Working with R

A University of Queensland Advanced Workshop

Session 7: Random Effects Extensions

Bill Venables, CSIRO/Data61, Dutton Park

Rhetta Chappell, Griffith University

2–5 February, 2021

Contents

1 An introductory example: petroleum extraction

The petrol data of N. L. Prater.

- *No* crude oil sample identification label. (Factor.)
- *SG* specific gravity, degrees API. (Constant within sample.)
- *VP* vapour pressure in pounds per square inch. (Constant within sample.)
- *V10* volatility of crude; ASTM 10% point. (Constant within sample.)
- *EP* desired volatility of gasoline. (The end point. Varies within sample.)
- *Y* yield as a percentage of crude.

For a description in **R**:

```
?petrol
petrol  ## print the whole data!
```

```
   No   SG  VP V10  EP    Y
1   A 50.8 8.6 190 205 12.2
2   A 50.8 8.6 190 275 22.3
3   A 50.8 8.6 190 345 34.7
4   A 50.8 8.6 190 407 45.7
....
28  I 32.2 2.4 284 351 14.0
29  I 32.2 2.4 284 424 23.2
30  J 31.8 0.2 316 365  8.5
31  J 31.8 0.2 316 379 14.7
32  J 31.8 0.2 316 428 18.0
```

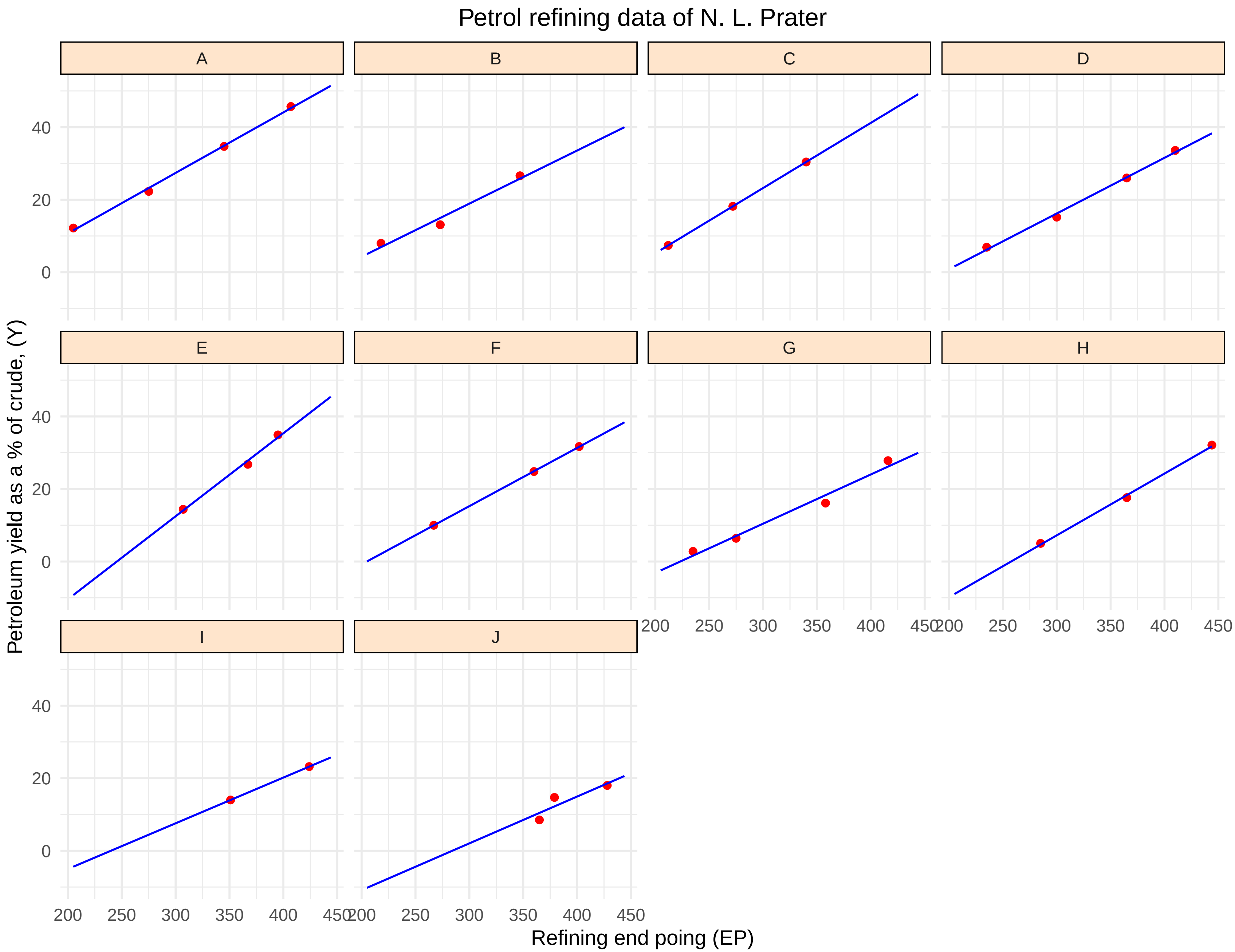
For a more complete description of the data and an alternative (somewhat fussy) analysis see the `betareg` package, (Cribari-Neto and Zeileis, 2010). *`?GasolineYield`*.

An initial look at the data:

```
ggplot(petrol) + aes(x = EP, y=Y) +
  geom_point(colour = "red") +
  stat_smooth(method = "lm", se = FALSE, fullrange = TRUE,
              size=0.5, colour = "blue",formula=y~x) + facet_wrap( ~ No) +
  labs(title = "Petrol refining data of N. L. Prater",
       x = "Refining end poing (EP)",
       y = "Petroleum yield as a % of crude, (Y)")
```

The plots are shown below It will be convenient later to have a centred version of *EP*:

```
petrol <- petrol %>%
  within(EPc <- scale(EP))  ### for convenience
Store(petrol)
```


Petrol refining data of N. L. Prater
A
B
C
D
E
F
G
H
I
J
Petroleum yield as a % of crude, (Y)
Refining end poing (EP)
0
20
40
200
250
300
350
400
450

1.1 Fixed or random?

A pure fixed effects model treats the crude oil samples as independent with the residual error as the only source of randomness.

A random effects model treats them as possibly dependent, in that they may share the value of a latent random variable, addition to the residual error.

The obvious candidate predictor to be regarded as injecting an additional source of randomness is the crude oil sample indicator, `No`.

