LegumeLegacy

SS7-II The statistical analysis of multi-site data (part 1-II)

Theory and R implementation

Name: Die Hu
Institute: Trinity College Dublin
Supervisor: Prof. Caroline Brophy

Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Funded by
the European Union

Overview

- Linear mixed-effects model
  - from linear models to mixed-effects models
  - random effects and variance structure
  - estimation methods: ML vs REML
- Linear mixed-effects model in R
  - fitting models using `nlme`
  - model comparison: ML vs REML
  - extracting predictions and variance components

Example dragon dataset

The relationship between dragon weight and carrying capacity appears to vary across mountain ranges.



Response: crop mass carried per day
predictor: dragon weight
grouping variable: mountain range

Linear modelling

Goal: fit a linear model for each mountain range

Dragon data:

- y_{ij}: carried crop mass
- x_{ij}: body weight of the ith dragon from mountain range j

Mountain range 1:

$$y_{i1} = \beta_{0,1} + \beta_{1,1}x_{i1} + \varepsilon_{i1}, \varepsilon_{i1} \sim N(0, \sigma_1^2)$$

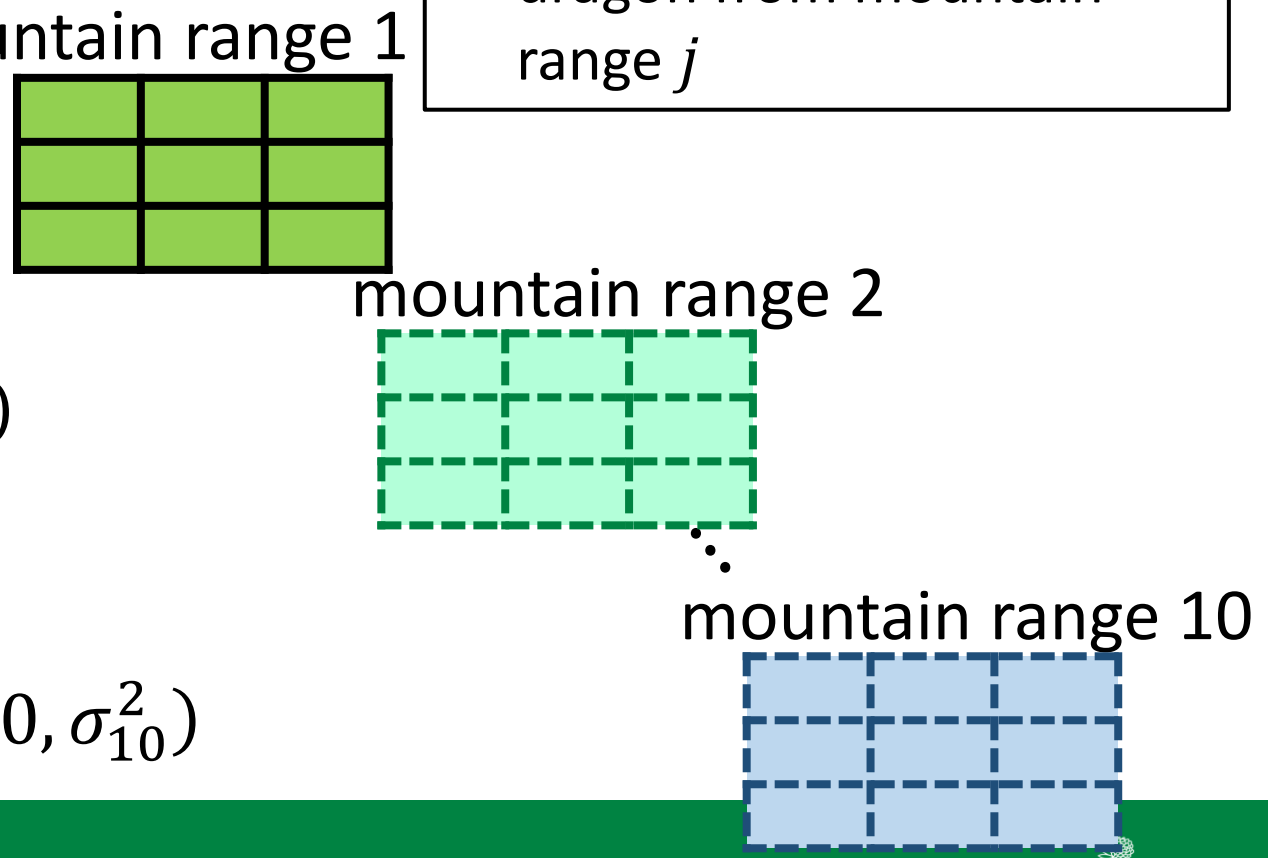
Mountain range 2:

$$y_{i2} = \beta_{0,2} + \beta_{1,2}x_{i2} + \varepsilon_{i2}, \ \varepsilon_{i2} \sim N(0, \sigma_2^2)$$

$$\vdots$$

Mountain range 10:

$$y_{i10} = \beta_{0,10} + \beta_{1,10}x_{i10} + \varepsilon_{i10}, \varepsilon_{i10} \sim N(0, \sigma_{10}^2)$$

