Chapter 5

Contrasts

For more detailed discussions of the use of contrasts by **R** see Chambers and Hastie (1992, Chapter 2) and Venables and Ripley (2002, Section 6.2).

1 An experiment with two factors

The following small data set was used by BHH = Box et al. (1978, Section 7.7; 8.1 in 2nd ed.). They borrowed the data from Box and Cox (1964, Section 4), a paper that contains an extensive discussion about how and why to transform data before feeding them into a standard analysis.

> *"[The data set] gives the survival times of animals in a 3 x 4 factorial experiment, the factors being (a) three poisons and (b) four treatments. Each combination of the two factors is used for four animals, the allocation to animals being completely randomized."*

```
##          A    B    C    D
## I.1   0.31 0.82 0.43 0.45
## I.2   0.45 1.10 0.45 0.71
## I.3   0.46 0.88 0.63 0.66
## I.4   0.43 0.72 0.76 0.62
## II.1  0.36 0.92 0.44 0.56
## II.2  0.29 0.61 0.35 1.02
## II.3  0.40 0.49 0.31 0.71
## II.4  0.23 1.24 0.40 0.38
## III.1 0.22 0.30 0.23 0.30
## III.2 0.21 0.37 0.25 0.36
## III.3 0.18 0.38 0.24 0.31
## III.4 0.23 0.29 0.22 0.33
```

BHH were using the data set to explain the virtues of data transformation. They argued that it was better to make a least squares fit for `rate = 1/time`, rather than fitting `time` itself. My purpose is different. I want to show you how different reparametrizations affect the output. The reparametrization is achieved by using different forms of ***contrasts*** for the factor variables.

For the purposes of analysis with **R**, we need to create a data frame with response variables `time` and `rate`, and factor variables `poison` and `treatment`:

```
BC <- read.table("boxcox.data", header=T,sep="\t")
BC$rate <- 1/BC$time # transformation suggested by BHH page 235
# poison and treatment are factors
print(BC[c(1,5,17,24:25,48),],digits=3)

##    time poison treatment rate
## 1  0.31      I         A 3.23
## 5  0.36     II         A 2.78
## 17 0.92     II         B 1.09
## 24 0.29    III         B 3.45
## 25 0.43      I         C 2.33
## 48 0.33    III         D 3.03
```

2 Linear models

Suppose $y_{i,j,k}$ denotes the response time for the kth replicate $(k = 1, \dots, 4)$ under the ith poison $(i = I, II, III)$ and the jth treatment $(j = A, B, C)$. In order to gain some understanding for how the two factors affect the response we could fit a model that treats the $y_{i,j,k}$'s as realizations of random variables

$$y_{i,j,k} = \theta_{i,j} + \xi_{i,j,k},$$

where the unobserved random errors $\xi_{i,j,k}$ have zero expected values, variances all equal to some unknown σ^2, and zero covariance between each pair of errors.

Linear models express the unknown mean $\theta_{i,j}$ as a linear function of unknown parameters determined by the factors. For example, an additive model would require

$$\theta_{i,j} = \mu + \alpha_i + \beta_j$$

Remark. Previously I wrote μ for the whole vector of expected values. Unfortunately that convention clashes with my other convention of using roman letters m, a_i, b_j as coefficients to describe a generic element of $\mathcal{X}$, with the corresponding greek letters as population parameters, and hatted roman letters $\widehat{m}, \widehat{a}_i, \widehat{b}_j$ for coefficients from least squares fits.

We could then estimate the parameters μ, α_i, β_j and σ^2 by least squares, decomposing y as residuals plus fitted values

$$\widehat{y}_{i,j,k} = \widehat{m} + \widehat{a}_i + \widehat{b}_j.$$

Of course we have to impose some constraints on the μ, α_i's and β_j's to ensure that they are uniquely determined by the $\theta_{i,j}$'s. The coefficients $\widehat{m}, \widehat{a}_i, \widehat{b}_j$ need to be similarly constrained.

Remark. As with all exercises in model fitting, the model does not claim to be a precise description of how the data were generated. It is merely a simple and convenient framework for thinking about the data. As you will see, the additive model for expected times does a poor job of approximating the observed data. Any conclusions derived from that model are unlikely to shed light on how `poison` and `treatment` affect `time`.

The triple subscript notation is unnecessarily cumbersome. It becomes particularly confusing when we need to think about the matrices involved in the least squares procedure. It is cleaner to write

$$\mathbb{E}(time \mid poison = i, treatment = j) = \theta_{i,j} = m + \alpha_i + \beta_j$$

for the way the expected values are parametrized. That notation fits well with the **R** way of describing the model,

```
out1 <- lm(time ~ poison + treatment, data=BC)
```

Similarly,

$$\theta_{i,j} = = m + \alpha_i + \beta_j + \gamma_{i,j}$$
$$\theta_{i,j} = m + \alpha_i + \beta_j$$
$$\theta_{i,j} = m + \alpha_i + \beta_j + \gamma_{i,j}$$

correspond to

```
out2 <- lm(time ~ poison * treatment, data=BC) # interactions
out3 <- lm(rate ~ poison + treatment, data=BC)
out4 <- lm(rate ~ poison * treatment, data=BC) # interactions
```

The $\gamma_{i,j}$ parameters are called ***interactions***. We need to impose some linear constraints on the unknown parameters to ensure that they are uniquely determined.

For the moment I'll put up with whatever reparametrization **R** has used. Here is the summary for `out1`:

```
## lm(formula = time ~ poison + treatment, data = BC)
## Residuals:
##      Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.25170 -0.09625 -0.01490  0.00000  0.06177  0.49830
## Rsquared:  0.65
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.452      0.056   8.088    0.000
## poisonII      -0.073      0.056  -1.308    0.198
## poisonIII     -0.341      0.056  -6.102    0.000
## treatmentB     0.363      0.065   5.614    0.000
## treatmentC     0.078      0.065   1.213    0.232
## treatmentD     0.220      0.065   3.407    0.001
```

For both factors in `out1` **R** has used treatment contrasts, which seems to be the default for my copy of the program. Look at `options()$contrasts` for the default on your machine. You can see from the summary that `poisonI` and `treatmentA` are missing from the list of coefficients. The constraint applied was $\alpha_1 = \beta_1 = 0$.

