A Brief Introduction to modelling and lme4 in R.

Introduction, Steps, and Initial Set-Up

This tutorial will take you through some basic steps to begin using statistical models in R. In layman's terms, a model is a way to make some kind of statement about how variables interact with each other. In the real world, data, especially ecological data, does not neatly follow the assumptions of classical statistical tests (ie t-tests, linear regression, ANOVA etc.), to compensate we can use broader statistical models. We will be discussing four different kinds of models including:

Linear regression – Models the relationship between a dependent and independent variable, assuming normality, homoscedasticity, and a linear relationship between variables.

Generalized Linear Model – Similar to the linear regression except response variables can have a non-normal error distribution.

Linear Mixed Model – A linear model that accounts for random effects

Generalized Linear Mixed Effects Model – A linear mixed model in which the response variable can have a non-normal error distribution.

The end goals of this tutorial are to become familiar with some of the basic functions of lme4 and to produce some plots and summary statistics to interpret your models. For plotting we will use base R along with the package 'sjPlot'.

Like anything else, work-flow should be organized and regimented. I recommend the following general steps:

1) Look at your data set and think about your research question and what variables can help answer that question.

2) Consider your options, you often have more than one solution available in modelling the relationship between variables.

3) Figure out the distribution of your variables of interest, consider if you will need to transform them in order to meet assumptions of normality or homoscedasticity.

4) Plot your response variable(s) against your variable(s) of interest, this can better inform you of an appropriate modelling procedure.

5) Code your model, ensuring that your variables are descriptive enough that you that you could open your R code in two months and know exactly what you did. In addition, be sure to use #comments judiciously.

6) Acquire the necessary summary statistics, AIC values, and other appropriate metrics.

7) Compare your models using the anova function, which can provide p-values for those inclined.

8) Visualize your models using base R plotting and sjPlot packages.

With all that in mind, let's get started. To begin clear your working environment if you haven't already with the function:

```
rm=(list=ls())#This removes everything, be careful!
```

Then, install and load the package using the following lines of code:

```
install.packages('lme4')
library(lme4)
```

We will also want to load package 'arm' for some additional functionality and 'sjPlot' to help visualize our models. Once installed, we can read in our data, in this tutorial I will be using the mpb data set obtained from the University of Calgary.

Simply point R to the directory with the data file, then read in the data.

```
setwd('Your/Working/Directory/')
data<-read.csv('mpb.csv')
```

You should see that the data set is now associated with an object called 'data' with 1149 observations of 23 variables. Let's take a look at the data using the 'structure' and 'head' functions. `str(data)` provides us with each variable and their class, in this data set we have a lot of factors and integer data. head(data) gives us the first few rows of data, which might be more useful if this was a smaller data set or if we decide to subset the data.

Initial Plotting and Visualization

Before we do any analyses it is important to look at our data set and think about what we want our variables to be. Our data set contains 23 variables and it wouldn't be a very good idea to place all of them into a single model. We'll start by doing a few linear regressions on a small selection of variables, these are appropriate for a surface exploration of the data.

For the first couple of models, let's use `drymass.mg.` as our response variable and see how it varies with respect to `sex`, `width.mm.`, and `volume`.

We can look at the distribution of dry mass these variables with the 'hist' and 'plot' functions (Fig. 1)

```
par(mfrow=(c(2,2))) #creates a 2 by 2 area to plot graphs

hist(data$drymass.mg.,xlab='Dry Mass (mg)',main=NA)
hist(data$width.mm., xlab= 'Width (mm)',main=NA)
hist(data$Volume, xlab= 'Volume (mm^3)',main=NA)
plot(data$sex, xlab= 'Sex',ylab='Frequency')#Use plot since 'sex' is a
      categorical variable
```

Let's create some exploratory plots to get an idea of what the data looks like before we do any modelling, feel free to change the colours and point graphics ('pch') at your leisure. (Fig. 2)

```
plot(data$sex,data$drymass.mg., xlab='sex',ylab='Drymass (mg)',pch=16,col='darkblue')
plot(data$Volume,data$drymass.mg.,xlab='volume (mg^3)',ylab='Drymass
      (mg)',pch=16,col='darkgreen')
plot(data$width.mm.,data$drymass.mg.,xlab='width (mm)',ylab='Drymass (mg)',pch=16,
      col='red')
```

We see here that mass seems to increase as a function of width and volume and female beetles are more massive.

