CHAPTER

10

Normal Two-Way ANOVA

OUTLINE

10.1 INTRODUCTION: MAIN AND INTERACTION EFFECTS

We now extend the one-way analysis of variance (ANOVA) model by adding another factor and arrive at a two-way ANOVA. We only consider fixed effects here. There are two ways in which the effects of two factors A and B can be combined, and the associated models are called main-effects and interaction-effects model. In the main-effects model, the effects of A and B are additive, i.e., the effect of one level of factor A, say a_1, does not depend on whether it is assessed at one level of B, say b_1, or at another, say b_2. In contrast, with an interaction between A and B, some or all effects depend on some or all of each other and the effect of a_1 may not be identical when assessed at b_1 or at b_2. Interaction is symmetric, so the effect of b_1 will in general also not be the same whether assessed at a_1 or at a_2. However, the interaction model is still linear, since effects are simply added together, only with an additional set of effects: those for the combination of each level of two or more factors. When factors are considered fixed, then depending on the model, not all effects will be estimable, see later (10.5.1). In contrast, in a random-effects model with interaction, all effects will in general be estimable.

In Chapter 6 we already saw the linear models for the two-way ANOVA with interaction. In short, the effects parameterization for a model with two factors A (with j levels) and B (with k levels) is this:

$$y_i = \alpha + \beta_{j(i)} * A_i + \delta_{k(i)} * B_i + \gamma_{jk(i)} * A_i * B_i + \varepsilon_i,$$

while the means parameterization is this:

$$y_i = \alpha_{jk(i)} * A_i * B_i + \varepsilon_i$$

In both cases, we need to assume a distribution for the residuals to complete the model description:

$$\varepsilon_i \sim Normal\,(0, \sigma^2).$$

We will use the beautiful mourning cloak (Fig. 10.1) as an illustration for this chapter and assume that we measured wing length of butterflies in each of three elevation classes in five different populations and that the effects of these factors interact. Table 10.1 shows the meaning of the coefficients in the linear model for the effects parameterization.

FIGURE 10.1 Mourning cloak (*Nymphalis antiopa*), Switzerland, 2006. (*Photo: T. Marent*)

TABLE 10.1 The 15 Parameters Estimated in the Effects Parameterization of a Two-Way ANOVA with Interaction for the Mourning Cloak Example.

Intercept	Elevation2	Elevation3
pop2	pop2.elevation2	pop2.elevation3
pop3	pop3.elevation2	pop3.elevation3
pop4	pop4.elevation2	pop4.elevation3
pop5	pop5.elevation2	pop5.elevation3

If the population factor has `n.pop` levels and the elevation factor `n.elev` levels, we have one intercept, `n.pop-1 = 4` effects for the population factor, `n.elev-1 = 2` effects for the elevation factor and `(n.pop-1)* (n.elev-1) = 8` effects for the interaction between population and elevation. This adds up to the 15 degrees of freedom that it will cost us to fit this model to a data set that contains observations in every cell (combination of levels) in the cross-classification of population and elevation.

10.2 DATA GENERATION

We assume five populations with 12 butterflies were measured in each and that of these 12, four butterflies were studied at each of three elevation classes (low, medium, high). Wing length differs with elevation, perhaps because butterflies hatch at different size at different elevation or because of different size-dependent predation owing to different bird communities at different elevations. Furthermore, the relationship between wing length and elevation class is not homogeneous among the five studied populations so there is a population-elevation interaction. Residual wing length standard deviation will be 3.

```
# Choose sample size
n.pop <- 5
n.elev <- 3
nsample <- 12
n <- n.pop * nsample

# Create factor levels
pop <- gl(n = n.pop, k = nsample, length = n)
elev <- gl(n = n.elev, k = nsample / n.elev, length = n)

# Choose effects
baseline <- 40                    # Intercept
pop.effects <- c(-10, −5, 5, 10) # Population effects
elev.effects <- c(5, 10)       # Elev effects
```