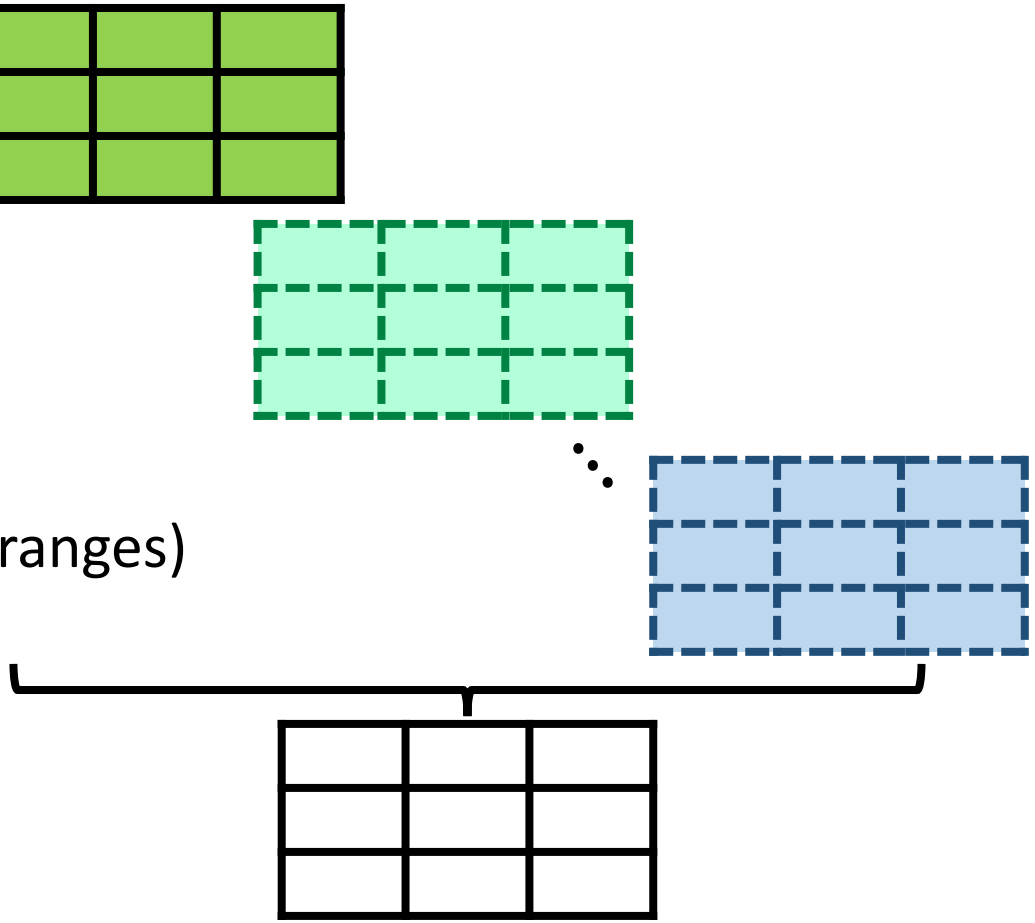


LegumeLegacy

Linear modelling

- Separate models for mountain range j
  - $y_{ij} = \beta_{0,j} + \beta_{1,j} x_{ij} + \varepsilon_{ij}$.
- Problems:
  - too many parameters (30 in total for 10 mountain ranges)

    $\{\beta_{0,1}, \dots, \beta_{0,10}; \beta_{1,1}, \dots, \beta_{1,10}; \sigma_1^2, \dots, \sigma_{10}^2\}$
  - no information sharing across mountain ranges