Linear Regression

To do a linear regression with the data we can use the 'lm' function with the following syntax:

lm(formula = response variable~predictor(s), data)

```
LM1<-lm(formula = drymass.mg.~sex,data=data)
LM2<-lm(formula = drymass.mg.~Volume.,data=data)
LM3<-lm(formula = drymass.mg.~width.mm.,data=data)
```

Now that we've stored our 3 regressions, we can use the display function. Display provides us with summary statistics of our model, let's try it with LM3.

```
Display(LM3)
lm(formula = drymass.mg. ~ width.mm., data = data)
            coef.est coef.se
Intercept) -4.57     0.19
width.mm.    4.26     0.10
---
n = 1149, k = 2
residual sd = 0.68, R-Squared = 0.63
```

We can also generate some diagnostic plots for our linear model (Fig. 3)

```
plot(LM3)#produces 4 diagnostic plots, par(mfrow=c(2,2))) is useful
```

We can see that width is a fairly strong correlate of dry mass. Depending on the kind of question you are attempting to answer, a linear regression might not be the right kind of model.

Generalized Linear Model

Now, let's turn to the GLM, this is where more advanced modelling techniques and the lme4 package come into play. The syntax is similar to the regression syntax with the addition of family to account for non-normal distributions: (formula = response variable ~ predictor(s), family, data) if family isn't specified it defaults to a normal/gaussian distribution.

We'll start by keeping our variables constant:

```
GLM1<-glm(formula = drymass.mg.~Volume+sex+width.mm.,data=data)
display(GLM1)
AIC(GLM1)
```

AIC is a new function here and stands for the Akaike Information Criterion, essentially a shorthand for estimating how good a fit your model is, a lower number indicates a better fit.

To demonstrate the family syntax, let's create another GLM using the variable procto, a measure of the abundance of a particular species of mite. Using hist(data$procto) we can see that the data is not normal at all. Therefore we need to specify a non gaussian distribution:

```
GLM2<-glm(formula = procto~drymass.mg.+Volume+sex+width.mm.,
       family='poisson',data=data)
summary(GLM2)
display(GLM2)
AIC(GLM2)
plot(GLM2)
```

Linear Mixed Model

In the analyses performed thus far, we have been neglecting the potential effects of the sampling year, park, and site. Since we can't be entirely sure how these may affect the study, it is appropriate to treat them as random effects in a mixed model. Put simply, these random effects will contribute some amount of error in constructing the model. Let's construct a linear mixed model with drymass.mg. as our response variable, width, sex, and volume as our fixed effects and year as our random effect. We can create varying intercept, varying slope, and varying slope and intercept models.

Let's begin with a varying intercept model:

```
M1<-lmer(formula = drymass.mg.~Volume+sex+width.mm.+(1|year),data=data)
```

Here, the (1|year) argument tells R to specify year as a random effect with a varying intercept. Let's look at the model in more detail:

```
display(M1)
```

```
ranef(M1)#provides the random effects
coef(M1)#provides the fixed effects
```

Display works as before, and we can see that this model fits a little better.

To fit the random slope model, we modify the random effect term slightly

```
M2<-lmer(formula = drymass.mg.~Volume+sex+width.mm.+
(0+width.mm.|year),data=data)
display(M2)
ranef(M2)
coef(M2)
```

Finally, we can fit a random slope and intercept:

```
M3<-lmer(formula = drymass.mg.~Volume+sex+width.mm.+
(1+width.mm.|year),data=data)
display(M3)
ranef(M3)
coef(M3)
```

We can see that this modestly improves the model's fit.

However, we have only looked at year, completely neglecting park and site. To rectify this, we'll need to nest our variables going from year to park then to site. All we need to do is make a few modifications to our syntax:

```
M4<-lmer(formula = drymass.mg.~Volume,sex,width.mm.+(1|year/park/site)
display(M4)
ranef(M4)
coef(M4)
```

We can see that this is a substantial improvement over our previous models.

Generalied Linear Mixed Model

We can also construct generalized linear mixed models using the glmer function which, like the glm, allows us to specify a non-gaussian distribution. As an example of some robust coding, calling lmer() without a family specified reverts to a glmer call, while calling glmer() without a family specified or if 'gaussian' (ie normal) is specified calls lmer.

