Session 11

More on mixed effects modelling

A Generalized Linear Mixed Model

- Recovery of the benthos after trawling on the Great Barrier Reef
- 6 Control plots, 6 Impact plots
- Visited 2 times prior to treatment and 4 times afterwards – longitudinal study
- Inspection is by Benthic sled. The total number of animals of each species is counted for each transect.
- Mean trawl intensity in the transect
- Total ‘swept area’ of the sled video camera
- Topography (Shallow or Deep)

Model

- For each species the total count for a control or pre-experiment treatment plot has a distribution:

$$Y_{pmd} \mid C_m, E_{mp} \sim \mathrm{Po}(\exp(\tau_d + C_m + E_{mp})), \quad C_m \sim N(0, \sigma_C^2), \quad E_{mp} \sim N(0, \sigma_E^2)$$

- For a post-experiment treatment plot the distribution is similar, but the Poisson mean has additional terms

$$Y_{pmd} \mid C_m, E_{mp} \sim \mathrm{Po}(\exp(\tau_d + n_s(t) + \gamma_m + C_m + E_{mp}))$$

  - These extra terms are a spline term in mean trawl intensity and a factor term to allow for recovery

Some functions and data

```
Species <- scan("SpeciesNames.csv", what = "")

match(Species, names(Benthos))

bin <- function(fac, all = F) {
#
# binary matrix for a factor
#
   fac <- as.factor(fac)
   X <- outer(fac, levels(fac), "==") + 0
   if(all) X else X[, -1]
}
nsi <- function(x, k = 3, Intensity = Benthos$Intensity) {
#
# natural spline in intensity
#
  knots <- as.vector(quantile(unique(Intensity), 0:k/k))
  ns(x, knots = knots[2:k], Boundary.knots = knots[c(1, k + 1)])
}

guard <- function(x) ifelse(x < -5, NA, x)
```

Setting up the prediction data

```
vars <- c("Cruise", "Plot", "Months", "Treatment",
          "Time", "Impact", "Topography", "Unity")
vals <- with(Benthos, tapply(Intensity, Impact, mean))
msa <- mean(Benthos$SweptArea)

newData <- transform(Benthos[, vars],
  Months = as.numeric(as.character(Months)),
  Intensity = vals[factor(Impact)],
  SweptArea = rep(msa, nrow(Benthos)))

newData <- newData[order(Benthos$Months), ]
```

The key species

```
print(Species, quote = F)
 [1] Alcyonacea                Annella.reticulata
 [3] Ctenocella.pectinata      Cymbastela.coralliophila
 [5] Dichotella.divergens      Echinogorgia.sp.
 [7] Hypodistoma.deeratum      Ianthella.flabelliformis
 [9] Junceella.fragilis        Junceella.juncea
[11] Nephtheidae               Porifera
[13] Sarcophyton.sp.           Scleractinia
[15] Solenocaulon.sp.          Subergorgia.suberosa
[17] Turbinaria.frondens       Xestospongia.testudinaria
```

- These are names of some columns in the `Benthos` data frame
- We need to fit the master model for each and produce plots of the fixed and random effects