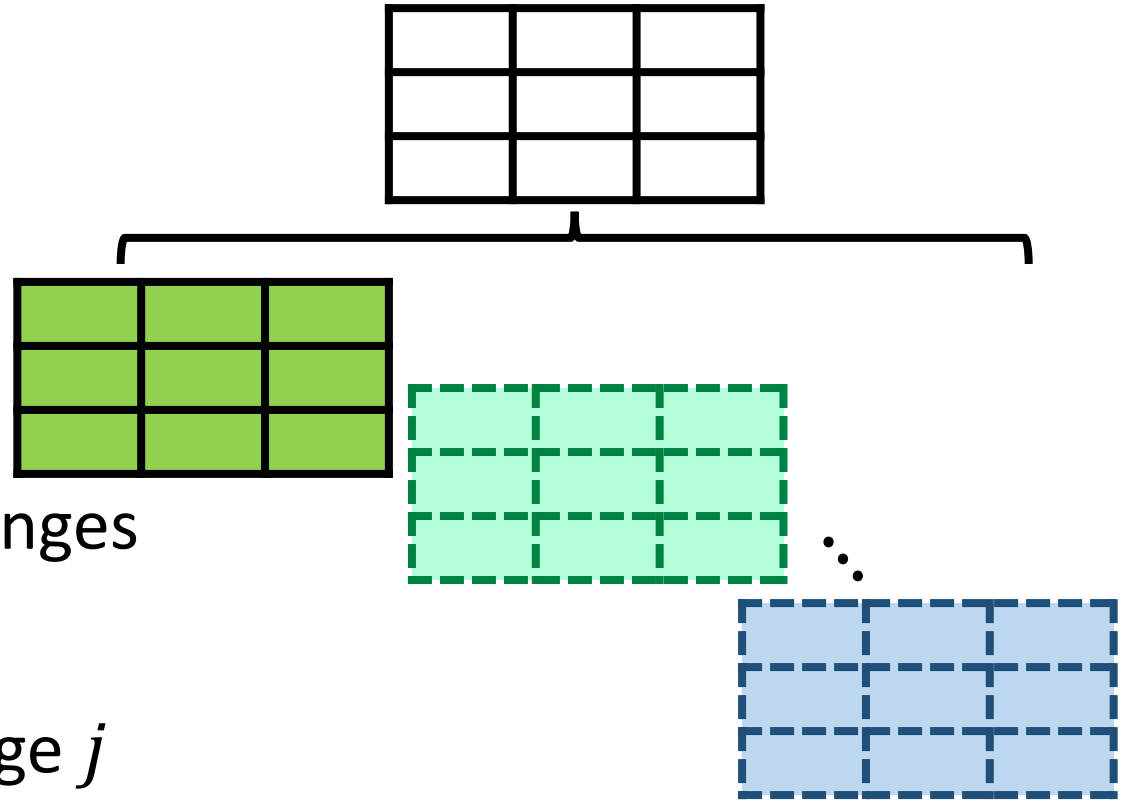


Linear mixed-effects model (random intercept)

$$y_{ij} = (\beta_0 + b_{0,j}) + \beta_1 x_{ij} + \varepsilon_{ij},$$

$$b_{0,j} \sim N(0, \sigma_{b0}^2), j = 1, \ldots, 10$$

$$\varepsilon_{ij} \sim N(0, \sigma_e^2)$$



- β_0, β_1: fixed effects across all mountain ranges
- ε_{ij}: residual
- $b_{0,j}$: random intercepts for mountain range j