In this case let's use totalmites as our response variable, incorporating length.mm. width.mm. (which likely interact) and drymass.mg.

```
M5<-glmer(formula = totalmites~length.mm.:width.mm.+drymass.mg.+
(1|year/park/site),data=data,family='poisson') #colon = interaction
```

```
display(M5)
ranef(M5)
```

If we wanted to look at sex as our response variable we need to do an additional step and tell R to treat it as numeric data like so:

```
M_s<-lmer(as.numeric(sex)~drymass.mg.+width.mm.+
            (1|year/park/site),data=data)
```

In the accompanying code, I have set up other models (M6-M9) looking at individual mite species in the data set, feel free to change around some of the variables.

Comparing and Graphing Models

We can compare models using the anova() function, note that this is case sensitive.

```
anova(M1,M4,test='F')

Data: data
Models:
M1: drymass.mg. ~ Volume + sex + width.mm. + (1 | year)
M4: drymass.mg. ~ Volume + sex + width.mm. + (1 | year/park/site)
   Df    AIC    BIC  logLik deviance  Chisq Chi Df Pr(>Chisq)
M1  6 1795.3 1825.5 -891.63   1783.3
M4  8 1565.9 1606.3 -774.96   1549.9 233.34      2  < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Here, you can see that model 4 is significantly different compared to model 1 and, judging by the output, a better fit.

To get a graphical representation of these models, we can use the package sjPlot:

```
sjp.lm(lm2)#plots the linear model with CIs
sjp.lmer(M4,type='fe.pred') #graph of regression lines w/ CI for
#each fixed effect
sjp.lmer(M4,type='fe.resid') #plots fitted values against residuals
```

We can visualize our GLM with the following function (Fig. 4)

```
sjp.glm(GLM2)#Plots incident rate ratios
```

The same exists for the generalized linear mixed effects model (Fig. 5 & Fig. 6):

```
sjp.glmer(M5,'re')#Plots random effects
sjp.glmer(M5,'fe')#Plots the fixed effects
```

This has only been a brief overview of the functions of lme4. Websites like R-bloggers and CRAN are a great repository for more detailed information. Before we go, don't forget to print your session info, this is convenient for when you want to go back to your code and remember what packages you used.

```
                              sessionInfo()
                  R version 3.2.3 (2015-12-10)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows >= 8 x64 (build 9200)

locale:
[1] LC_COLLATE=English_United States.1252  LC_CTYPE=English_United States.1252
LC_MONETARY=English_United States.1252
[4] LC_NUMERIC=C                           LC_TIME=English_United States.1252