The trick: fitting multiple similar models

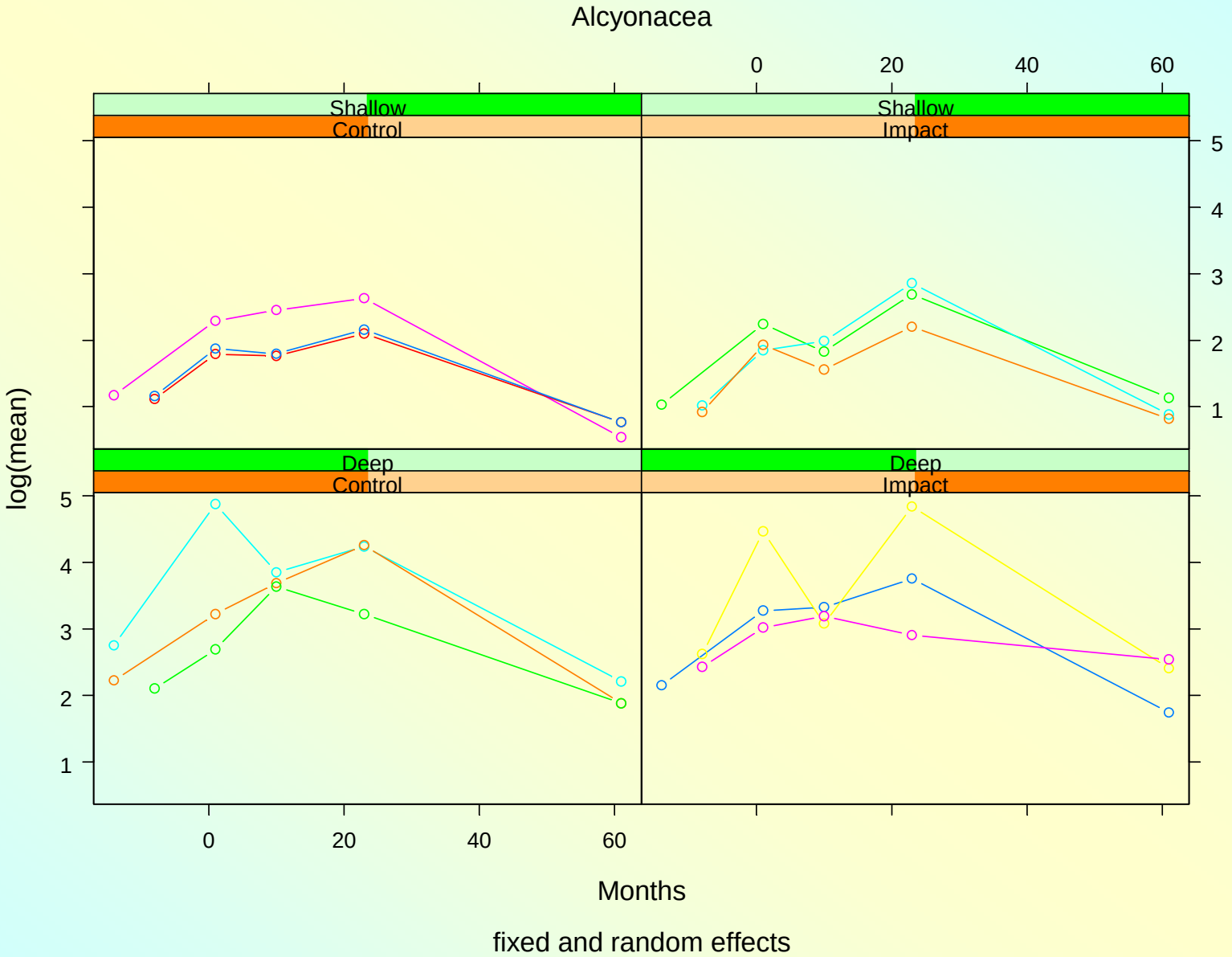
```
fitCommand <- Quote({
fm <- glmmPQL(Species ~ Topography +
  I(Impact * cbind(nsi(Intensity),
  bin(Time))) +
  offset(log(SweptArea)),
  random = ~1|Cruise/Plot,
  family = poisson, data = Benthos,
  verbose = T)
})
```

