glmmPQL	yes a	yes f	no g	no	yes
lmer	yes b	yes	yes d,e	yes	yes
blmer	yes b	yes	yes d,e	yes	yes
glmmadmb	no	yes h	no i	yes j	yes
glmmLasso	no **k**	yes **l**	no	no	no
lmm	no	yes	no	no	no
MCMCglmm	yes	yes	no	no	no
inla	yes	yes	no	no	no
brm	yes	yes	no	no	yes

Table 11. Generic functions provided for objects of different classes of LMM analysis objects.

k. `No smooth terms to plot!`

l. No residual term reported, so Standard Error of fixed effect estimates is `NA`.

From citation("glmmLasso")

glmmLasso: Variable Selection for Generalized Linear Mixed Models by L1-Penalized Estimation, A. Groll, *https://CRAN.R-project.org/package=glmmLasso*. (2016).

L_1 is a metric space based on absolute differences, while L_2 is based on squared differences. Thus, we would not expect functions that depend on sums of squares for variance estimates (i.e. anova) to be supported for `glmmLasso` objects.

	plot	summary	anova(.)	anova(.,.)	update
lm	yes	yes	yes	yes	yes
lme	yes a	yes	yes c	yes	yes
glmmPQL	yes a	yes f	no g	no	yes
lmer	yes b	yes	yes d,e	yes	yes
blmer	yes b	yes	yes d,e	yes	yes
glmmadmb	no	yes h	no i	yes j	yes
glmmLasso	no k	yes l	no	no	no
lmm	no	no **m**	no	no	no
MCMCglmm	yes	yes	no	no	no
inla	yes	yes	no	no	no
brm	yes	yes	no	no	yes

Table 12. Generic functions provided for objects of different classes of LMM analysis objects.

m. summary.list is applied

If an explicit summary function is not provided, R will use a base object-appropriate function, i.e one for lists:

```
summary(Table5.7.minque)
      Length Class  Mode
reml  5      -none- list
ALPHA 1      -none- numeric

str(Table5.7.minque)
List of 2
 $ reml :List of 5
  ..$ Var         :List of 1
  .. ..$ obs:'data.frame':      2 obs. of  4 variables:
  .. .. ..$ Est    : num [1:2] 0.891 1.318
  .. .. ..$ SE     : num [1:2] 0.936 0.517
  .. .. ..$ Chi_sq : num [1:2] 0.907 6.501
  .. .. ..$ P_value: num [1:2] 0.17049 0.00539
  ..$ FixedEffect :List of 1
  .. ..$ obs:'data.frame':      7 obs. of  4 variables:
  .. .. ..$ Est    : num [1:7] 5.2749 -0.4024 0.2926 1.7751 0.0376 ...
  .. .. ..$ SE     : num [1:7] 0.562 0.559 0.559 0.559 0.559 ...
  .. .. ..$ z_value: num [1:7] 9.3836 -0.7192 0.523 3.1728 0.0673 ...
```

```
 .. .. ..$ P_value: num [1:7] 0 0.47205 0.60094 0.00151 0.94636 ...
 ..$ RandomEffect:List of 1
 .. ..$ obs:'data.frame':      4 obs. of  4 variables:
 .. .. ..$ Pre    : num [1:4] -1.2388 1.0389 -0.0392 0.239
 .. .. ..$ SE     : num [1:4] 0.718 0.718 0.727 0.727
 .. .. ..$ z_value: num [1:4] -1.7254 1.4471 -0.0539 0.3286
 .. .. ..$ P_value: num [1:4] 0.0844 0.1479 0.957 0.7424
 ..$ Residual    : num [1:22, 1] -0.204 0.921 0.659 -1.274 -1.006 ...
 .. ..- attr(*, "dimnames")=List of 2
 .. .. ..$ : NULL
 .. .. ..$ : chr "obs"
 ..$ Fitted      : num [1:22, 1] 3.63 4.33 5.81 4.07 3.67 ...
 .. ..- attr(*, "dimnames")=List of 2
 .. .. ..$ : NULL
 .. .. ..$ : chr "obs"
$ ALPHA: num 0.05
```

	plot	summary	anova(.)	anova(.,.)	update
lm	yes	yes	yes	yes	yes
lme	yes a	yes	yes c	yes	yes
glmmPQL	yes a	yes f	no g	no	yes
lmer	yes b	yes	yes d,e	yes	yes
blmer	yes b	yes	yes d,e	yes	yes
glmmadmb	no	yes h	no i	yes j	yes
glmmLasso	no k	yes l	no	no	no
lmm	no	no m	no	no	no
MCMCglmm	yes **n**	yes	no	no	no
inla	yes **o**	yes	no	no	no
brm	yes **p**	yes	no	no	yes

Table 13. Generic functions provided for objects of different classes of LMM analysis objects.

