Mixed-effects models

Remko Duursma[1], *Jeff Powell*[1]
[1]Hawkesbury Institute for the Environment
Western Sydney University

WESTERN SYDNEY
UNIVERSITY

Hawkesbury Institute
for the Environment

September 14, 2016

Preface

This book is a collection of advanced topics in R. References to previous sections, as well as descriptions of datasets, are to the book by Duursma, Powell and Stone: *Data analysis and visualization with R*.

Contents

Chapter 1

Mixed-effects models

1.1 Introduction

In Chapter 7, we used linear models to estimate 'fixed' effects, which consist of specific and repeatable categories/variables that are representative of an entire population (e.g., species, age). In longitudinal studies (repeated measures) and in studies using hierarchical (nested) sampling, it is also possible to estimate effects associated with individuals sampled at random from the population of interest. These are 'random' effects and convey information about the degree that individuals in a population differ but not how or why they differ.

One way to differentiate fixed and random effects is that fixed effects contain levels that are informative beyond the current analysis (e.g., a species of tree or a specific management type) while random effects contain levels that are not informative beyond the current analysis (e.g., a group of trees within an observation plot or a field under specific management). Another way to understand the difference is that fixed effects influence the mean of the response while random effects influence the variance of the response.

Mixed-effects models estimate both fixed and random effects and are particularly useful when dealing with potential pseudoreplication and unbalanced designs. Including random effects can also account for variation that could mask patterns in an analysis considering only fixed effects.

Two commonly used packages for fitting mixed-effects models are `nlme` and `lme4`. In this chapter, we will use the newer `lme4` package, but note that more complex correlation structures are only possible with the `nlme` package.

Rather than to present the theory underlying mixed-effects models, which is very complex, we will treat this topic by example, and thus aim at a practical application. An example of this is given in Section 1.2.

1.1.1 A note about p-values

The `lme4` package does not report p-values. The developers made this decision because p-values require calculating degrees of freedom. Random effects don't necessarily have to expend the same degrees of freedom as treating them as fixed effects, so the package developers have decided not to fudge this by calculating them for you.

In this chapter, we use two approaches to calculate p-values of fixed effects. To obtain the p-values of all fixed effects in a model, we use the Kenward-Roger approximation to the degrees of freedom, as recommended by Halekoh and Höjsgaard (2014) [1]. This approximation is most conveniently implemented in the `Anova` function from the `car` package (not `anova`!), and used like this (if you have a fitted model `mymodel` returned by `lmer`),

```
library(car)
Anova(mymodel, test="F")
```

Note that we cannot use `test="F"` for generalized linear mixed-effects models (i.e. fit with `glmer`), and some caution needs to be applied with the `Anova` default use of the chi-squared test statistic.

The second approach is to use likelihood-ratio tests to test the significance of a single fixed effect in a model. To do this, we fit two models (one with, and one without the fixed effect of interest), and use the `KRmodcomp` function from the `pbkrtest` package (again using the Kenward-Roger approximation as mentioned above). Note that the results are usually slightly different, and sometimes very different, if you use the standard likelihood-ratio test with the `anova` function.

Finally, you may wish to inspect p-values for individual coefficients in the `summary` statement of the fitted model. With the `lmerTest` package loaded, fit the model with `lmer`, which will now show p-values in the `summary`. This even works for `glmer`, but the z-score approximation is probably quite poor. Use with caution.

We will demonstrate the use of each of these functions in the examples that follow.

Further reading See the help page `?pvalues` for references to several options to calculate p-values with `lme4`.

1.2 Example: individual-level variation in tree canopy gradients

The following example uses the `pref` data (not yet described in Appendix A, download the file 'prefdata.csv'). The dataset contains measurements of leaf mass per area (LMA), and distance from the top of the tree (dfromtop) on 35 trees of two species. We want to know whether LMA decreases with `dfromtop`, as expected, and whether this decrease in LMA with distance from top differs by species.

```
# Read the data and inspect the first few rows, and the species factor variable.
pref <- read.csv("prefdata.csv")
head(pref)

##     ID         species dfromtop totheight height      LMA    narea
## 1 FP11 Pinus ponderosa     8.88     22.40  13.52 319.4472 2.779190
## 2 FP11 Pinus ponderosa     0.62     22.40  21.78 342.7948 4.010700
## 3 FP11 Pinus ponderosa     4.72     22.40  17.68 329.5399 3.365579
## 4 FP15 Pinus ponderosa     2.74     27.69  24.95 312.4467 3.682907
## 5 FP15 Pinus ponderosa     5.48     27.69  22.21 278.4037 2.524224
## 6 FP15 Pinus ponderosa     8.40     27.69  19.29 255.9716 2.351546

levels(pref$species)

## [1] "Pinus monticola" "Pinus ponderosa"
```

Before we fit mixed-effects models, let's start with a linear regression that includes `dfromtop` and `species` as the predictor variables to observe the general patterns. We use `visreg` to quickly visualize the fitted linear model. The following code produces Fig. 1.1.

[1] Ulrich Halekoh and Sören Höjsgaard, 2014, A Kenward-Roger Approximation and Parametric Bootstrap Methods for Tests in Linear Mixed Models – The R Package pbkrtest, J. Stat. Software 59, 1-30

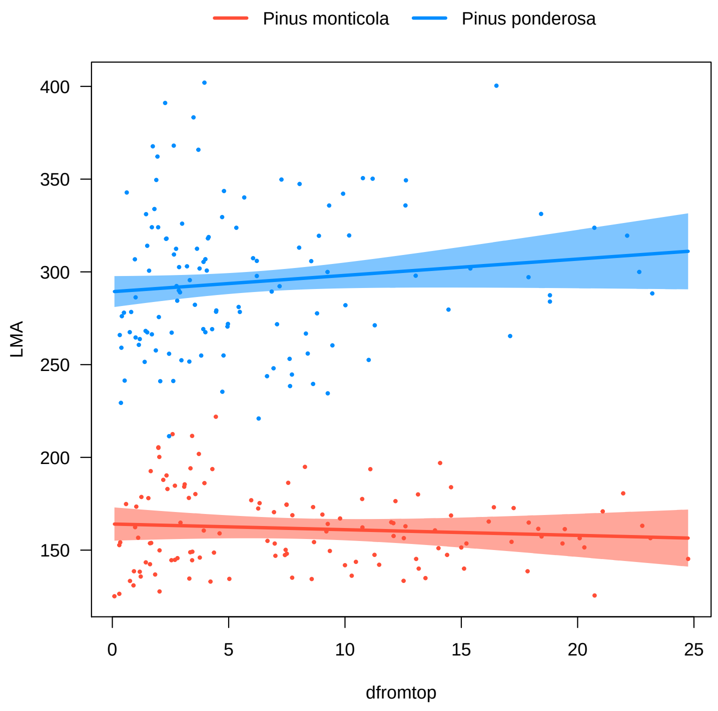

Figure 1.1: Leaf mass per area as a function of tree species (two colours) and the distance from the top of each tree, as fitted with a simple linear model and visualized with visreg.

```
# Fit a linear regression by species (ignoring individual-level variation)
lm1 <- lm(LMA ~ species + dfromtop + species:dfromtop, data=pref)

# Plot predictions
library(visreg)
visreg(lm1, "dfromtop", by="species", overlay=TRUE)
```

As we can see in Fig. 1.1, there is a strong effect of species, but it appears that `LMA` and `dfromtop` are not significantly correlated.

Try this yourself Look at the `anova` table for the fitted linear regression model from the example above to confirm that LMA does not significantly change with dfromtop.

To see whether the relationship between `LMA` and `dfromtop` potentially varies from tree to tree, we fit a linear regression separately for each tree using the `lmList` function in the `lme4` package and then plot the outcome (in Fig. 1.2).

```
# For the lmList function (Note: the nlme package also includes the lmList function)
library(lme4)

# fit linear regression by tree ('ID')
lmlis1 <- lmList(LMA ~ dfromtop | ID, data=pref)

# Extract coefficients (intercepts and slopes) for each tree
liscoef <- coef(lmlis1)
```

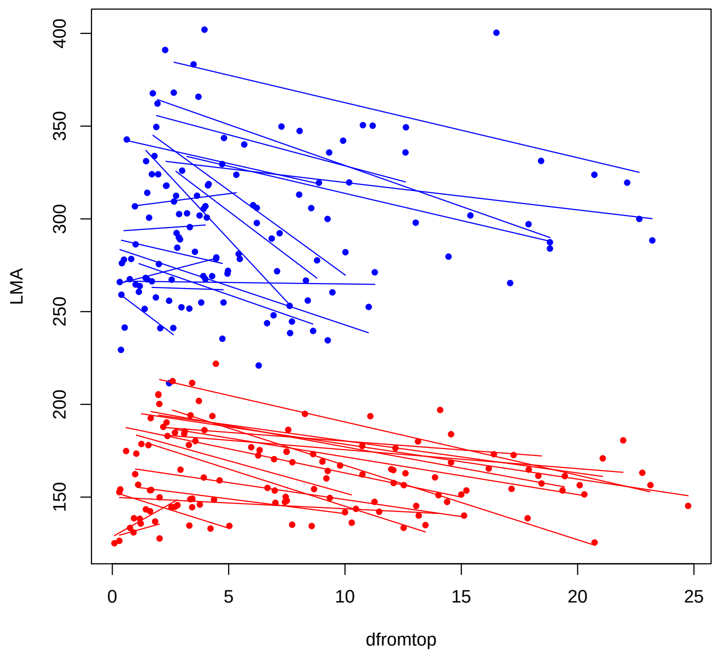

Figure 1.2: Leaf mass per area as a function of tree species (two colours) and the distance from the top of each tree. The solid lines represent the slope of the relationship for each individual tree.

```r
# load plottix for the 'ablineclip' function, which clips lines within the range of x
library(plotrix)

# split pref by tree (prefsp is a list)
prefsp <- split(pref, pref$ID)

# Plot
palette(c("red","blue"))
with(pref, plot(dfromtop, LMA, col=species, pch=16, cex=0.8))
for(i in 1:length(prefsp)){

  # Find min and max values of dfromtop, to send to ablineclip
  xmin <- min(prefsp[[i]]$dfromtop)
  xmax <- max(prefsp[[i]]$dfromtop)

  # add regression lines
  ablineclip(liscoef[i,1], liscoef[i,2], x1=xmin, x2=xmax,
             col=prefsp[[i]]$species)
}
```

From the figure we can conclude (informally) that:

1. Intercepts vary a lot between trees
2. There seems to be a negative relationship for many trees
3. It seems there is less variation between slopes than intercepts

We know the individual data points are not independent, as they are nested within trees (that is, multiple samples were collected for each tree). To properly account for this non-independence, we have to use a mixed-effects model. In the example below, we will fit two models: one with a random intercept only, and one with a random intercept and slope.

1.2.1 A simple equation for a mixed-effects model

Before we fit a mixed-effects model, let's write down an equation that illustrates the roles of the fixed effects and random effects in the model.

For a simple linear regression of Y versus X (where X is numeric), we fit the model:

$$Y = \beta_0 + \beta_1 \cdot X$$

where the β's are the intercept and the slope of the fitted regression line.

A mixed-effects model additionally fits two random parameters, and we can write the model as:

$$Y = (\beta_0 + b_0) + (\beta_1 + b_1) \cdot X$$

where b_0 is the random intercept (which is normally distributed with mean zero, and some standard deviation), and b_1 is the random slope (also assumed to be normally distributed with mean of zero and some standard deviation).

When we fit a mixed-effects model, not only do we get an estimate of the variance (or standard deviation) of the random effects, we also get estimates of this random effect for each individual. These estimates are known as the BLUPs (best linear unbiased predictors).

1.2.2 Fitting the mixed-effects model

To specify random effects with `lmer`, we add it to the formula in the right-hand side. For example, a random intercept for 'ID' (that is, the intercept will vary randomly among ID's) is coded as `(1|ID)`. If we also allow the slope of the relationship to vary, we specify it as `(dfromtop|ID)` so that the slope and intercept of the relationship between LMA and dfromtop will vary randomly between tree ID's.

```
# Random intercept only
pref_m1 <- lmer(LMA ~ species + dfromtop + species:dfromtop + (1|ID), data=pref)

# Random intercept and slope
pref_m2 <- lmer(LMA ~ species + dfromtop + species:dfromtop + (dfromtop|ID), data=pref)

# The AIC and a likelihood-ratio test tell us that we don't need a random slope.
# lower AIC indicates that model fit is better (more efficient)
AIC(pref_m1, pref_m2)
```

```
##         df      AIC
## pref_m1  6 2251.997
## pref_m2  8 2255.735
```

```
# Likelihood ratio test : the more complex model is not supported by the data.
# Note: the models will be re-fitted with ML instead of REML; this is necessary
# when performing likelihood-ratio tests.
anova(pref_m1, pref_m2)
```

```
## Data: pref
## Models:
## pref_m1: LMA ~ species + dfromtop + species:dfromtop + (1 | ID)
## pref_m2: LMA ~ species + dfromtop + species:dfromtop + (dfromtop | ID)
##         Df    AIC    BIC  logLik deviance  Chisq Chi Df Pr(>Chisq)
## pref_m1  6 2263.2 2284.3 -1125.6   2251.2
## pref_m2  8 2267.1 2295.2 -1125.5   2251.1 0.1274      2     0.9383
```

```
# Output from the random intercept model - not shown (inspect yourself!)
# summary(pref_m1)

# Using Anova from car, we get p-values for the main effects.
library(car)
Anova(pref_m1)
```

```
## Analysis of Deviance Table (Type II Wald chisquare tests)
##
## Response: LMA
##                    Chisq Df Pr(>Chisq)
## species          147.475  1     <2e-16 ***
## dfromtop          86.832  1     <2e-16 ***
## species:dfromtop   2.531  1     0.1116
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Note that in the above, we use the familiar `anova` function to test between models with different random effects structure. Since we use `anova` on an object returned by `lmer`, this command invokes the `anova.merMod` function, from the `lme4` package.

Try this yourself Inspect the help file `?anova.merMod`, and find out that you can name the two models, simplifying the interpretation of the command `anova(lme1, lme2)` from the above example.

We now conclude that `LMA` decreases significantly with `dfromtop`. Compare this with the fixed-effects model we started with:

```
summary(lm1)
```

```
##
## Call:
## lm(formula = LMA ~ species + dfromtop + species:dfromtop, data = pref)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -80.047 -21.737  -2.908  17.231 109.207
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    164.0749     4.5800  35.824   <2e-16 ***
## speciesPinus ponderosa         125.2444     6.2581  20.013   <2e-16 ***
## dfromtop                        -0.3062     0.4406  -0.695   0.4878
## speciesPinus ponderosa:dfromtop   1.1855     0.6923   1.712   0.0881 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 31.34 on 245 degrees of freedom
## Multiple R-squared:  0.8211,Adjusted R-squared:  0.8189
## F-statistic: 374.9 on 3 and 245 DF,  p-value: < 2.2e-16
```

Ignoring the tree-to-tree variance thus resulted in drawing the wrong conclusion from our data. When we accounted for this variation with a mixed-effects model, we did find a significant overall relationship between LMA and dfromtop. The reason for this discrepancy is that the large variation in intercepts between the individual trees (between-subject variation) masked the relationship between the two variables within individuals (within-subject variation).

Finally, we may be interested in quantifying the variation between individuals in terms of the intercept and slope. These are the standard deviations of the random effects, and can be extracted with the `VarCorr` function (it is also shown in the `summary` statement of the mixed-effects model).