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

loaded via a namespace (and not attached):
 [1] minqa_1.2.4     MASS_7.3-45     Matrix_1.2-3    tools_3.2.3     Rcpp_0.12.3
      splines_3.2.3   nlme_3.1-126
 [8] grid_3.2.3      nloptr_1.0.4    lme4_1.1-11     lattice_0.20-33
```

Figures

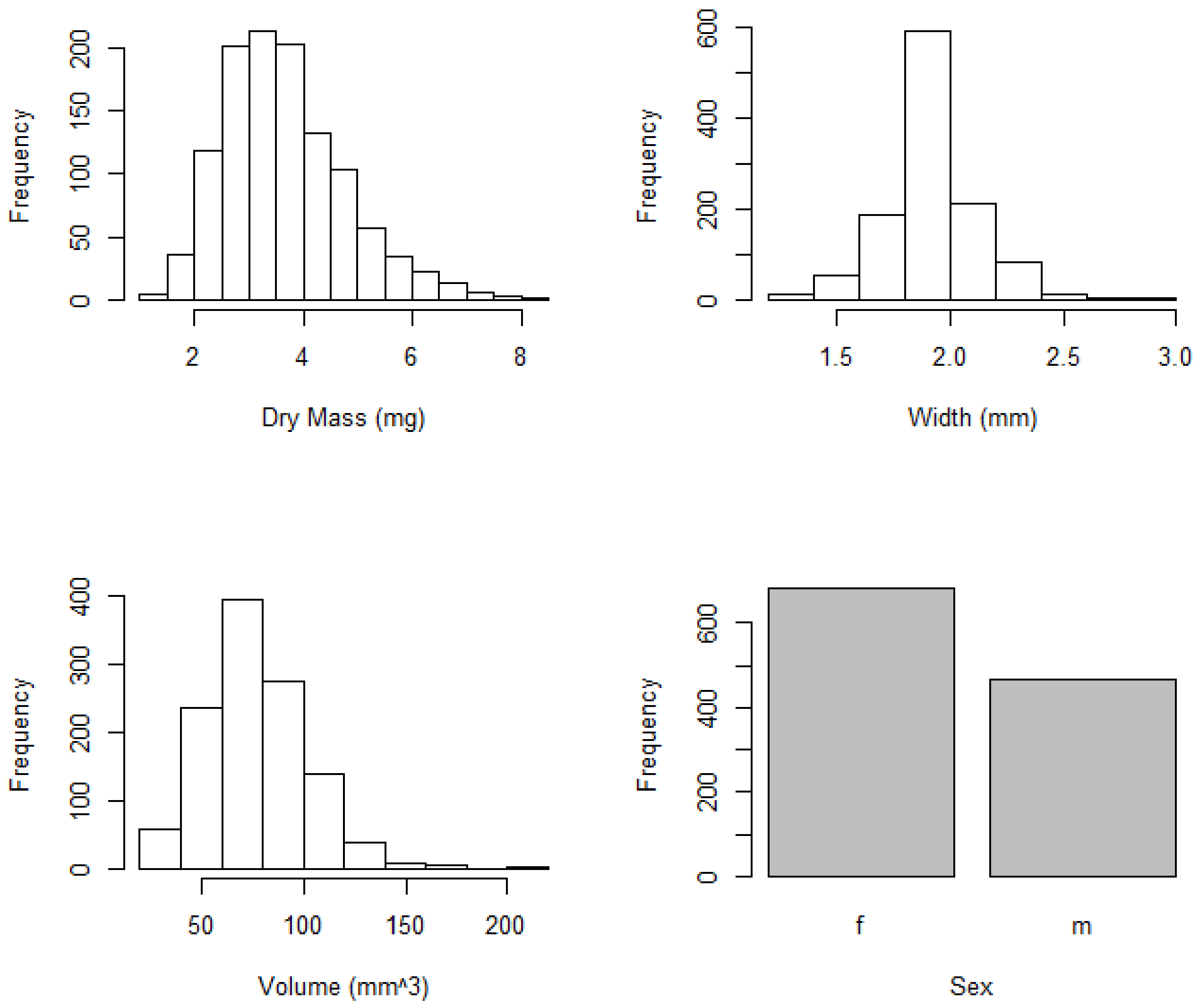