n. requires `Hit <Return> to see next plot:`

o. opens several plot windows simultaneously

p. ggplot based graphics

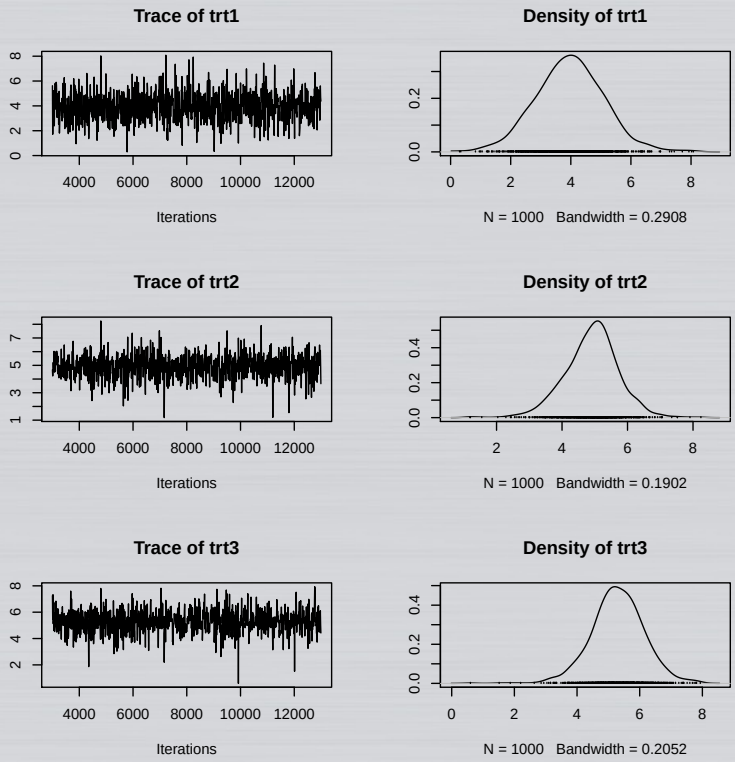


Figure 1. Representative diagnostic plot from `MCMCglmm`, applied to Example 1 data.

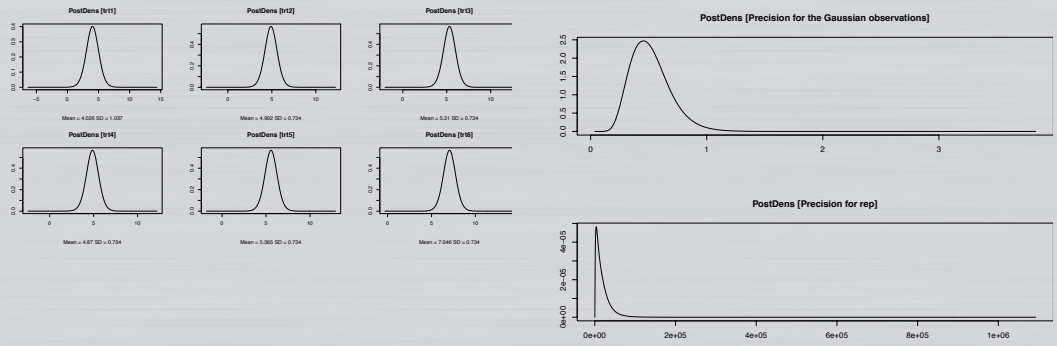


Figure 2. Representative `inla` diagnostic plots, applied to Example 1 data.

`inla` diagnotic plots are more difficult to use outside an active session.

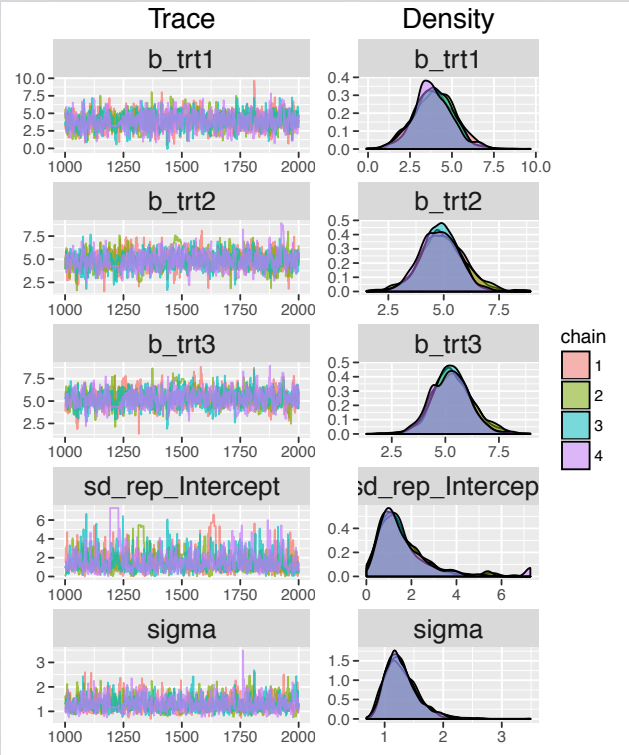


Figure 3. Partial diagnostics plot, `brm`, Example 1.