```
# Standard deviation of the intercept and slope ('dfromtop') between individuals,
# as estimated from the random effects.
VarCorr(pref_m2)

##  Groups   Name        Std.Dev. Corr
##  ID       (Intercept) 31.36023
##           dfromtop     0.47658 -0.304
##  Residual             17.96337
```

Try this yourself Try using `ranef` on `lme1` and `lme2`; the first will show intercepts for each of the random effects (trees), while the second will show both estimated intercepts and slopes for the random effects. Repeat the above example looking at the relationship between `narea` (leaf nitrogen per unit area), `dfromtop`, and `species`.

1.3 Example: individual-level variation in metabolic rate of mice

In this example we will look at a dataset containing metabolic rate measured on mice at three different temperatures. Measurements on every mouse were repeated three times. We are interested in estimating the change in metabolic rate with temperature, and whether this relationship changes with body mass, sex (male or female), whether the mice were fed on the day of measurements, and whether they were using a wheel.

First we read and inspect the dataset (data not yet described in the Appendix).

```
mouse <- read.csv("wildmousemetabolism.csv")

# Make sure the individual label ('id') is a factor variable
mouse$id <- as.factor(mouse$id)
```

As a first step let's look at the data. We would like to visualize whether temperature has an effect on metabolic rate, and whether individuals were very different in terms of metabolic rate (`rmr`) at a fixed temperature. The following code makes Fig. 1.3.

```
# Raw data of rmr versus temperature (temp). Note use of jitter()
# to minimize overlap between data.
with(mouse, plot(jitter(temp),rmr, pch=21, bg="#BEBEBE99", ylim=c(0,0.6)))

# Take a subset, and reorder the 'id' factor levels by rmr.
```

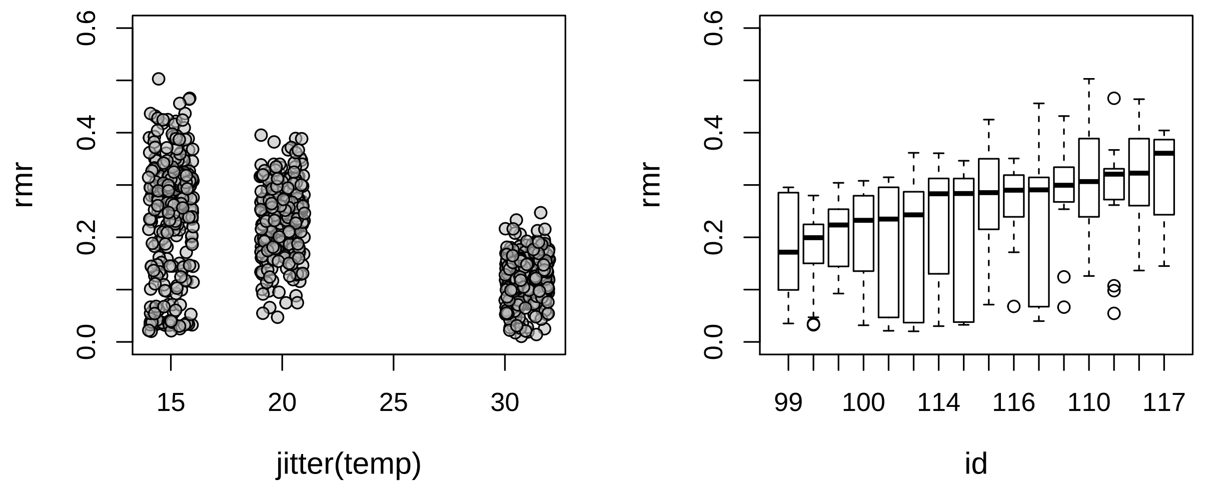

Figure 1.3: Resting metabolic rate (rmr) for individual mice. Left panel: raw data against temperature, demonstrating increase in rmr with decreasing temperature, and larger variance at lower temperature. Right panel: rmr at a measurement temperature of 15C, by individual (id).

```r
mouse15 <- subset(mouse, temp == 15)
mouse15$id <- with(mouse15, reorder(id, rmr, median, na.rm=T))

# A simple boxplot showing variation in rmr across and within individuals
boxplot(rmr ~  id, data=mouse15, xlab="id", ylab="rmr",ylim=c(0,0.6))
```

A practical way to test for multiple variables is to start with a simple model that includes the most important variables, and gradually increase model complexity. Every time we add one variable to a model, the significance can be tested with a likelihood ratio test. As discussed, we can use the built-in `anova` function, but probably more precise is the use of `KRmodcomp` from the `pbkrtest` package. We will demonstrate its use in this example but note that all conclusions will be the same when using `anova`.

We know for sure that resting metabolic rate should increase when the temperature decreases, so let's start with a model that only includes `temp` and appropriate random effects. Since each individual was measured multiple times (indicated by the `run` variable), we specify a `run` random effect nested within individual (`id`).

```r
mouse_m0 <- lmer(rmr ~ temp + (1|id/run), data=mouse )
```

In this first model we only allow the intercept to vary between individuals (`1|id/run`). A side-effect of the nested random effect is that we can quickly quantify the variation between individuals, as well as runs within individuals, using the `VarCorr` function:

```r
VarCorr(mouse_m0)

##  Groups   Name        Std.Dev.
##  run:id   (Intercept) 0.029825
##  id       (Intercept) 0.027776
##  Residual             0.071433
```

This demonstrates that the variation between individuals (`id`) is in fact similar to variation between runs within individuals (run:id). It is quite obvious that the fixed effect temp is significant in this model (inspect the Anova table yourself). We could add a random slope effect by id, but since there are few data points for every individual, this causes some problem that we would like to avoid (but feel free to try this for yourself).

Next, we can see in the summary statement of the mouse_m1 model that the slope and intercept are highly correlated. This can be a problem when we want to interpret those estimates. Note that the intercept of the model is defined as the value of the response variable (rmr) when the predictor (temp) is zero. This is not very informative since we did not measure metabolic rate at zero C. In a case like this, it is advisable to recenter the data so that the intercept is more meaningful. We will subtract 31C from temp, so that the intercept can now be interpreted as the metabolic rate at 31C. This is no choice of convenience, but rather has a biological justification: 31C approximates the lower critical limit of the thermal neutral zone (the range of temperature where metabolic rate does not vary). Below this temperature, metabolic rate increases to counter heat loss.

We will make a new variable and add it to the dataset, and refit the model.

```
mouse$temp31 <- mouse$temp - 31

mouse_m1 <- lmer(rmr ~ temp31 + (1|id/run), data=mouse )
```

Compare the summary statement from this model with mouse_m0, and also note that the two models are identical in goodness of fit (compare the AIC, for example). We are only expressing the intercept differently, allowing biological interpretation of the coefficients.

Next we include body mass as a predictor, as well as the interaction with temperature. It is well known that body mass is an important determinant of metabolic rate, so we should not be surprised by a strong significant effect:

```
mouse_m2 <- lmer(rmr ~ temp31*bm + (1|id/run), data=mouse )

library(pbkrtest)
KRmodcomp(mouse_m2, mouse_m1)

## F-test with Kenward-Roger approximation; computing time: 0.28 sec.
## large : rmr ~ temp31 * bm + (1 | id/run)
## small : rmr ~ temp31 + (1 | id/run)
##          stat    ndf    ddf F.scaling   p.value
## Ftest 13.429  2.000 57.968   0.98865 1.613e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The F-test shows that body mass is highly significant overall. The Anova shows that both the main effect and interaction are significant:

```
Anova(mouse_m2, test="F")

## Analysis of Deviance Table (Type II Wald F tests with Kenward-Roger df)
##
## Response: rmr
##                 F Df Df.res    Pr(>F)
## temp31    465.240  1 784.03 < 2.2e-16 ***
## bm         17.146  1  22.63 0.0004074 ***
## temp31:bm  10.021  1 784.03 0.0016075 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

An interesting side effect of including body mass as a fixed effect is that the estimated variance of the id is now much smaller. This makes sense because body mass varies between individuals, and thus including it in the model reduces the individual-level variation *after having accounted for body mass*.

Compare the standard deviation for id for the mouse_m2 model (see below) with that estimated for the mouse_m0 model above.

```
# Random effects for the mixed-effects model including body mass
VarCorr(mouse_m2)
```

```
##  Groups   Name        Std.Dev.
##  run:id   (Intercept) 0.029872
##  id       (Intercept) 0.012096
##  Residual             0.071025
```

Next we test whether individuals vary significantly in terms of their response to temperature, by adding a random slope effect. The models above only allowed the intercept (i.e. rmr at 31C) to vary between individuals. To test whether a more complex random effects structure is supported by the data, we fit two models and compare them with a likelihood ratio test (with anova), like so:

```
# Like mouse_m2, but with random slope (temp31)
mouse_m3 <- lmer(rmr ~ temp31*bm + (temp31|id/run), data=mouse )

anova(mouse_m3, mouse_m2)
```

```
## refitting model(s) with ML (instead of REML)
```

```
## Data: mouse
## Models:
## ..1: rmr ~ temp31 * bm + (1 | id/run)
## object: rmr ~ temp31 * bm + (temp31 | id/run)
##        Df     AIC     BIC logLik deviance  Chisq Chi Df Pr(>Chisq)
## ..1     7 -1962.4 -1929.3  988.2  -1976.4
## object 11 -1981.4 -1929.4 1001.7  -2003.4 26.998      4   1.99e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The test shows that the more complex model is better, providing evidence that individuals differ substantially in their response to temperature.

As always, when we have a significant main effect and interaction, it is not easy to see *how* they affect the response variable. It is most convenient to plot the model predictions with the visreg package. The following code makes Fig. 1.4. The figure shows that rmr increases with body mass (bm) and temperature (temp), and the temperature effect increases with body mass.

```
# Model predictions as a function of body mass, for the three temperatures.
# The argument 'partial=FALSE' turns off the partial residuals, producing a cleaner plot.
visreg(mouse_m3, "bm", by="temp31",overlay=TRUE, partial=FALSE, ylim=c(0,0.4))
```

Next, we test for the effect of three additional fixed effects on metabolic rate. As before, we test these effects one by one. The following code shows tests of sex, food and wheel on the rmr response variable.

```
# No effect of sex
mouse_m4 <- lmer(rmr ~ bm*temp31 + sex + (temp31|id/run), data=mouse)
KRmodcomp(mouse_m4, mouse_m3)
```

```
## F-test with Kenward-Roger approximation; computing time: 0.44 sec.
## large : rmr ~ bm * temp31 + sex + (temp31 | id/run)
## small : rmr ~ temp31 * bm + (temp31 | id/run)
##          stat     ndf     ddf F.scaling p.value
## Ftest  0.2303  1.0000 26.1885         1  0.6353
```

```
# We add 'wheel' only as an additive effect. The interaction cannot be estimated because
```

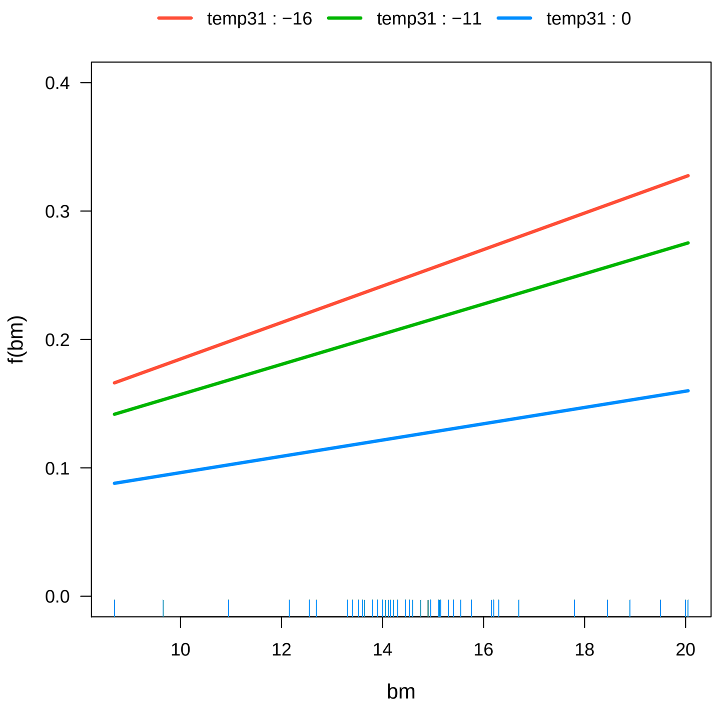

Figure 1.4: Plot fitted effects of the mouse m2 model, demonstrating significant effects of body mass and temperature on resting metabolic rate (rmr).

```
# the only cases where 'wheel=No' were at a temperature of 31C:
with(mouse, table(temp,wheel))

##      wheel
## temp   No Yes
##   15    0 288
##   20    0 288
##   31  144 144

mouse_m5 <- lmer(rmr ~ bm*temp31 + wheel + (temp31|id/run), data=mouse)
KRmodcomp(mouse_m5, mouse_m3)

## F-test with Kenward-Roger approximation; computing time: 0.45 sec.
## large : rmr ~ bm * temp31 + wheel + (temp31 | id/run)
## small : rmr ~ temp31 * bm + (temp31 | id/run)
##          stat     ndf     ddf F.scaling   p.value
## Ftest  46.551   1.000 740.237         1 1.857e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We finish with the standard anova table, showing p-values for the main effects.