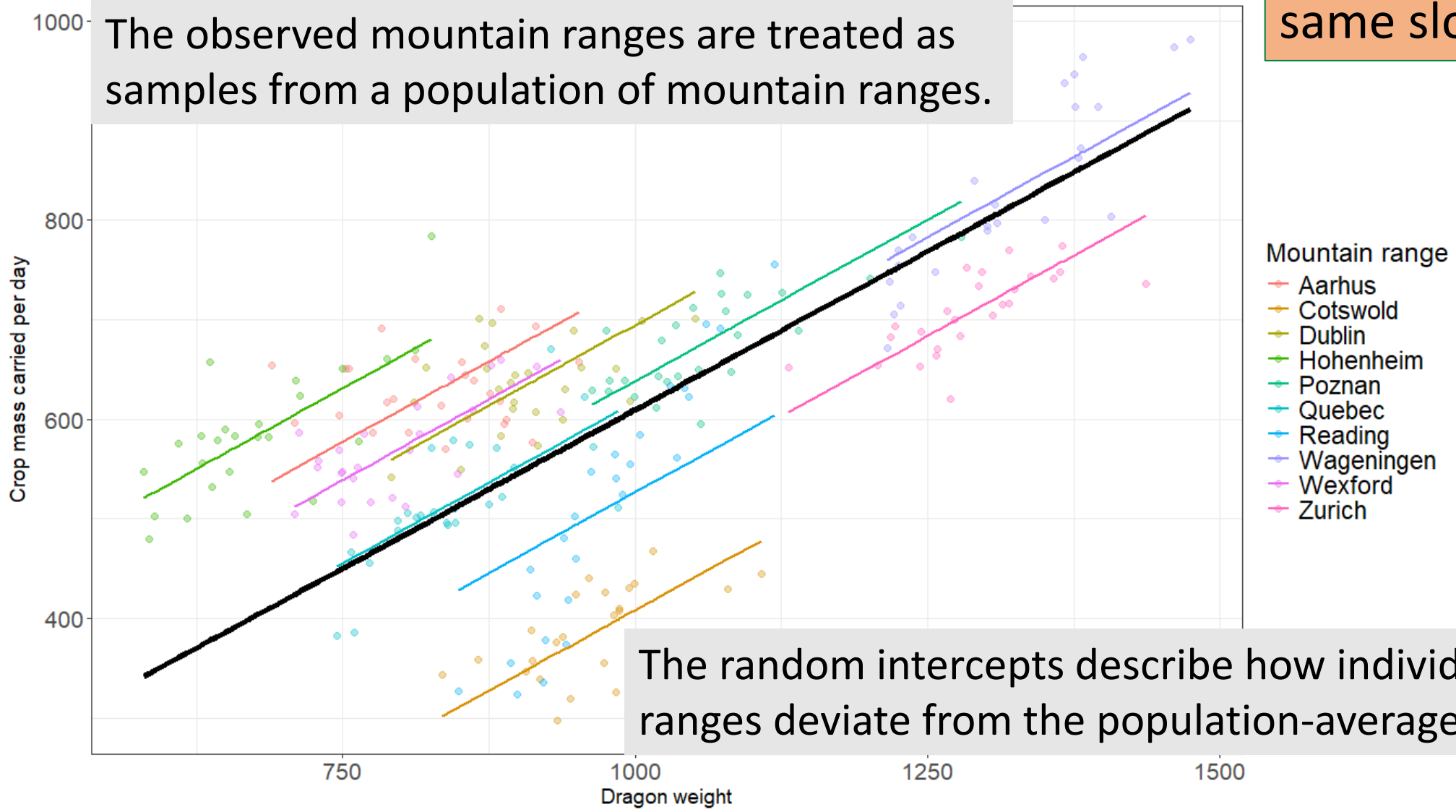
Only **4** parameters are estimated: $\{\beta_0, \beta_1; \sigma_{b0}^2, \sigma_e^2\}$

Linear mixed-effects model (random intercept)

$$y_{ij} = (\beta_0 + b_{0,j}) + \beta_1 x_{ij} + \varepsilon_{ij}$$

different intercepts same slope

The observed mountain ranges are treated as samples from a population of mountain ranges.



The random intercepts describe how individual mountain ranges deviate from the population-average intercept.

Linear mixed-effects model (random intercept and slope)

- $y_{ij} = (\beta_0 + b_{0,j}) + (\beta_1 + b_{1,j})x_{ij} + \varepsilon_{ij}$,
  - $b_{0,j} \sim N(0, \sigma_{b0}^2), b_{1,j} \sim N(0, \sigma_{b1}^2), j = 1, \ldots, 10$
  - we assume $\mathrm{cov}(b_{0,j}, b_{1,j}) = 0$ here, but it can be assumed to be nonzero
  - $\varepsilon_{ij} \sim N(0, \sigma_e^2)$
- $b_{0,j}, b_{1,j}$: random effects for mountain range j