`plot.brm` allows more control over diagnostic plots. This graph was generated by calling

```
plot(Table5.7.brms,pars=c(“b_trt1”,”b_trt2”,”b_trt3”,”sd_rep_Intercept”,
”sigma”))
```

	Block		Residual	
	σ_b^2	σ_b	σ^2	σ
Expected	1		1	
Published	0.89		1.32	
lm	?	?	1.3146	*1.1466*
lme	*0.8908*	0.9438	*1.3185*	1.1483
glmmPQL	0.67012	0.8186	0.9534	0.9764
lmer	*0.8908*	0.9438	*1.3185*	1.1483
blmer	2.335	1.528	1.168	1.081
glmmadmb	0.6701	0.8186		0.9764
lmm	0.8908	*0.9438*	1.3185	*1.1483*
glmmLasso	*2.1412*	1.4633	NA	NA
MCMCglmm	$<$0.0001	*0.0092*	2.5011	*1.5815*
inla	18590.8084	136.3481	0.5174	*0.7193*
brm	*2.25*	1.5	*1.64*	1.28

Table 14. Variance estimates for Example 1, Model 1. Some packages report variances, some report standard deviations, some both. Reported values are regular font, values calculated from `summary` are in italics.

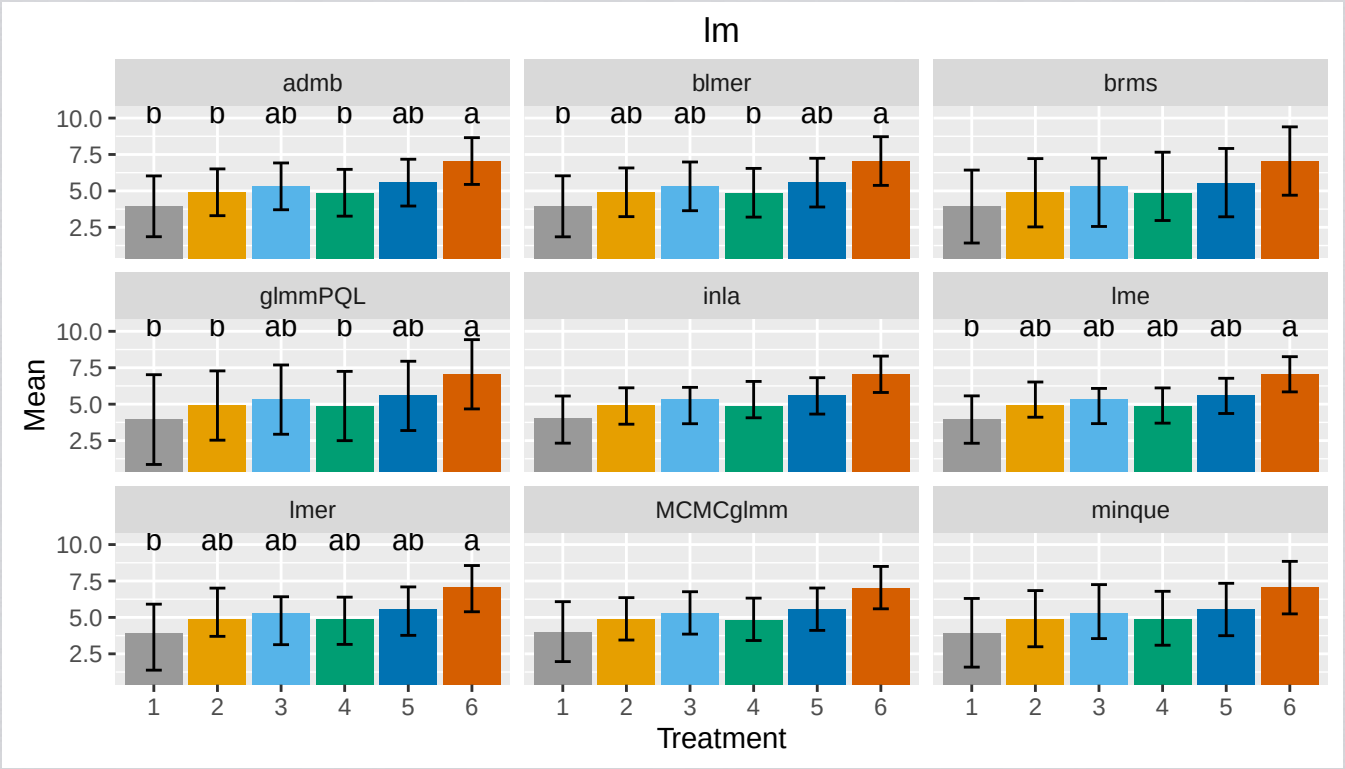


Figure 4. Mean summary plots. Plots with letters were generated using lsmeans and cld/glht to produce mean estimates, error bars and compact letter display. Means and error bars in plots without letters were calculated manually from summary of the associated model object.