From now on, to conserve space, I'll show only an abbreviated part of the summary: the call, the coefficients, and their estimated standard errors. For example, the display for `out1` would be reduced to:

```
## lm(formula = time ~ poison + treatment, data = BC)
##         (Int)    pII   pIII    tB    tC    tD
## Est    0.452 -0.073 -0.341 0.363 0.078 0.220
## StdErr 0.056  0.056  0.056 0.065 0.065 0.065
```

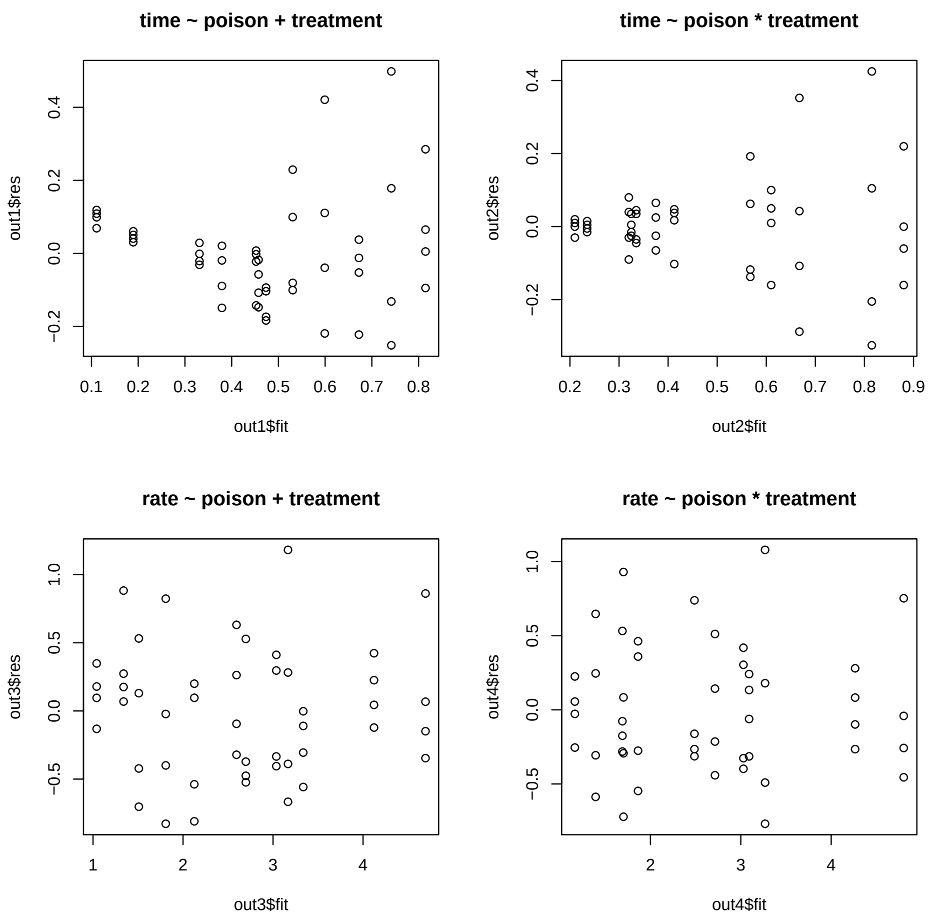
The summary for `out4` would be:

```
## lm(formula = rate ~ poison * treatment, data = BC)
##         (Int)  pII pIII    tB    tC    tD pII:tB pIII:tB pII:tC pIII:tC
## Est     2.49 0.78 2.32 -1.32 -0.62 -0.80  -0.55   -0.45   0.07   0.086
## StdErr  0.24 0.35 0.35  0.35  0.35  0.35   0.49    0.49   0.49   0.490
##        pII:tD pIII:tD
## Est     -0.77   -0.91
## StdErr   0.49    0.49
```

The coefficient labelled `pII:tB` corresponds to the interaction parameter $\gamma_{i,j}$ with $i = II$ and $j = B$.

Remark. In case you are interested, the function to produce the smaller summaries is contained in 312.R, which is in the Handouts directory.

The plots of residuals against fitted values certainly suggest that the assumption of constant variance is hard to believe for the `time` variable. Notice the strange pattern in the first two residual plots.



The interaction term slightly improves the additive fit for `time`, but not by much. The fit with interaction terms for `rate` also shows only a slight improvement over the additive fit. You should peruse the indivual summaries for more details.

From now on I'll ignore the issue of how to choose a transformation and focus on the interpretation of the coefficients for `lm(rate~)`.

3 Least squares with a single factor

Let me start with the simpler case of just one factor, first (`out5`) with the intercept explicitly excluded and then (`out6`) with an intercept. In this section, y denotes the `rate` response vector.

```
## lm(formula = rate ~ -1 + treatment, data = BC)
##         tA  tB  tC  tD
## Est    3.5 1.9 2.9 2.2
## StdErr 0.3 0.3 0.3 0.3
```

```
## lm(formula = rate ~ 1 + treatment, data = BC)
##        (Int)   tB   tC   tD
## Est      3.5 -1.7 -0.6 -1.4
## StdErr   0.3  0.4  0.4  0.4
```

The factor `BC$treatment` implicitly defines four dummy predictors, $\mathbb{1}_A$, $\mathbb{1}_B$, $\mathbb{1}_C$, and $\mathbb{1}_D$. The dummy $\mathbb{1}_A$ contains a 1 wherever `BC$treatment` contains an A and zeros elsewhere; and so on. In **R** you could manufacture `dummyT` $= (\mathbb{1}_A, \mathbb{1}_B, \mathbb{1}_C, \mathbb{1}_D)$ by

```
dummyT <- outer(BC$treat,levels(BC$treat),"==")+0
# the zero converts from Boolean to numeric
dimnames(dummyT)[[2]] <- levels(BC$treatment)
dummyT[c(1:2,13:14,25:26,37:38),]       # Why did I choose these rows?

##      A B C D
## [1,] 1 0 0 0
## [2,] 1 0 0 0
## [3,] 0 1 0 0
## [4,] 0 1 0 0
## [5,] 0 0 1 0
## [6,] 0 0 1 0
## [7,] 0 0 0 1
## [8,] 0 0 0 1
```

If I only showed rows $1, 13, 25, 37$ you would see I_4, the 4×4 identity matrix. Why?

The model matrix for `out5` is `dummyT` itself. It produces a fitted vector of the form

$$\widehat{y} = \widehat{b}_A \mathbb{1}_A + \widehat{b}_B \mathbb{1}_B + \widehat{b}_C \mathbb{1}_C + \widehat{b}_D \mathbb{1}_D,$$

which corresponds to projection of y onto the 4-dimensional subspace $\mathcal{X}$ of $\mathbb{R}^{48}$ spanned by $\mathbb{1}_A, \mathbb{1}_B, \mathbb{1}_C, \mathbb{1}_D$. The least squares estimates $\widehat{b}_A, \dots, \widehat{b}_D$ are just the means over responses at the same level of the `BC$treatment` factor.

```
print( rbind(out5$coeff,tapply(BC$rate,BC$treatment,mean)),digits=4)

##      treatmentA treatmentB treatmentC treatmentD
## [1,]      3.519      1.862      2.947      2.161
## [2,]      3.519      1.862      2.947      2.161
```

The model matrix for `out6` is $X = (\mathbb{1}, \mathbb{1}_B, \mathbb{1}_c, \mathbb{1}_D)$. It produces

$$\widehat{y} = \widehat{a}_0\mathbb{1} + \widehat{a}_2\mathbb{1}_B + \widehat{a}_3\mathbb{1}_C + \widehat{a}_4\mathbb{1}_D.$$

The fitted vector is the same as for `out5` because the linearly independent column vectors $\mathbb{1}, \mathbb{1}_B, \mathbb{1}_C, \mathbb{1}_D$ span the same $\mathcal{X}$. Only the parametrizations differ. In case you have doubts:

`round( max(abs( out5$fit - out6$fit )) ,6) =` 0.