Fixed effects only.

```
options(show.signif.stars = FALSE)
m3 <- lm(Y ~ 0 + No/EPc, petrol)              ## 10 ints + 10 slopes
m2 <- lm(Y ~ 0 + No+EPc, petrol)              ## 10 ints + 1 slope
m1 <- lm(Y ~ 1 + SG+VP+V10+EPc, petrol)       ## (1 int + 3 coeffs) + 1 slope
anova(m1, m2, m3)

Analysis of Variance Table


Model 1: Y ~ 1 + SG + VP + V10 + EPc
Model 2: Y ~ 0 + No + EPc
Model 3: Y ~ 0 + No/EPc
  Res.Df     RSS Df Sum of Sq      F  Pr(>F)
1     27 134.804
2     21  74.132  6    60.672 4.0009 0.01965
3     12  30.329  9    43.803 1.9257 0.14390
```

Parallel regressions, but differences between samples cannot quite be explained by regression on the other variables.

Random effects alternatives:

```
requireData(lme4)      ## alt. nlme
Rm1 <- lmer(Y ~ 1 + SG+VP+V10 + EPc + (1|No),     data = petrol)
Rm2 <- lmer(Y ~ 1 + SG+VP+V10 + EPc + (1+EPc|No), data = petrol)
anova(Rm1, Rm2)        ## automatic re-fitting

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

Data: petrol
Models:
Rm1: Y ~ 1 + SG + VP + V10 + EPc + (1 | No)
Rm2: Y ~ 1 + SG + VP + V10 + EPc + (1 + EPc | No)
    npar    AIC    BIC  logLik deviance  Chisq Df Pr(>Chisq)
Rm1    7 149.38 159.64 -67.692   135.38
Rm2    9 153.34 166.54 -67.672   135.34 0.0385  2     0.9809
```

Emphatically different slopes are not needed!

NB:

- Default fitting method “*REML*” produces good estimates of variance components
- To compare models using LR, models must be fitted with method “*ML*”
- *anova* detects if this has happened and updates the object if necessary.

Inspecting the random effects fit:

```
print(summary(Rm1), correlation = FALSE)
```

```
Linear mixed model fit by REML ['lmerMod']
Formula: Y ~ 1 + SG + VP + V10 + EPc + (1 | No)
   Data: petrol

REML criterion at convergence: 143.9

Scaled residuals:
    Min      1Q  Median      3Q     Max
-1.7807 -0.6064 -0.1068  0.4572  1.7811

Random effects:
 Groups   Name        Variance Std.Dev.
 No       (Intercept) 2.087    1.445
 Residual             3.505    1.872
Number of obs: 32, groups:  No, 10

Fixed effects:
            Estimate Std. Error t value
(Intercept) 46.06264   14.69241   3.135
```

```
SG              0.21940    0.14694   1.493
VP              0.54586    0.52053   1.049
V10            -0.15424    0.03996  -3.860
EPc            10.96402    0.38980  28.128
```

```r
print(summary(Rm2), correlation = FALSE)
```

```
Linear mixed model fit by REML ['lmerMod']
Formula: Y ~ 1 + SG + VP + V10 + EPc + (1 + EPc | No)
   Data: petrol

REML criterion at convergence: 143.9

Scaled residuals:
     Min       1Q   Median       3Q      Max
-1.77775 -0.59744 -0.07372  0.46067  1.76997

Random effects:
 Groups   Name        Variance Std.Dev. Corr
 No       (Intercept) 2.13847  1.4624
          EPc         0.06278  0.2506   0.06
 Residual             3.43304  1.8528
Number of obs: 32, groups:  No, 10

Fixed effects:
            Estimate Std. Error t value
(Intercept) 45.93487   14.77738   3.108
SG           0.21923    0.14770   1.484
```

```
VP              0.55213     0.52349   1.055
V10            -0.15380     0.04023  -3.823
EPc            10.96966     0.39677  27.648
```

Use *fixef* for fixed effect estimates and *ranef* for BLUPs:

```
cbind(Rm1 = ranef(Rm1)$No, Rm2 = ranef(Rm2)$No)

  (Intercept) Rm2.(Intercept)      Rm2.EPc
A -0.05943867     -0.04207529  0.05301543
B -0.21857133     -0.21900738 -0.02073023
C  1.92029758      1.96452890  0.01941871
D -1.92760817     -1.95646887 -0.03046897
E -0.21650182     -0.22737149  0.06972143
F  0.56931165      0.57476410  0.01571855
G  0.06701596      0.05019416 -0.10419793
H  0.19194473      0.18788703  0.04550772
I -0.40278068     -0.40719019 -0.03461858
J  0.07633074      0.07473900 -0.01336613
```

Variances and correlations

```
VarCorr(Rm2)           ## an unhelpful print method.

 Groups   Name        Std.Dev. Corr
 No       (Intercept) 1.46235
          EPc         0.25056  0.057
 Residual             1.85285

names(VarCorr(Rm2)) %>% noquote()

[1] No

VarCorr(Rm2)[["No"]]  ## The pieces are all accessible

            (Intercept)        EPc
(Intercept)  2.13847440 0.02098379
EPc          0.02098379 0.06278021
attr(,"stddev")
(Intercept)         EPc
  1.4623524   0.2505598
attr(,"correlation")
            (Intercept)        EPc
(Intercept)  1.00000000 0.05726913
EPc          0.05726913 1.00000000
```

2 An extended example: going fishing

The `Headrope` data set gives catch and effort data from a prawn fishery.

- The fishery has 7 *Stock* regions *Tig1*, ..., *Tig7*, West to East.
- The data is for 20 seasons (*YearF*) 1987, ..., 2006. (*Y2K* = year - 2000.)
- There are 236 *Vessel*s, which visit one or more stock regions within a season, each for one or more *Days*.
- The *response* for which a model is required is the total *Catch* in kg, by a vessel within a stock region for a season.
- Additionally the vessels have *Hull* size, engine *Power* and the *Head*rope length they were using recorded. (These are constant within season, but may change between seasons.)

```r
dim(Headrope)
```

```
[1] 8594   13
```

```r
head(Headrope, 2)
```

```
     YearF Y2K Stock Vessel Days Head Hull Power Catch Banana Tiger
0001  1987 -13  Tig1   V008   20   20  133   350  4355   2509   975
0002  1987 -13  Tig1   V012   13   20  134   336  4746   3612   252
     Endeavour King
0001       871    0
0002       882    0
```

```r
Headrope <- Headrope %>% within(YearF <- factor(YearF)) ## needed
Store(Headrope)
```

The purpose of the study was to gain some insight on the marginal effect of headrope length on the catch.