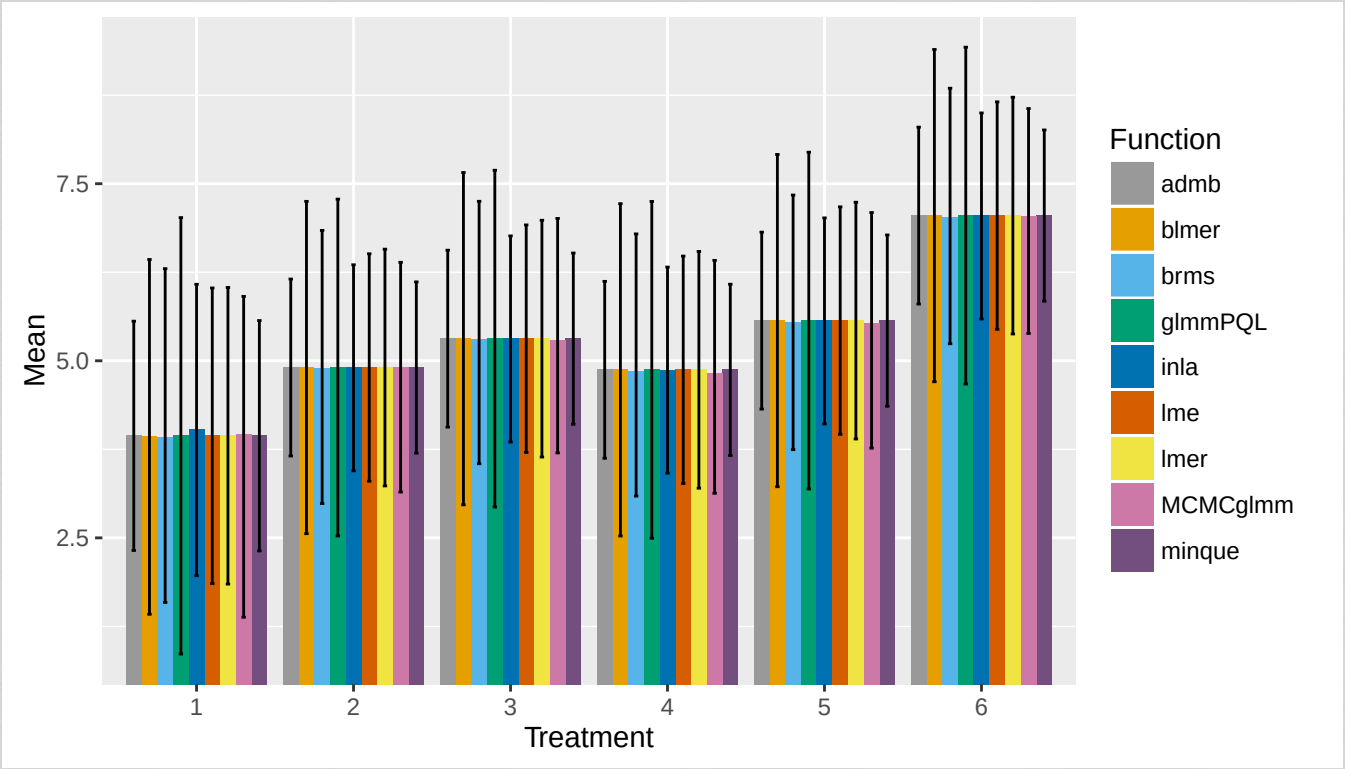


Figure 5. Combined means plot, Example 1, Model 1.

All packages tend to agree on means estimates, but the Bayesian packages, `brm`, `inla` and `MCMCglmm` tend to produce larger estimates for the standard error of means.

	Mixed Model Formula
lm	`Yield ~ Location:Block + Location*Strain`
lme	`Yield ~ Location*Strain, random = ~ 1 | LBlock`
glmmPQL	`Yield ~ Location*Strain, random = ~ 1 | LBlock, family=gaussian`
lmer	`Yield ~ Strain + (1 | Location/Rep) + (1 | Location:Strain)`
blmer	`Yield ~ Strain + (1 | Location/Rep) + (1 | Location:Strain)`
glmmadmb	`Yield ~ Strain, family = "gaussian'',` `random = ~ (1 | Location/Block) +(1 | Location:Strain)`
glmmLasso	`Yield ~ Strain,` `rnd = list(Location=~1, LBlock=~1, Interaction=~1)`
lmm	`Yield ~ Strain | Location/Block + Strain:Location`
MCMCglmm	`Yield ~ Strain,` `random = ~ Location + Location:Block + Location:Strain`
inla	`Yield ~ Strain + f(Location, model="iid") +` `f(Location:Block,model="iid") + f(Location:Strain,model="iid`
brm	`Yield ~ Strain + (1 | Location/Block) + (1 | Location:Strain)`

Table 15. R syntax for an analysis of a series of similar experiments.