The generic element z of $\mathcal{X}$ can be expressed in two ways:

$$z = b_A\mathbb{1}_A + b_B\mathbb{1}_B + b_C\mathbb{1}_C + b_D\mathbb{1}_D = a_0\mathbb{1} + a_2\mathbb{1}_B + a_3\mathbb{1}_C + a_4\mathbb{1}_D.$$

There is a one-to-one correspondence between the column vectors of coefficients $b = [b_A, b_B, b_C, b_D]$ and $a = [a_0, a_2, a_3, a_4]$, namely $b_A = a_0$ and $b_B = a_0 + a_2$ and $b_C = a_0 + a_3$ and $b_D = a_0 + a_4$. In matrix terms $b = Ka$ where K is the matrix

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    0    0    0
## [2,]    1    1    0    0
## [3,]    1    0    1    0
## [4,]    1    0    0    1
```

In particular, $\widehat{b} = K\widehat{a}$ and $\text{var}(\widehat{b}) = K\text{var}(\widehat{a})K^T$. The `summary.lm()` function, which gets called when we ask for `summary()` of an lm object, actually calculates the estimated matrix of variances and covariances for the estimated coefficients. We should be able to reproduce part of `out5` from `out6`.

```
K <- matrix(c(1,0,0,0,1,1,0,0,1,0,1,0,1,0,0,1),byrow=T,ncol=4)
ahat <- out6$coeff
newbhat <- K %*% ahat
V6 <- summary(out6)$cov # estimate of var(ahat)
V5 <- K %*% V6 %*% t(K)
OUT5 <- BC.coeff(out5,do.print=F)
#print(OUT5£CALL)
print( rbind( OUT5$COEFF,t(newbhat), stderr=sqrt(diag(V5))),digits=3)

##           tA    tB    tC    tD
## Est    3.519 1.862 2.947 2.161
## StdErr 0.292 0.292 0.292 0.292
##        3.519 1.862 2.947 2.161
## stderr 0.289 0.289 0.289 0.289
```

The first two rows give the `out5` values; the next two rows give the same values calculated from `out6`. It works, with just a little bit of round-off error.

4 One factor with Helmert contrasts and intercept

Now add another column to `BC`, a copy of `BC$treatment` with its contrasts changed from the default (treatment) to Helmert. You won't see any difference between `BC$treatment` and `BC$Ht` if you just print out `BC`, but it does make a difference to the coefficients for the least squares fit because of a difference in the contrast matrices:

```
BC$Ht <- C(BC$treatment,helmert)
out7 <- lm(rate ~ 1+ Ht, BC)
C.treat <- contrasts(BC$treatment)
C.Htreat <- contrasts(BC$Ht)
dimnames(C.Htreat)[[2]] <- names(out7$coef)[-1]
# insert a column of 1's before C.Htreat
data.frame(C.treat, ... =" ", int= 1, .. =" ", C.Htreat)

##   B C D ... int .. Ht1 Ht2 Ht3
## A 0 0 0       1     -1  -1  -1
## B 1 0 0       1      1  -1  -1
## C 0 1 0       1      0   2  -1
## D 0 0 1       1      0   0   3
```

Notice that the columns of the contrast matrix `C.Htreat` are orthogonal to $\mathbb{1}_4$.

Again we get a reparametrization of the same fitted vector:

`round( max(abs( out6$fit - out7$fit )) ,6) =` 0.

Of course the summary looks very different from the summaries for `out5` and `out6`:

```
## lm(formula = rate ~ 1 + Ht, data = BC)
## Residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -1.6310 -0.7443 -0.2581  0.0000  0.8479  2.0360
## Rsquared:  0.312
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    2.622      0.146  17.947    0.000
## Ht1           -0.829      0.207  -4.010    0.000
## Ht2            0.086      0.119   0.717    0.477
## Ht3           -0.154      0.084  -1.823    0.075
```

A lot of interesting stuff has happened inside the least squares calculation. We need to be careful interpreting the coefficients. The model matrix used for `out7` is

$X = (\mathbb{1}_{48}, \texttt{dummyT \%*\%}\ C_4)$ where $C_4 =$ `C.Htreat = contr.helmert(4)`.

```
typical <- c(1,13,25,37)
blank <- rep("",4)
mm <- model.matrix(out7)
dimnames(mm)[[2]][1] <- "Int"
# Insert columns of blanks to separate the four matrices.
data.frame(dummyT[typical,], .=blank, mm[typical,],
..=blank,BC[typical,c(2,5)], ...=blank, C.Htreat)