```
Anova(mouse_m5, test="F")

## Analysis of Deviance Table (Type II Wald F tests with Kenward-Roger df)
##
## Response: rmr
##                   F Df Df.res    Pr(>F)
## bm          12.1872  1   31.45  0.001449 **
## temp31     106.7154  1   35.02 3.579e-12 ***
## wheel       46.5513  1 740.24 1.857e-11 ***
## bm:temp31    4.4921  1   36.61  0.040899 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Finally we visualize the individual-level variation in resting metabolic rate and its response to temperature, as estimated by the mixed-effects model. We use the `mouse_m2` model to visualize the predicted effects for every individual, ignoring effects of `wheel` and `food`. One interesting aspect of the data is that the variance in `rmr` was clearly higher for lower `temp`. The mixed-effects model is able to partially account for this as a result of a) the variation in body mass between individuals (which, as you recall, interacted with the temperature response), and b) the random variation in intercepts and slopes between individuals. This is a good result because it is otherwise very difficult to model changes in variance with response variables.

The following, more advanced, code produces Fig. 1.5.

```
# Make a dataframe with all combinations of temp and id, for run 1 only
pred_dfr <- expand.grid(temp31=c(-16,-11,0),
                        id=levels(mouse$id), run=1)

# Get average body mass by individual, merge onto pred_dfr
library(doBy)
bmid <- summaryBy(bm ~ id, FUN=mean, data=mouse, keep.names=TRUE)
pred_dfr <- merge(pred_dfr, bmid)

# Predict rmr for every id and temp, from the mouse_m3 model
# The default behaviour is to make predictions including the random
# effects (i.e. id and run:id)
pred_dfr$rmr_pred <- predict(mouse_m3, pred_dfr)

# Plot the data for run 1
with(subset(mouse, run==1), plot(jitter(temp),rmr, pch=21, bg="#BEBEBE99", ylim=c(0,0.6)))

# Add a prediction line for every individual. This is an alternative implementation,
# avoiding a for loop. The use of invisible() avoids lapply from printing output.
invisible(lapply(split(pred_dfr, pred_dfr$id), function(x)lines(x$temp31 + 31, x$rmr_pred)))
```

1.4 Example: blocked designs in the litter decomposition data

Blocked designs are often used in field experiments to account for known or suspected environmental gradients at the study site. By blocking the experimental design, the effect of the environmental gradient can be separated from the effect of the treatment(s) of interest increasing the ability to detect significant effects. Let's look at the effects of herbicide and profile on soybean litter decomposition as a function of agricultural management (herbicide usage), microenvironment, and time. In this experiment, herbicide treatments were applied at the level of whole plots, with both treatments represented within each of four blocks. Both levels of variety and profile were each represented within each plot, with six replicates of each treatment added to each plot. The data description can be found in Section A.19.

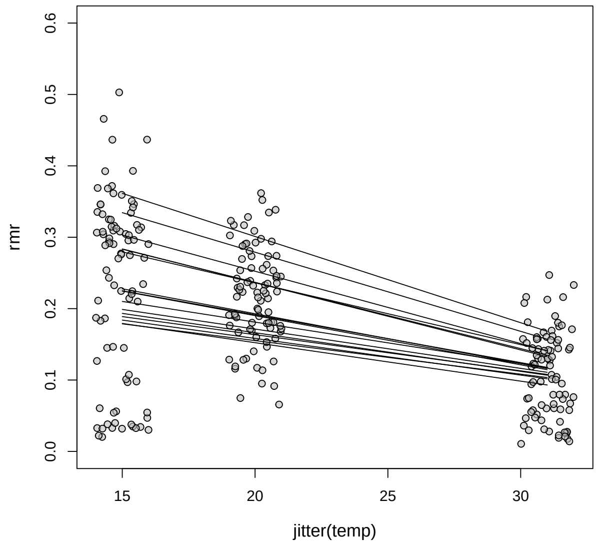

Figure 1.5: Measured (symbols) and predicted (lines) resting metabolic rate for the mouse dataset. Lines are predictions from the mouse m3 model for the individuals in the dataset, using the random variation in intercepts and slopes as well as the individual level variation in body mass.

The following code prepares the dataset for analysis, and produces Fig. 1.6.

```
# Read data
litter <- read.csv("masslost.csv")

# Make sure the intended random effects (plot and block) are factors
litter$plot <- as.factor(litter$plot)
litter$block <- as.factor(litter$block)

# Represent date as number of days since the start of the experiment
library(lubridate)
litter$date <- as.Date(mdy(litter$date))
litter$date2 <- litter$date - as.Date("2006-05-23")

# Quickly visualize the data to look for treatment effects
library(lattice)
bwplot(masslost ~ factor(date) | profile:herbicide, data=litter)
```

From inspecting Fig. 1.6, the buried litter appears to be decomposing faster than the surface litter (`masslost` is higher for `buried` compared to `surface`). If there are effects of herbicide (`gly` vs. `conv`), they are not immediately clear from the figure.

The blocking can be treated as a fixed effect or random effect. However, the design is unbalanced because some litter bags were lost, resulting in a variable number of litter bags recovered from each treatment. This affects the calculation of sums of squares, which vary depending on the order the terms are introduced to the model. It is therefore more appropriate to treat the blocking factor as a random effect, and use a mixed-effects model.

We first fit a simple linear model which ignores some details of the experimental design, and use block as a fixed effect. It is often very useful to start with a linear model, perhaps on subsets of the data, to gradually try to make sense of the data.

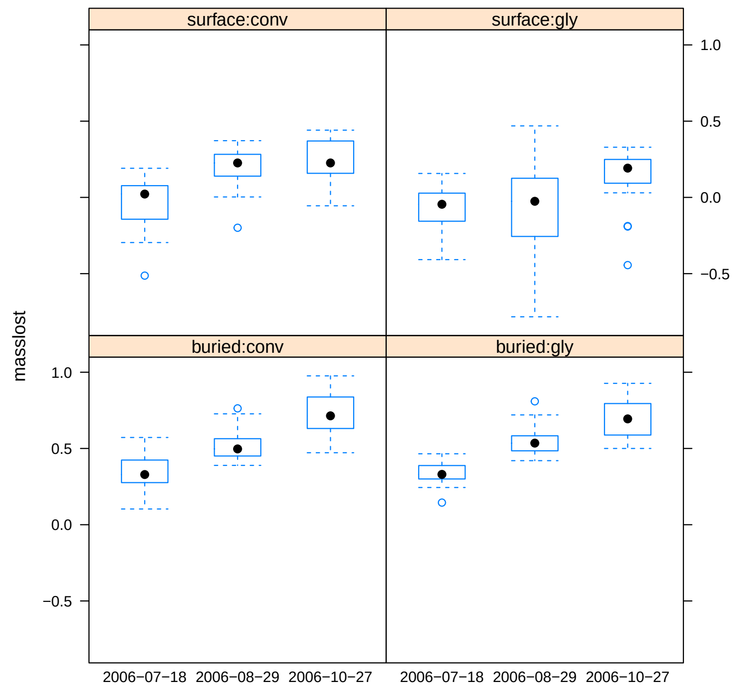

Figure 1.6: Proportion of litter mass lost from bags during field incubation as a function of microenvironment and herbicide program.

```
# Count the data to confirm that the design is unbalanced (ignore blocks for brevity)
ftable(xtabs(~ date2 + profile + herbicide, data=litter))

##                  herbicide conv gly
## date2 profile
## 56    buried                 22  22
##       surface                21  21
## 98    buried                 23  23
##       surface                20  20
## 157   buried                 19  16
##       surface                18  21

# Simple linear model with 'herbicide' as the first predictor in the model,
m1fix <- lm(masslost ~ date2 + herbicide * profile + block, data = litter)
anova(m1fix)

## Analysis of Variance Table
##
## Response: masslost
##                    Df  Sum Sq Mean Sq  F value    Pr(>F)
## date2               1  3.1696  3.1696 122.0249 < 2.2e-16 ***
## herbicide           1  0.4327  0.4327  16.6579 6.113e-05 ***
## profile             1 13.7225 13.7225 528.3018 < 2.2e-16 ***
## block               3  0.4674  0.1558   5.9987 0.0005891 ***
## herbicide:profile   1  0.3207  0.3207  12.3475 0.0005284 ***
## Residuals         238  6.1820  0.0260
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# ... or listing 'profile' first in the model.
m2fix <- lm(masslost ~ date2 + profile * herbicide + block, data = litter)
anova(m2fix)
```

```
## Analysis of Variance Table
##
## Response: masslost
##                    Df  Sum Sq Mean Sq  F value    Pr(>F)
## date2               1  3.1696  3.1696 122.0249 < 2.2e-16 ***
## profile             1 13.8335 13.8335 532.5749 < 2.2e-16 ***
## herbicide           1  0.3217  0.3217  12.3848 0.0005184 ***
## block               3  0.4674  0.1558   5.9987 0.0005891 ***
## profile:herbicide   1  0.3207  0.3207  12.3475 0.0005284 ***
## Residuals         238  6.1820  0.0260
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The sums of squares and p-values differ for `profile` and `herbicide` across the two fits (although they are still highly significant). Note that the order of the variables entered in the model matters because each next term is tested against a model that includes *all terms preceding it* (so-called Type-I tests). These standard tests with `anova` are sequential tests, which is perhaps not the most intuitive behaviour.

In many cases it is more intuitive to use so-called Type-II tests, in which each main effect is tested against a model that includes *all other terms*. We can use `Anova` (from the `car` package) to do this.

```
library(car)
Anova(m1fix, test="F")
```

```
## Anova Table (Type II tests)
##
## Response: masslost
##                    Sum Sq  Df  F value    Pr(>F)
## date2               3.7056   1 142.6601 < 2.2e-16 ***
## herbicide           0.3330   1  12.8198 0.0004157 ***
## profile            13.6594   1 525.8732 < 2.2e-16 ***
## block               0.4933   3   6.3311 0.0003793 ***
## herbicide:profile   0.3207   1  12.3475 0.0005284 ***
## Residuals           6.1820 238
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Anova(m2fix, test="F")
```

```
## Anova Table (Type II tests)
##
## Response: masslost
##                    Sum Sq  Df  F value    Pr(>F)
## date2               3.7056   1 142.6601 < 2.2e-16 ***
## profile            13.6594   1 525.8732 < 2.2e-16 ***
## herbicide           0.3330   1  12.8198 0.0004157 ***
## block               0.4933   3   6.3311 0.0003793 ***
## profile:herbicide   0.3207   1  12.3475 0.0005284 ***
## Residuals           6.1820 238
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

If the data were in fact balanced, the sequential (Type-I) and Type-II tests would be identical.

We also have not yet accounted for the fact that multiple litter bags were placed within single plots and that the 'herbicide' treatment was applied at the level of the plots, not the individual bags, which further complicates the analysis. Treating `block` and `plot` as random effects addresses both the imbalance and the hierarchical nature of the design.

In this example, we specify the nested nature of the data (plots within blocks) in the formula for the random effects as `(1|block/plot)`.

```
# fit model with random effects, plots nested within blocks
litter_m1 <- lmer(masslost ~ date2 + herbicide * profile + (1|block/plot),
           data = litter)
Anova(litter_m1, test="F")
```

```
## Analysis of Deviance Table (Type II Wald F tests with Kenward-Roger df)
##
## Response: masslost
##                          F Df  Df.res    Pr(>F)
## date2             142.317  1 235.489 < 2.2e-16 ***
## herbicide          12.738  1   2.997 0.0376270 *
## profile           525.467  1 235.564 < 2.2e-16 ***
## herbicide:profile  12.177  1 235.270 0.0005774 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# As you can see for yourself with anova(m1, m2), plot explains
# essentially zero variance.
litter_m2 <- lmer(masslost ~ date2 + herbicide * profile + (1|block),
           data = litter)
Anova(litter_m2, test="F")
```

```
## Analysis of Deviance Table (Type II Wald F tests with Kenward-Roger df)
##
## Response: masslost
##                          F Df Df.res    Pr(>F)
## date2             142.509  1 238.04 < 2.2e-16 ***
## herbicide          12.746  1 238.03 0.0004315 ***
## profile           526.217  1 238.03 < 2.2e-16 ***
## herbicide:profile  12.184  1 238.09 0.0005743 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Note that in the model, we used `date2` as a numeric variable, which assumes that the relationship between `masslost` and `date2` is more or less linear. Figure 1.7 shows that this is a resonable assumption However, in the case where you have a timeseries where no transformation exists to linearize the relationship, you will have to represent your time variable as a factor. We return to this issue in Section 1.5.1, after we treat another repeated measures example where time was continuous in Section 1.5.1.

Finally we visualize the fit, to make sense of the fitted coefficients, and to make sure we draw the right conclusions as to the direction of the significant effects. As we saw in Chapter 7, we can use `visreg` to quickly visualize the fitted model.

```
library(visreg)

# Because we have three fixed effects, we can make two plots to visualize
# certain pairs of combinations.
visreg(litter_m2, "date2", by="profile", overlay=TRUE)
```

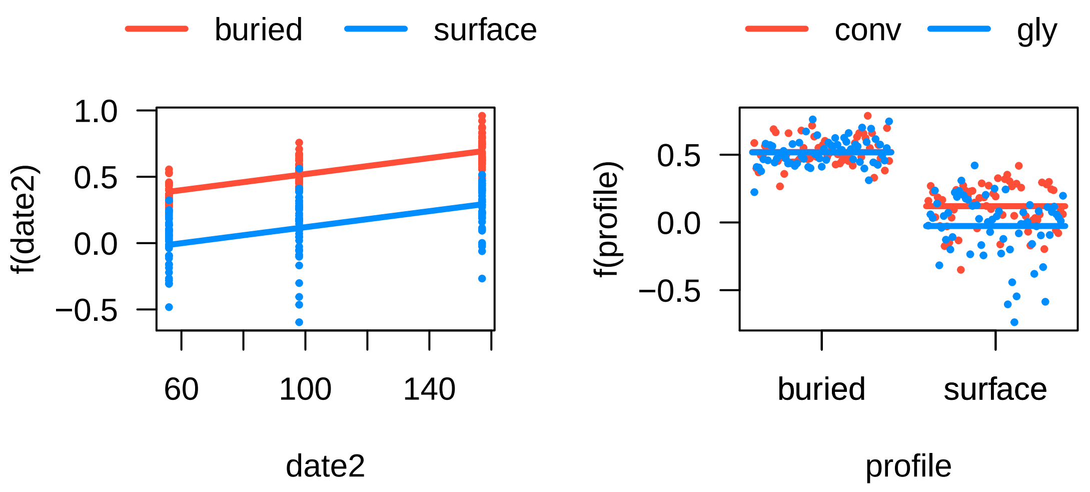

Figure 1.7: Visualization of the fixed effects of the mixed-effects model fit to the litter decomposition data.