A multiplicative (log-linear) model was suggested, with additive random effects for a) vessel and b) stock regions over seasons.

Two random effects models: the first is the simpler

```r
HRmodel1 <- lmer(log(Catch) ~ 0 + log(Days) + Y2K + log(Head) +
                 log(Power) + log(Hull) + Stock +
                 (1|Vessel) + (1|YearF/Stock), data = Headrope,
                 control = lmerControl(optimizer = "bobyqa"))
HRmodel1_ML <- update(HRmodel1, REML = FALSE)
Store(HRmodel1, HRmodel1_ML, lib = .Robjects)
```

The second has a more elaborate random effect structure:

```r
HRmodel2 <- lmer(log(Catch) ~ 0 + log(Days) + Y2K + log(Head) +
                 log(Power) + log(Hull) + Stock +
                 (1|Vessel) + (0+Stock|YearF), data = Headrope,
                 control = lmerControl(optimizer = "bobyqa"))

boundary (singular) fit:  see ?isSingular

HRmodel2_ML <- update(HRmodel2, REML = FALSE)

boundary (singular) fit:  see ?isSingular

Store(HRmodel2, HRmodel2_ML, lib = .Robjects)
```

The more elaborate model seems justified by AIC, but not BIC!

```
anova(HRmodel1_ML, HRmodel2_ML)

Data: Headrope
Models:
HRmodel1_ML: log(Catch) ~ 0 + log(Days) + Y2K + log(Head) + log(Power) + log(Hull)
HRmodel1_ML:     Stock + (1 | Vessel) + (1 | YearF/Stock)
HRmodel2_ML: log(Catch) ~ 0 + log(Days) + Y2K + log(Head) + log(Power) + log(Hull)
HRmodel2_ML:     Stock + (1 | Vessel) + (0 + Stock | YearF)
            npar   AIC   BIC  logLik deviance  Chisq Df Pr(>Chisq)
HRmodel1_ML   16 11676 11788 -5821.8    11644
HRmodel2_ML   42 11613 11910 -5764.5    11529 114.48 26  4.501e-13
```

The fixed effects estimates are very similar:

```
cbind(m1 = fixef(HRmodel1), m2 = fixef(HRmodel2))
```

```
                    m1          m2
log(Days)   1.16175473 1.16092771
Y2K         0.02742599 0.03477483
log(Head)   0.30269966 0.30166004
log(Power)  0.11566567 0.11414072
log(Hull)   0.20684792 0.20812242
StockTig1   2.84888065 2.88900893
StockTig2   2.39961501 2.43790967
StockTig3   2.14663311 2.18115993
StockTig4   2.32495254 2.35986363
StockTig5   2.41038427 2.44534551
StockTig6   2.55091856 2.56126561
StockTig7   2.19453754 2.17294182
```

Some notes:

- The coefficient on `log(Days)` is slightly larger than 1, (but significantly). A coefficient of 1 would imply that, *mutatis mutandis*, catch is proportional to "effort" (measured in boat days).

- The coefficient of `Y2K` suggests an average fishing power increase in the order of 2.5%–3.5% per year. This looks about right, but it is confounded with change in the stock abundance. Essentially the job of disentangling this confounding is what stock assessment is all about (and why it is so hard).

For reference we include a copy of the summary of the more elaborate model below.

```
print(summary(HRmodel2), correlation = FALSE)

Linear mixed model fit by REML ['lmerMod']
Formula:
log(Catch) ~ 0 + log(Days) + Y2K + log(Head) + log(Power) + log(Hull) +
    Stock + (1 | Vessel) + (0 + Stock | YearF)
   Data: Headrope
Control: lmerControl(optimizer = "bobyqa")

REML criterion at convergence: 11593

Scaled residuals:
     Min       1Q   Median       3Q      Max
-16.3368  -0.3270   0.0071   0.4091   4.9784

Random effects:
 Groups   Name        Variance Std.Dev. Corr
 Vessel   (Intercept) 0.01028  0.1014
 YearF    StockTig1   0.16147  0.4018
          StockTig2   0.11352  0.3369    0.30
          StockTig3   0.02282  0.1511    0.11  0.38
          StockTig4   0.01923  0.1387    0.20  0.76  0.55
          StockTig5   0.02021  0.1422    0.16  0.62  0.62  0.93
```

```
          StockTig6   0.15939  0.3992   -0.07  0.15  0.36  0.22  0.53
          StockTig7   0.17844  0.4224    0.03  0.35  0.26  0.06  0.30
 Residual             0.21124  0.4596

  0.80

Number of obs: 8594, groups:  Vessel, 236; YearF, 20

Fixed effects:
           Estimate Std. Error t value
log(Days)  1.160928   0.004801 241.786
Y2K        0.034775   0.004265   8.153
log(Head)  0.301660   0.049988   6.035
log(Power) 0.114141   0.037573   3.038
log(Hull)  0.208122   0.031699   6.566
```

```
StockTig1  2.889009   0.195859  14.750
StockTig2  2.437910   0.188492  12.934
StockTig3  2.181160   0.175105  12.456
StockTig4  2.359864   0.174618  13.514
StockTig5  2.445346   0.175125  13.963
StockTig6  2.561266   0.193472  13.238
StockTig7  2.172942   0.196103  11.081
optimizer (bobyqa) convergence code: 0 (OK)
boundary (singular) fit: see ?isSingular
```

2.1 A brief look at generalized linear/additive mixed models

Software for GLMMs is still somewhat developmental.

- *glmmPQL* in `MASS` is based on `nlme`, but handles general cases.
- *glmer* from the `lme4` package handles some GLMMs but is restricted in the families it can take. (In particular, *quasipoisson* is NOT included.)

The software for GAMMs also uses a linear ME engine.

- *gamm* from the `mgcv` package uses `nlme` engine,
- *gamm4* from he `gamm4` package (Wood, 2011), uses `lme4` engine, (and so has the same limitations).

Both *gamm* and *gamm4* return a composite object with an `lme` and a `gam` component. Manipulation is tricky.

To illustrate, we construct a GLMM and a GAMM for the Tiger Prawn species split example. The model structure is slightly simplified relative to the working model.

We use two helper functions, *`Hyear`* and *`twoWay`* which will be defined at the end.