##    A B C D . Int Ht1 Ht2 Ht3 .. poison Ht ... Ht1.1 Ht2.1 Ht3.1
## 1  1 0 0 0     1  -1  -1  -1          I  A       -1    -1    -1
## 13 0 1 0 0     1   1  -1  -1          I  B        1    -1    -1
## 25 0 0 1 0     1   0   2  -1          I  C        0     2    -1
## 37 0 0 0 1     1   0   0   3          I  D        0     0     3
```

In effect, **R** has coded each appearance of level A by the first row in `C.Htreat`, and so on.

4.1 General interpretation of a fit with Helmert contrasts

Consider the general case of an $n \times 2$ data frame $(y, \mathbb{F})$, where $\mathbb{F}$ is a factor taking values in a set $\mathcal{L}$ of k different labels. For simplicity you could assume $\mathcal{L} = [k] = \{1, 2, \dots, k\}$. Write $F = (F_t : t \in \mathcal{L})$ for the $n \times k$ matrix of dummy vectors corresponding to $\mathbb{F}$. That is, F_t has a 1 in those rows for which $\mathbb{F}$ takes the value t, zeros elsewhere.

For `out7` we have $\mathbb{F} =$ `BC$Ht`, with $n = 48, k = 4$ and $\mathcal{L} = \{A, B, C, D\}$. The matrix F corresponds to `dummyT` $= (\mathbb{1}_A, \mathbb{1}_B, \mathbb{1}_C, \mathbb{1}_D)$.

If $\mathbb{F}$ has Helmert contrasts then the model matrix for `lm(y ~ X)` is

$$X = (\mathbb{1}_n, FC_k) = F(\mathbb{1}_k, C_k) \qquad \text{where } \texttt{C\_k = contr.helmert(k)}.$$

The second form for X comes from the fact that $F\mathbb{1}_k = \sum_{t \in \mathcal{L}} F_t = \mathbb{1}_n$. The columns of the $k \times (k-1)$ matrix C_k form a basis for the subspace of $\mathbb{R}^k$ orthogonal to $\mathbb{1}_k$. The columns of the $k \times k$ matrix $K_k = (\mathbb{1}_k, C_k)$ form a basis of $\mathbb{R}^k$; the matrix K_k is non-singular, with inverse L_k, so that

$$X = FK_k \quad \text{AND} \quad F = XL_k.$$

The columns of F span the same k-dimensional subspace of $\mathbb{R}^n$ as the columns of X.

Remark. Here I am tacitly assuming that F is of rank k, which is true provided none of the F_t's is a vector of zeros. That is, $\mathcal{X}$ has dimension k provided each of the levels in $\mathcal{L}$ appears somewhere in $\mathbb{F}$.

The generic element z of $\mathcal{X}$ is now represented by a $k \times 1$ column vector of "raw" coefficients $\mathfrak{p} = [m, f_1, \dots, f_{k-1}]$ with

$$z = X\mathfrak{p} = m\mathbb{1}_n + FC_k f \qquad \text{where } f = [f_1, \dots, f_{k-1}].$$

If we define a $k \times 1$ column vector $a = C_k f$ then we have $\mathbb{1}_k^T a = \mathbb{1}_k^T C_k f = 0$ and

$$z = m\mathbb{1}_n + \sum\nolimits_{t \in \mathcal{L}} a_t F_t \qquad \text{with } \sum\nolimits_{t \in \mathcal{L}} a_t = 0.$$

In matrix notation,

$$\begin{pmatrix} m \\ a \end{pmatrix} = M_{\mathbb{F}}\mathfrak{p} \qquad \text{where } M_{\mathbb{F}} = \begin{array}{c c} & \begin{array}{cc} 1 & k-1 \end{array} \\ \begin{array}{c} 1 \\ k \end{array} & \begin{bmatrix} 1 & 0 \\ 0 & C_k \end{bmatrix} \end{array}.$$

There is no harm in using the same m on the left-hand side of the equality because it represents the same number. In particular,

$$\begin{pmatrix} \widehat{m} \\ \widehat{a} \end{pmatrix} = M_{\mathbb{F}}\widehat{\mathfrak{p}} \qquad \text{so that var} \begin{pmatrix} \widehat{m} \\ \widehat{a} \end{pmatrix} = M_{\mathbb{F}} \text{var}(\widehat{\mathfrak{p}}) M_{\mathbb{F}}^T.$$

```
M.treat <- bdiag(1,C.Htreat)
raw.hat <- out7$coeff
ma.hat <- as.vector(M.treat %*% raw.hat)
names(ma.hat) <- c("int",levels(BC$Ht))
V.ma.hat <- M.treat %*% summary(out7)$cov %*% t(M.treat)
ma.hat.stderr <- sqrt(diag(V.ma.hat))
est <- rbind(ma.hat = ma.hat,std.err=ma.hat.stderr)
print(est,3)

##           int     A     B     C      D
## ma.hat  2.622 0.897 -0.76 0.325 -0.461
## std.err 0.144 0.250  0.25 0.250  0.250
```

The fitted model is now represented by an estimated average effect of the treatments (coeff 2.62) together with estimated deviations (0.9, -0.76, 0.32, -0.46) of the individual treatments from that average.

Just as a check, make sure that the ma.hat coefficients satisfy the constraints and they do give the same fit as `out5`:

```
round( sum(ma.hat[2:5]),5)

## [1] 0
```

```
print(rbind(out5$coef, ma.hat[1] + ma.hat[2:5]),4)

##      treatmentA treatmentB treatmentC treatmentD
## [1,]      3.519      1.862      2.947      2.161
## [2,]      3.519      1.862      2.947      2.161
```

Each of `out5`, `out6`, and `out7`, together with the `ma.hat` give the same fitted vector $\widehat{y}$. They express $\widehat{y}$ using different estimated coefficients:

```
BC.coeff(out6,3)

## lm(formula = rate ~ 1 + treatment, data = BC)
##         (Int)     tB     tC     tD
## Est    3.519 -1.657 -0.572 -1.358
## StdErr 0.292  0.413  0.413  0.413

BC.coeff(out5,3)

## lm(formula = rate ~ -1 + treatment, data = BC)
##           tA    tB    tC    tD
## Est    3.519 1.862 2.947 2.161
## StdErr 0.292 0.292 0.292 0.292

BC.coeff(out7,3)

## lm(formula = rate ~ 1 + Ht, data = BC)
##         (Int)    Ht1    Ht2     Ht3
## Est    2.622 -0.829 0.0855 -0.1538
## StdErr 0.146  0.207 0.1193  0.0844

print(est,3) # ma.hat derived from out7

##           int     A     B     C      D
## ma.hat  2.622 0.897 -0.76 0.325 -0.461
## std.err 0.144 0.250  0.25 0.250  0.250
```

Which is easiest to interpret?

5 Two factors: additive effects

The story gets more complicated when both the poison and treatment factors are used as predictors. Let me again consider a more general situation where we have two factors, represented as an $n \times 1$ vector $\mathbb{F}$ with elements taken from the set $[k] = \{1, 2, \ldots, k\}$ and an $n \times 1$ vector $\mathbb{G}$ with elements taken from the set $[\ell] = \{1, 2, \ldots, \ell\}$. Because **R** doesn't allow fancy symbols as variable names, the first five rows of the part of the data frame corresponding to the factors and the intercept would look something like:

```
##   intercept F.factor G.factor
## 1         1        1        7
## 2         1        3        9
## 3         1        8        6
## 4         1        2       11
## 5         1        4        2
```

Perhaps I should have made up more suggestive names for the levels, such as $\{A, B, C, \dots\}$ and $\{a, b, c, \dots\}$ so that you could more easily distinguish between the two factors.

The factor $\mathbb{F}$ corresponds to an $n \times k$ matrix $F = (F_1, \dots, F_k)$ of dummy variables, where F_i indicates where $\mathbb{F} = i$. That is, $F[r, i] = 1$ if and only if $\mathbb{F}[r] = i$, zero otherwise. Similarly $\mathbb{G}$ corresponds to an $n \times \ell$ matrix $G = (G_1, \dots, G_k)$.

Again let me assume that both factors have Helmert contrasts. The **R** command `lm(y ~ F.factor + G.factor)` then projects y onto the subspace $\mathfrak{X}_{\mathbb{F}+\mathbb{G}}$ spanned by $\mathbb{1}_n, F_1, \dots, F_k, G_1, \dots, G_\ell$.

For the moment let me assume that each of the $k\ell$ factor combinations $\mathbb{F}[r] = i, \mathbb{G}[r] = j$, for $i \in [k]$ and $j \in [\ell]$, appears at least once amongst the rows of the $n \times (k + \ell)$ matrix (F, G). As you will show on Homework 5, that assumption implies that $\mathfrak{X}_{\mathbb{F}+\mathbb{G}}$ has dimension $k + \ell - 1$. Equivalently, it implies that each z in $\mathfrak{X}_{\mathbb{F}+\mathbb{G}}$ has a unique representation

<5.1> $$z = m\mathbb{1}_n + Fa + Gb \qquad \text{with } \sum\nolimits_{i\in[k]} a_i = 0 = \sum\nolimits_{j\in[\ell]} b_j.$$

In particular, both

$$\begin{aligned} \mathbb{E}y &= \mu\mathbb{1} + \sum\nolimits_i \alpha_i F_i + \sum\nolimits_j \beta_j G_j &&\text{with } \sum\nolimits_i \alpha_i = 0 = \sum\nolimits_j \beta_j \\ \widehat{y} &= \widehat{m}\mathbb{1} + \sum\nolimits_i \widehat{a}_i F_i + \sum\nolimits_j \widehat{b}_j G_j &&\text{with } \sum\nolimits_i \widehat{a}_i = 0 = \sum\nolimits_j \widehat{b}_j \end{aligned}$$

have unique solutions. It then makes sense to think of $\widehat{m}$ as an estimator for μ, and so on.

R solves the least squares problem using the model matrix

$$X = (\mathbb{1}, FC_k, GC_\ell)$$