`lme` does not provide simple syntax for crossed random effects, so location is fit as a fixed effect. `lme` will accept `Location/Block` as a nested random effect, but not `Location:Block`, so a dummy variable `LBlock` is required.

	Mixed Model Formula
lm	`Yield ~ Location:Block + Location*Strain`
lme	`Yield ~ Location*Strain, random = ~ 1 | LBlock`
glmmPQL	`Yield ~ Location*Strain, random = ~ 1 | LBlock, family=gaussian`
lmer	`Yield ~ Strain + (1 | Location/Rep) + (1 | Location:Strain)`
blmer	`Yield ~ Strain + (1 | Location/Rep) + (1 | Location:Strain)`
glmmadmb	`Yield ~ Strain, family = "gaussian'',` `random = ~ (1 | Location/Block) +(1 | Location:Strain)`
glmmLasso	`Yield ~ Strain,` `rnd = list(Location=~1, LBlock=~1, Interaction=~1)`
lmm	`Yield ~ Strain | Location/Block + Strain:Location`
MCMCglmm	`Yield ~ Strain,` `random = ~ Location + Location:Block + Location:Strain`
inla	`Yield ~ Strain + f(Location, model="iid") +` `f(Location:Block,model="iid") + f(Location:Strain,model="iid`
brm	`Yield ~ Strain + (1 | Location/Block) + (1 | Location:Strain)`

Table 16. R syntax for an analysis of a series of similar experiments.

`glmmLasso` syntax is difficult for crossed random effects, so dummy variables `LBlock` and `Interaction` are required.

	Trial σ^2_l	Interaction $\sigma^2_{\tau \times l}$	Block $\sigma^2_{b(l)}$	Residual σ^2_ϵ
lm			*-680*	19708
lme	-	-	*0.00008*	*19027*
glmmPQL	-	-	*0.00004*	*12684*
lmer	42572	1732	0	19027
blmer	31253	2079	1486	18590
glmmadmb	28356	0.0122	0.699	*18123*
lmm	49799	1505	-0.680	19708
glmmLasso	*338061*	*7693*	*36726*	NA
MCMCglmm	265650	229	3.462	20372
inla	13902	68094	20427	0
brm	*64313*	*1773*	*560*	*19656*

Table 17. Variance estimates, Example 2 and Model 2

The key difference is the estimate of block variance, but there is also considerable disagreement about trial and interaction variances.