First, the GLMM:

```
TModelGLMM <- glmmPQL(Psem/Total ~ ns(Coast, 6) + ns(Sea, 5) +
                      twoWay(DayOfYear, Sea) + ns(Depth, k=5) +
                      Hyear(DayOfYear, 2) + ns(Mud, k=5),
                      random = ~1|Survey,
                      family = quasibinomial, data= Tigers,
                      niter = 40, weights = Total)

iteration 1

iteration 2

iteration 3

iteration 4

Store(TModelGLMM, lib = .Robjects)
```

Note that the random component is defined separately from the main formula, in `nlme` style.

For a GAM with smoothed terms:

```
requireData(mgcv)
TModelGAMM <- gamm(formula = Psem/Total ~ s(Coast, k=5) + s(Sea,k=5) +
                   twoWay(DayOfYear, Sea) +
                   s(DayOfYear, k=5, bs="cc") + s(Depth,k=5) +
                   s(Mud, k=5),
                   knots = list(DayOfYear = seq(0, 364.25, length = 5)),
                   random = list(Survey = ~1),
                   family = quasibinomial, data = Tigers,
                   niterPQL = 40,
                   weights = Total)


 Maximum number of PQL iterations:  40

iteration 1

iteration 2

iteration 3

iteration 4

iteration 5

iteration 6
```

```
iteration 7

Store(TModelGAMM, lib = .Robjects)
```

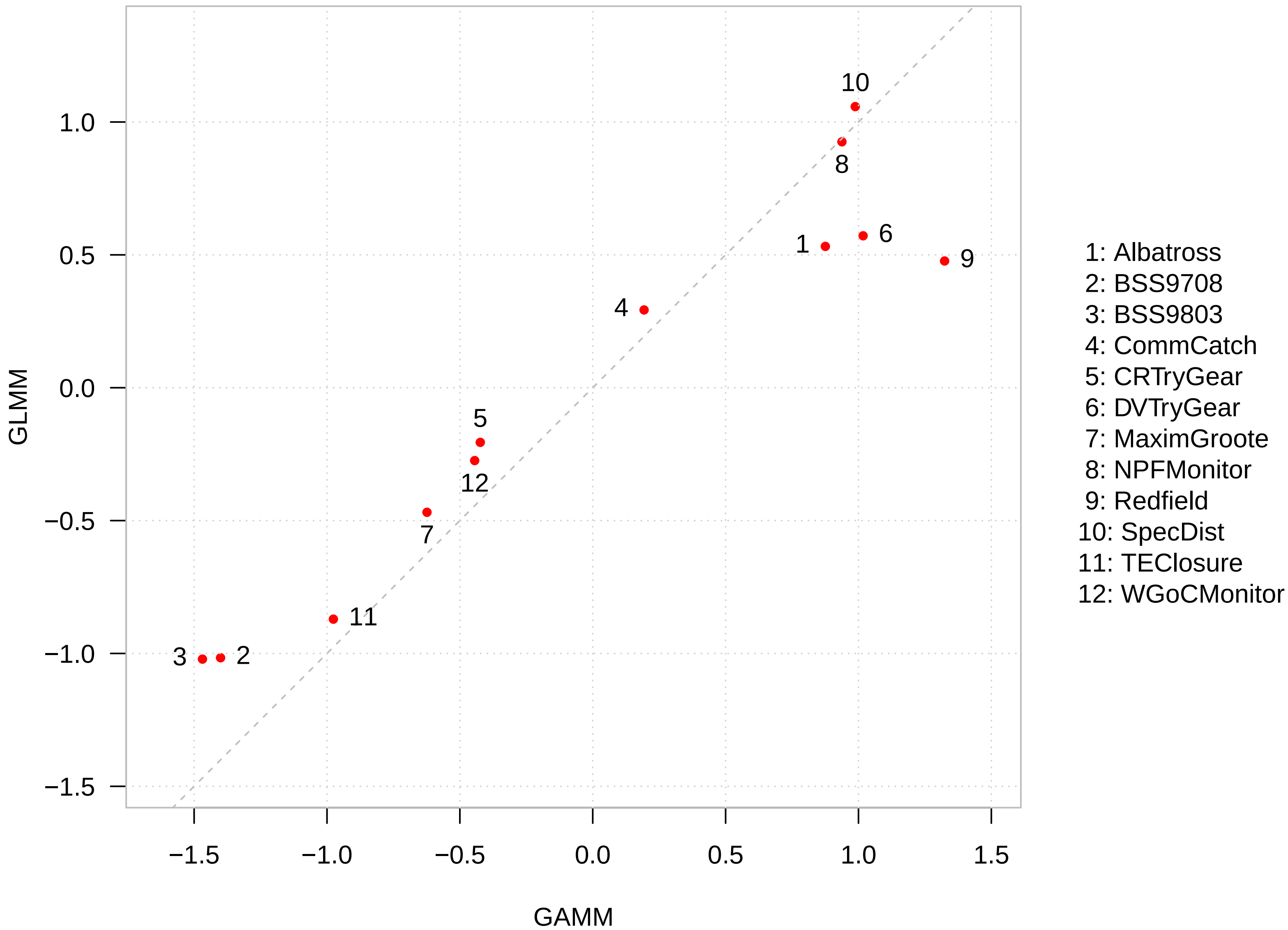
The random effects from these different models are quite similar. We illustrate below.

```r
re1 <- setNames(ranef(TModelGLMM), "GLMM")
re2 <- setNames(ranef(TModelGAMM$lme)$Survey, "GAMM")  ## obscure
(re12 <- cbind(re1, re2))
```

```
                  GLMM       GAMM
Albatross    0.5318898  0.8753641
BSS9708     -1.0162057 -1.4004097
BSS9803     -1.0212852 -1.4685617
CommCatch    0.2927734  0.1931905
CRTryGear   -0.2056465 -0.4231140
DVTryGear    0.5717618  1.0175587
MaximGroote -0.4686942 -0.6236857
NPFMonitor   0.9254647  0.9378444
Redfield     0.4769554  1.3240899
SpecDist     1.0579519  0.9878992
TEClosure   -0.8708619 -0.9759418
WGoCMonitor -0.2741036 -0.4442340
```