```
BC$Hp <- C(BC$poison,helmert)
out.HtHp <- lm(rate ~ Ht + Hp,BC)
look(out.HtHp)
```

```
## lm(formula = rate ~ Ht + Hp, data = BC)
## Residuals:
##      Min.   1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.82760 -0.37620  0.02116  0.00000  0.27570  1.18200
## Rsquared:  0.844
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    2.622      0.071  36.842    0.000
## Ht1           -0.829      0.101  -8.233    0.000
## Ht2            0.086      0.058   1.472    0.149
## Ht3           -0.154      0.041  -3.742    0.001
## Hp1            0.234      0.087   2.688    0.010
## Hp2            0.587      0.050  11.670    0.000
```

Once again **R** has created names for the coefficients $\widehat{f}$ and $\widehat{g}$ that appear in the representation

$$\widehat{y} = \widehat{m}\mathbb{1}_{48} + \texttt{dummyT \%*\%}\ C_4\widehat{f} + \texttt{dummyG \%*\%} C_3\widehat{g}.$$

In the general case, each vector z in $\mathcal{X}_{\mathbb{F}+\mathbb{G}}$ has a unique representation

$$z = m\mathbb{1}_n + FC_kf + GC_\ell g \qquad \text{with } f = [f_1, \dots, f_{k-1}] \text{ and } g = [g_1, \dots, g_{\ell-1}].$$

The vectors $a = C_kf$ and $b = C_\ell g$ then satisfy <5.1>.

The correspondence between the vector of "raw" coefficients $\mathfrak{p}^T = (m, f^T, g^T)$ for which $z = X\mathfrak{p}$ and the constrained coefficients for which $z = m\mathbb{1}_n + Fa + Gb$ is given by

$$\begin{pmatrix} m \\ a \\ b \end{pmatrix} = M_{\mathbb{F}+\mathbb{G}}\mathfrak{p} \qquad \text{where } M_{\mathbb{F}+\mathbb{G}} = \begin{array}{c|ccc} & 1 & k-1 & \ell-1 \\ \hline 1 & 1 & 0 & 0 \\ k & 0 & C_k & 0 \\ \ell & 0 & 0 & C_\ell \end{array}$$

In particular

$$\begin{pmatrix} \widehat{m} \\ \widehat{a} \\ \widehat{b} \end{pmatrix} = M_{\mathbb{F}+\mathbb{G}}\widehat{\mathfrak{p}} \qquad \text{so that var} \begin{pmatrix} \widehat{m} \\ \widehat{a} \\ \widehat{b} \end{pmatrix} = M_{\mathbb{F}+\mathbb{G}}\text{var}(\widehat{\mathfrak{p}})M_{\mathbb{F}+\mathbb{G}}^T.$$

```
CHt <- contrasts(BC$Ht)
CHp <- contrasts(BC$Hp)
M.HtHp <- bdiag(1,CHt,CHp)
raw.hat <- out.HtHp$coeff
mab.hat <- as.vector(M.HtHp %*% raw.hat)
names(mab.hat) <- c("int",levels(BC$Ht),levels(BC$Hp))
```

```
V.mab.hat <- M.HtHp %*% summary(out.HtHp)$cov %*% t(M.HtHp)
mab.hat.stderr <- sqrt(diag(V.mab.hat))
est.HtHp <- rbind(mab.hat = mab.hat,std.err=mab.hat.stderr)
print(est.HtHp,3)

##           int     A     B     C      D      I     II   III
## mab.hat 2.622 0.897 -0.76 0.325 -0.461 -0.822 -0.353 1.175
## std.err 0.144 0.250  0.25 0.250  0.250  0.204  0.204 0.204
```

The fitted model is now represented by an estimated average effect of the treatments (coeff 2.62) together with estimated deviations from that average due to an additive effect of treatment and poison.

The BC data are balanced, in the sense that there is the same number of replicates for every treatment/poison combination. For that reason, the least squares estimates have a simple representation, $\widehat{m} = \overline{y}_{...}$ and $\widehat{a}_i = \overline{y}_{i..} - \overline{y}$ and $\widehat{b}_j = \overline{y}_{.j.} - \overline{y}$:

```
grand.mean <- mean(BC$rate)
meanT <- tapply(BC$rate,BC$treatment,mean) - grand.mean
meanP <- tapply(BC$rate,BC$poison,mean)  - grand.mean
means <- c(grand=grand.mean,meanT, meanP)
print(rbind(means,mab.hat),3)

##         grand     A     B     C      D      I     II  III
## means    2.62 0.897 -0.76 0.325 -0.461 -0.822 -0.353 1.17
## mab.hat  2.62 0.897 -0.76 0.325 -0.461 -0.822 -0.353 1.17
```

Remark. You might be wondering why the estimated standard errors in the previous **R** display were all the same (up to rounding error). The representation as centered means explains the equality. If the data set were not balanced it would be most surprising to have such equalities.

If some of the y_{ijk}'s were lost, for some reason, the design would probably not be balanced. The least squares estimates would no longer take such a simple form.

Conceptually, the least squares procedure has decomposed the table `BC$rate` (the first matrix in the next **R** display) into an additive fit (the second matrix) plus a residual (the third matrix):