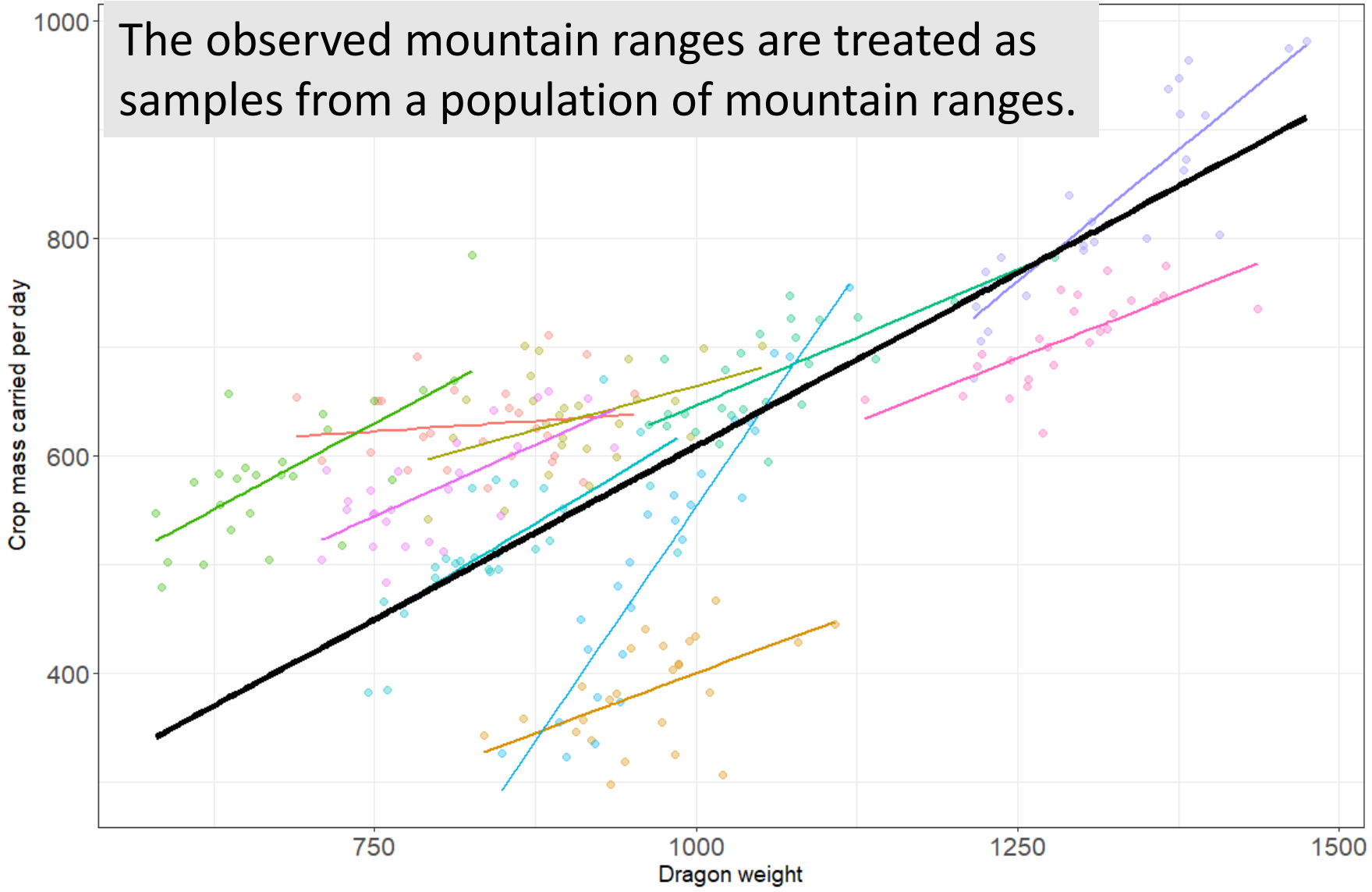
Only **5** parameters to be estimated: $\{\beta_0, \beta_1; \sigma_{b0}^2, \sigma_{b1}^2, \sigma_e^2\}$

Different mountain ranges can have different intercepts and slopes.

Linear mixed-effects model (random intercept and slope)

$$y_{ij} = (\beta_0 + b_{0,j}) + (\beta_1 + b_{1,j})x_{ij} + \varepsilon_{ij}$$

The observed mountain ranges are treated as samples from a population of mountain ranges.



different intercepts
different slopes

group ×predictor interactions

Linear mixed-effects model

$y_{ij} = (\beta_0 + b_{0,j}) + (\beta_1 + b_{1,j})x_{ij} + \varepsilon_{ij}$ can be written as:

$$y_{ij} = \beta_0 + \beta_1 x_{ij} + b_{0,j} + b_{1,j} x_{ij} + \varepsilon_{ij}$$

Fixed effects

Random effects

Mixed-effects modelling

- β_0, β_1: fixed effects across all mountain ranges
- $b_{0,j}, b_{1,j}$: random effects for mountain range j
- ε_{ij} : error term

Random effects assumption

$$y_{ij} = \beta_0 + \beta_1 x_{ij} + b_{0,j} + b_{1,j} x_{ij} + \varepsilon_{ij}$$

Assumptions:

$$b_{0,j} \sim N\left(0, \sigma_{b0}^2\right), b_{1,j} \sim N\left(0, \sigma_{b1}^2\right), j = 1, \ldots, 10$$

- Random effects represent variation across mountain ranges.