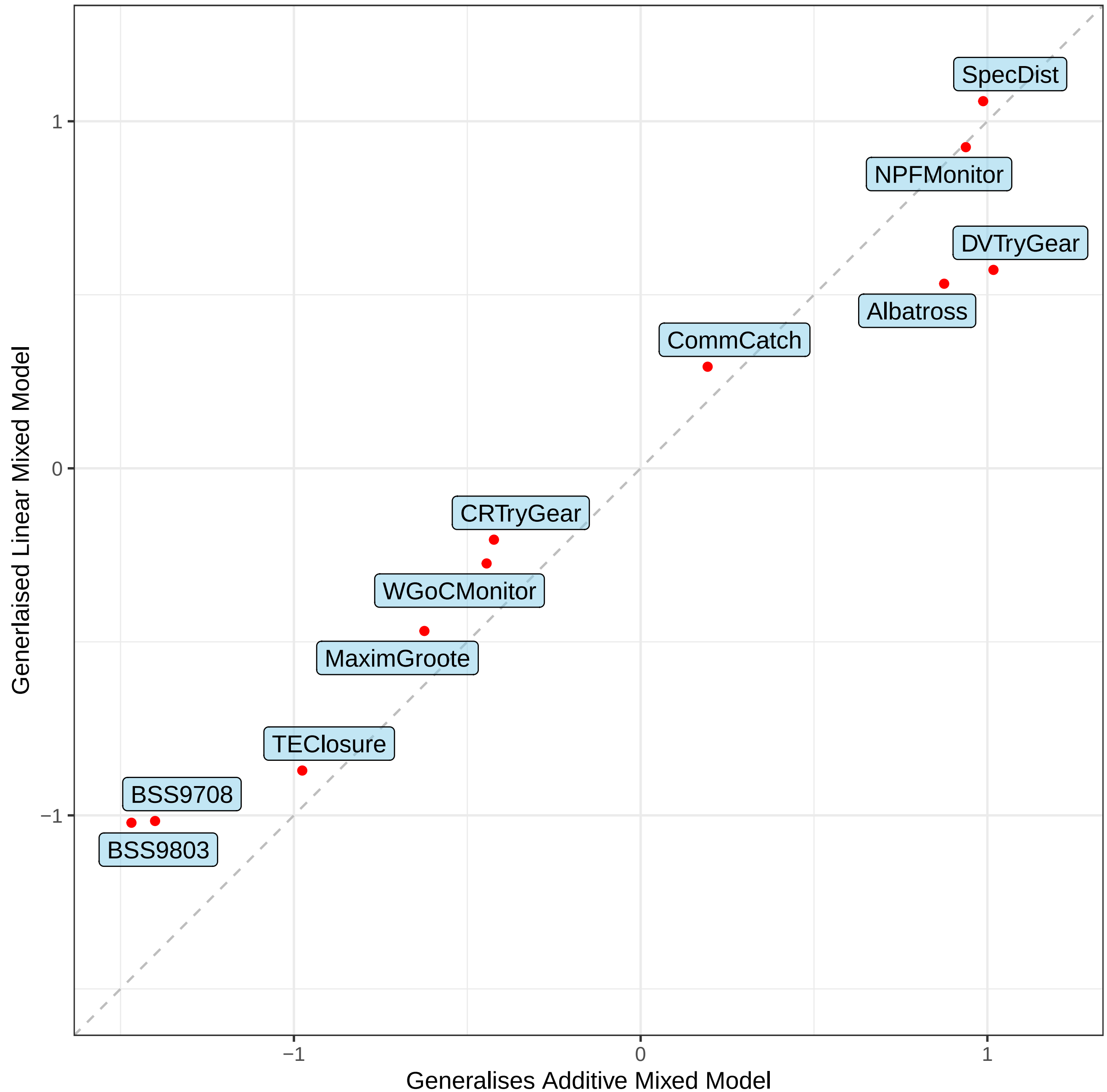
Displaying the correspondence “by hand” with base graphics and plot tricks.

```r
layout(rbind(1:2), widths = c(3.5,0.5), heights = c(3.5, 3.5),
       respect = TRUE)
with(re12, {
  nos <- seq_along(GLMM)
  lim <- range(GLMM, GAMM)
  plot(GLMM ~ GAMM, pch = 20, col="red", bty="n",
       xlim = lim, ylim = lim, asp = 1)
  abline(0, 1, col = "grey", lty = "dashed")
  box(col = "grey")
  grid()
  pos <- avoid(GAMM, GLMM) ## minimise clashes
  text(GLMM ~ GAMM, labels = nos, xpd = NA, pos=pos)
  par(mar = c(0,0,0,0), xpd = NA)
  frame()
  legend("center", paste(format(nos), rownames(re12),
                         sep = ": "), bty="n")
})
```


GLMM
GAMM
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
10
8
1
6
9
4
5
12
7
11
3
2
1: Albatross
2: BSS9708
3: BSS9803
4: CommCatch
5: CRTryGear
6: DVTryGear
7: MaximGroote
8: NPFMonitor
9: Redfield
10: SpecDist
11: TEClosure
12: WGoCMonitor

But wait, there's more.

```r
re12 <- re12 %>% within({
  Survey <- factor(rownames(re1)) %>% reorder(-(GLMM+GAMM), I)
})
ggplot(re12) + aes(x = GAMM, y = GLMM) +
    coord_equal() + xlim(-1.5, 1.2) + ylim(-1.5, 1.2) +
    geom_abline(intercept = 0, slope = 1,
                linetype = "dashed", colour = "grey") +
    geom_point(colour = "red")  +
    ggrepel::geom_label_repel(aes(label = Survey),
                              fill = alpha("sky blue", 0.5)) +
    labs(x = "Generalises Additive Mixed Model",
         y = "Generlaised Linear Mixed Model") +
    theme_bw() +
    theme(text = element_text(family = "sans"))
```


SpecDist
NPFMonitor
DVTryGear
Albatross
CommCatch
CRTryGear
WGoCMonitor
MaximGroote
TEClosure
BSS9708
BSS9803
1
0
−1
−1
0
1
Generlaised Linear Mixed Model
Generalises Additive Mixed Model

3 Count response variables

Three most important models for frequency responses are

Binomial: variance less than the mean,

$$\mathsf{V}[Y] = \mu(1 - \mu/n).$$

Poisson: variance equal to the mean,

$$\mathsf{V}[Y] = \mu.$$

Negative Binomial: variance greater than the mean,

$$\mathsf{V}[Y] = \mu(1 + \mu/\theta).$$

It is clear when a Binomial model is appropriate.

Poisson models are mostly used in *surrogate* models for multinomial situations.

3.1 Negative Binomial models: the `quine` example

Negative Binomial models have proved to be a good option for situations where the response is a count variable and the variance is clearly larger than the mean, but the standard deviation is *roughly proportional* to the mean.

This behaviour often results when the count comes from a process where there are clumps or clusters in the items being counted.

The `quine` data

- Response: `Days` away from school by a student in one school year.
- Predictors: Four classifying factors, `Age`, `Lrn`, `Sex` and `Eth`.