```
##         A   B   C   D .. A.1 B.1 C.1 D.1 ...  A.2  B.2  C.2  D.2
## I.1   3.2 1.2 2.3 2.2    2.7 1.0 2.1 1.3      0.5  0.2  0.2  0.9
## I.2   2.2 0.9 2.2 1.4    2.7 1.0 2.1 1.3     -0.5 -0.1  0.1  0.1
## I.3   2.2 1.1 1.6 1.5    2.7 1.0 2.1 1.3     -0.5  0.1 -0.5  0.2
## I.4   2.3 1.4 1.3 1.6    2.7 1.0 2.1 1.3     -0.4  0.3 -0.8  0.3
## II.1  2.8 1.1 2.3 1.8    3.2 1.5 2.6 1.8     -0.4 -0.4 -0.3  0.0
## II.2  3.4 1.6 2.9 1.0    3.2 1.5 2.6 1.8      0.3  0.1  0.3 -0.8
## II.3  2.5 2.0 3.2 1.4    3.2 1.5 2.6 1.8     -0.7  0.5  0.6 -0.4
## II.4  4.3 0.8 2.5 2.6    3.2 1.5 2.6 1.8      1.2 -0.7 -0.1  0.8
## III.1 4.5 3.3 4.3 3.3    4.7 3.0 4.1 3.3     -0.1  0.3  0.2  0.0
## III.2 4.8 2.7 4.0 2.8    4.7 3.0 4.1 3.3      0.1 -0.3 -0.1 -0.6
## III.3 5.6 2.6 4.2 3.2    4.7 3.0 4.1 3.3      0.9 -0.4  0.0 -0.1
## III.4 4.3 3.4 4.5 3.0    4.7 3.0 4.1 3.3     -0.3  0.4  0.4 -0.3
```

Of course the four fitted values are the same within each treatment/poison cell. I could have reduced the fit to a much simpler table:

```
fits2 <- out.HtHp$fit[seq(1,48,by=4)]
MM <- Matrix(fits2,ncol=4)
dimnames(MM) <- list(levels(BC$poison),levels(BC$treat))
round(MM,3)

## 3 x 4 Matrix of class "dgeMatrix"
##         A     B     C     D
## I   2.698 1.040 2.126 1.339
## II  3.166 1.509 2.594 1.808
## III 4.694 3.037 4.122 3.336
```

For the sake of comparison, here is the table explicitly calculated from the estimated coefficients in the constrained additive fit, with the coefficients $\widehat{m}, \widehat{a}_1, \widehat{a}_2, \widehat{a}_3, \widehat{a}_4$ added as a top row and the coefficients $\widehat{b}_1, \widehat{b}_2, \widehat{b}_3$ filling out the rest of the first column. The south-east 3×4 submatrix agrees with the previous **R** display.

```
row.hat <- mab.hat[6:8]
col.hat <- mab.hat[2:5]
fitted.HtHp <- outer(row.hat,col.hat,"+") + mab.hat[1]
additive <- as.matrix(rbind(mab.hat[1:5],cbind(mab.hat[6:8],fitted.HtHp)))
round(additive,3)

##        int     A      B     C      D
##      2.622 0.897 -0.760 0.325 -0.461
## I   -0.822 2.698  1.040 2.126  1.339
## II  -0.353 3.166  1.509 2.594  1.808
## III  1.175 4.694  3.037 4.122  3.336
```

5.1 The effect of missing data

Suppose I were careless and lost some of the BC data rows. Instead of getting the estimates from the full data set I would then get estimates based on only a subset of the data. That is, instead of

```
BC.coeff(out.HtHp)

## lm(formula = rate ~ Ht + Hp, data = BC)
##         (Int)  Ht1  Ht2   Ht3  Hp1  Hp2
## Est      2.62 -0.8 0.09 -0.15 0.23 0.59
## StdErr   0.07  0.1 0.06  0.04 0.09 0.05

# or
print(est.HtHp,2)

##          int    A     B    C     D     I    II III
## mab.hat 2.62 0.90 -0.76 0.32 -0.46 -0.82 -0.35 1.2
## std.err 0.14 0.25  0.25 0.25  0.25  0.20  0.20 0.2
```

I would have something like

```
lost <- c(3,5,19,22)
out.lost <- lm(rate ~ Ht + Hp, BC, subset = -lost)
raw.hat <- out.lost$coeff
mab.hat <- as.vector(M.HtHp %*% raw.hat)
names(mab.hat) <- c("int",levels(BC$Ht),levels(BC$Hp))
V.mab.hat <- M.HtHp %*% summary(out.lost)$cov %*% t(M.HtHp)
mab.hat.stderr <- sqrt(diag(V.mab.hat))
est.lost <- rbind(mab.hat = mab.hat,std.err=mab.hat.stderr)
print(est.lost,2)

##          int    A     B    C     D     I    II  III
## mab.hat 2.64 0.97 -0.80 0.31 -0.48 -0.80 -0.38 1.18
## std.err 0.15 0.27  0.27 0.25  0.25  0.21  0.22 0.21
```

Notice that the calculations are essentially the same as in the balanced case, once we have the raw fit. The lack of balance has no effect on the interpretation of the fit with the constrained parametrization.

The file 312.R contains the code for an **R** function

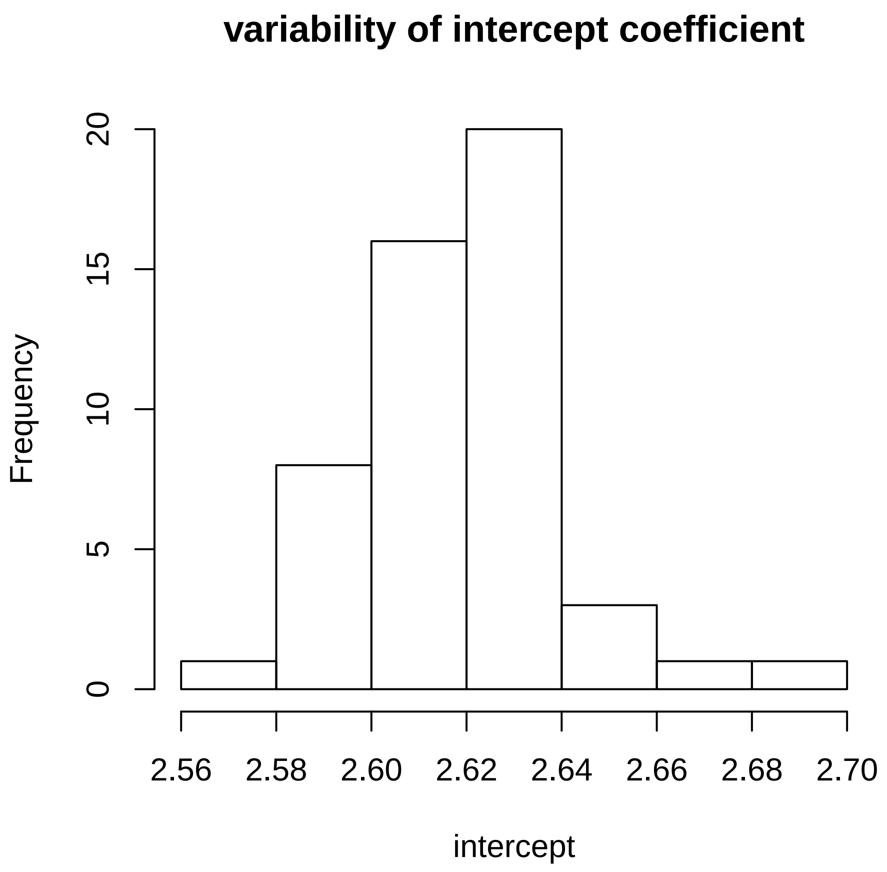
`lose.some(num.samples=5,lose=4)`.

The default sets the number of lost observations equal to 4. Also by default, the function calculates the fit for the original full data set plus 5 randomly chosen subsets.