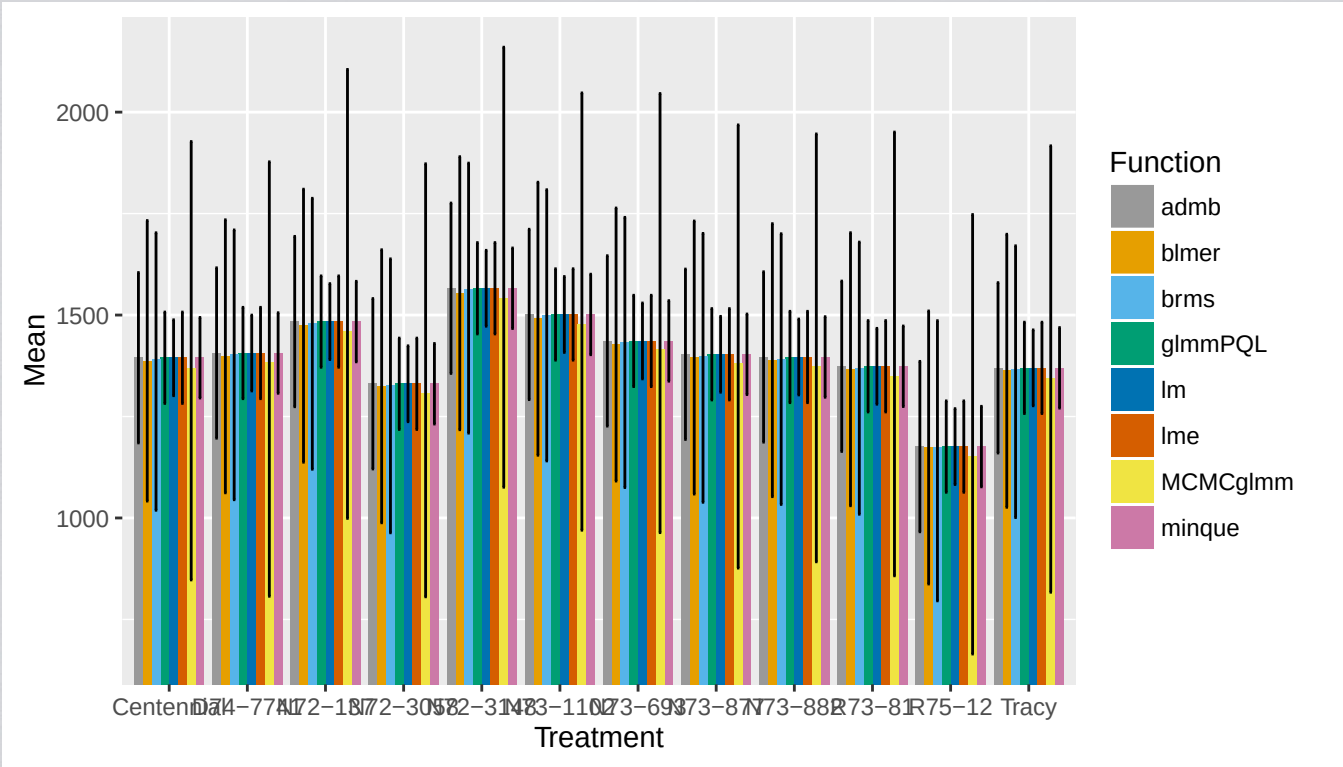


Figure 6. Combined means plot, Example 2, Model 2.

Error bars for `lm`, `lme` and `glmmPQL` are smaller, since these models were fitted with Trial as a fixed effect. Bayesian (MCMC) based packages produce larger error terms because block effects are non-trivial.

	Mixed Model Formula
lm	NA
lme	`YLD ~ Loca*Entry, random = ~ 1 | LBlock, weights = varIdent(form = ~ Plot | Loca)`
glmmPQL	`YLD ~ Loca*Entry, random = ~ 1 | LBlock, weights = varIdent(form = ~ Plot | Loca)`
lmer	`YLD ~ Entry + (1 | Loca/Repe) + (0 + Entry | Loca)`
blmer	`YLD ~ Entry + (1 | Loca/Repe) + (0 + Entry | Loca)`
glmmadmb	`YLD ~ Entry, random= ~ (1 | Loca/Repe) + (0 + Entry | Loca)`
glmmLasso	`YLD ~ Entry, rnd = list(Loca = ~1, LBlock = ~1,Loca=~ 0 + Entry)`
lmm	NA
MCMCglmm	`YLD ~ Entry, random = ~ Loca + idh(Entry):Loca + Loca:Repe, rcov=~idh(Loca):units`
inla	NA
brm	`YLD ~ Entry + (1 | Loca:Repe) + (Entry | Loca)`

Table 18. R syntax for Model 3, Example 3. Example 3 is also analyzed using Model 2 as previously described. An appropriate model syntax to express Model 3 could not be found for all packages.

	Model 2		Model 3	
	self	total	self	total
lm	.20	.20	NA	NA
lme	5.62	5.94	100.38	113.08
glmmPQL	11.50	12.60	-	-
lmer	.32	.44	4243	4308
blmer	.16	.74	-	-
glmmadmb	.18	.52	-	-
glmmLasso	2.82	4.58	2.82	3.76
lmm	14.52	15.68	NA	NA
MCMCglmm	9.76	10.02	17.20	17.84
inla	.24	1.12	NA	NA
brm	374.38	375.56	-	-

Table 19. Function timing generated by calling, e.g.

```
Rprof("meta.lm.prof")
meta.lm <- lm(YLD ~ Loca:Repe + Entry*Loca, data=rcbd.dat)
Rprof(NULL)
summaryRprof("META.lm.prof")
```

for the data from Example 3.

self.time refers to time spend on local computations, total.time includes time spent in called R routines.

	Model 2		Model 3	
	self	total	self	total
lm	.20	.20	NA	NA
lme	5.62	5.94	100.38	113.08
glmmPQL	11.50	12.60	-	-
lmer	.32	.44	4243	4308
blmer	.16	.74	-	-
glmmadmb	.18	.52	-	-
glmmLasso	2.82	4.58	2.82	3.76
lmm	14.52	15.68	NA	NA
MCMCglmm	9.76	10.02	17.20	17.84
inla	.24	1.12	NA	NA
brm	374.38	375.56	-	-

Table 20. Function timing

`lmer` (and by extension, `blmer`) use Laplace approximation for maximum likelihood estimate, which tends to be a fast algorithm. Since `glmmadmb` and `inla` both invoke external processes, timings may be misleading. `summary(meta.inla)` reports a total time of 7.62 for Model 2.

Execution time for `brm` is slow because execution requires compilation of C code by the stan system. Once code is compiled, repeated executions may be much faster, within the family of Bayesian estimation.