The first clue that this is *not* well modelled as a Poisson response is that the deviance far exceeds the residual degrees of freedom:

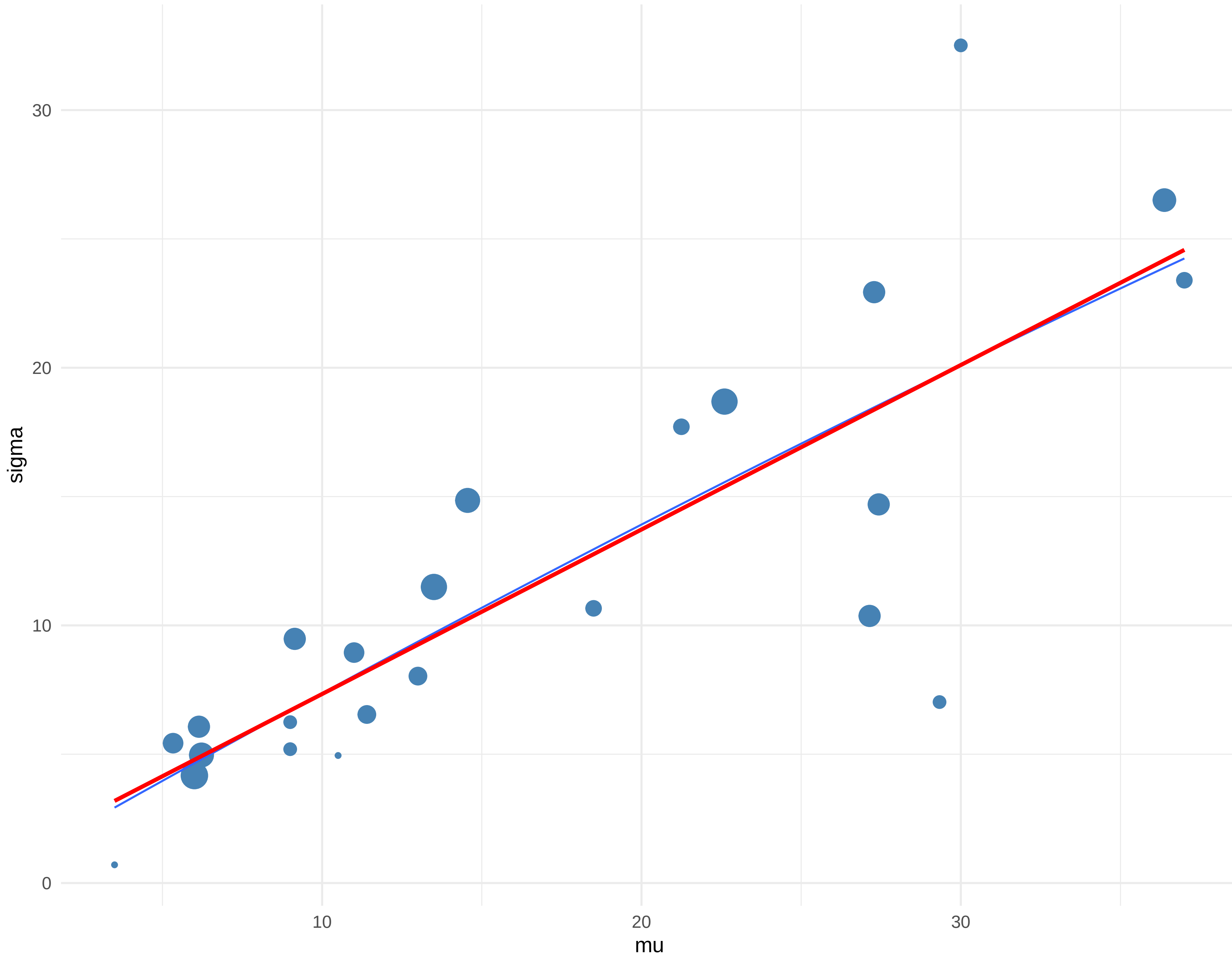
```
head(quine, 2)

  Eth Sex Age Lrn Days
1   A   M  F0  SL    2
2   A   M  F0  SL   11

q0 <- glm(Days ~ Age*Sex*Eth*Lrn, poisson, quine)
c(deviance = deviance(q0), "residual d.f." = df.residual(q0))

     deviance residual d.f.
     1173.899       118.000
```

Now check the form of any mean-variance relationship:

```
quine %>%
  group_by(Age, Sex, Eth, Lrn) %>%
  summarise(n = n(), mu = mean(Days), sigma = sd(Days), .groups = 'drop') %>%
  na.omit %>%
  unclass %>%
  data.frame -> quine_sum
head(quine_sum, 4)
```

```
  Age Sex Eth Lrn n    mu     sigma
1  F0   F   A  AL 4 21.25 17.708284
2  F0   F   N  AL 4 18.50 10.661457
3  F0   M   A  AL 5 13.00  8.031189
4  F0   M   A  SL 3  9.00  6.244998
```

```
ggplot(quine_sum) + aes(x = mu, y = sigma, size = n) +
  geom_point(colour = "steel blue") +
  geom_smooth(method = "lm", formula = y ~ poly(x, 2), se = FALSE, size = 0.5) +
  geom_smooth(method = "lm", formula = y ~ x, se = FALSE, colour = "red") +
  theme(legend.position = "none")
```



30
20
10
0
sigma
10
20
30
mu

Modelling

Negative binomial model has heuristic and empirical support. Let's fit one and see how it goes.

```
quine_full <- glm.nb(Days ~ Age*Sex*Eth*Lrn, quine)
quine_step <- step_BIC(quine_full) ## BIC penalty
dropterm(quine_step) %>%  booktabs(digits = c(0,0,2,2,6))
```

	Df	BIC	LRT	Pr(Chi)
<none>		1132.80		
Age:Sex	3	1136.46	18.62	0.000328
Sex:Eth:Lrn	1	1140.23	12.42	0.000424

(Ex.: check diagnostics to verify things look OK.)

3.2 Gladstone Bream recruitment index

Data from a monitoring survey of bream recruitment in the Gladstone harbour region.