Figure 1: A plot of our main variables

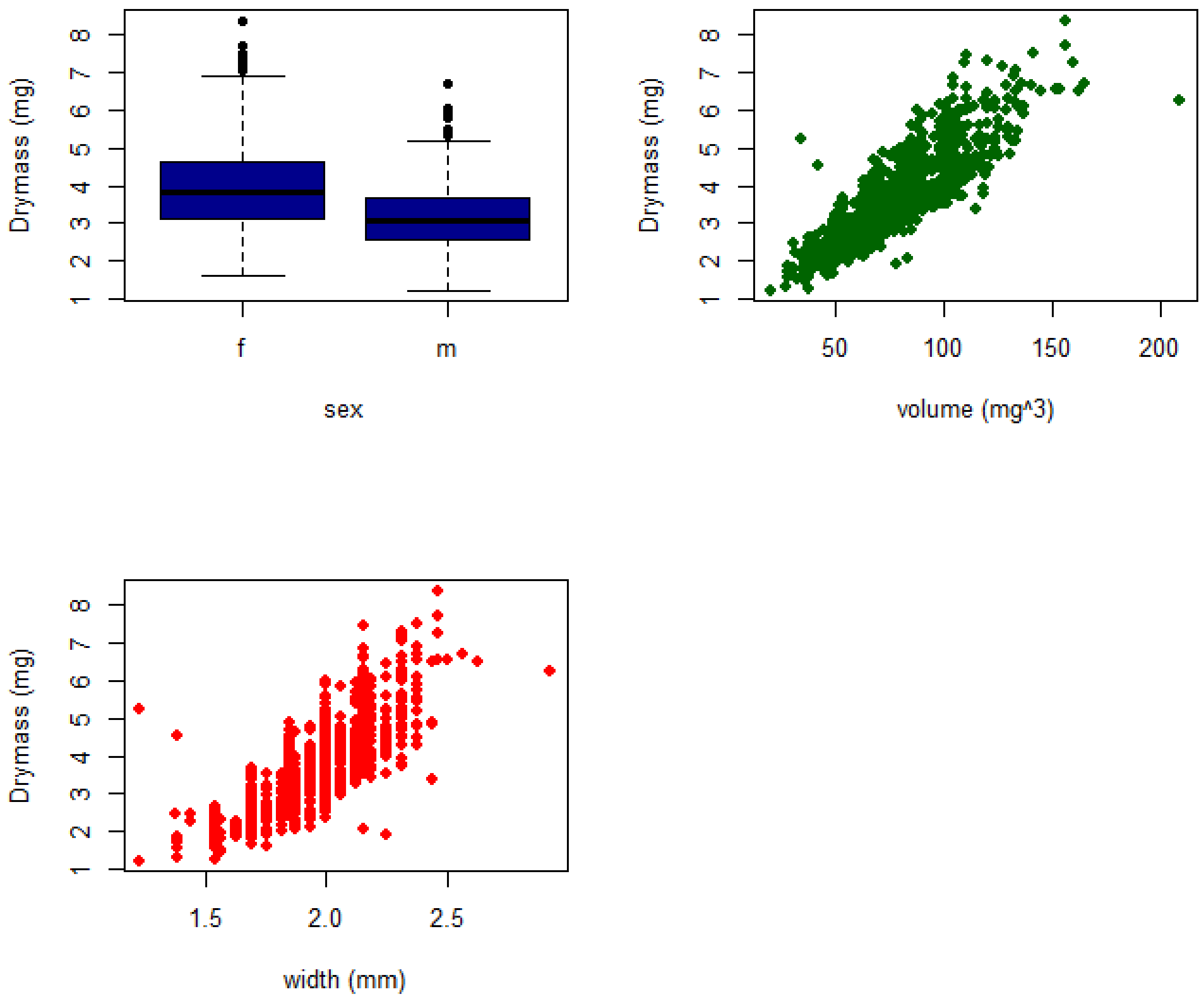

Figure 2: The relationship between drymass and the variables above