```
round( lose.some(),2)

## Error in lose.some():  object 'mab.hat' not found
```

The spread in a larger number of randomly chosen subsets is more interesting:

```
COEFF <- lose.some(num.samples=100,lose=4)

## Error in lose.some(num.samples = 100, lose = 4):  object 'mab.hat' not found

hist(COEFF[,"int"],xlab="intercept",main="variability of intercept coefficient")
```



Compare with the estimated standard error 0.15.

6 Two factors: interactions

Once again suppose factor $\mathbb{F}$ is a factor with levels $[k]$, which corresponds to an $n \times k$ matrix $F = (F_1, \dots, F_k)$ of dummy variables. And $\mathbb{G}$ is a factor with levels $[\ell]$, which corresponds to an $n \times \ell$ matrix $G = (G_1, \dots, G_k)$ of dummy variables.

Conceptually, the pair of factors together define a new factor with $k\ell$ levels—which I write as $\mathbb{F} * \mathbb{G}$ by analogy with **R** notation—taking values in the set $\{(i,j) : i \in [k], j \in [\ell]\}$. It is represented by the set of $k\ell$ dummy variables $F_i * G_j$, where the $*$ denotes component-wise multiplication of vectors, as in **R**: the rth element of $F_i * G_j$ equals 1 if $\mathbb{F}[r] = i$ and $\mathbb{G}[r] = j$, and is 0 otherwise. If $F_i * G_j \neq 0$ for all i and j, the set of all kr dummy variables span a $k\ell$-dimensional subspace $\mathcal{X}_{\mathbb{F}*\mathbb{G}}$ of $\mathbb{R}^n$. The challenge is to write each vector z in that subspace as

<5.2> $$z = m\mathbb{1}_n + \sum\nolimits_{i \le k} a_i F_i + \sum\nolimits_{j \le \ell} b_j G_j + \sum\nolimits_{i \le k, j \le \ell} d_{i,j} F_i * G_j$$

with constraints $\sum_i a_i = 0$ and $\sum_j b_j = 0$ and $\sum_j d_{i,j} = 0$ for each i and $\sum_i d_{i,j} = 0$ for each i.

Remark. We have $1 + k + \ell + k\ell$ parameters subject to $k + \ell + 1$ linearly independent constraints (a tricky calculation), which leaves $k\ell$ degrees of freedom. That agrees with the fact that the $k\ell$ factors $F_i * G_j$ are linearly independent if $F_i * G_j \neq 0$ for all i and j. For the moment don't worry too much about the calculation. Homework 5 will step you through the solution.

The subspace $\mathcal{X}_{\mathbb{F}*\mathbb{G}}$ can also have dimension $k\ell$ without the assumption that every pair of possible factor levels is represented in $\mathbb{F} * \mathbb{G}$.

Nevertheless, I'll tacitly assume the stronger condition throughout this Section.

Under Helmert contrasts, the factor $\mathbb{F}$ has a $k \times (k-1)$ contrast matrix C_k whose columns span the part of $\mathbb{R}^k$ that is orthogonal to $\mathbb{1}_k$. The $k \times k$ matrix $K_k = (\mathbb{1}_k, C_k)$ has columns that form a basis for $\mathbb{R}^k$; the matrix is non-singular with $k \times k$ inverse L_k. Define an $n \times k$ matrix

$$\Phi = (\phi_1, \ldots, \phi_k) = FK_k = (\mathbb{1}_n, FC_k).$$

The correspondence between Φ and F is one-to-one, with $F = \Phi L_k$. The columns of Φ span the same subset of $\mathbb{R}^n$ as the columns of F. More explicitly,

<5.3> $$F_i = \sum\nolimits_{r \in [k]} L_k[r,i]\phi_r \qquad \text{for each } i \text{ in } [k].$$

Similarly, under Helmert contrasts, the factor $\mathbb{G}$ has an $\ell \times (\ell-1)$ contrast matrix for which $K_\ell = (\mathbb{1}_\ell, C_\ell)$ is non-singular with $\ell \times \ell$ inverse L_ℓ, and the columns of the $n \times \ell$ matrix

$$\Psi = (\psi_1, \ldots, \psi_\ell) = GK_\ell = (\mathbb{1}_n, GC_\ell)$$

span the same subspace of $\mathbb{R}^n$ as the columns of G, with

<5.4> $$G_j = \sum\nolimits_{s\in[k]} L_\ell[s,j]\psi_s \qquad \text{for each } j \text{ in } [\ell].$$

The component-wise product of the factor dummy variables is given by

$$\begin{aligned} F_i * G_j &= \left(\sum\nolimits_{r\in[k]} L_k[r,i]\phi_r\right) * \left(\sum\nolimits_{s\in[k]} L_\ell[s,j]\psi_s\right) \\ &= \sum\nolimits_{r,s} L_k[r,i]L_\ell[s,j]\phi_r * \psi_s. \end{aligned}$$

Every linear combination of the $F_i * G_j$'s can be written as a linear combination of the $\phi_r * \psi_s$'s. The vectors $\{\phi_r * \psi_s : r \in [k], s \in [\ell]\}$ provide another basis for $\mathcal{X}_{\mathbb{F}*\mathbb{G}}$. More explicitly, for any $k \times \ell$ matrix M,

$$\begin{aligned} &\sum\nolimits_{i,j} M[i,j]F_i * G_j \\ &\quad = \sum\nolimits_{i,j} M[i,j] \sum\nolimits_{r,s} L_k[r,i]L_\ell[s,j]\phi_r * \psi_s \\ &\quad = \sum\nolimits_{r,s} \left(\sum\nolimits_{i,j} M[i,j]L_k[r,i]L_\ell[s,j]\right) \phi_r * \psi_s \end{aligned}$$

<5.5> $$\quad = \sum_{r,s} B[r,s]\phi_r * \psi_s \qquad \text{where } B[r,s] = \sum_{i,j} M[i,j]L_k[r,i]L_\ell[s,j].$$

That is, $B = L_r M L_\ell^T$. Equivalently, $M = K_k B K_\ell^T$. The correspondence is, indeed, one-to-one.

6.1 Interactions with constrained coefficients

The **R** command out.FG = `lm(y ~ F.factor*G.factor)` uses the $n \times (k\ell)$ model matrix X that has columns $\phi_i * \psi_j$. Because $\phi_1 = \mathbb{1}_n = \psi_1$, the leading columns of X are just $\mathbb{1}_n, \phi_1, \dots, \phi_k, \psi_1, \dots, \psi_\ell$. To shorten the display I print out only the unique rows of X for the BC example.