Putting the trick to work

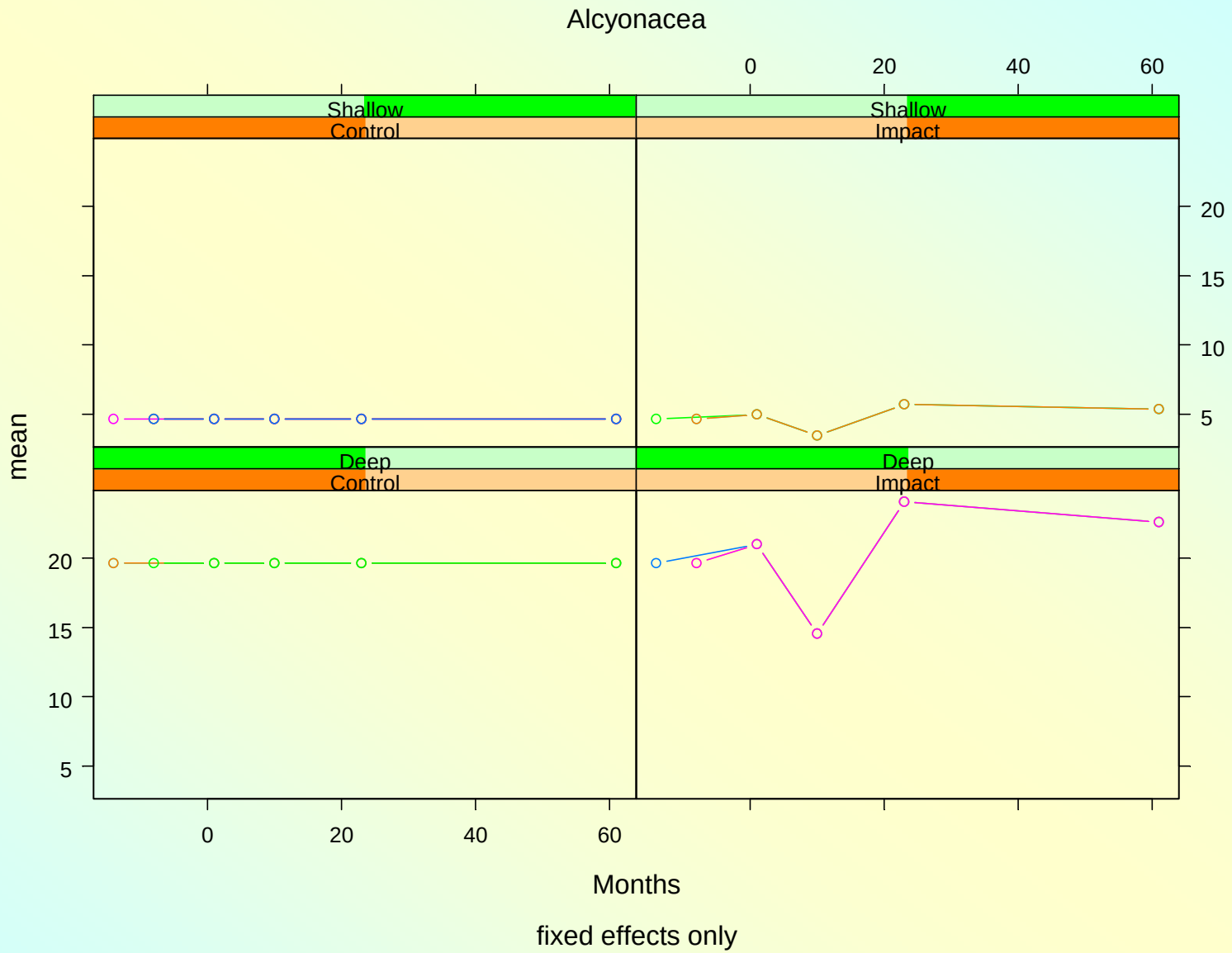
```
spno <- 1

### --- start of main loop

mname <- paste("GLMM", Species[spno], sep=".")
sname <- Species[spno]
thisFitCommand <- do.call("substitute", list(fitCommand,
   list(fm = as.name(mname), Species = as.name(sname))))

eval(thisFitCommand)
print(spno <- spno + 1)

### --- end of main loop
objects(pat = "GLMM.*")
[1] "GLMM.Alcyonacea"
[2] "GLMM.Annella.reticulata"
[3] "GLMM.Ctenocella.pectinata"
[4] "GLMM.Cymbastela.coralliophila"
```