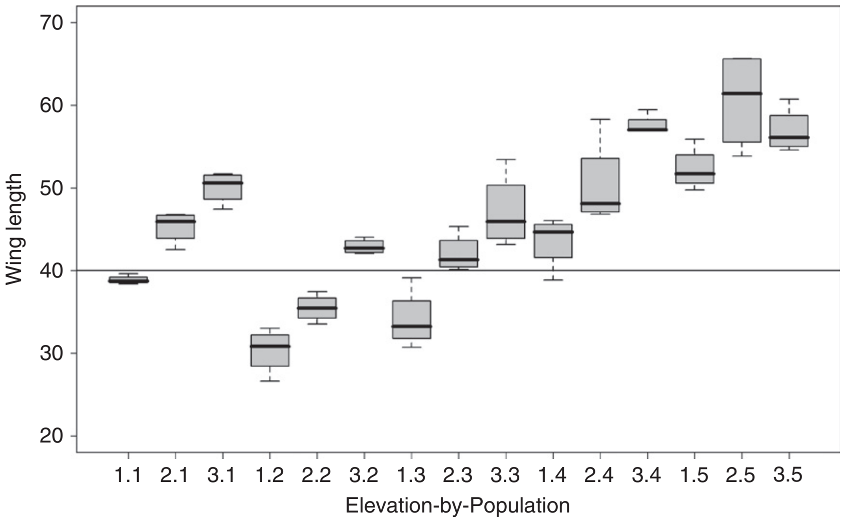

FIGURE 10.2 Mean wing length of mourning cloaks at each of three elevations and in each of five populations. Boxplots are ordered first by elevation and second by population and their identity is recognizable from the tick labels on the x-axis. For instance, the boxplot labelled 3.2 on the x-axis shows the mean wing length in population 2 at elevation 3.

```
interaction.effects <- c(−2, 3, 0, 4, 4, 0, 3, −2)     # Interaction effects
all.effects <- c(baseline, pop.effects, elev.effects, interaction.effects)

sigma <- 3
eps <- rnorm(n, 0, sigma)                 # Residuals

X <- as.matrix(model.matrix(~ pop*elev) )    # Create design matrix
X                                         # Have a look at that
```

Use matrix multiplication to assemble all components for the final wing length measurements y which we inspect in a grouped boxplot (Fig. 10.2).

```
wing <- as.numeric(as.matrix(X) %*% as.matrix(all.effects) + eps)
    # NOTE: as.numeric is ESSENTIAL for WinBUGS later
boxplot(wing ~ elev*pop, col = "grey", xlab = "Elevation-by-Population", ylab =
"Wing length", main = "Simulated data set", las = 1, ylim = c(20, 70))  # Plot of
generated data
abline(h = 40)
```

We have generated data for which the wing length–elevation relationship varies considerably among the five populations. This can also be nicely seen in a useful conditioning plot, which can be drawn using the function `xyplot()` in the lattice package. The data can be viewed in two ways; both plots show that the effects of population and elevation are not independent.

```
library("lattice")           # Load the lattice library
xyplot(wing ~ elev | pop, ylab = "Wing length", xlab = "Elevation", main =
"Population-specific relationship between wing and elevation class")
xyplot(wing ~ pop | elev, ylab = "Wing length", xlab = "Population", main =
"Elevation-specific relationship between wing and population")
```

10.3 ASIDE: USING SIMULATION TO ASSESS BIAS AND PRECISION OF AN ESTIMATOR

Let's quickly compare our parameter estimates with what we put into the data:

```
lm(wing ~ pop*elev)
all.effects

> lm(wing ~ pop*elev)

Call:
lm(formula = wing ~ pop * elev)

Coefficients:
(Intercept)        pop2        pop3        pop4        pop5       elev2
     38.859      -8.543      -4.793       4.702      13.410       6.437
      elev3  pop2:elev2  pop3:elev2  pop4:elev2  pop5:elev2  pop2:elev3
     11.213      -1.284       1.530       0.332       1.857       1.359
 pop3:elev3  pop4:elev3  pop5:elev3
      1.820       2.835      -6.609
> all.effects
[1]  40  -10  -5  5  10  5  10  -2  3  0  4  4  0  3  -2
```

The coefficient estimates don't necessarily resemble very much the parameters from which we simulated these data; after all, our sample size is rather small. So, to reassure ourselves that these differences are simply due to sampling variation, we repeat this data generation-analysis cycle 1000 times and average over the random sampling variation to convince ourselves that the estimators from the linear model are indeed unbiased. Simulations of this kind can be done easily in R, and this is one of the great strengths of R.

```
n.iter <- 1000          # Desired number of iterations
estimates <- array(dim = c(n.iter, length(all.effects)))  # Data structure to
hold results

for(i in 1:n.iter) {    # Run simulation n.iter times
  print(i)              # Optional
  eps <- rnorm(n, 0, sigma)         # Residuals
```

```
    y <- as.numeric(as.matrix(X) %*% as.matrix(all.effects) + eps) # Assemble data
    fit.model <- lm(y ~ pop*elev)                              # Break down data
    estimates[i,] <- fit.model$coefficients          # Save estimates of coefs.
  }
```