	Model 2		Model 3	
	self	total	self	total
lm	.20	.20	NA	NA
lme	5.62	5.94	100.38	113.08
glmmPQL	11.50	12.60	- **a**	-
lmer	.32	.44	4243	4308
blmer	.16	.74	-	-
glmmadmb	.18	.52	-	-
glmmLasso	2.82	4.58	2.82	3.76
lmm	14.52	15.68	NA	NA
MCMCglmm	9.76	10.02	17.20	17.84
inla	.24	1.12	NA	NA
brm	374.38	375.56	-	-

Table 21. Function timing.

a. `Error in model.frame.default(data = rcbd.dat, weights = varIdent(form = ~Plot | : variable lengths differ (found for '(weights)')`

	Model 2		Model 3	
	self	total	self	total
lm	.20	.20	NA	NA
lme	5.62	5.94	100.38	113.08
glmmPQL	11.50	12.60	- a	-
lmer	.32	.44	4243	4308 **b**
blmer	.16	.74	-	- **b**
glmmadmb	.18	.52	-	-
glmmLasso	2.82	4.58	2.82	3.76
lmm	14.52	15.68	NA	NA
MCMCglmm	9.76	10.02	17.20	17.84
inla	.24	1.12	NA	NA
brm	374.38	375.56	-	-

Table 22. Function timing.

b. Both `lmer` and `blmer` reported convergence issues for Model 3, but only `lmer` returned a result.

	Model 2		Model 3	
	self	total	self	total
lm	.20	.20	NA	NA
lme	5.62	5.94	100.38	113.08
glmmPQL	11.50	12.60	- a	-
lmer	.32	.44	4243	4308 b
blmer	.16	.74	-	- b
glmmadmb	.18	.52	- **c**	-
glmmLasso	2.82	4.58	2.82	3.76
lmm	14.52	15.68	NA	NA
MCMCglmm	9.76	10.02	17.20	17.84
inla	.24	1.12	NA	NA
brm	374.38	375.56	-	-

Table 23. Function timing.

c. `The function maximizer failed (couldn't find parameter file) Troubleshooting steps include (1) run with 'save.dir' set and inspect output files; (2) change run parameters: see '?admbControl';(3) re-run with debug=TRUE for more information on failure mode`

	Model 2		Model 3	
	self	total	self	total
lm	.20	.20	NA	NA
lme	5.62	5.94	100.38	113.08
glmmPQL	11.50	12.60	- a	-
lmer	.32	.44	4243	4308 b
blmer	.16	.74	-	- b
glmmadmb	.18	.52	- c	-
glmmLasso	2.82	4.58	2.82	3.76
lmm	14.52	15.68	NA	NA
MCMCglmm	9.76	10.02	17.20	17.84
inla	.24	1.12	NA	NA
brm	374.38	375.56	- **d**	-

Table 24. Function timing.

d.

SAMPLING FOR MODEL 'gaussian(identity) brms-model' NOW (CHAIN 1).
[1] "Rejecting initial value:"
[2] " Log probability evaluates to log(0), i.e. negative infinity."
[3] " Stan can't start sampling from this initial value."
[4] "Error : model6ae1142a794_filed6ae473b77fd_namespace::write_array: y is not positive definite."
error occurred during calling the sampler; sampling not done

	Model	lm	lme	lmer	glmmLasso	MCMCglmm
Trial σ^2_l	2			8.4472	1.8924	10.19
Trial σ^2_l	3			8.0412	1.8921	11.11
Interaction $\sigma^2_{\tau\times l}$	2			0.4322	0.2094	0.4318
Interaction						
$\sigma^2_{\tau_1\times l}$	3			0.1574	0.0408	0.0003
$\sigma^2_{\tau_2\times l}$	3			0.4715	0.0401	0.0017
$\sigma^2_{\tau_3\times l}$	3			0.2159	0.0422	0.0048
Block $\sigma^2_{b(l)}$	2		*0.1416*	0.1417	0.7392	0.1569
Block $\sigma^2_{b(l)}$	3		*0.0610*	0.1494	0.7390	0.0737
Residual σ^2_ϵ	2	0.94	*0.9411*	0.9412		0.9435
Residual σ^2_ϵ	3		*0.4534*			
$\sigma^2_{\epsilon_1}$	3		a *0.4534*			0.5753
$\sigma^2_{\epsilon_2}$	3		*0.2954*			0.3489
$\sigma^2_{\epsilon_3}$	3		*2.4707*			2.4028

Table 25. Variances estimates from Example 3, Model 3. Stability variances for the first three treatments and residual variances for the first three trials shown were applicable.

a. lme reports standard deviations per strata, the display values are calculated from

Does the added complexity justify computational difficulties?