Plots

```
pfm <- predict(GLMM.Alcyonacea, newData) + log(msa)
pfm0 <- predict(GLMM.Alcyonacea, newData, level=0) + log(msa)
graphics.off()

trellis.device()
xyplot(guard(pfm) ~ Months|Treatment*Topography, newData,
  groups = Plot, type="b",
   main = "Alcyonacea", ylab = "log(mean)",
   sub = "fixed and random effects")


xyplot(exp(pfm0) ~ Months|Treatment*Topography, newData,
  groups = Plot, type="b",
   main = "Alcyonacea", ylab = "mean",
   sub = "fixed effects only")
```



fixed and random effects



Variance components

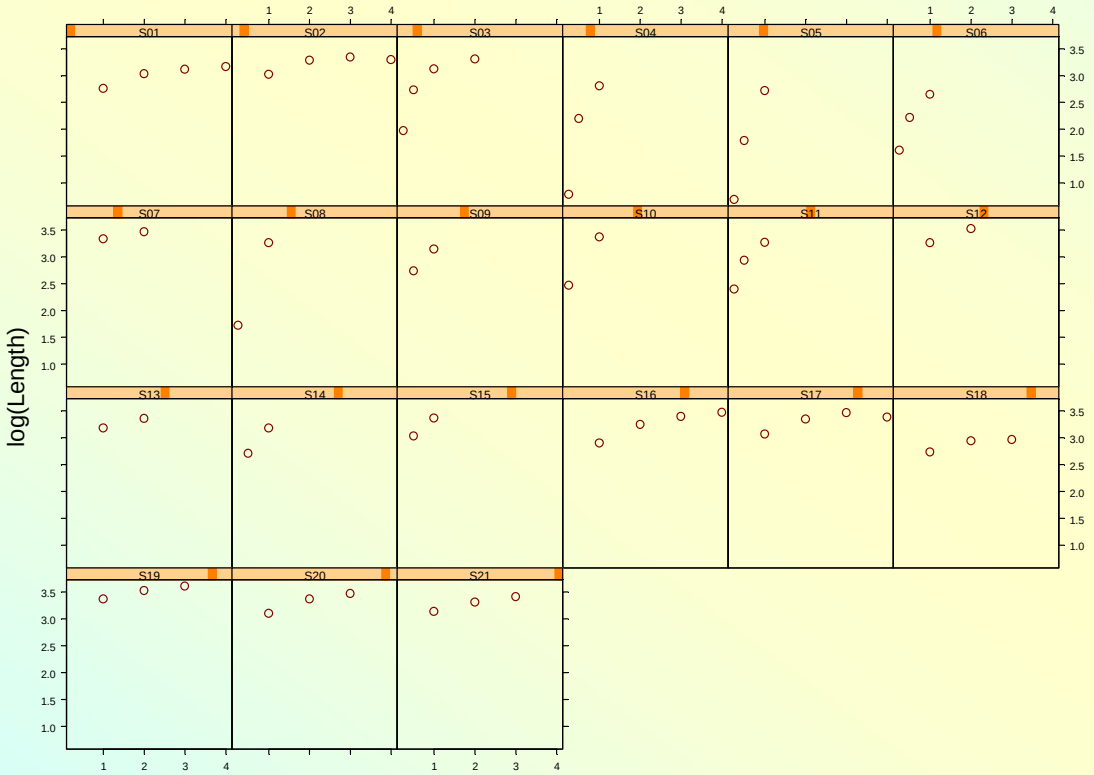
```
summary(GLMM.Alcyonacea, corr = F)
Linear mixed-effects model fit by maximum likelihood
 Data: Benthos
       AIC      BIC    logLik
  338.8218 371.0074 -157.4109

Random effects:
 Formula:  ~ 1 | Cruise
        (Intercept)
StdDev:   0.6677526

 Formula:  ~ 1 | Plot %in% Cruise
        (Intercept) Residual
StdDev:   0.5896012 2.509969

Variance function:
 Structure: fixed weights
 Formula:  ~ invwt
Fixed effects: Alcyonacea ~ Topography + I(Impact *
   cbind(nsi(Intensity),    bin(Time))) + offset(log(SweptArea)) …
```

Comments

- In this case, produces a result where the components of variance are smaller than the underlying Poisson component.
- A useful way of isolating various parts of the model to reveal (perhaps speculatively!) the fixed pattern
- A good way of handling overdispersion – close connection between GLMMs and Negative Binomial Models
- Still somewhat controversial. The inference process is still a matter of debate.