Covariance structure:

$$\begin{pmatrix} b_{0,j} \\ b_{1,j} \end{pmatrix} \sim N(\mathbf{0}, \mathbf{G}), \mathbf{G} = \begin{pmatrix} \sigma_{b0}^2 & 0 \\ 0 & \sigma_{b1}^2 \end{pmatrix}.$$

- Here we assume the random intercepts and slopes are independent.
- More complex covariance structures can also be used. e.g. $\begin{pmatrix} \sigma_{b0}^2 & \sigma_{b01} \\ \sigma_{b01} & \sigma_{b1}^2 \end{pmatrix}$

Error term assumption

$$y_{ij} = \beta_0 + \beta_1 x_{ij} + b_{0,j} + b_{1,j} x_{ij} + \varepsilon_{ij}$$

Assumption: $\varepsilon_{ij} \sim N(0, \sigma_e^2)$.

- Error term represent dragon-level variation within mountain ranges.

Across all dragons:

$$\boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_{11} \\ \vdots \\ \varepsilon_{1n} \\ \varepsilon_{21} \\ \vdots \\ \varepsilon_{10n} \end{pmatrix} \sim N(\mathbf{0}, \mathbf{R}), \mathbf{R} = \begin{pmatrix} \sigma_e^2 & 0 & 0 & 0 \\ 0 & \sigma_e^2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \sigma_e^2 \end{pmatrix}$$

- where $\mathbf{R}$ assumes constant variance and independent error terms.
- More flexible $\mathbf{R}$ structures can model, e.g. heteroskedasticity.

Variance components

$$y_{ij} = \beta_0 + \beta_1 x_{ij} + b_{0,j} + b_{1,j} x_{ij} + \varepsilon_{ij}, \begin{pmatrix} b_{0,j} \\ b_{1,j} \end{pmatrix} \sim N(\mathbf{0}, \mathbf{G}), \boldsymbol{\varepsilon} \sim N(\mathbf{0}, \mathbf{R})$$

With $\boldsymbol{Y} = \begin{pmatrix} y_{11} \\ \vdots \\ y_{1n} \\ y_{21} \\ \vdots \\ y_{10n} \end{pmatrix}, \mathbf{X} = \begin{pmatrix} 1 & x_{11} \\ \vdots & \vdots \\ 1 & x_{1n} \\ 1 & x_{21} \\ \vdots & \vdots \\ 1 & x_{10n} \end{pmatrix}, \mathbf{Z} = \begin{pmatrix} 1 & x_{11} \\ \vdots & \vdots \\ 1 & x_{1n} \\ 1 & x_{21} \\ \vdots & \vdots \\ 1 & x_{10n} \end{pmatrix}$

$$\mathrm{Var}(\boldsymbol{Y}) = \mathbf{Z}\mathbf{G}\mathbf{Z}^T + \mathbf{R}$$

How does variance affect estimation?

$$\mathrm{Var}(\boldsymbol{Y}) = \mathbf{ZGZ}^T + \mathbf{R}$$

- it is different from that in linear regression
- it makes the estimation more complex

Estimation method:

- ML (maximum likelihood)
- REML (restricted maximum likelihood)

When to use ML vs REML?

- ML
  - estimates fixed effects and variance together
  - tends to have biased variance
- REML
  - adjusts for estimation of fixed effects
  - provides less biased for variance estimates

- Use **ML** when comparing models **with different fixed effects**
- Use **REML** when comparing models **with different random effects**
- Use **REML** for final model estimates

Fitting LMM in R (nlme)

Step 0: load package and inspect toy data

```
install.package("nlme")
library(nlme)
# load dragon data
dragon_dat<-
readRDS("dragon_data.rds")


# Look at the first six rows of the dataset
head(dragon_dat)
# Check variables
str(dragon_dat)
```

```
> head(dragon_dat)
  dragon_id  range   weight weight_c100 carry_mass_day
1         1 Aarhus 912.3811 -0.37618916       576.2716
2         2 Aarhus 851.8823 -0.98117674       657.7297
3         3 Aarhus 854.7494 -0.95250641       644.4076
4         4 Aarhus 834.4431 -1.15556852       613.4063
5         5 Aarhus 787.7865 -1.62213522       617.4209
6         6 Aarhus 951.7793  0.01779277       657.9056
```

Response: crop mass carried per day
predictor: dragon weight
grouping variable: mountain range