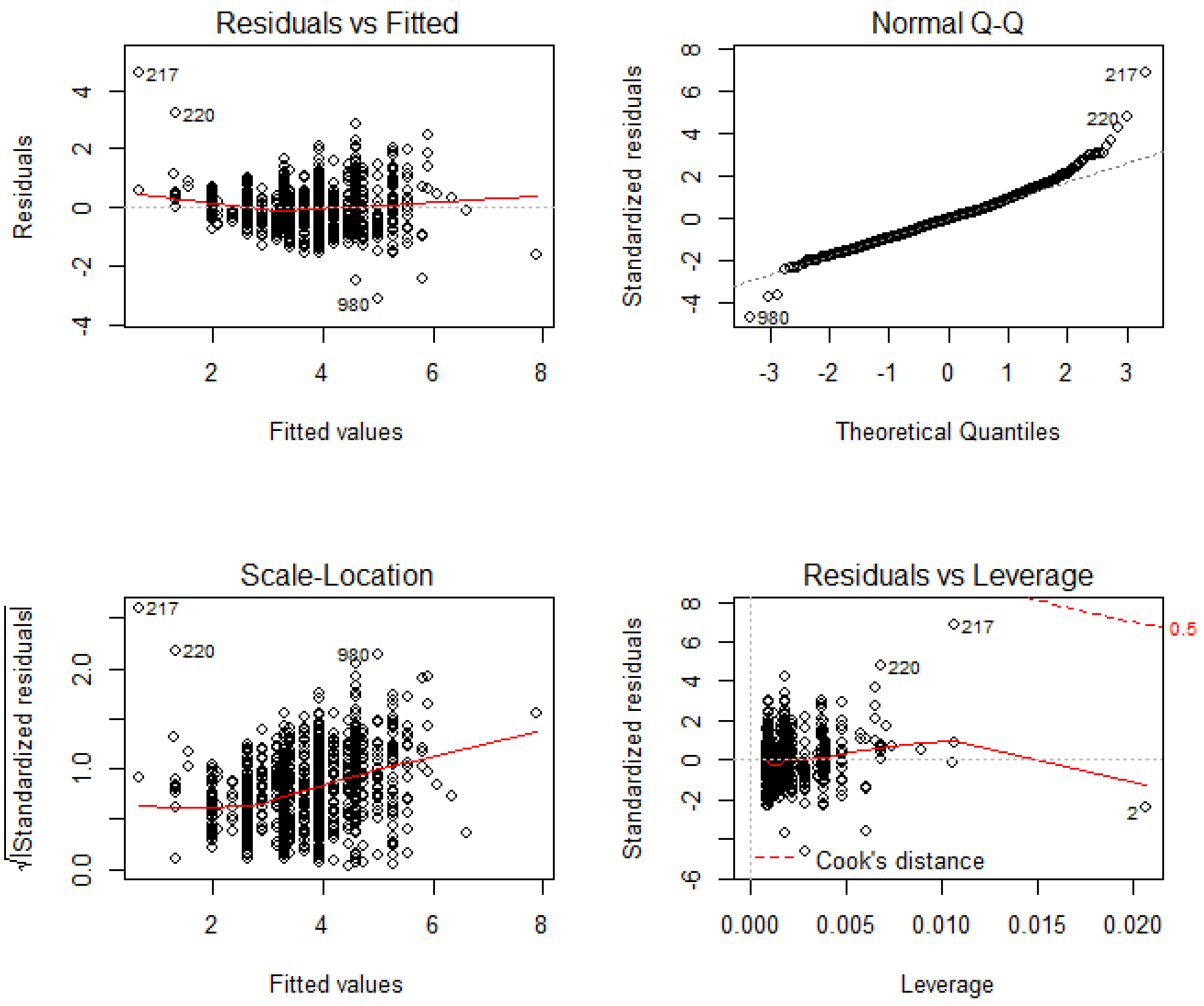

Figure 3: Some of the diagnostic plots generated for LM3

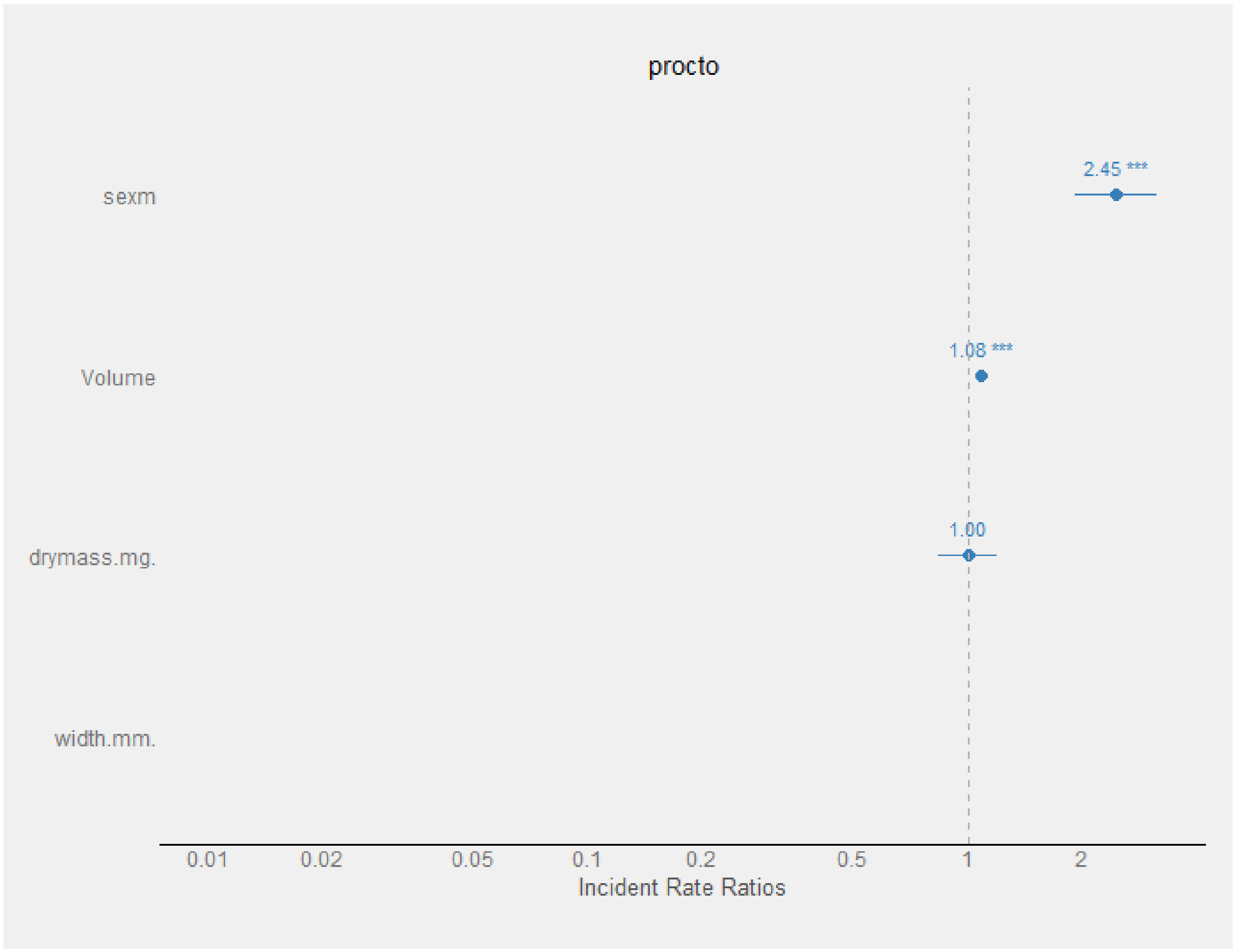

Figure 4: The Incident Rate Ratios from GLM2