- Four monthly surveys of 26 Sites, December – March
- 20 casts per visit. Response: total bream recruits per visit
- Every site monitored in 2015-16 & 2016-17. Some sporadic visits from 2011-12.
- Some environmental variables recorded, but very roughly.
- Aim: an annual site index for the GHHP health card for each Site.

```
Bream <- subset(GladstoneBream, select = c(Trip, Site:Rock, Casts, Bream))
names(Bream) %>% noquote
```

```
 [1] Trip   Site   Date   Year   Month  Tidal  Depth  Sand   Mud    Gravel
[11] Rock   Casts  Bream
```

Models

Some experience has led to a set of useful fixed effects and the needs of the index specify a random effect structure. First estimate the NB model including θ.

```
NB_Bream0 <- glmer.nb(Bream ~ offset(log(Casts)) + Depth + Rock + Month +
                        (1|Site) + (1|Year/Site), data = Bream, nAGQ = 0)
getME(NB_Bream0, "glmer.nb.theta")
```

```
[1] 2.10841
```

There are some advantages in fixing θ at some reasonable value and estimating the parameters of interest with more accuracy.

```
NB_Bream <- glmer(Bream ~ offset(log(Casts)) + Depth + Rock + Month + (1|Site) +
                    (1|Year/Site), family = negative.binomial(theta = 2),
                  mustart = fitted(NB_Bream0), data = Bream, nAGQ = 1)
```

Variance components

Our first step is to extract the variance components.

```r
(v <- VarCorr(NB_Bream) %>% unlist ) ## these are variances!
```

```
 Site:Year       Site       Year 
0.12782639 0.68712447 0.09655783 
```

```r
sigS <- sqrt(v[["Site"]])
sigYS <- sqrt(v[["Year"]] + v[["Site:Year"]])
c(Site = sigS, Year_x_Site = sigYS)
```

```
       Site Year_x_Site 
  0.8289297   0.4736921 
```

The score will be made up of the `Year` and `Site:Year` random effects (BLUPs), but we will produce a `Site` score as well, which gives some idea of the relative productivities of the sites themselves.

```r
(RE <- ranef(NB_Bream)) %>% names
```

```
[1] "Site:Year" "Site"      "Year"     
```

```r
x <- RE[["Site"]]
(S <- data.frame(Site = factor(rownames(x), levels = rownames(x)),
                 Score = pnorm(x[["(Intercept)"]], sd = sigS)))  %>% head(2)
```

```
             Site     Score
1 Ramsay Crossing 0.8387646
2  Munduran Creek 0.3076080
```

```r
x <- RE[["Year"]]
(Y <- data.frame(Year = factor(rownames(x), levels = rownames(x)),
                 Y_BLUP = x[["(Intercept)"]]))                   %>% head(2)
```

```
   Year     Y_BLUP
1 11-12  0.2316966
2 12-13 -0.3604189
```

```r
(x <- RE[["Site:Year"]])                                         %>% head(3)
```

```
                      (Intercept)
Ramsay Crossing:15-16   0.2985097
Ramsay Crossing:16-17  -0.1459388
Munduran Creek:11-12    0.1878135
```

This is more involved.

```r
YS <- data.frame(nam = rownames(x), stringsAsFactors = FALSE) %>%
  separate(nam, c("Site", "Year"), sep = ":") %>%
  within({
    Site <- factor(Site, levels = levels(S$Site))
    Year <- factor(Year, levels = levels(Y$Year))
    YS_BLUP <- x[["(Intercept)"]]
  }) %>%
  left_join(Y, by = "Year") %>%
  within({
    YS <- pnorm(Y_BLUP + YS_BLUP, sd = sigYS)
    Y_BLUP <- YS_BLUP <- NULL
  }) %>% arrange(Year) %>%
  pivot_wider(names_from = Year, values_from = YS) %>%
  arrange(Site)
booktabs(YS)
```

	Site	11–12	12–13	13–14	14–15	15–16	16–17
1	Ramsay Crossing					0.58	0.62
2	Munduran Creek	0.81	0.24	0.33	0.49	0.37	0.71
3	Black Swan				0.67	0.06	0.97
4	Targinnie Creek	0.67	0.18		0.87	0.16	0.74
5	Graham Creek				0.77	0.31	0.56
6	Hobble Gully				0.56	0.49	0.62
7	Mud Island					0.17	0.79
8	Boat Creek		0.21	0.43	0.80	0.28	0.47
9	Little Enfield Creek				0.76	0.28	0.71
10	Barney Point Pond		0.22	0.44	0.70	0.24	0.57
11	Beecher Creek	0.84	0.12	0.40	0.66	0.29	0.76
12	Old Bruce Highway Bridge				0.45	0.35	0.83
13	Callemondah	0.61	0.08	0.39	0.68	0.49	0.91
14	Farmers Point					0.09	0.91
15	Gatcombe Anchorage					0.34	0.53
16	Wappentake Creek		0.22	0.49	0.59	0.32	0.67
17	South Trees					0.42	0.64
18	Crematorium Pool					0.27	0.85
19	Old Boyne	0.69	0.30		0.61	0.44	0.71
20	Boyne Highway				0.54	0.49	0.78
21	Broadacres					0.36	0.73
22	Iveragh					0.41	0.70
23	Oaky					0.49	0.76
24	7 Mile					0.50	0.75
25	Worthington					0.27	0.69
26	Sandy Bridge					0.39	0.80

Finally, the main result:

```
Scores <- merge(S, YS, by = "Site") %>%
  within({
    Site <- factor(as.character(Site), levels = levels(Bream$Site))
  }) %>% arrange(Site)
Scores
```

Site	Score	11–12	12–13	13–14	14–15	15–16	16–17
Ramsay Crossing	0.84					0.58	0.62
Munduran Creek	0.31	0.81	0.24	0.33	0.49	0.37	0.71
Black Swan	0.68				0.67	0.06	0.97
Targinnie Creek	0.53	0.67	0.18		0.87	0.16	0.74
Graham Creek	0.31				0.77	0.31	0.56
Hobble Gully	0.37				0.56	0.49	0.62
Mud Island	0.16					0.17	0.79
Boat Creek	0.09		0.21	0.43	0.80	0.28	0.47
Little Enfield Creek	0.60				0.76	0.28	0.71
Barney Point Pond	0.05		0.22	0.44	0.70	0.24	0.57
Beecher Creek	0.51	0.84	0.12	0.40	0.66	0.29	0.76
Old Bruce Highway Bridge	0.35				0.45	0.35	0.83
Callemondah	0.78	0.61	0.08	0.39	0.68	0.49	0.91
Farmers Point	0.25					0.09	0.91
Gatcombe Anchorage	0.06					0.34	0.53
Wappentake Creek	0.26		0.22	0.49	0.59	0.32	0.67
South Trees	0.48					0.42	0.64
Crematorium Pool	0.77					0.27	0.85
Old Boyne	0.86	0.69	0.30		0.61	0.44	0.71
Boyne Highway	0.81				0.54	0.49	0.78
Broadacres	0.60					0.36	0.73
Iveragh	0.62					0.41	0.70
Oaky	0.93					0.49	0.76
7 Mile	0.94					0.50	0.75
Worthington	0.19					0.27	0.69
Sandy Bridge	0.87					0.39	0.80