A third non-linear example: the muscle data

```
xyplot(log(Length) ~ Conc | Strip, muscle, as.table = T,
    xlab = "CaCl concentration")
```



An initial fit: fixed effects only

- Suggested model:

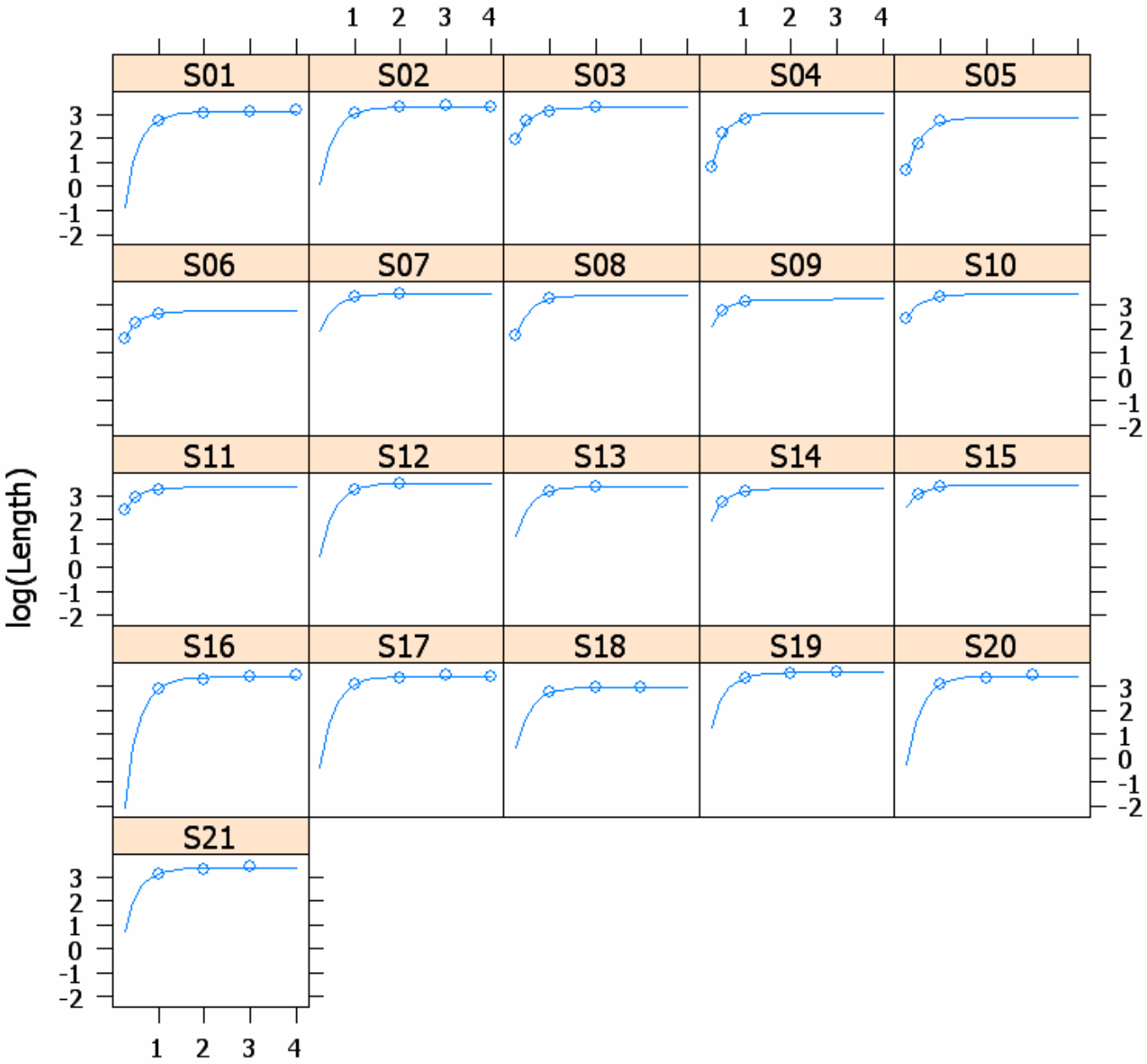
$$\log(\ell_{ij}) = \alpha_j + \beta_j \rho^{c_{ij}} + \varepsilon$$