Compare the input (i.e., the chosen effects) and the output when averaged over sampling variation:

```
print(apply(estimates, 2, mean), dig = 2)
all.effects

> print(apply(estimates, 2, mean), dig = 2)
 [1]  40.0102  -10.0452  -5.0717  4.9687  9.9524  5.0417  10.0419  -1.9761
 [9]   2.9493   -0.0226   4.0031  3.9407 -0.0039  3.0234  -1.9838
> all.effects
 [1]  40  -10  -5  5  10  5  10  -2  3  0  4  4  0  3  -2
```

These are much closer to our input parameters. Depending on the number of iterations, we can get arbitrarily close to the input. Alternatively, we could increase the sample size from 12 to 12000, and we would get estimates that are still closer to the input values.

10.4 ANALYSIS USING R

We continue with the analysis of our data set and use R to fit the main-effects model first.

```
mainfit <– lm(wing ~ elev + pop)
mainfit

> mainfit
[ ... ]
Coefficients:
(Intercept)       elev2       elev3        pop2
     38.736       6.924      11.095      -8.518
       pop3        pop4        pop5
     -3.676       5.757      11.826
```

Then, we fit the means parameterization of the interaction model.

```
intfit <- lm(wing ~ elev*pop-1-pop-elev)
intfit

> intfit <- lm(wing ~ elev*pop-1-pop-elev)
> intfit
[ ... ]
```

```
Coefficients:
elev1:pop1   elev2:pop1   elev3:pop1
     38.86        45.30        50.07
elev1:pop2   elev2:pop2   elev3:pop2
     30.32        35.47        42.89
elev1:pop3   elev2:pop3   elev3:pop3
     34.07        42.03        47.10
elev1:pop4   elev2:pop4   elev3:pop4
     43.56        50.33        57.61
elev1:pop5   elev2:pop5   elev3:pop5
     52.27        60.56        56.87
```

10.5 ANALYSIS USING WinBUGS

10.5.1 Main-Effects ANOVA Using WinBUGS

We fit the main-effects model in the effects parameterization because I find that easier to code. One minor feature in this analysis is the way in which we specify the priors for the elements of the parameter vectors: instead of looping over each of them, we now write them all out, since we have to set to zero the first (or another) level of each factor to make this fixed-effects model identifiable.

```
# Define model
sink("2w.anova.txt")
cat("
model {

# Priors
 alpha ~ dnorm(0, 0.001)              # Intercept
 beta.pop[1] <- 0                     # set to zero effect of 1st level
 beta.pop[2] ~ dnorm(0, 0.001)
 beta.pop[3] ~ dnorm(0, 0.001)
 beta.pop[4] ~ dnorm(0, 0.001)
 beta.pop[5] ~ dnorm(0, 0.001)
 beta.elev[1] <- 0                    # ditto
 beta.elev[2] ~ dnorm(0, 0.001)
 beta.elev[3] ~ dnorm(0, 0.001)
 sigma ~ dunif(0, 100)

# Likelihood
for (i in 1:n) {
  wing[i] ~ dnorm(mean[i], tau)
  mean[i] <- alpha + beta.pop[pop[i]] + beta.elev[elev[i]]
}
# Derived quantities
tau <- 1 / ( sigma * sigma)
}
",fill=TRUE)
sink()

# Bundle data
win.data <- list(wing=wing, elev = as.numeric(elev), pop = as.numeric(pop), n =
length(wing))

# Inits function
inits <- function(){ list(alpha = rnorm(1), sigma = rlnorm(1) )}

# Parameters to estimate
params <- c("alpha", "beta.pop", "beta.elev", "sigma")

# MCMC settings
ni <- 1200
nb <- 200
nt <- 2
nc <- 3

# Start Gibbs sampling
out <- bugs(win.data, inits, params, "2w.anova.txt", n.thin=nt, n.chains=nc,
n.burnin=nb, n.iter=ni, debug = TRUE)

# Print estimates
print(out, dig = 3)
> print(out, dig = 3)
Inference for Bugs model at "2w.anova.txt", fit using WinBUGS,
 3 chains, each with 1200 iterations (first 200 discarded), n.thin = 2
 n.sims = 1500 iterations saved

                mean    sd    2.5%     25%     50%     75%   97.5%  Rhat n.eff
alpha         38.701 1.216  36.355  37.870  38.760  39.510  40.945 1.006   680
beta.pop[2]   -8.447 1.476 -11.325  -9.492  -8.442  -7.428  -5.547 1.001  1500
beta.pop[3]   -3.567 1.458  -6.360  -4.512  -3.586  -2.664  -0.670 1.001  1500
beta.pop[4]    5.820 1.470   3.040   4.825   5.871   6.788   8.713 1.003   780
beta.pop[5]   11.841 1.454   9.090  10.850  11.860  12.770  14.725 1.001  1500
beta.elev[2]   6.901 1.156   4.714   6.106   6.923   7.671   9.184 1.004   560
beta.elev[3] 11.077 1.119   8.988  10.300  11.040  11.850  13.220 1.002   970
sigma          3.561 0.360   2.954   3.306   3.538   3.788   4.323 1.000  1500
deviance     320.983 4.308 314.900 317.900 320.400 323.200 331.600 1.001  1500
[ ... ]
DIC info (using the rule, pD = Dbar-Dhat)
pD = 7.8 and DIC = 328.8
DIC is an estimate of expected predictive error (lower deviance is better).
```