```
visreg(litter_m2, "profile", by="herbicide", cond=list(date2=100), overlay=TRUE)
```

1.4.1 More about p-values

As we mentioned at the top of this chapter, p-values in `lme4` are a bit controversial. We here used the `Anova` function, but another option is to use the `LMERConvenienceFunctions` package. This package includes functions to estimate of p-values from *conservative* and *liberal* assumptions about random effect degrees of freedom, and thus gives a range of possible p-values.

```
library(LMERConvenienceFunctions)

# To use the function below, we must fit with ML, not REML.
litter_m2ml <- update(litter_m2, REML=FALSE)

# calculate upper- and lower-bounds on p-values
pamer.fnc(litter_m2ml)

##                    Df  Sum Sq Mean Sq  F value upper.den.df upper.p.val
## date2               1  3.1860  3.1860 124.7135          241       0e+00
## herbicide           1  0.4439  0.4439  17.3742          241       0e+00
## profile             1 13.6726 13.6726 535.2013          241       0e+00
## herbicide:profile   1  0.3153  0.3153  12.3406          241       5e-04
##                    lower.den.df lower.p.val expl.dev.(%)
## date2                       237      0e+00      13.1139
## herbicide                   237      0e+00       1.8269
## profile                     237      0e+00      56.2776
## herbicide:profile           237      5e-04       1.2976
```

Each main effect and interaction has two p-values: one assuming that each random effect accounts for one degree of freedom (`lower.p.val`) or no degrees of freedom (`upper.p.value`). The 'true' p-value will be somewhere in between these two bounds.

In this particular case, the results are highly significant (both lower and upper p-values are very small) because the effect size is quite large, but this will not always be the case.

Try this yourself Use `pamer.fnc` on the model above that contains random effects for both `Block` and `Plot`. What effect does this have on the degrees of freedom and p-value calculations? Why?

Another approach is to evaluate the importance of a term by comparing models that contain or do not contain that term using likelihood ratio tests (as already mentioned in Section 1.1.1). The recommended method for this is the `KRmodcomp` function (from the `pbkrtest` package), but the familiar `anova` can also be used (though p-values are approximate since likelihood ratios don't quite fit a chi-square distribution). You could also rely on model selection based on `AIC` (the lower the better). The following code shows all three approaches, and shows, as is often the case, that all methods show similar results.

```
# remove the interaction term from the model
litter_m2.int <- lmer(masslost ~ date2 + herbicide + profile + (1|block), data = litter)

# 1. anova
# Note that anova() will refit the models with ML (not REML) automatically,
# this is necessary when comparing models with different fixed or random effects terms.
anova(litter_m2, litter_m2.int)

## refitting model(s) with ML (instead of REML)

## Data: litter
## Models:
## ..1: masslost ~ date2 + herbicide + profile + (1 | block)
## object: masslost ~ date2 + herbicide * profile + (1 | block)
##        Df     AIC     BIC logLik deviance Chisq Chi Df Pr(>Chisq)
## ..1     6 -173.68 -152.64 92.838  -185.68
## object  7 -183.70 -159.16 98.848  -197.70 12.02      1  0.0005263 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

# 2. KRmodcomp
library(pbkrtest)
KRmodcomp(litter_m2ml, litter_m2.int)

## F-test with Kenward-Roger approximation; computing time: 0.09 sec.
## large : masslost ~ date2 + herbicide + profile + (1 | block) + herbicide:profile
## small : masslost ~ date2 + herbicide + profile + (1 | block)
##          stat     ndf     ddf F.scaling   p.value
## Ftest  12.184   1.000 238.087         1 0.0005743 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

# 3. AIC
AIC(litter_m2ml, litter_m2.int)

##               df       AIC
## litter_m2ml    7 -183.6958
## litter_m2.int  6 -141.5371
```

The model that includes the interaction provides the much better model fit. We can tell this by the significant p-value from `anova` and `KRmodcomp` result and by the lower AIC score for the model that includes an interaction. When an interaction is significant, the automatic follow-up question is 'what is the source of this interaction?'. Again inspecting Fig. 1.7, it appears bthat the herbicide treatments affected decomposition differently, but only on the surface of the soil. To further understand the nature of the interactions, it is useful to combine variables into a single variable, as the following example illustrates.

```
# Create a new variable containing the combinations of herbicide and profile.
# This new variable will have 4 levels
litter$combtrt <- paste(litter$herbicide, litter$profile, sep='-')

# Lump all observations for which the bags were buried into a single level,
# we now have just three levels in the new combined variable.
litter$combtrt[litter$profile == 'buried'] <- 'buried'

# Make this new variable into a factor
litter$combtrt <- as.factor(litter$combtrt)

# Fit a models using this new factor (3 levels) and a model without herbicide (2 levels)
litter_m3 <- lmer(masslost ~ date2 + combtrt + (1|block), data = litter, REML=FALSE)
litter_m3.herb <- lmer(masslost ~ date2 + profile + (1|block), data = litter, REML=FALSE)

# Compare the models by AIC (lower is 'better')
AIC(litter_m2ml, litter_m3, litter_m3.herb)

##                df       AIC
## litter_m2ml     7 -183.6958
## litter_m3       6 -185.6860
## litter_m3.herb  5 -163.6908
```

The model with the lowest AIC is `m3`, which is the model describing the relationship where herbicide affected decomposition rates only at the soil surface (because for that model, we combined all 'buried' litter samples into one level, regardless of the herbicide application).

1.5 Example: repeated measures in tree measurements

This example shows a very common use of mixed-effects models in repeated measurements. The basic idea is that when you have measurements on the same individuals (or plots, or some other unit) over time, you cannot treat the measurements as independent because that would be pseudo-replication, inflation of your sample size, and anti-conservative conclusions about significant effects.

We use data from the Hawkesbury Forest Experiment irrigation by fertilisation experiment (HFEIF, see Section A.16 for description of the data). In this experiment, sixteen plots of 72 *Eucalyptus saligna* trees were remeasured 20 times for height and diameter (although on a number of dates, not all trees were measured). Four treatments were applied (control, irrigated, fertilised, irrigated + fertilised). We ask in the following example whether tree height differs by treatment.

It is important that you recognize that the experimental unit in this example is the plot, not the tree, because the treatments were applied at a plot level. We therefore have to take into account the fact that trees are nested in plots, to avoid pseudoreplication.

```
# Read data, make proper date and make sure the intended factor variables are factors.
hfeif <- read.csv("HFEIFbytree.csv")
hfeif$Date <- as.Date(hfeif$Date)

# Make sure plot number (plotnr) is a factor; it is read in as a numeric variable.
hfeif$plotnr <- as.factor(hfeif$plotnr)
```

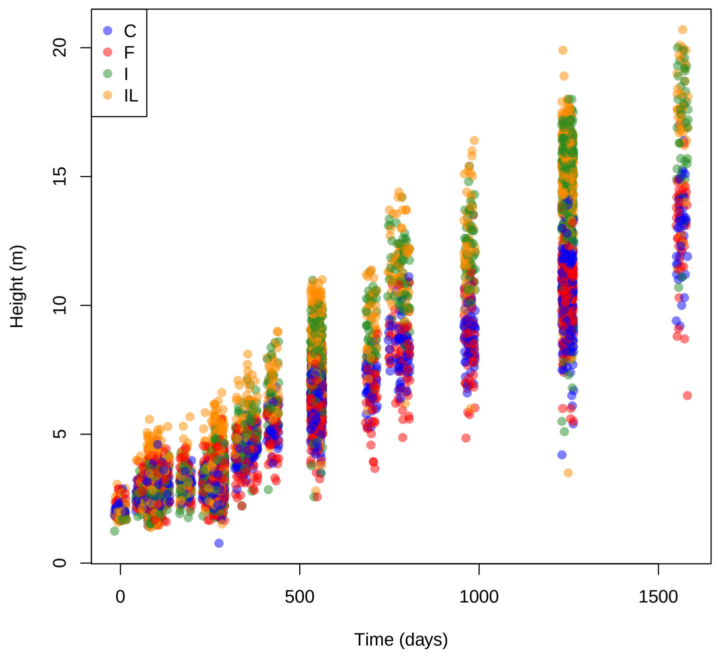

Figure 1.8: Plot tree height over time for the HFE irrigation x fertilisation experiment.

```r
# Days since start of experiment
# The as.numeric statement converts this into a simple numeric variable
hfeif$Time <- as.numeric(with(hfeif, Date - min(Date)))
```

Before we do anything, always explore the data with a few simple figures. Here we show tree height over time, colored by treatment (Fig. 1.8). The data show some separation between at least some of the treatments over time. Also note that the increase in height over time is perhaps not exactly linear, but we will ignore this in the remainder of the example (and further note that no straightforward transformation exists in this case).

Note the use of `jitter` in the example below, this adds some random noise to the Time variable to avoid excessive overlap of data points on each Date. We also use `sample` to randomly reorder the rows of the dataset to avoid the final treatment in the dataset to be plotted on top (this way, it is easier to see treatment differences).

The following code produces Fig. 1.8.

```r
# For the alpha() function (transparency)
library(scales)

# Set colours, transparent
palette(alpha(c("blue","red","forestgreen","darkorange"), 0.5))
with(hfeif[sample(nrow(hfeif)),],
     plot(jitter(Time,3), height, col=treat, pch=19,
                 xlab="Time (days)", ylab="Height (m)"))
legend("topleft", levels(hfeif$treat), pch=19, col=palette())
```

Again, the reason we want to use mixed-effects models in this case is because we want to use the correct number of degrees of freedom to test for the treatment effect. If you are not sure what that should be, let's start with a linear model on the data from just one date, when we have averaged the data by plot (our experimental unit). In this case we can simply use `lm`, as follows.

The following example produces Fig. 1.9.

```
# Take subset of data at last Date
hfeif_last <- subset(hfeif, Date == max(Date))

# Average all variables by plot (and include the 'treat' factor variable in the result)
library(doBy)
hfeif_last_plot <- summaryBy(. ~ Date + plotnr, data=hfeif_last,
                             FUN=mean, na.rm=TRUE,
                             id=~treat, keep.names=TRUE)

# Linear model with treatment only.
lm_last <- lm(height ~ treat, data=hfeif_last_plot)

# Note that height is highly significant, and that we use 3 numerator df
# to test for treatment effects
anova(lm_last)

## Analysis of Variance Table
##
## Response: height
##           Df Sum Sq Mean Sq F value    Pr(>F)
## treat      3 74.160 24.7201  19.137 7.226e-05 ***
## Residuals 12 15.501  1.2917
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

# A quick visualization of the fitted model shows much taller trees
# in I and IL,
library(visreg)
visreg(lm_last, "treat", xlab="Treatment", ylab="Tree height (m)")
```

The above shows 3 numerator degrees of freedom in the F-test, which makes sense because we have 4 levels of our treatment (thus df = 4 - 1). You can do no such simple check for the denominator degrees of freedom, but it's a useful check nonetheless.

To account for the repeated measures nature of the data as well as the fact that the experimental unit is the plot, not the tree, all we need is to specify the plot as the random effect. We will fit two models, one without and one with the interaction between `Time` and `treat`, and again use `Anova` (from the `car` package) to test for significant effects.

Note that we specify `Time` as a numeric variable, which assumes that the relationship between `height` and `Time` is more or less linear, which according to Figure 1.8 is a reasonable assumption. We return to this issue in Section 1.5.1.

```
# Effect of treatment on intercept only.
lmeif1 <- lmer(height ~ treat + Time + (1|plotnr), data=hfeif)
Anova(lmeif1)

## Analysis of Deviance Table (Type II Wald chisquare tests)
##
## Response: height
##           Chisq Df Pr(>Chisq)
## treat    27.752  3  4.095e-06 ***
## Time  48339.942  1  < 2.2e-16 ***
```

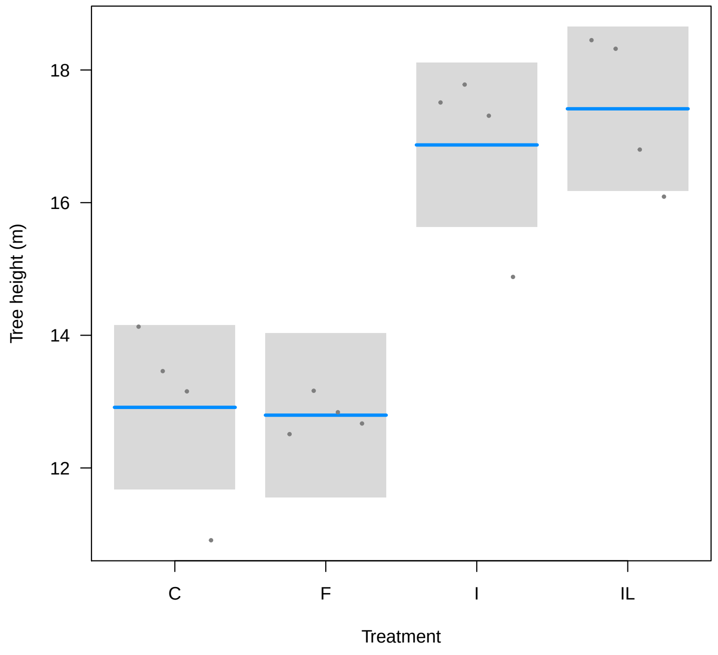

Figure 1.9: Simple visualization of fitted linear model (with lm) of tree height on the last date of the HFE IF data.

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

# Effect of treatment on intercept and slope (i.e. main effect + interaction)
lmeif2 <- lmer(height ~ treat*Time + (1|plotnr), data=hfeif)
Anova(lmeif2)

## Analysis of Deviance Table (Type II Wald chisquare tests)
##
## Response: height
##                 Chisq Df Pr(>Chisq)
## treat          27.389  3  4.879e-06 ***
## Time        72342.647  1  < 2.2e-16 ***
## treat:Time   3208.442  3  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

# A likelihood ratio tests shows the interaction is highly significant
anova(lmeif1, lmeif2)

## refitting model(s) with ML (instead of REML)

## Data: hfeif
## Models:
## object: height ~ treat + Time + (1 | plotnr)
## ..1: height ~ treat * Time + (1 | plotnr)
##        Df   AIC   BIC logLik deviance  Chisq Chi Df Pr(>Chisq)
## object  7 22654 22702 -11320    22640
```