```
out.TP <- lm(rate ~ Ht*Hp,BC)
C4 <- contrasts(BC$Ht); C3 <- contrasts(BC$Hp)
Phi <- dummyT %*% cbind(1,C4);  Psi <- dummyP %*% cbind(1,C3)
mm.TP <- unique(model.matrix(out.TP)) # the unique rows of X
dimnames(mm.TP)[[2]][1] <- "int" # shorten to make display fit page width
```

If you look carefully you should be able to see many copies of C_4 and C_3 hiding in the matrix `mm.TP[,2:6]`. You should also note that the columns for the interactions are just component-wise products of the columns for the individual factors. For example,

`mm.TP[,"Ht3:Hp2"]` is equal to `mm.TP[,"Ht3"] * mm.TP[,"Hp2"]`

```
# You should check that mm.TP[,"Ht3:Hp2"]-mm.TP[,"Ht3"] * mm.TP[,"Hp2"] is zero
print(unique(Phi))

##      [,1] [,2] [,3] [,4]
## [1,]    1   -1   -1   -1
## [2,]    1    1   -1   -1
## [3,]    1    0    2   -1
## [4,]    1    0    0    3

print(unique(Psi))

##      [,1] [,2] [,3]
## [1,]    1   -1   -1
## [2,]    1    1   -1
## [3,]    1    0    2

print(mm.TP)

##    int Ht1 Ht2 Ht3 Hp1 Hp2 Ht1:Hp1 Ht2:Hp1 Ht3:Hp1 Ht1:Hp2 Ht2:Hp2 Ht3:Hp2
## 1    1  -1  -1  -1  -1  -1       1       1       1       1       1       1
## 5    1  -1  -1  -1   1  -1      -1      -1      -1       1       1       1
## 9    1  -1  -1  -1   0   2       0       0       0      -2      -2      -2
## 13   1   1  -1  -1  -1  -1      -1       1       1      -1       1       1
## 17   1   1  -1  -1   1  -1       1      -1      -1      -1       1       1
## 21   1   1  -1  -1   0   2       0       0       0       2      -2      -2
## 25   1   0   2  -1  -1  -1       0      -2       1       0      -2       1
## 29   1   0   2  -1   1  -1       0       2      -1       0      -2       1
## 33   1   0   2  -1   0   2       0       0       0       0       4      -2
## 37   1   0   0   3  -1  -1       0       0      -3       0       0      -3
## 41   1   0   0   3   1  -1       0       0       3       0       0      -3
## 45   1   0   0   3   0   2       0       0       0       0       0       6
```

Before you read further you should look at the output from

```
alph <- LETTERS[1:24]
Malph <- matrix(alph,nrow=4)
print(alph); print(Malph); print(as.vector(Malph))
```

You will see how **R** turns a vector `alph` of length 24 into a 4×6 matrix by `matrix(p,nrow=k)`: it chops `alph` into subvectors of length 4 then makes those pieces the columns of a matrix.

Remark. In fact **R** stores a $k \times \ell$ matrix as a vector of length $k\ell$.

For the BC model with interactions:

```
raw <- out.TP$coeff
X <- model.matrix(out.TP)
dd <- dimnames(X)[[2]]
print(rbind(names(raw),dd)); print(matrix(names(raw),nrow=4))

##    [,1]          [,2]  [,3]  [,4]  [,5]  [,6]  [,7]      [,8]
##    "(Intercept)" "Ht1" "Ht2" "Ht3" "Hp1" "Hp2" "Ht1:Hp1" "Ht2:Hp1"
## dd "(Intercept)" "Ht1" "Ht2" "Ht3" "Hp1" "Hp2" "Ht1:Hp1" "Ht2:Hp1"
##    [,9]      [,10]     [,11]     [,12]
##    "Ht3:Hp1" "Ht1:Hp2" "Ht2:Hp2" "Ht3:Hp2"
## dd "Ht3:Hp1" "Ht1:Hp2" "Ht2:Hp2" "Ht3:Hp2"
##      [,1]          [,2]      [,3]
## [1,] "(Intercept)" "Hp1"     "Ht3:Hp1"
## [2,] "Ht1"         "Hp2"     "Ht1:Hp2"
## [3,] "Ht2"         "Ht1:Hp1" "Ht2:Hp2"
## [4,] "Ht3"         "Ht2:Hp1" "Ht3:Hp2"
```

In the general case, with factors $\mathbb{F}$ and $\mathbb{G}$, each vector z in $\mathcal{X}_{\mathbb{F}*\mathbb{G}}$ has a unique representation as $z = X\mathfrak{p}$, where $\mathfrak{p}$ is a $(kl) \times 1$ vector of coefficients. It helps to think of $\mathfrak{p}$ also as the $k \times \ell$ matrix B from <5.5>, which can be partitioned as

$$B = matrix(\mathfrak{p}, nrow = k) = \begin{array}{c c} & \begin{array}{cc} 1 & \ell{-}1 \end{array} \\ \begin{array}{c} 1 \\ k{-}1 \end{array} & \left[\begin{array}{cc} m & g^T \\ f & N \end{array}\right] \end{array}.$$

Then we have $\mathfrak{p} = $ `as.vector(B)` and $z = \sum_{r,s} B[r,s]\phi_r * \psi_s$. By <5.2>, we also have $z = \sum_{i,j} M[i,j]F_i * G_j$, where

$$\begin{aligned} M &= K_k B K_\ell^T \\ &= (\mathbb{1}_k, C_k)\begin{pmatrix} m & g^T \\ f & N \end{pmatrix}\begin{pmatrix} \mathbb{1}_\ell^T \\ C_\ell^T \end{pmatrix} \\ &= m\mathbb{1}_k\mathbb{1}_\ell^T + C_k f \mathbb{1}_\ell^T + \mathbb{1}_k g^T C_\ell^T + C_k N C_\ell^T. \end{aligned}$$

Define $a = C_k f$ and $b = C_\ell g$ and $D = C_k N C_\ell^T$. Note that $\mathbb{1}_k^T a = 0 = \mathbb{1}_\ell^T b$ and $\mathbb{1}_k^T D = 0$ and $D\mathbb{1}_\ell = 0$. Also

$$M[i,j] = m + a_i + b_j + D_{i,j},$$

so that

$$z = \sum\nolimits_{i,j} M[i,j]F_i * G_j = m\mathbb{1}_n + Fa + Gb + \sum\nolimits_{i,j} D[i,j]F_i * G_j,$$

as in <5.2>.

If I were not so worn out after all those calculations I would now derive the matrix that defines the transformation from G-coordinates to constrained m, a, b coordinates; then I would produce tables of estimated constrained coeffients and their estimated standard errors.

Maybe next time.

References

Box, G. E. and D. R. Cox (1964). An analysis of transformations. *Journal of the Royal Statistical Society. Series B (Methodological) 26*(2), 211–252.

Box, G. E. P., W. G. Hunter, and J. S. Hunter (1978). *Statistics for Experimenters: An Introduction to Design, Data Analysis, and Model Building.* New York: Wiley.

Chambers, J. M. and T. J. Hastie (Eds.) (1992). *Statistical Models in S.* Wadsworth.

Venables, W. N. and B. D. Ripley (2002). *Modern Applied Statistics with S* (4th ed.). Springer-Verlag.