We get estimates that are fairly similar with the MLEs above. To see the estimate of the residual, you can type `summary(mainfit)`.

10.5.2 Interaction-Effects ANOVA Using WinBUGS

We will specify the means parameterization for ease of coding and show how parameters in WinBUGS can be arrays with two (or more) dimensions. This is handy when organizing an analysis.

```
# Write model
sink("2w2.anova.txt")
cat("
model {

# Priors
  for (i in 1:n.pop){
    for(j in 1:n.elev) {
       group.mean[i,j] ~ dnorm(0, 0.0001)
    }
  }
sigma ~ dunif(0, 100)

# Likelihood
  for (i in 1:n) {
     wing[i] ~ dnorm(mean[i], tau)
     mean[i] <- group.mean[pop[i], elev[i]]
  }

# Derived quantities
  tau <- 1 / ( sigma * sigma)
}
",fill=TRUE)
sink()

# Bundle data
win.data <- list(wing=wing, elev = as.numeric(elev), pop = as.numeric(pop), n =
length(wing), n.elev = length(unique(elev)), n.pop = length(unique(pop)))

# Inits function
inits <- function(){list(sigma = rlnorm(1) )}

# Parameters to estimate
params <- c("group.mean", "sigma")

# MCMC settings
ni <- 1200
nb <- 200
nt <- 2
nc <- 3
```

```
# Start Gibbs sampling
out <- bugs(win.data, inits, params, "2w2.anova.txt", n.thin=nt, n.chains=nc,
n.burnin=nb, n.iter=ni, debug = TRUE)
# Print estimates
print(out, dig = 3)
> print(out, dig = 3)
Inference for Bugs model at "2w2.anova.txt", fit using WinBUGS,
 3 chains, each with 1200 iterations (first 200 discarded), n.thin = 2
 n.sims = 1500 iterations saved
                  mean    sd   2.5%    25%    50%    75%  97.5% Rhat n.eff
group.mean[1,1] 38.864 1.644 35.500 37.790 38.885 39.982 42.040 1.001  1500
group.mean[1,2] 45.263 1.600 42.135 44.187 45.250 46.300 48.506 1.000  1500
group.mean[1,3] 50.062 1.645 46.784 48.987 50.080 51.110 53.330 1.000  1500
group.mean[2,1] 30.330 1.589 27.150 29.297 30.320 31.370 33.495 1.000  1500
group.mean[2,2] 35.405 1.637 32.284 34.340 35.390 36.532 38.495 1.004   720
group.mean[2,3] 42.905 1.564 39.708 41.865 42.930 43.950 45.850 1.003   780
group.mean[3,1] 34.071 1.634 30.975 32.977 34.065 35.120 37.346 1.002  1500
group.mean[3,2] 41.932 1.609 38.714 40.897 41.940 43.010 45.041 1.001  1500
group.mean[3,3] 47.077 1.628 43.855 46.017 47.040 48.160 50.290 1.004   500
group.mean[4,1] 43.508 1.663 40.345 42.370 43.440 44.580 46.895 1.001  1500
group.mean[4,2] 50.403 1.635 47.239 49.310 50.365 51.520 53.590 1.004   540
group.mean[4,3] 57.630 1.617 54.525 56.570 57.640 58.720 60.770 1.002  1500
group.mean[5,1] 52.246 1.608 49.054 51.177 52.260 53.350 55.411 1.002  1100
group.mean[5,2] 60.538 1.606 57.384 59.457 60.580 61.630 63.625 1.000  1500
group.mean[5,3] 56.893 1.621 53.670 55.877 56.870 57.962 60.095 1.002  1300
sigma             3.227 0.358  2.617  2.980  3.193  3.447  4.014 1.001  1500
deviance        309.326 6.734 299.100 304.500 308.350 313.000 324.952 1.001  1500
[...]
DIC info (using the rule, pD = Dbar-Dhat)
pD = 15.7 and DIC = 325.1
DIC is an estimate of expected predictive error (lower deviance is better).
```