Notes

- The offset of `log(Casts)` allows for minor fluctuations from the nominal 20 casts per visit and recognises that, other things being equal, catch should be proportional to effort.
- The `Month` fixed effect allows for systematic changes in productivity over the calendar year.
- The `Site` score is an estimate of the (relative) productivity of the site for Bream recruitment overall, allowing for the minimal effect of `Depth` and `Rock` in the fixed effects.
- The `Year` × `Site` scores give a measure of the *change* in catch rate from year to year *relative* to what might have been expected given the productivity of the site.
- Converting scores to a $(0, 1)$ scale is something required by the Health Card protocol; this is only *one possible* way to do it.

A Theme functions

ggplot2 release 2.0.0 (or later) has a wide range of pre-defined themes, as well as theme generating and theme modifying functions. A list is:

```
apropos("^(theme$|theme_)") %>% noquote

 [1] theme                  theme_base             theme_bw
 [4] theme_calc             theme_classic          theme_clean
 [7] theme_dark             theme_economist        theme_economist_white
[10] theme_excel            theme_excel_new        theme_few
[13] theme_fivethirtyeight theme_foundation       theme_gdocs
[16] theme_get              theme_gray             theme_grey
[19] theme_hc               theme_igray            theme_light
[22] theme_linedraw         theme_map              theme_minimal
[25] theme_pander           theme_par              theme_replace
[28] theme_set              theme_solarized        theme_solarized_2
[31] theme_solid            theme_stata            theme_test
[34] theme_tufte            theme_update           theme_void
[37] theme_wsj
```

B Two helper functions

These are needed to define harmonic terms and interactions.

```
Harm <- function (theta, k = 4) {
  X <- matrix(0, length(theta), 2 * k)
  nam <- as.vector(outer(c("c", "s"), 1:k, paste, sep = ""))
  dimnames(X) <- list(names(theta), nam)
  m <- 0
  for (j in 1:k) {
    X[, (m <- m + 1)] <- cos(j * theta)
    X[, (m <- m + 1)] <- sin(j * theta)
  }
  X
}
Hyear <- function(x, k = 4)
    Harm(2*base::pi*x/365.25, k)
twoWay <- local({
  `%star%` <- function(X, Y) { ## all column-products
    X <- as.matrix(X)
    Y <- as.matrix(Y)
    stopifnot(is.numeric(X), is.numeric(Y),
```

```r
              nrow(X) == nrow(Y))
    XY <- matrix(NA, nrow(X), ncol(X)*ncol(Y))
    k <- 0
    for(i in 1:ncol(X))
        for(j in 1:ncol(Y)) {
          k <- k+1
          XY[, k] <- X[, i] * Y[, j]
        }
    XY
  }
  function(day, sea, k = c(3,2)) {
    Hyear(day, k[1]) %star% splines::ns(sea, k[2])
  }
})
Store(Harm, Hyear, twoWay, lib = .Robjects)
```

References

Cribari-Neto, F. and A. Zeileis (2010). Beta regression in **R**. *Journal of Statistical Software 34*(2), 1–24.

Venables, W. N. and B. D. Ripley (2002). *Modern Applied Statistics with* **S** (Fourth ed.). New York: Springer. ISBN 0-387-95457-0.

Wood, S. (2011). *gamm4: Generalized additive mixed models using* `mgcv` *and* `lme4`. CRAN. R package version 0.1-5.

Session information

Date: 2021-01-29

- R version 4.0.3 (2020-10-10), `x86_64-pc-linux-gnu`
- Running under: `Ubuntu 20.04.1 LTS`
- Matrix products: default
- BLAS: `/usr/lib/x86_64-linux-gnu/blas/libblas.so.3.9.0`
- LAPACK: `/usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.9.0`
- Base packages: base, datasets, graphics, grDevices, methods, stats, utils
- Other packages: dplyr 1.0.3, english 1.2-5, forcats 0.5.1, ggplot2 3.3.3, ggthemes 4.2.4, gridExtra 2.3, knitr 1.31, lattice 0.20-41, lme4 1.1-26, Matrix 1.3-2, patchwork 1.1.1, purrr 0.3.4, readr 1.4.0, scales 1.1.1, SOAR 0.99-11, stringr 1.4.0, tibble 3.0.5, tidyr 1.1.2, tidyverse 1.3.0, WWRCourse 0.2.3, WWRData 0.1.0, WWRGraphics 0.1.2, WWRUtilities 0.1.2, xtable 1.8-4
- Loaded via a namespace (and not attached): assertthat 0.2.1, backports 1.2.1, boot 1.3-26, broom 0.7.3, cellranger 1.1.0, cli 2.2.0, colorspace 2.0-0, compiler 4.0.3, crayon 1.3.4, DBI 1.1.1, dbplyr 2.0.0, digest 0.6.27, ellipsis 0.3.1, evaluate 0.14, fansi 0.4.2, farver 2.0.3, fractional 0.1.3, fs 1.5.0, generics 0.1.0, ggrepel 0.9.1, glue 1.4.2, grid 4.0.3, gtable 0.3.0, haven 2.3.1, highr 0.8, hms 1.0.0, httr 1.4.2, iterators 1.0.13, jsonlite 1.7.2, labeling 0.4.2, lazyData 1.1.0, lifecycle 0.2.0, lubridate 1.7.9.2, magrittr 2.0.1, MASS 7.3-53, mgcv 1.8-33, minqa 1.2.4, modelr 0.1.8, munsell 0.5.0, nlme 3.1-151, nloptr 1.2.2.2, parallel 4.0.3, PBSmapping 2.73.0, pillar 1.4.7, pkgconfig 2.0.3, R6 2.5.0, randomForest 4.6-14, Rcpp 1.0.6, readxl 1.3.1, reprex 1.0.0, rlang 0.4.10, rpart 4.1-15, rstudioapi 0.13, rvest 0.3.6, splines 4.0.3, statmod 1.4.35, stringi 1.5.3, tidyselect 1.1.0, tools 4.0.3, vctrs 0.3.6, withr 2.4.1, xfun 0.20, xml2 1.3.2