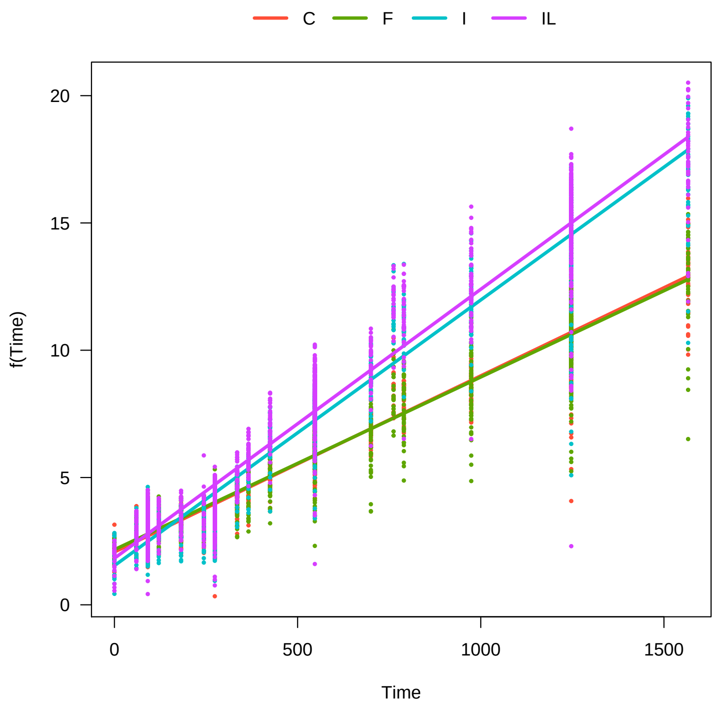

Figure 1.10: Visualized effect of Time on height, by treatment for the HFE IxF dataset, fitted with a linear mixed-effects model.

```
## ..1     10 20054 20122 -10017    20034 2605.9      3  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Try this yourself Repeat the likelihood-ratio test with the more accurate `KRmodcomp` function from the `pbkrtest` package.

Note that in the above, we have correctly used 3 numerator degrees of freedom to test for the effect of treatment on height. The individual `Anova` statements summarize the significance of the fixed effects in each model, whereas the `anova` of the two models uses a likelihood-ratio test on the two models. In this case, it effectively tests for significance of the interaction (because the only difference between the two models was the inclusion of the `treat` by `Time` interaction in `lmeif2`). The interaction is overwhelmingly significant.

The final step is to try to understand this interaction, how large is the effect size, and which direction does it point? The time by treatment interaction is significant, but in which way? It is never sufficient in an analysis to state that an interaction was 'significant', we must make more sense of it. One simple approach is to use the `visreg` package to visualize the fit (see Fig. 1.10).

```
library(visreg)
visreg(lmeif2,"Time", by="treat", overlay=TRUE)
```

In this case it is abundantly clear that irrigated (I) and irrigated + fertilised (IL) have a steeper slope of height with Time (that is, they have a faster height growth), compared to control (C) and fertilised (F). If there was no significant interaction (or a small effect size), the lines would be parallel to each other.

We can further look at the p-values for the individual effects (slopes and intercepts by treatment). Note that p-values in the `summary` statement are only computed if we have loaded the `lmerTest` package before fitting the model. Consider this example,

```
# Loading this package first affects both summary and anova methods
library(lmerTest)

# ... we must refit the model after loading lmerTest
lmeif2 <- lmer(height ~ treat*Time + (1|plotnr), data=hfeif)

# Print just the coefficients table from the summary
summary(lmeif2)$coefficients
```

```
##                  Estimate   Std. Error         df      t value     Pr(>|t|)
## (Intercept)   2.073764669 2.992535e-01   12.32252    6.9297919 1.377196e-05
## treatF        0.093074081 4.233121e-01   12.33460    0.2198711 8.295684e-01
## treatI       -0.535359455 4.232604e-01   12.32858   -1.2648466 2.293128e-01
## treatIL      -0.246322580 4.232390e-01   12.32609   -0.5819940 5.710624e-01
## Time          0.006921244 6.399533e-05 6453.01255 108.1523211 0.000000e+00
## treatF:Time  -0.000141759 9.111223e-05 6453.02451   -1.5558723 1.197875e-01
## treatI:Time   0.003514427 9.095049e-05 6453.01005   38.6411001 0.000000e+00
## treatIL:Time  0.003645960 9.085136e-05 6453.02770   40.1310445 0.000000e+00
```

Try this yourself The `lmerTest` package also modifies the `anova` function, so that it calculates p-values for a fitted model with `lmer`. Compare `anova(lmeif2)` with `Anova(lmeif2)`, these will rarely be exactly the same as they use different methods to approximate the degrees of freedom of the random effects.

Looking at the interaction terms (treat:Time),the summary table shows that `treatF:Time` is not significantly different from the first level (`treatC:Time`), that is, there is no difference between fertilized and control in terms of the interaction with Time. But both irrigated (I) and irrigated + fertilised (IF) are highly significant, again, this comparison is in relation to the first level of the factor (control, C).

Although there is a significant main effect of treatment, none of the levels are actually different from the first (the control). This shows that the intercept itself is different from zero, but the treatments are not actually different in terms of the intercept. This makes sense, because seedlings were planted at time zero before any treatment was applied.

1.5.1 Repeated measures: is time numeric or factor?

In both examples in this chapter where we used time as a predictor in our models, we treated time as a continuous (numeric) variable. This was appropriate in both cases because the relationship between the dependent variable (`masslost` or `height`) showed a nearly linear relationship with time, allowing us to estimate and interpret an intercept and a slope of the variable with time. In the example with the tree height measurements, the slope of `height` with time can actually be interpreted as the height growth rate.

But there are many cases in which it would be more appropriate to use time as a factor variable. These include cases where the relationship is highly non-linear and cannot be transformed, or you only have two or three dates of measurements. The example below shows a simple example for the latter case. (see also Fig. 1.11).

```
# A repeated measures example with only two dates of measurement.
# Though it is possible to have time as a continuous variable, it is much
# more useful to code it as a factor.
```

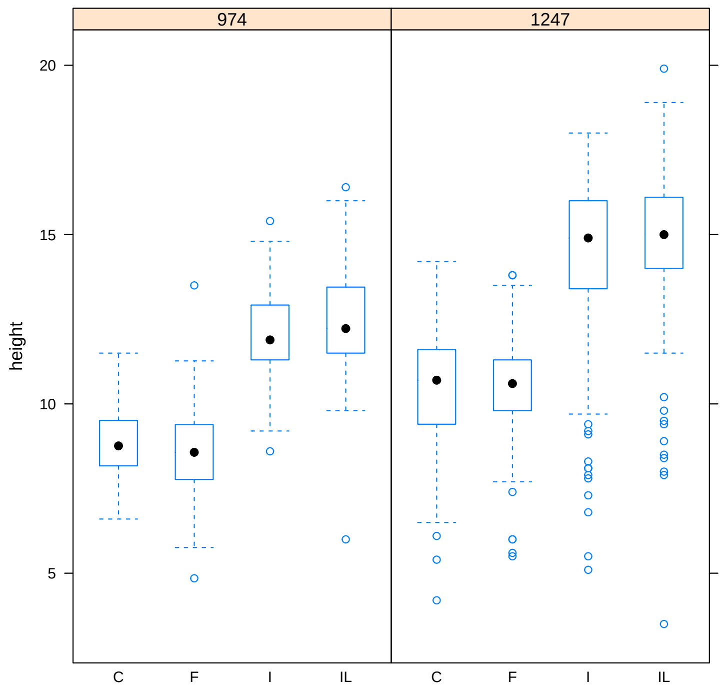

Figure 1.11: Boxplots of height vs. treatment and time (days since start of experiment, shown in the panel label) for a subset of the HFE IF data.

```
# We take a subset of hfeif.
hfeif2 <- subset(hfeif, Date %in% as.Date(c("2010-09-01","2011-06-01")))

# Convert Time to a factor
hfeif2$Time_fac <- as.factor(hfeif2$Time)

# As before, we can quickly use bwplot to inspect the data
library(lattice)
bwplot(height ~ treat | Time_fac, data=hfeif2)
```

Try this yourself Repeat the above example, but using the entire dataset (rather than a subset of the data for two dates). Inspect the model with `Anova`, and also look at the `summary` statement.

Let's fit the mixed-effects model on this small subset of the data to test whether a) treatment affects height, b) there is an effect of time on height, c) there is an interaction (i.e. height response to treatment depends on time).

```
# Fit the model
lmeif4 <- lmer(height ~ treat*Time_fac + (1|plotnr), data=hfeif2)

# Overall significance shows no interaction
Anova(lmeif4, test="F")

## Analysis of Deviance Table (Type II Wald F tests with Kenward-Roger df)
##
```

```
## Response: height
##                       F Df Df.res    Pr(>F)
## treat           23.0111  3     12  2.89e-05 ***
## Time_fac       359.1285  1   1333 < 2.2e-16 ***
## treat:Time_fac   2.8048  3   1333   0.03859 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Try this yourself Use `visreg` to visualize the fit, and compare it to the box plots produced above.

1.6 Example: repeated measures in leaf photosynthesis

In this example we look at another case of repeated measurements, this time for measurements of leaf photosynthesis at the EucFACE. Measurements were repeated during four campaigns in 2013 (labelled simply as A-D), in all six rings at the EucFACE (three of which have an ambient CO_2 concentration, and three an elevated CO_2 concentration), on a few trees per ring. The random effects naturally follow from the experimental design, as measurements were performed on trees nested within rings.

In this example we are particulary interested in comparing the four dates with each other - we would like to know not only if leaf photosynthesis was enhanced by elevated CO_2, but also whether this enhancement differed between the campaigns.

We start out with a simple barplot showing averages by campaign and treatment. The following code produces Fig. 1.12. Note that the error bars cannot be used for judging differences between treatments, as these are simply calculated across all the individual data points.

```
eucgas <- read.csv("eucface_gasexchange.csv")

library(sciplot)
bargraph.CI(Date, Photo, CO2, data=eucgas, legend=TRUE,
            ylab="Photo")
```

Next, we fit the mixed-effects model. We have just two fixed effects in the model: `Date` and `CO2`, and the random effects structure is tree within ring (`1|Ring/Tree`). We can test whether we need to include a Date x CO2 interaction by fitting two models and comparing them with a likelihood ratio test, as before.

We also load the `lmerTest` package here because we would like to inspect the p-values for the individual coefficients printed in the `summary` statement, in what follows.

```
library(lmerTest)

eucgas_m0 <- lmer(Photo ~ Date + CO2 + (1|Ring/Tree), data=eucgas)
eucgas_m1 <- lmer(Photo ~ Date * CO2 + (1|Ring/Tree), data=eucgas)

# Compare the two models - this tests for the significance of Date:CO2
library(pbkrtest)
KRmodcomp(eucgas_m0, eucgas_m1)

## F-test with Kenward-Roger approximation; computing time: 0.17 sec.
## large : Photo ~ Date * CO2 + (1 | Ring/Tree)
## small : Photo ~ Date + CO2 + (1 | Ring/Tree)
##          stat     ndf     ddf F.scaling p.value
## Ftest  2.2889  3.0000 57.2479   0.99996 0.08806 .
```

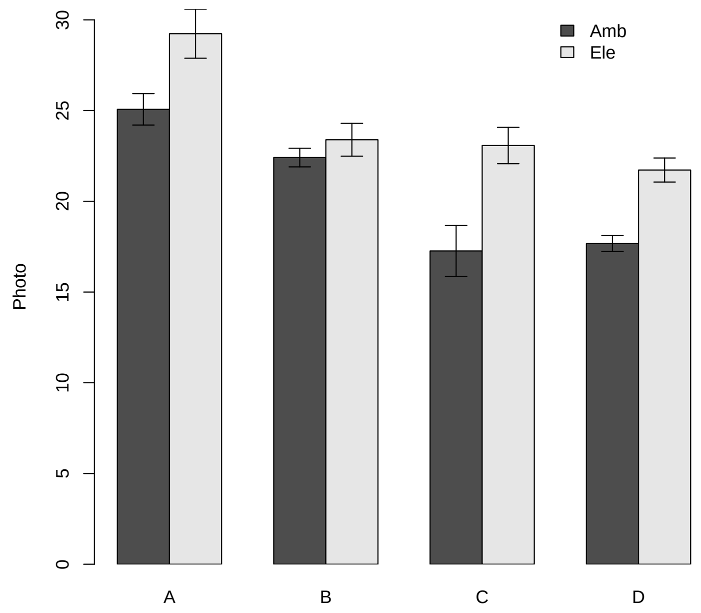

Figure 1.12: Average net leaf photosynthesis (Photo) by Date and CO2 treatment for the `eucgas` dataset.

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The F-test shows that the interaction is at best marginally significant (p = 0.088). Make sure that you understand the difference between these two models: the interaction allows for a different CO2 response at different dates. The model fits are visualized in Fig. 1.13, produced by the following code.

```
visreg(eucgas_m0, "Date", by="CO2", overlay=TRUE)
visreg(eucgas_m1, "Date", by="CO2", overlay=TRUE)
```

Note: if you are using an older version of `visreg`, you will see some warnings printed that you can ignore.