We find the usual similarity between the Bayes and the maximum likelihood solution above (do `summary(intfit)`) and note in passing that the estimated number of parameters (pD) is pretty close to what we would expect it to be.

10.5.3 Forming Predictions

Let's present the Bayesian inference for the interaction-effects model in a graph showing the predicted response, analogous to least-square means in a classical analysis, for each combination of elevation and population (Fig. 10.3). This plot corresponds to the boxplot of the data set (Fig. 10.2); `or` selects the order of the predictions to match that in Fig. 10.2.

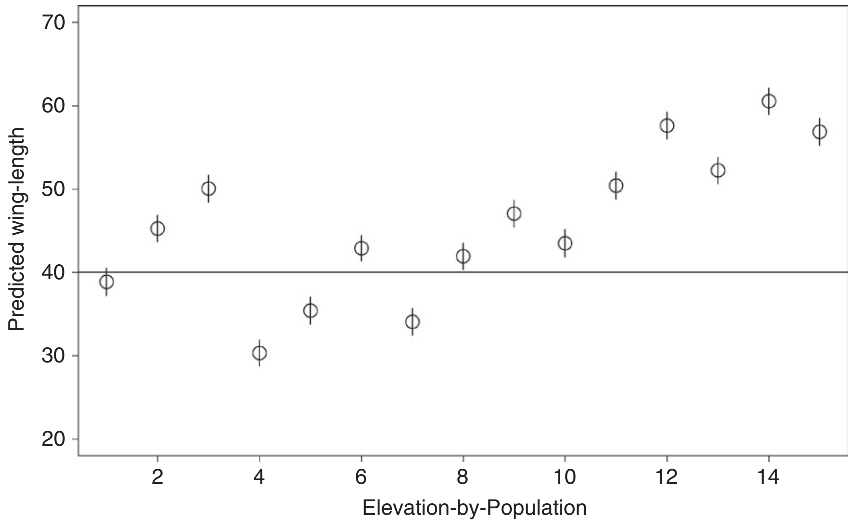

FIGURE 10.3 Predicted wing length of mourning cloaks for each elevation-population combination. Error bars are 1 SE.

```
or <- c(1,4,7,10,13,2,5,8,11,14,3,6,9,12,15)
plot(or, out$mean$group.mean, xlab = "Elev-by-Population", las = 1, ylab =
"Predicted wing-length", cex = 1.5, ylim = c(20, 70))
segments(or, out$mean$group.mean, or, out$mean$group.mean + out$sd$group.mean,
col = "black", lwd = 1)
segments(or, out$mean$group.mean, or, out$mean$group.mean - out$sd$group.mean,
col = "black", lwd = 1)
abline(h = 40)
```

10.6 SUMMARY

We have introduced the concepts of main and interaction effects and used R and WinBUGS to fit the corresponding two-way ANOVA models. In an aside, we have illustrated R's flexibility to conduct simulations to verify the effects of sampling variation on the parameter estimates.

EXERCISES

1. *Toy snake example*: Fit a two-way ANOVA with interaction to the toy example of Chapter 6 and see what happens to the nonidentifiable parameter.
2. *Swiss hare data*: Fit an ANOVA model to mean hare density to decide whether the effect of grassland and arable land use is the same in all regions. Regions and land use are somewhat confounded, but we ignore this here.