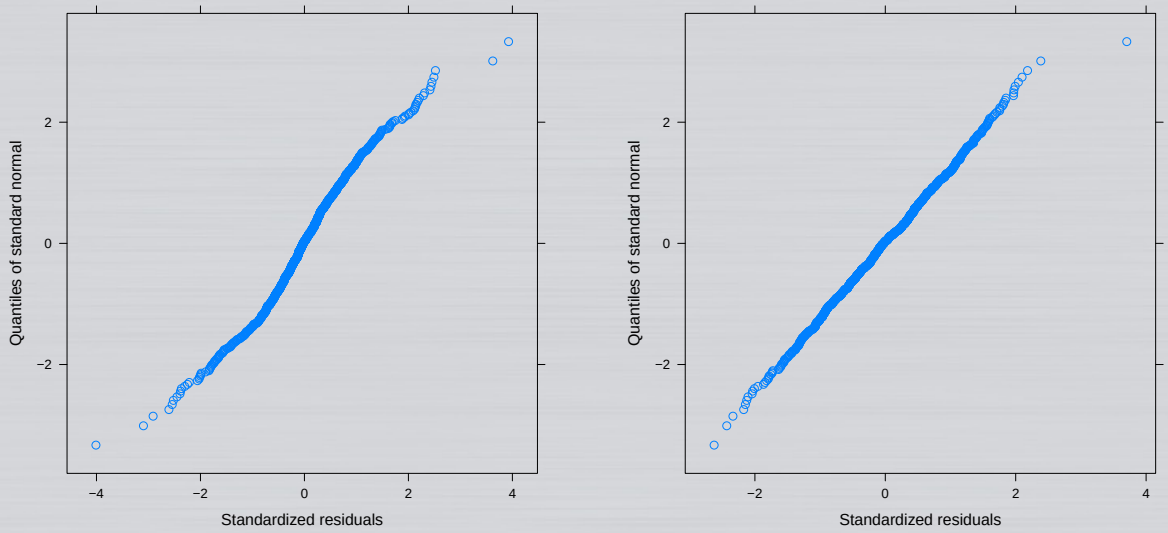


Figure 7. QQ-plots from Model 2 and Model 3 fit using lme on Example 3 data.

We can compare the two models more formally using

```
> anova(meta.lme,cotes.lme)
```

The model with smaller or more negative AIC and BIC values are interpreted to an improvement on the model with larger AIC or BIC values. AIC and BIC are calculated from log-likelihood and provide different penalties for the number of parameters in a model.

We can compare the two models more formally using

```
> anova(meta.lme,cotes.lme)
```

```
          Model  df      AIC      BIC    logLik   Test L.Ratio p-value
meta.lme      1 386 3369.047 5161.55 -1298.524
cotes.lme     2 397 3108.315 4951.90 -1157.158 1 vs 2 282.732  <.0001
```

The model with smaller or more negative AIC and BIC values are interpreted to an improvement on the model with larger AIC or BIC values. AIC and BIC are calculated from log-likelihood and provide different penalties for the number of parameters in a model.

We can't use `anova(.,.)` to compare the two models fit using `MCMCglmm`, but we can perform a rough comparison by examining DIC:

```
summary(meta.mcmc)$DIC
[1] 3465.136
> summary(cotes.mcmc)$DIC
[1] 3151.771
> summary(meta.mcmc)$DIC - summary(cotes.mcmc)$DIC
[1] 313.3646
```

We don't get an explicit p-value with this method, but we might remember that the differences in AIC/BIC for `lme` was ~260. DIC is a generalization of AIC and BIC.

lmer did not converge on a solution for Model 3, so we can't make a valid comparison; however, let this be a warning (my bold):

```
anova(meta.lmer,cotes.lmer)
refitting model(s) with ML (instead of REML)
Data: rcbd.dat
Models:
meta.lmer: YLD ~ Entry + (1 | Loca/Repe) + (1 | Loca:Entry)
cotes.lmer: YLD ~ Entry + (1 | Loca/Repe) + (0 + Entry | Loca)
            Df    AIC    BIC  logLik deviance Chisq Chi Df Pr(>Chisq)
meta.lmer   36 3683.6 3865.4 -1805.8   3611.6
cotes.lmer 563 4313.4 7156.1 -1593.7   3187.4 424.2    527     0.9996
Warning messages:
1: In commonArgs(par, fn, control, environment()) :
  maxfun < 10 * length(par)^2 is not recommended.
2: In optwrap(optimizer, devfun, x@theta, lower = x@lower, calc.derivs
= TRUE,  :
  convergence code 1 from bobyqa: bobyqa -- maximum number of function
evaluations exceeded
```

Remember that profiling found that lmer required >4000 seconds before returning; this results in an unexpected computing delay.

The presentation is a literate programming document formatted in TEXMACS using the `beamer` theme.

GNU TEXMACS: a scientific editing platform, J. van der Hoeven, *www.texmacs.org*. (2006).