Comparing the two panels in Fig. 1.13, we are not quite yet convinced that the CO2 response was the same for all dates. For Date B, the response seems particularly small. Also, although the F-test above did not give us evidence for a significant interaction, the model with the interaction does have a lower AIC, suggesting it does improve the model somewhat:

```
AIC(eucgas_m0, eucgas_m1)

##             df      AIC
## eucgas_m0  8 437.1154
## eucgas_m1 11 427.9704
```

Before we continue, it is a good idea as always to check the model diagnostics. The following code makes a plot (Fig. 1.14) of the residuals versus fitted, and a quantile-quantile plot of the residuals. These diagnostics look quite good, so we are happy to continue.

```
plot(fitted(eucgas_m0), residuals(eucgas_m0))
abline(h=0)
```

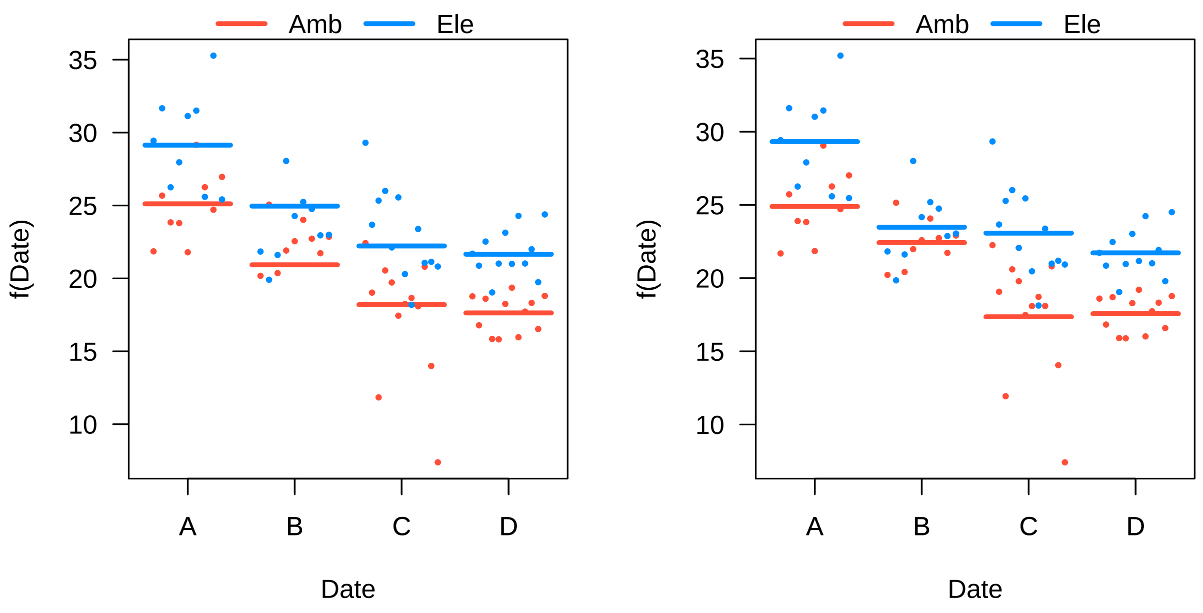

Figure 1.13: A model for net leaf photosynthesis without (left) and with (right) an interaction between Date and CO2 treatment.

```
library(car)
qqPlot(residuals(eucgas_m0))
```

To address our question, we are interested to find out whether there was a significant CO2 response on each of the four dates. With the standard way that **R** codes the factor levels, this is not straightforward to address. Recall that each factor level is tested against the first level, as we can see in the coefficients printed as part of the `summary` table. Here is a subset of the output of `summary(eucgas_m1)`:

```
##                Estimate Std. Error       df    t value   Pr(>|t|)
## (Intercept)  24.8925064   1.037901 73.47517 23.9835135 < 2.22e-16 ***
## DateB        -2.4697297   1.369405 47.05959 -1.8035060   0.077713   .
## DateC        -7.5340656   1.283741 48.75124 -5.8688380 3.7846e-07 ***
## DateD        -7.3220721   1.283741 48.75124 -5.7037007 6.7726e-07 ***
## CO2Ele        4.4314124   1.469861 72.57121  3.0148506   0.003541  **
## DateB:CO2Ele -3.3755079   1.921530 43.25287 -1.7566769   0.086051   .
## DateC:CO2Ele  1.2836021   1.808643 46.55320  0.7097047   0.481427
## DateD:CO2Ele -0.2763754   1.808643 46.55320 -0.1528082   0.879210
```

In this table, `(Intercept)` is the estimated `Photo` for the first Date (A), for the first level of CO2 (`Amb`). All other values are expressed relative to this value. Thus, `CO2Ele` tests the CO2 effect for the first Date. It shows that for Date A, `Photo` was 4.43 units higher in elevated compared to the ambient treatment. The next three parameters (`DateB:CO2Ele` and so on) are expressed relative to the first date, so that `DateB:CO2Ele` tests the difference in CO2 response between Date B and Date A.

This table therefore does not address our question: was CO2 significant on each date? A straightforward way to compute the right tests is to refit the model without an intercept, so that the individual coefficients are not tested against the first level, but against zero.

```
# Refit the model, without an intercept ("-1"), and including CO2 only as
# the interaction
eucgas_m2 <- lmer(Photo ~ Date  + Date:CO2 -1 + (1|Ring/Tree), data=eucgas)
```

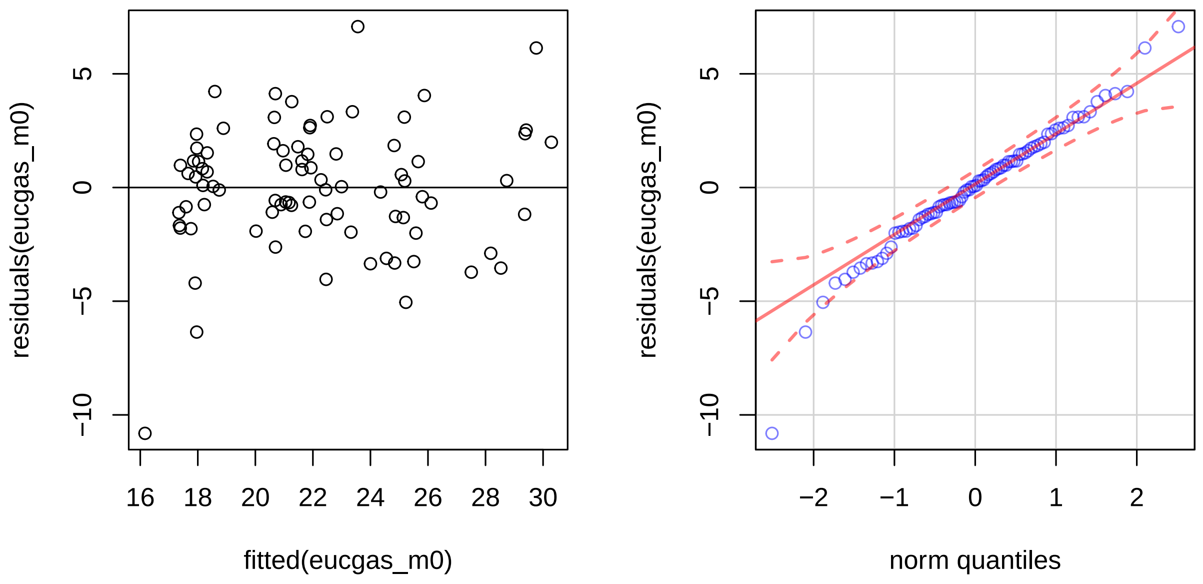

Figure 1.14: Diagnostic plots for the eucgas_mo model.

```
# Note that this model is exactly the same as before - see the AIC for example
AIC(eucgas_m1, eucgas_m2)
```

```
##           df      AIC
## eucgas_m1 11 427.9704
## eucgas_m2 11 427.9704
```

The new model expresses the coefficients very differently, as we can see from this extract from `summary(eucgas_m2)`:

```
##                 Estimate Std. Error       df    t value   Pr(>|t|)
## DateA          24.892506  1.0379007 73.47516 23.9835135 < 2.22e-16 ***
## DateB          22.422777  1.0386388 73.12648 21.5886184 < 2.22e-16 ***
## DateC          17.358441  0.9050977 67.83184 19.1785280 < 2.22e-16 ***
## DateD          17.570434  0.9050977 67.83184 19.4127496 < 2.22e-16 ***
## DateA:CO2Ele    4.431412  1.4698613 72.57121  3.0148506  0.0035410  **
## DateB:CO2Ele    1.055904  1.4703826 72.37632  0.7181155  0.4749978
## DateC:CO2Ele    5.715014  1.2814115 66.59206  4.4599370 3.2301e-05 ***
## DateD:CO2Ele    4.155037  1.2814115 66.59206  3.2425470  0.0018526  **
```

This time, the first four rows give the estimates for the four dates in the ambient CO2 treatment, together with the p-values of a test against zero (not very meaningful in this case). The next four rows give the actual CO2 response, that is, the difference between elevated and ambient CO2 for each of the four dates. Since we loaded the `lmerTest` package before fitting this model, the p-values are computed for each of these effects. The results show that, as we suspected, there was no significant CO2 effect on Date B ($p = 0.47$).

Finally we show another method to test individual coefficients against zero, using the `glht` function from the `multcomp` package. We have met this package before for performing Tukey tests on fitted models.

To test coefficients against zero, we can do:

```
library(multcomp)
summary(glht(eucgas_m2, linfct=c("DateA:CO2Ele = 0",
                                 "DateB:CO2Ele = 0",
                                 "DateC:CO2Ele = 0",
                                 "DateD:CO2Ele = 0")))

##
##   Simultaneous Tests for General Linear Hypotheses
##
## Fit: lme4::lmer(formula = Photo ~ Date + Date:CO2 - 1 + (1 | Ring/Tree),
##     data = eucgas)
##
## Linear Hypotheses:
##                   Estimate Std. Error z value Pr(>|z|)
## DateA:CO2Ele == 0    4.431      1.470   3.015  0.01020 *
## DateB:CO2Ele == 0    1.056      1.470   0.718  0.91947
## DateC:CO2Ele == 0    5.715      1.281   4.460  < 1e-04 ***
## DateD:CO2Ele == 0    4.155      1.281   3.243  0.00472 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- single-step method)
```

Note that the p-values will be different from the ones computed above because a) `lmerTest` approximates the degrees of freedom and uses a t-test, whereas `glht` relies on a normal approximation (the two will be closer together for larger sample sizes), and b) `glht` adjusts the p-values for multiple comparisons. This time, though, the conclusions are no different using either method.

Using the `glht` function, we can also test various parameter combinations quite easily. This final example shows how we can test whether the CO2 response was different on Date C from Date D:

```
summary(glht(eucgas_m2, linfct="DateC:CO2Ele - DateD:CO2Ele = 0"))

##
##   Simultaneous Tests for General Linear Hypotheses
##
## Fit: lme4::lmer(formula = Photo ~ Date + Date:CO2 - 1 + (1 | Ring/Tree),
##     data = eucgas)
##
## Linear Hypotheses:
##                                Estimate Std. Error z value Pr(>|z|)
## DateC:CO2Ele - DateD:CO2Ele == 0    1.560      1.657   0.942    0.346
## (Adjusted p values reported -- single-step method)
```

From this test we can conclude that there is no evidence to suggest that CO2 increased photosynthesis more on Date C compared to Date D.

1.7 Example: analysis of count data with the groundcover data (glmer)

We have seen how to fit generalised linear models (Section 7.5) and linear mixed models (above). Once you know how to fit these models in **R**, fitting GLMMs is fairly easy using the `glmer` function in the `lme4` package.

The EucFACE ground cover dataset (see Section A.20) contains estimates of plant and litter cover within

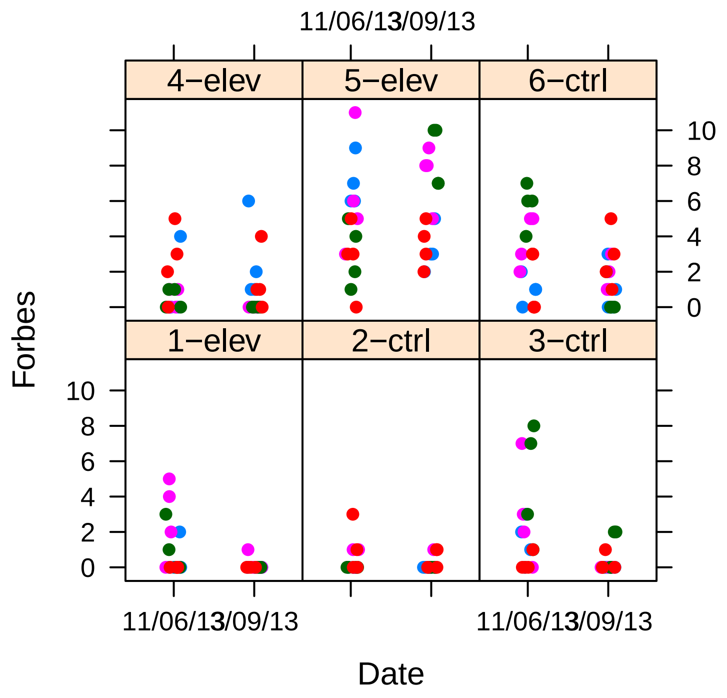

Figure 1.15: Counts of forbs within plant communities exposed to ambient or elevated carbon dioxide concentrations at two dates. Points represent estimates within subplots, subplots common to the same plot within each ring have a common colour.

the rings of the EucFACE experiment, evaluating forest ecosystem responses to elevated CO_2, on two dates. There are six rings (`Ring`), three for each treatment (`Trt`; ambient and elevated CO_2). Within each ring are four plots (`Plot`) and within each plot are four 1m by 1m subplots (`Sub`). Here we will test for an interaction between `Trt` and `Date` on ground cover measurements of `Forbes` (these are count data).

The following code produces Fig. 1.15, showing a plot of the raw groundcover data.

```
# read data and convert random effects to factors
eucface <- read.csv("eucfaceGC.csv")
eucface$Ring <- as.factor(paste(eucface$Ring, eucface$Trt, sep='-'))
eucface$Plot <- as.factor(eucface$Plot)
eucface$Sub <- as.factor(eucface$Sub)

# load packages
library(lme4)
library(lattice)

# A quick plot to visualize ground cover by Date and Ring.
# Colours represent 'Plot'.
xyplot(Forbes~Date|Ring,groups=Plot,data=eucface,pch=16,jitter.x=T)
```

Since the data are count data, it is usually appropriate to use the Poisson distribution. The following code fits a `glmer` with the `poisson` error family, and produces diagnostic plots in Fig. 1.16.