- 43 parameters with only 61 points. Special care needs to be taken with fitting the model
- We use the "plinear" algorithm both to simplify the model specification and to make the fitting process more robust
- Need to specify the non-linear parameters only.
- Then re-fit the model in a standard way to simplify predictions from it

```
X <- model.matrix(~Strip - 1, muscle)

muscle.nls1 <- nls(log(Length) ~ cbind(X,
   X*rho^Conc), muscle, start = c(rho = 0.1),
   algorithm = "plinear", trace = T)
......
b <- as.vector(coef(muscle.nls1))

init <- list(rho = b[1],
   alpha = b[2:22], beta = b[23:43])
muscle.nls2 <- nls(log(Length) ~ alpha[Strip] +
   beta[Strip]*rho^Conc, muscle, start = init,
   trace = T)
......
```

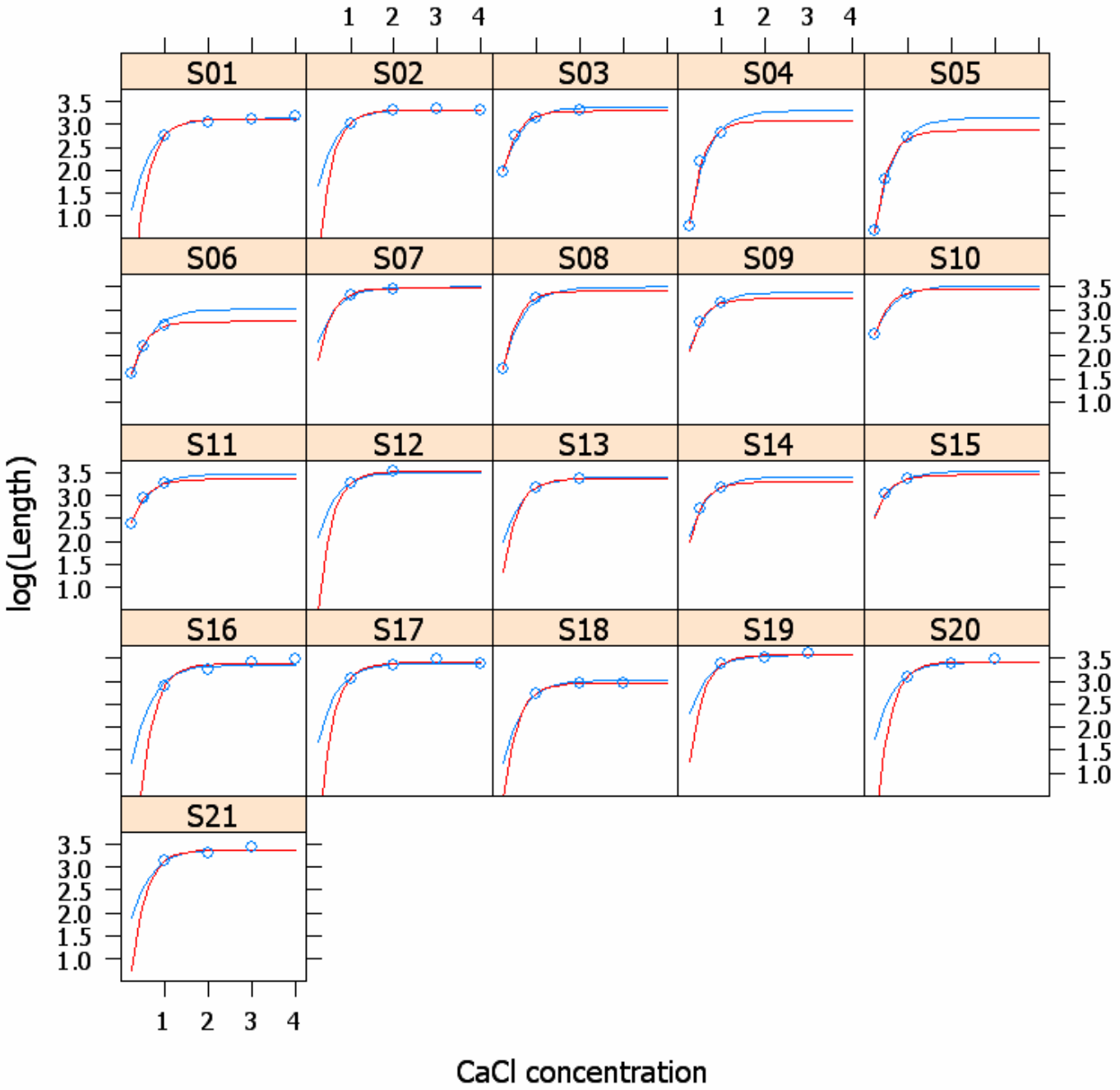
Prediction and presentation of the fit

```
Muscle <- expand.grid(Conc = seq(0.25, 4, len=20),
       Strip = levels(muscle$Strip), Length = 0)
Muscle <- rbind(muscle, Muscle)
Muscle <- Muscle[order(Muscle$Strip, Muscle$Conc), ]
Muscle$fit <- predict(muscle.nls2, Muscle)

xyplot(fit ~ Conc|Strip, Muscle, type = "l",
   subscripts = T,
panel = function(x, y, subscripts, ...) {
   panel.xyplot(x, y, ...)
   panel.points(x, log(Muscle$Length[subscripts]))
}, as.table = T, ylab = "log(Length)",
   xlab = "CaCl concentration")
```