Step 1: fit a linear-mixed model

Assumptions:

$$\boldsymbol{b}_j = \begin{pmatrix} b_{0,j} \\ b_{1,j} \end{pmatrix} \sim N(\mathbf{0}, \mathbf{G}),\ \boldsymbol{\varepsilon} \sim N(\mathbf{0}, \mathbf{R})$$

here $\mathbf{G} = \begin{pmatrix} \sigma_{b0}^2 & \sigma_{b01} \\ \sigma_{b01} & \sigma_{b1}^2 \end{pmatrix}$ and

$$\mathbf{R} = \begin{pmatrix} \sigma_e^2 & 0 & 0 & 0 \\ 0 & \sigma_e^2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \sigma_e^2 \end{pmatrix}.$$

$$y_{ij} = \beta_0 + \beta_1 x_{ij} + b_{0,j} + b_{1,j} x_{ij} + \varepsilon_{ij}$$

```
lmm_fit1<-lme(
fixed=carry_mass_day~1+weight,
random=~1+weight|range,
data=dragon_dat,
method="ML"
)

summary(lmm_fit1)
```

"REML"

Step 1-1: fit a linear mixed-effects model

Diagonal:

$$\mathbf{G} = \begin{pmatrix} \sigma_{b0}^2 & 0 \\ 0 & \sigma_{b1}^2 \end{pmatrix}$$

```
list(range=pdDiag(~1+weight))
```

Compound symmetric:

$$\mathbf{G} = \begin{pmatrix} \sigma_b^2 & \sigma \\ \sigma & \sigma_b^2 \end{pmatrix}$$

```
list(range=pdCompSymm(~1+weight))
```

Step 1-2: fit a linear mixed-effects model

```
lmm_fit1 <- lme(
fixed=carry_mass_day~1+weight,
random=~1+weight|range,
data=dragon_dat,
method = "ML",
weights=varIdent(form=~1|range)
)
```

To allow different residual variances across mountain ranges

$\varepsilon_{ij} \sim N\left(0, \sigma_{ej}^2\right)$ and $\sigma_{ej}^2 \neq \sigma_{ei}^2$ when $j \neq i$.

Step 1-3: fit a linear mixed-effects model

```
lmm_fit1 <- lme(
    ...,
    control = lmeControl(
      maxIter = 50,
      msMaxIter = 50,
      msMaxEval = 200,
      tolerance = 1e-6,
      msTol = 1e-7,
      opt = "optim"
    )
)
```

These settings improve convergence stability but do not change the statistical model.

Step 2-1: extract coefficients and variances

```
summary(lmm_fit1)
# to extract fixed effects
fixef(lmm_fit1)
# to extract random effects
ranef(lmm_fit1)
# to extract fixed effects + random effects
coef(lmm_fit1)
```

```
> fixef(lmm_fit1)
(Intercept)      weight
-27.9967430   0.6374153
```

β_0, β_1

```
> ranef(lmm_fit1)
            (Intercept)       weight
Aarhus        592.41483 -0.559573244
Cotswold      -10.51617 -0.198406253
Dublin        369.92192 -0.314583231
Hohenheim     182.23315 -0.002213349
Poznan        175.40674 -0.137775885
Quebec        -55.81009  0.073397235
Reading     -1143.45877  1.087955116
Wageningen   -421.59783  0.330909950
Wexford       176.54107 -0.109072999
Zurich        134.86516 -0.170637340
```

b_{0j}, b_{1j}

```
> coef(lmm_fit1)
            (Intercept)      weight
Aarhus        564.41809 0.07784205
Cotswold      -38.51291 0.43900905
Dublin        341.92517 0.32283207
Hohenheim     154.23640 0.63520195
Poznan        147.40999 0.49963941
Quebec        -83.80684 0.71081253
Reading     -1171.45551 1.72537041
Wageningen   -449.59457 0.96832525
Wexford       148.54432 0.52834230
Zurich        106.86841 0.46677796
```

$\beta_0 + b_{0j}$, $\beta_1 + b_{1j}$

Step 2-2: extract variances

```
# to extract random effects variance-covariance matrix
VarCorr(lmm_fit1)


# or you can use the code to generate the matrix G
restruct <- lmm_fit1$modelStruct$reStruct
matrix_data<-as.matrix(restruct$range)*lmm_fit1$sigma^2
```

```
> matrix_data
             (Intercept)       weight
(Intercept) 245314.6813 -224.2627183
weight         -224.2627    0.2147693
```

Step 3: compare models

```
lmm_fit1<-lme(
fixed=carry_mass_day~1+weight,
random=~1+weight|range,
...
)
```

```
lmm_fit2 <- lme(
fixed=carry_mass_day~1+weight,
random=~1|range,
...
)
```

```
lmm_fit3 <- lme(
fixed=carry_mass_day~1,
random=~1|range,
...
)
```

```
MuMIn::AICc(lmm_fit1, lmm_fit2)
# lower AICc indicates a better model fit
anova(lmm_fit2, lmm_fit3)
# fitted with ML when comparing fixed effects
```

	fit 1 vs fit 2	fit 2 vs fit 3	fit 1 vs fit 3
Fixed-effects structure	same	different	different
Random-effects structure	different	same	different
REML or ML	**REML**	**ML**	**×**