To usual way to decide on the appropriate family is to inspect the diagnostic plots, especially a plot of residuals versus the fitted values. In the following, we fit the model three times, first with normal

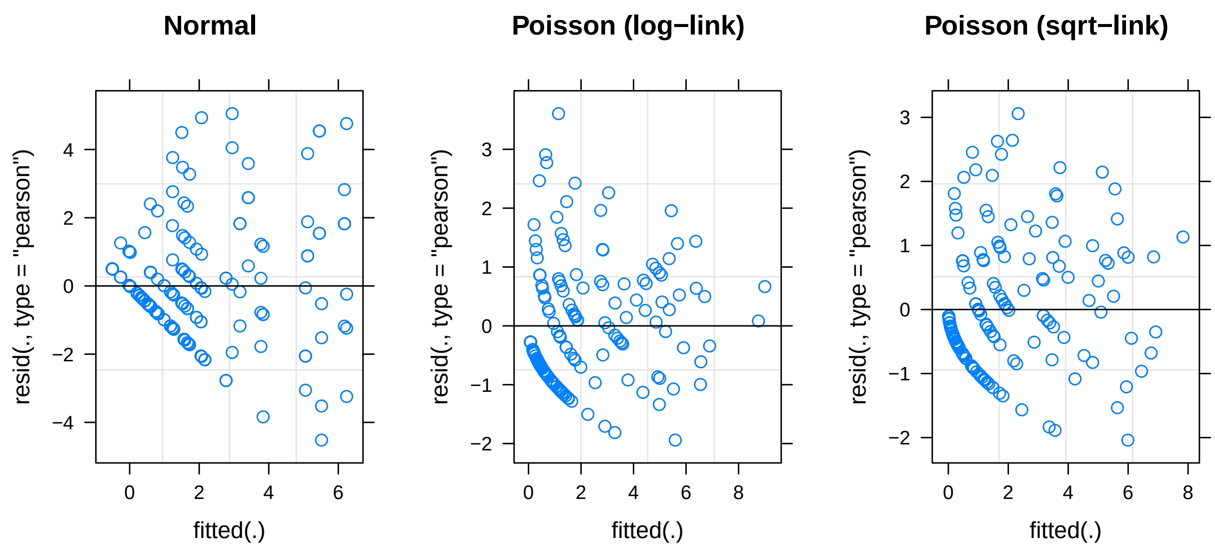

Figure 1.16: Diagnostic plots for three model fits of the EucFACE groundcover data.

errors (using `lmer`), then with Poisson errors (the distribution we expect to be appropriate), and a Poisson distribution with square-root link function.

The following code produces three diagnostic plots (Fig. 1.16).

```r
forb.norm <- lmer(Forbes~Date*Trt+(1|Ring/Plot/Sub),
                  data=eucface)

forb.pois <- glmer(Forbes~Date*Trt+(1|Ring/Plot/Sub),
                  family=poisson, data=eucface)

forb.pois.sqrt <- glmer(Forbes~Date*Trt+(1|Ring/Plot/Sub),
                       family=poisson('sqrt'), data=eucface)

# The standard plot() on an (g)lmer object makes a 'grid-based'
# plot. To arrange plots side-by-side we have to use gridExtra, like so.
library(gridExtra)
p1 <- plot(forb.norm, main="Normal")
p2 <- plot(forb.pois, main="Poisson (log-link)")
p3 <- plot(forb.pois.sqrt, main="Poisson (sqrt-link)")
grid.arrange(p1,p2,p3,ncol=3)
```

Note that the syntax of `glmer` is identical to that of `lmer`, with the exception of the `family` argument.

Inspecting Fig. 1.16, the residuals look better for the models fit with the Poisson family compared to the gaussian error (as assumed by `lmer`). The residuals improve a bit more when using the `sqrt` link function in the Poisson family.

Note that the fit of the model may be improved further by using one of the other plant cover variables as a covariate or by incorporating other data from the site, but for the purpose of this chapter the fit is good enough.

Something else to consider when it comes to poisson errors is that overdispersion can result in the underestimation of error terms for the model coefficients. Overdispersion occurs when you have a large number of zeros in the data and/or an important predictor is not accounted for. Testing hypotheses from models where overdispersion is evident is dangerous as the probability of Type I error is increased. A good model fit should result in the ratio of residual deviance to degrees of freedom being close to one.

```
# Calculate the residual deviance
sum(resid(forb.pois.sqrt, type='pearson')^2)

## [1] 171.408

# Excerpt of the model summary, showing only the AIC and related parameters.
# From here we learn that the df.resid is 185
summary(forb.pois.sqrt)$AICtab

##       AIC       BIC    logLik  deviance  df.resid
##  612.5734  635.3758 -299.2867  598.5734  185.0000
```

Overdispersion does not appear to be a problem here as the ratio (171.4 / 185) is less than one. If it was, we could use a quasipoisson family as for GLMs (but unfortunately `glmer` does not support that family). It has also been suggested that including individual-level random effects in the model could alleviate the problem of overdispersion. We try this in the following example.

```
# Create a variable for individual-level random effects
eucface$Ind <- as.factor(1:nrow(eucface))

# fit the model and look at the model summary
forb.pois.ind <- glmer(Forbes~Date*Trt+(1|Ring/Plot/Sub/Ind),
                       family=poisson(sqrt), data=eucface)

# The estimated variances of the random effects
VarCorr(forb.pois.ind)

##  Groups                Name        Std.Dev.
##  Ind:(Sub:(Plot:Ring)) (Intercept) 0.26215
##  Sub:(Plot:Ring)       (Intercept) 0.10949
##  Plot:Ring             (Intercept) 0.35224
##  Ring                  (Intercept) 0.57035

# Inspect the entire summary yourself:
# summary(forb.pois.ind)
```

The random effects block indicates that a small amount of variance is accounted for by the individual level random effects. This is expected as we did not observe overdispersion.

We can use `Anova` function from the `car` package to calculate significance associated with the main effects and interaction. Note that is is only one way of many to calculate p-values, as we discussed above linear mixed models.

```
library(car)
Anova(forb.pois.ind)

## Analysis of Deviance Table (Type II Wald chisquare tests)
##
## Response: Forbes
##            Chisq Df Pr(>Chisq)
## Date     16.8793  1  3.983e-05 ***
## Trt       0.3659  1    0.54527
## Date:Trt  4.4273  1    0.03537 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

There appears to be a significant interaction between `Date` and `Trt`. Again, when an interaction is

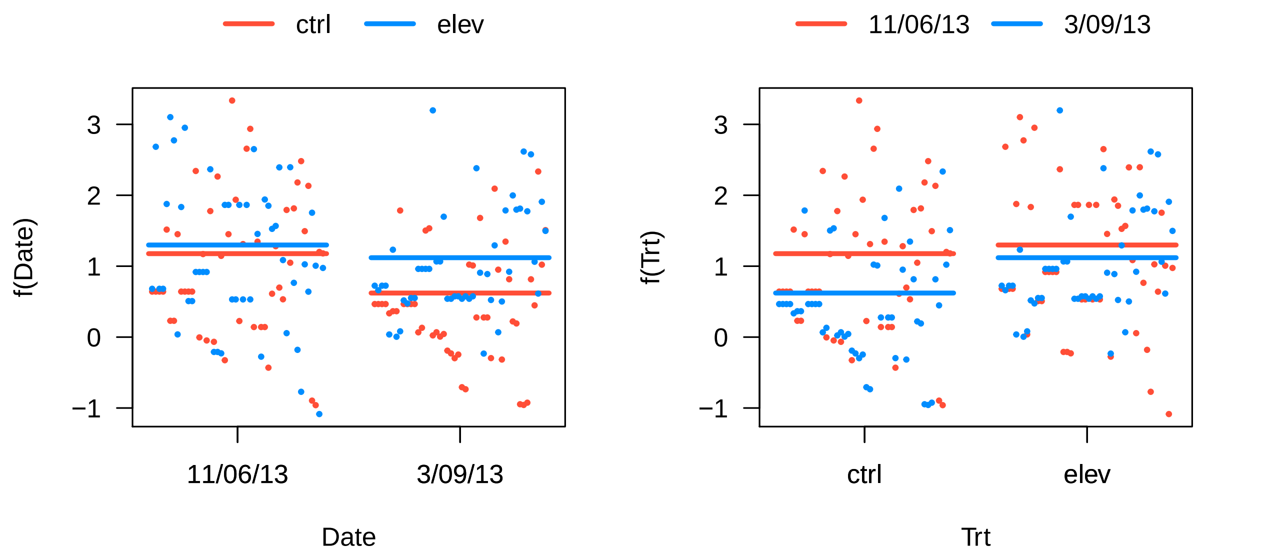

Figure 1.17: Model predictions, looking at treatment effects by date (left) and date effects by treatment (right).

significant we must dig deeper to understand the source of the interaction. Looking at Fig. 1.17, it appears that :

1. Forbs decreased in frequency between the two dates in the control but not in the elevated CO_2 treatment
2. Forb abundance was lower in the control than in the elevated CO_2 treatment on the second date, but not the first date.

```
# set up graphics window for two plots
par(mfrow=c(1, 2))

# plot model preditions and data
visreg(forb.pois.ind, 'Date', 'Trt', overlay=TRUE)
visreg(forb.pois.ind, 'Trt', 'Date', overlay=TRUE)
```

We can evaluate each of these hypotheses through model selection after combining treatment levels.

```
# create a three level factor that combines both dates in the 'elev' treatment
eucface$trtcomb.elev <- with(eucface, paste(Trt, Date, sep='-'))
eucface$trtcomb.elev[eucface$Trt == 'elev'] <- 'elev'
eucface$trtcomb.elev <- as.factor(eucface$trtcomb.elev)
levels(eucface$trtcomb.elev)

## [1] "ctrl-11/06/13" "ctrl-3/09/13"  "elev"

# create a three level factor that combines both treatments for the '11/06/13' sampling
eucface$trtcomb.date <- with(eucface, paste(Trt, Date, sep='-'))
eucface$trtcomb.date[eucface$Date == '11/06/13'] <- '11/06/13'
eucface$trtcomb.date <- as.factor(eucface$trtcomb.date)
levels(eucface$trtcomb.date)

## [1] "11/06/13"     "ctrl-3/09/13" "elev-3/09/13"
```

```
# fit model with three level combination factors
m2.elev <- glmer(Forbes ~ trtcomb.elev + (1|Ring/Plot/Sub/Ind),
                 family=poisson('sqrt'), data=eucface)

m2.date <- glmer(Forbes ~ trtcomb.date + (1|Ring/Plot/Sub/Ind),
                 family=poisson('sqrt'), data=eucface)

# fit models with only one main effect (Date or Trt)
m3.Trt <- glmer(Forbes~Date+(1|Ring/Plot/Sub/Ind), family=poisson(sqrt), data=eucface)
m3.Date <- glmer(Forbes~Trt+(1|Ring/Plot/Sub/Ind), family=poisson(sqrt), data=eucface)

# compare models (model with the lowest AIC is the most efficient at predicting the response)
AIC(forb.pois.ind, m2.elev, m2.date, m3.Trt, m3.Date)

##               df      AIC
## forb.pois.ind  8 611.9524
## m2.elev        7 611.9420
## m2.date        7 610.0109
## m3.Trt         6 612.6342
## m3.Date        6 627.4332
```

The three-level model in which date effects are estimated in the control treatment but not the elevated treatment (`m2.elev`) is not a substantial improvement over the fully factorial model (`forb.pois.ind`), so we do not consider it further. The three-level model in which treatment effects are estimated on the second date but not the first date (`m2.date`) has the lowest AIC score in comparison to the fully factorial model (`forb.pois.ind`) and to the two-level models that do not estimate an effect of treatment (`m3.Trt`) or date (`m3.Date`), so is the model that we select as the one that best predicts forb dynamics.

1.8 Example: logistic regression with the ground cover data (glmer)

Something that wasn't mentioned regarding this EucFACE ground cover dataset is that vegetation was assessed at a maximum of 16 points within each subplot and, therefore, the maximum number of observations per plot is constrained. This does not affect our analysis of forb abundances because these are generally low (less than ten in almost all plots) and so interpreting these as count data is appropriate. This is not the case for other response variables. As an example, let's look at grass cover in Fig. 1.18.

```
# A quick plot to visualize ground cover by Date and Ring.
# Colours represent 'Plot'.
xyplot(Grass~Date|Ring,groups=Plot,data=eucface,pch=16,jitter.x=T)
```

The data are clearly bounded at both the lower and upper range. We can treat these as binomial distributed, with the presence or absence of grass assessed at each of the sixteen points within each subplot. Do do this, we use `cbind` to create a two-column response matrix indicating the number of presences and absences within each subplot, as we did for logistic regression in Section 7.5.1. We also specify the random effects as we did for the analysis of forb abundance above.

```
# fit model with binomial error distribution
grass.binom <- glmer(cbind(Grass, 16-Grass)~Date*Trt+(1|Ring/Plot/Sub),
                     family=binomial, data=eucface)

# test significance of main effects and interaction
```

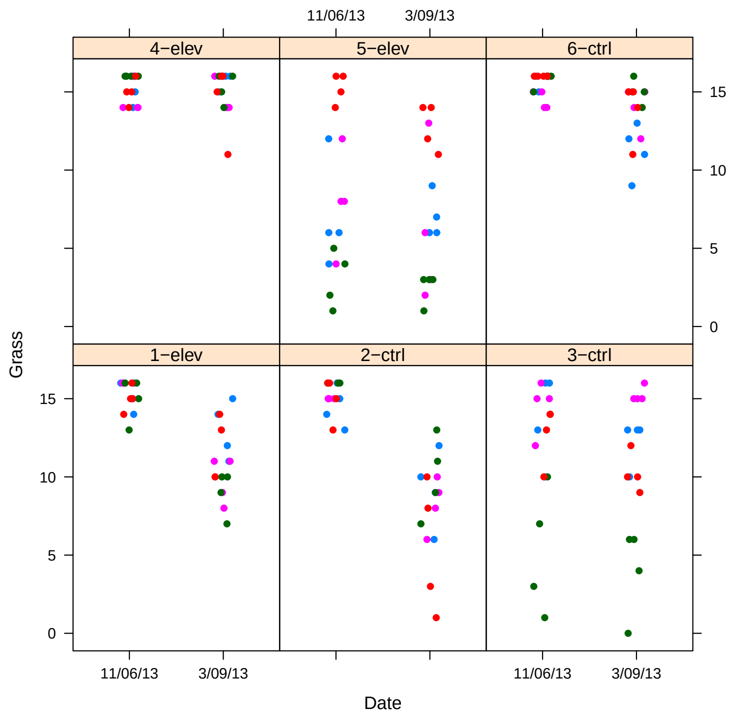

Figure 1.18: Grass cover within plant communities exposed to ambient or elevated carbon dioxide concentrations at two dates. Cover was assessed based on presence at each of sixteen locations within a subplot. Points represent estimates within subplots, subplots common to the same plot within each ring have a common colour.

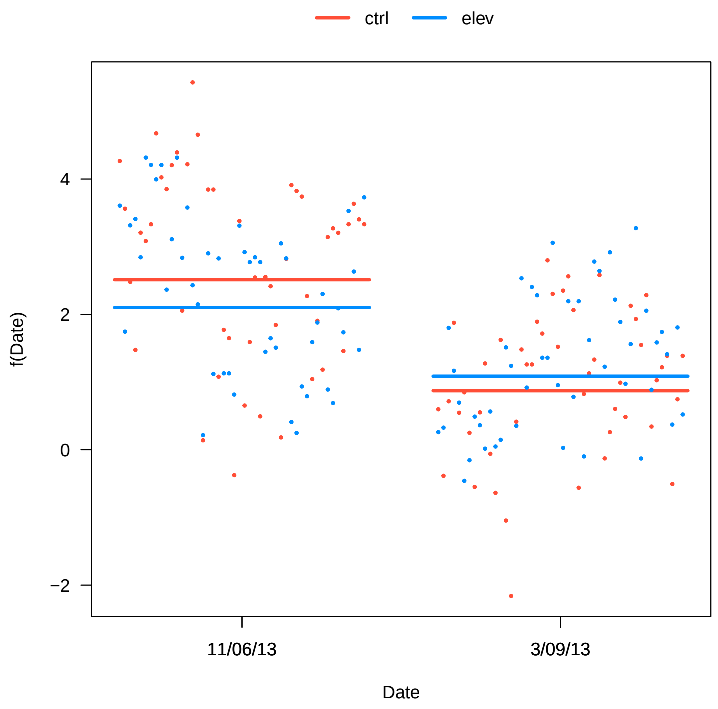

Figure 1.19: Model predictions, looking at treatment effects by date.