S01
S02
S03
S04
S05
S06
S07
S08
S09
S10
S11
S12
S13
S14
S15
S16
S17
S18
S19
S20
S21
log(Length)
CaCl concentration

Fitting a mixed effects model

```
muscle.nlme <- nlme(log(Length) ~ alpha +
   beta*rho^Conc, muscle,
  fixed = rho+alpha+beta ~ 1,
  random = alpha + beta ~ 1|Strip,
  start = ival)

Muscle$RandomFit <- predict(muscle.nlme, Muscle)

xyplot(RandomFit ~ Conc|Strip, Muscle, type = "l",
   subscripts = T,
panel = function(x, y, subscripts, ...) {
   panel.xyplot(x, y, ...)
   panel.lines(x, Muscle$fit[subscripts], col = "red")
   panel.points(x, log(Muscle$Length[subscripts]))
}, as.table = T, ylab = "log(Length)",
       xlab = "CaCl concentration")
```


S01
S02
S03
S04
S05
S06
S07
S08
S09
S10
S11
S12
S13
S14
S15
S16
S17
S18
S19
S20
S21
log(Length)
CaCl concentration

```
summary(muscle.nlme)
Nonlinear mixed-effects model fit by maximum likelihood
  Model: log(Length) ~ alpha + beta * rho^Conc
Random effects:
 Formula: list(alpha ~ 1, beta ~ 1)
 Level: Strip
 Structure: General positive-definite, Log-Cholesky parametrization
         StdDev     Corr
alpha    0.16991613 alpha
beta     0.88268573 0.35
Residual 0.07692169

Fixed effects: rho + alpha + beta ~ 1
          Value  Std.Error DF  t-value p-value
rho    0.094083 0.01349832 37  6.96996       0
alpha  3.363546 0.04486974 37 74.96247       0
beta  -2.817441 0.29503865 37 -9.54940       0
 Correlation:
      rho   alpha
alpha 0.371
beta  0.546 0.284

Standardized Within-Group Residuals:
        Min          Q1         Med          Q3         Max
-1.66713635 -0.34229465 -0.08582487  0.35210281  2.84334516

Number of Observations: 60
Number of Groups: 21
```

Final comments

- Non-linear regression offers a way of integrating empirical and “process” models
- If kinds of non-linear regression are to be repeatedly done, some investment in selfStart models pays off.
- The “plinear” algorithm offers an effective way of using the simplicity of linear parameters
- Random effects can be very tricky, but offer a way of “borrowing strength” from one group to another, in order to gain a more holistic representation of the process generating the data.