Step 4: refit final model using REML and get predictions

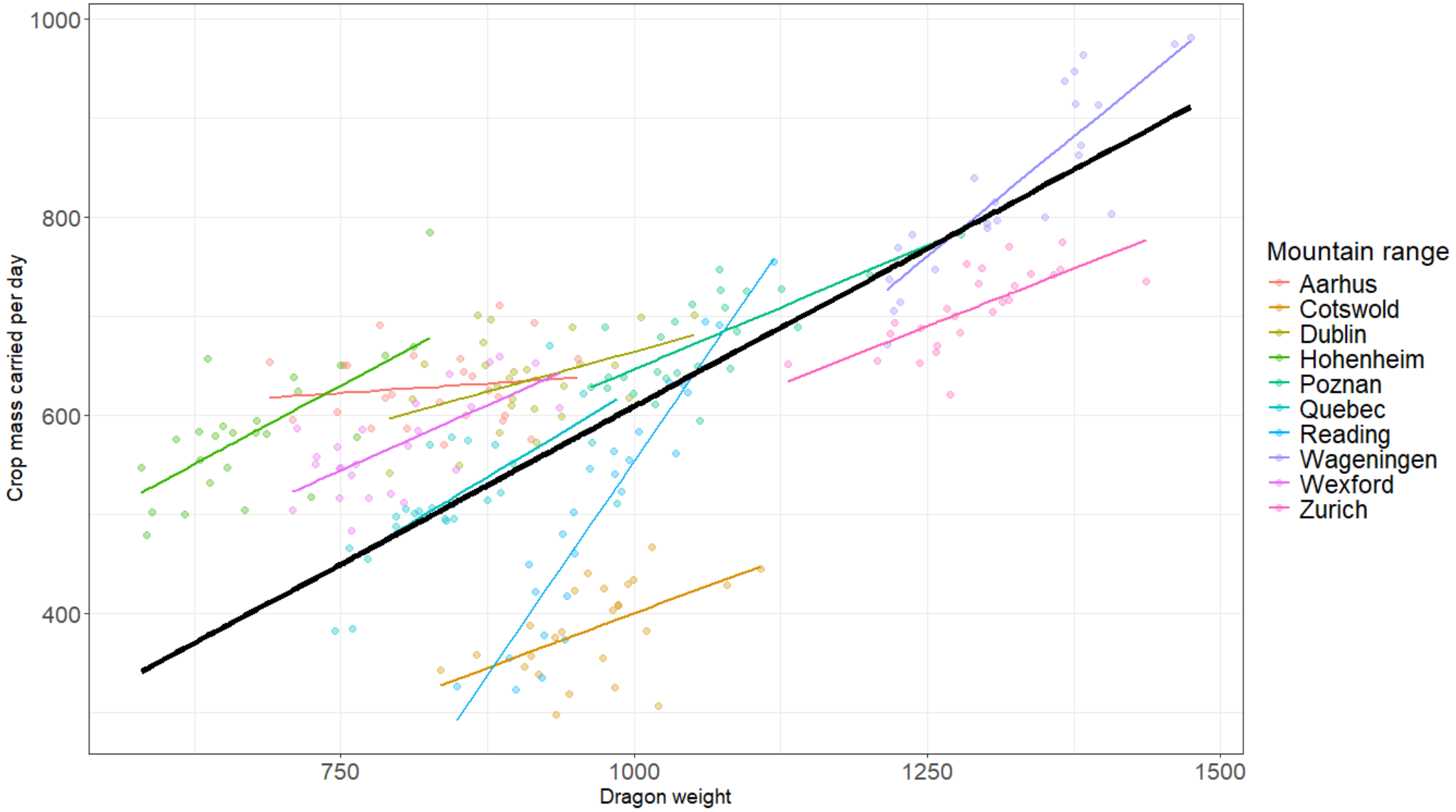
```
lmm_final<-
update(lmm_fit1,method="REML")
# population-level predictions
predict(lmm_final,newdata=dragon_dat,
level=0)

# group-specific predictions
predict(lmm_final,newdata=dragon_dat,
level=1)
```

Only use estimates of fixed effects

Use both estimates of fixed effects and random effects

Step 4: refit final model using REML and get predictions



Lab 1.2 – Analysing the dragon data