```
Anova(grass.binom)

## Analysis of Deviance Table (Type II Wald chisquare tests)
##
## Response: cbind(Grass, 16 - Grass)
##             Chisq Df Pr(>Chisq)
## Date     145.3645  1  < 2.2e-16 ***
## Trt        0.0000  1   0.997748
## Date:Trt   8.3183  1   0.003925 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The results suggest a highly significant effect of `date` and a signficant `date` by `treatment` interaction. The model predictions are shown in Fig. 1.19. A similar approach could be used to tease apart the source of the interaction as was used for forb abundances.

```
# plot model preditions and data
visreg(grass.binom, 'Date', 'Trt', overlay=TRUE)
```

1.9 Example: logistic regression with seed germination data (glmer)

In this example we will take another look at logistic regression when we have random effects, using a generalized linear mixed effects model. We have to related datasets on germination success of seeds. Seeds of four Tea Tree (*Melaleuca* sp.) species were collected at 9 sites, and subjected to either a 'fire

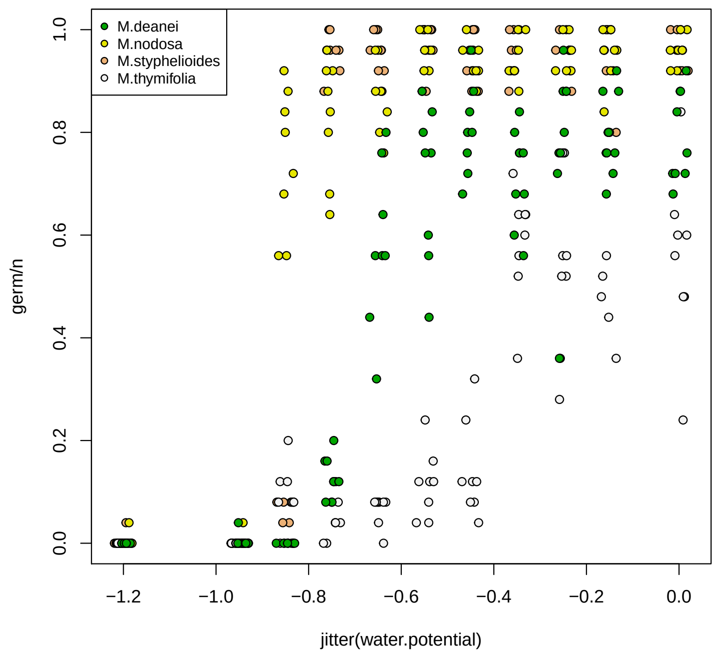

Figure 1.20: Germination success by species and as a function of water potential for the seedwater data.

cue' treatment (the `seedfire` dataset), or a dehydration treatment (`seedwater` data). Multiple cabinets were used for the experiment in the `seedfire` data, but not for the `seedwater` data. Each replicate contains ca. 20 seeds (given by the `n` variable), for which the germinated seeds were counted after some time. We thus have tabulated data, which we can use in a generalized linear model with a simple trick (we encountered this already in Section 7.5.1.1).

First we read the data, and make a simple plot of germination success (number of germinated seeds divided by sample size) for the `seedwater` data, producing Fig. 1.20.

```
seedfire <- read.csv("germination_fire.csv")

# Make sure temperature treatment is a factor
seedfire$temp <- as.factor(seedfire$temp)

seedwater <- read.csv("germination_water.csv")

palette(terrain.colors(4))
with(seedwater, plot(jitter(water.potential), germ/n, pch=21, bg=species))
legend("topleft", levels(seedwater$species), pch=21, pt.bg=palette(), cex=0.8)
```

We will first work with the `seedfire` dataset to test for temperature effects, and whether the fire cue treatment had any effect on germination success. We choose to treat the `site` variable as a random effect, since we are at this point not interested in specific site-to-site comparisons, but we do wish to account for the source of variation. It is also possible to treat `site` as a fixed effect, demonstrating that a variable can be either fixed or random depending on the question.

As mentioned above, we use a trick to allow the use of tabulated data in the `glmer` specification, using `cbindgerm, n - germ`, it represents a matrix with number of 'successes' and 'failures' tabulated in the two columns. We here treat the temperature treatment (`temp`) as a factor since the response is very non-linear, as we will see.

```
# A simple first fit with species and temperature
firefit1 <- glmer(cbind(germ, n-germ) ~ species + temp +
                  (1|site), data=seedfire, family=binomial)
Anova(firefit1)
```

```
## Analysis of Deviance Table (Type II Wald chisquare tests)
##
## Response: cbind(germ, n - germ)
##           Chisq Df Pr(>Chisq)
## species  271.18  3  < 2.2e-16 ***
## temp    2368.78  5  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Clearly both species and temp are highly significant. Next we test whether the interaction between species and temp is significant, as well as the fire cue treatment. We prefer to test additional variables in steps, as follows.

```
# Fire cue is not significant
firefit2 <-  glmer(cbind(germ, n-germ) ~ species + temp + fire.cues +
                  (1|site), data=seedfire, family=binomial)

# fire.cues is not significant
anova(firefit1, firefit2)
```

```
## Data: seedfire
## Models:
## firefit1: cbind(germ, n - germ) ~ species + temp + (1 | site)
## firefit2: cbind(germ, n - germ) ~ species + temp + fire.cues + (1 | site)
##          Df    AIC    BIC  logLik deviance  Chisq Chi Df Pr(>Chisq)
## firefit1 10 2712.6 2756.1 -1346.3   2692.6
## firefit2 11 2714.5 2762.4 -1346.3   2692.5 0.0367      1     0.8481
```

```
# Include the interaction between species and temperature.
firefit3 <-  glmer(cbind(germ, n-germ) ~ species * temp +
                  (1|site), data=seedfire, family=binomial)

# The interaction is very significant (also note large decrease in AIC)
anova(firefit1, firefit3)
```

```
## Data: seedfire
## Models:
## firefit1: cbind(germ, n - germ) ~ species + temp + (1 | site)
## firefit3: cbind(germ, n - germ) ~ species * temp + (1 | site)
##          Df    AIC    BIC  logLik deviance  Chisq Chi Df Pr(>Chisq)
## firefit1 10 2712.6 2756.1 -1346.3   2692.6
## firefit3 25 2266.2 2375.1 -1108.1   2216.2 476.36     15  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Note: you will get a warning when fitting the `firefit3` model. For the purpose of this example, you can ignore the warning.

We can now conclude that there is ample evidence that the different species respond very differently to temperature, but that there is no evidence for the fire cue to have any effect on germination success.

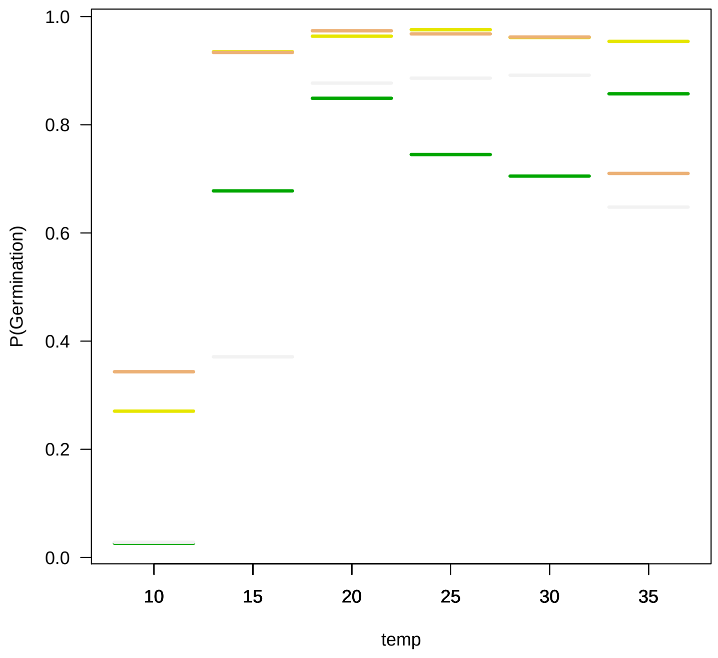

Figure 1.21: The firefit3 model visualized.

Finally we visualize the effects estimated with the `visreg` package. When visualizing a fitted `glmer` model, we have to specify `scale="response"` to convert the link function back to the original scale. The following code makes Fig. 1.21. Note the various settings to simplify the plot, and the use of `line.par` to change the colour of the lines.

```
visreg(firefit3, "temp", by="species", overlay=TRUE, band=FALSE,partial=FALSE,scale="response",
       ylab="P(Germination)", line.par=list(col=terrain.colors(4)), legend=FALSE, rug=FALSE)
```

Next, we will use the `seedwater` dataset to test for effects of dehydration on germination success. The degree of dehydration is measured as the water potential, where more negative values indicate drier seed. The model fitted will be similar as for the seedfire data, except our main predictor (in addition to `species`) is a numeric variable, not a factor.

```
fitwater1 <- glmer(cbind(germ, n-germ) ~ species + water.potential +
                     (1|site), data=seedwater, family=binomial)
fitwater2 <- glmer(cbind(germ, n-germ) ~ species * water.potential +
                     (1|site), data=seedwater, family=binomial)

anova(fitwater1, fitwater2)

## Data: seedwater
## Models:
## fitwater1: cbind(germ, n - germ) ~ species + water.potential + (1 | site)
## fitwater2: cbind(germ, n - germ) ~ species * water.potential + (1 | site)
##           Df    AIC    BIC  logLik deviance  Chisq Chi Df Pr(>Chisq)
## fitwater1  6 2623.9 2647.1 -1306.0   2611.9
## fitwater2  9 2482.8 2517.6 -1232.4   2464.8 147.14      3  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

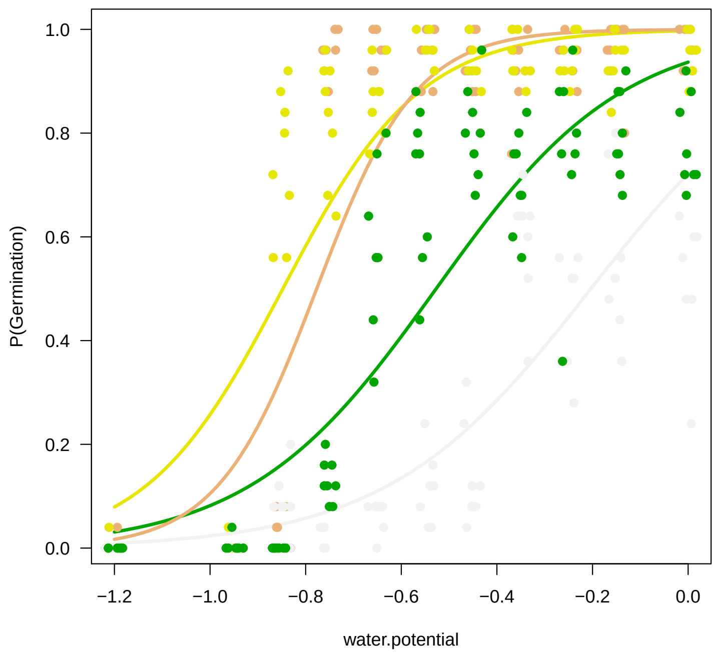

Figure 1.22: The fitwater2 model visualized, with the raw data superimposed

The following code makes Fig. 1.22, and shows a simple way to include the raw data alongside the fitted model. Note that the model does not fit equally well for all species - the response is steeper than expected for at least two of the species. This suggests some transformation of water potential would be appropriate, but we have not included further analyses here.

```r
visreg(fitwater2, "water.potential", by="species",
       overlay=TRUE, rug=FALSE, legend=FALSE,
       line.par=list(col=terrain.colors(4)),
       scale="response", ylab="P(Germination)")
with(seedwater, points(jitter(water.potential), germ/n, pch=19, col=species))
```

1.10 Exercises

In these exercises, we use the following colour codes:

- ■ **Easy**: make sure you complete some of these before moving on. These exercises will follow examples in the text very closely.
- ⧫ **Intermediate**: a bit harder. You will often have to combine functions to solve the exercise in two steps.
- ▲ **Hard**: difficult exercises! These exercises will require multiple steps, and significant departure from examples in the text.

We suggest you complete these exercises in an **R** markdown file. This will allow you to combine code chunks, graphical output, and written answers in a single, easy-to-read file.

1.10.1 PREF Canopy data

1. ⧫ In the analysis of the `pref` data, use model selection (`AIC`, `anova`) to evaluate the importance of `species` and `dfromtop`.

1.10.2 Litter decomposition data

1. ⧫ The `litter` data contain a factor (`variety`) describing whether the litter is derived from a genetically modified (`gm`) or conventional (`nongm`) soy variety. Plot the data to observe the effect of `variety`. Use `lmer` to test the effect of `variety`, in addition to the other significant variables, on litter decomposition.

1.10.3 EucFACE ground cover data

The file `eucfaceGC.csv` contains estimates of plant and litter cover within the rings of the EucFACE experiment, evaluating forest ecosystem responses to elevated CO2, on two dates; the data description can be found in Section A.20 (p. 271).

1. ⧫ Convert the variables indicating the nested sampling design to factors, then use `glmer` in `lme4` to test for an interaction between `Trt` and `Date` on `Grass` and `Litter` cover. Grass cover represents a frequency across a maximum of 16 points within a quadrat (use the `binomial` family), while litter cover represents counts (use the `poisson` family).
2. ▲ Following on from exercise 3, generate subsets to determine the sources of the interactions (i.e., does the treatment effect differ between the two dates or does the date effect differ between the two treatments?).

Index