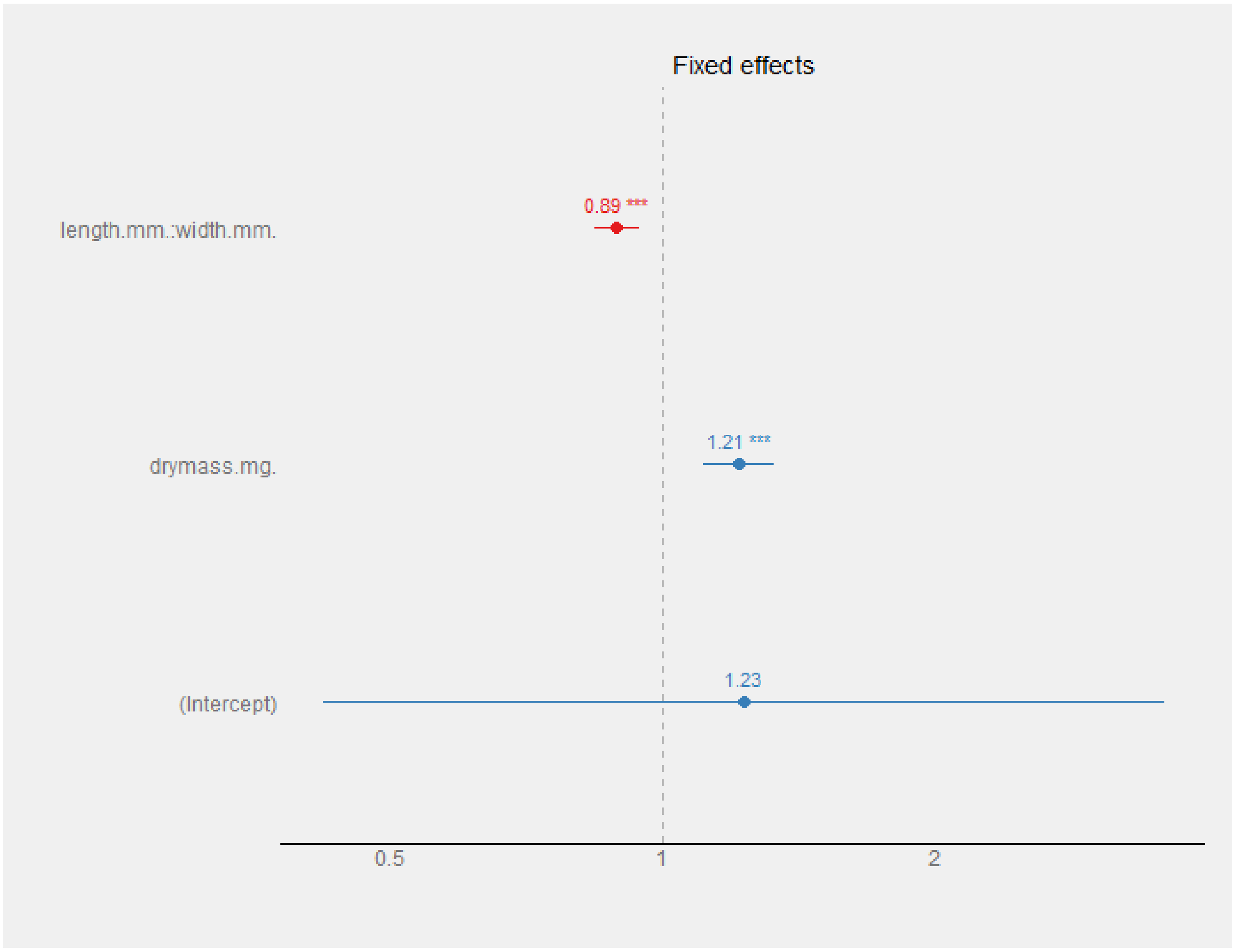

Figure 5: The fixed effects from M5

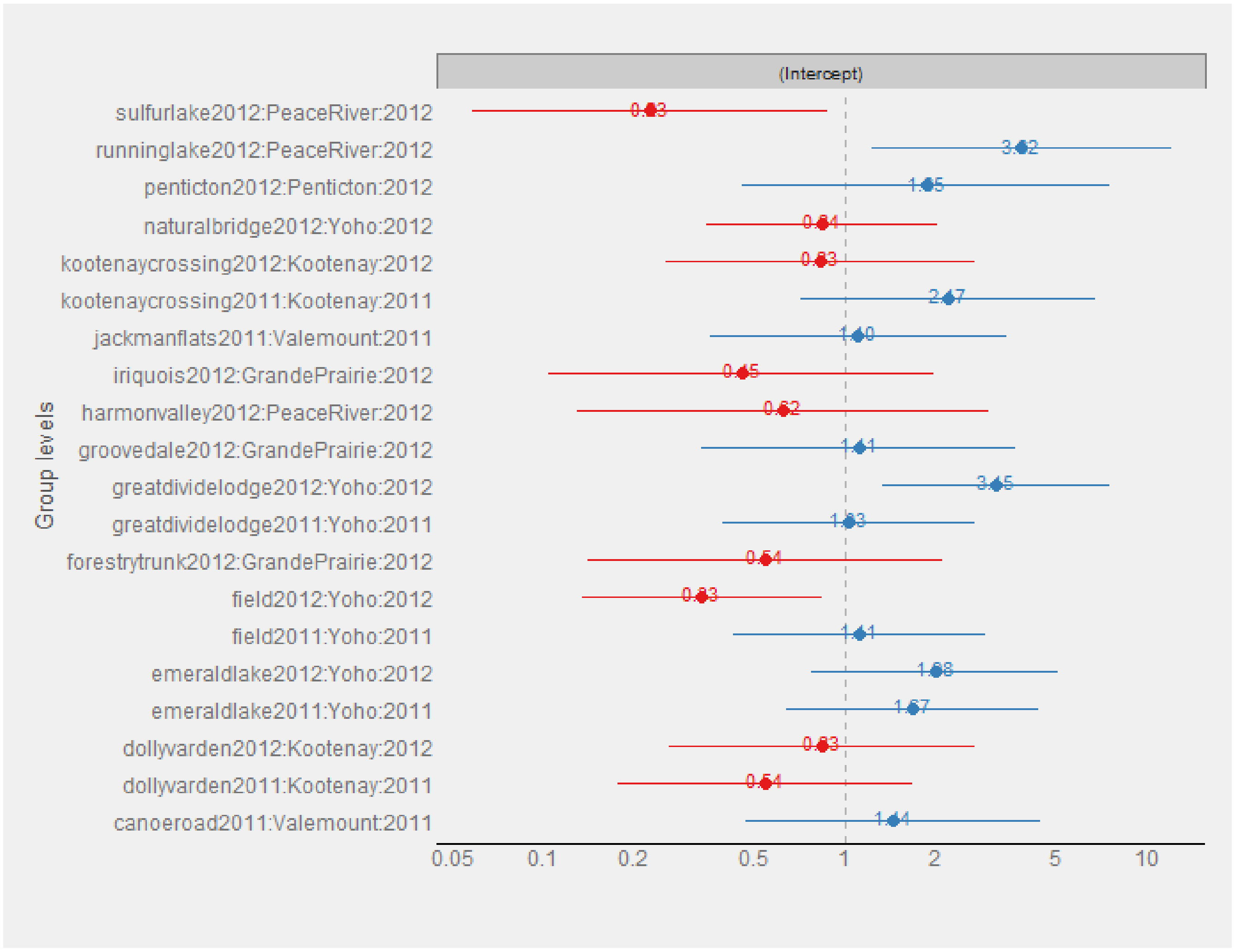

Figure 6: A random effect plot from M5