FURTHER TOPICS IN ANALYSIS OF VARIANCE

14

ANOVA models can be expanded, for example, into multivariate analysis, split-plot designs, and repeated measures.

14.1 Model Coefficients and Contrasts **417**
14.2 Summarizing ANOVA Results **421**
  Splitting Treatment Sums of Squares into Contrast Terms 422
  Treatment Means and Standard Errors 423
  Balanced Designs 423
  2k Factorial Designs 426
  Unbalanced Designs 427
14.3 Multivariate Analysis of Variance **430**
14.4 Split-plot Designs **431**
14.5 Repeated-Measures Designs **433**
14.6 Rank Tests for One-Way and Two-Way Layouts **436**
  The Kruskal-Wallis Rank Sum Test 436
  The Friedman Rank Sum Test 436
14.7 Variance Components Models **437**
  Estimating the Model 437
  Estimation Methods 438
  Random Slope Example 439
14.8 References **440**

FURTHER TOPICS IN ANALYSIS OF VARIANCE

The previous chapter, Designed Experiments and Analysis of Variance, describes the basic techniques for using S-PLUS for analysis of variance. This chapter extends the concepts to several related topics:

- Multivariate analysis of variance (MANOVA), discussed in the section Multivariate Analysis of Variance, page 430.
- Split-plot designs (the section Split-plot Designs, page 431).
- Repeated measures (the section Repeated-Measures Designs, page 433).
- Nonparametric tests for one-way and blocked two-way designs (the section Rank Tests for One-Way and Two-Way Layouts, page 436)
- Variance components models (the section Variance Components Models, page 437)

These topics are preceded by a discussion of model coefficients and contrasts (the section Model Coefficients and Contrasts, page 417); this information is important in interpreting the available ANOVA summaries, described in the section Summarizing ANOVA Results, page 421.

14.1 MODEL COEFFICIENTS AND CONTRASTS

This section explains what the coefficients mean in ANOVA models, and how to get more meaningful coefficients for particular cases.

Suppose we have 5 measurements of a response variable `scores` for each of three treatments, `"A"`, `"B"`, and `"C"`, as shown below:

```
> scores
[1] 4 5 4 5 4 10 7 7 7 7 7 7 8 7 6
> scores.treat
[1] A A A A A B B B B B C C C C C
```

In solving the basic ANOVA problem, we are trying to solve the following simple system of equations:

$$\hat{\mu}_A = \hat{\mu} + \hat{\alpha}_A$$
$$\hat{\mu}_B = \hat{\mu} + \hat{\alpha}_B$$
$$\hat{\mu}_C = \hat{\mu} + \hat{\alpha}_C$$

The sample means $\hat{\mu}_A$, $\hat{\mu}_B$, and $\hat{\mu}_C$ can be calculated directly from the data:

$$\hat{\mu}_A = (4+5+4+5+4)/5 = 4.4$$
$$\hat{\mu}_B = (10+7+7+7+7)/5 = 7.6$$
$$\hat{\mu}_C = (7+7+8+7+6)/5 = 7.0 \quad .$$

This leaves the following three equations in four unknowns:

$$4.4 = \hat{\mu} + \hat{\alpha}_A$$
$$7.6 = \hat{\mu} + \hat{\alpha}_B$$
$$7.0 = \hat{\mu} + \hat{\alpha}_C .$$

Like all ANOVA models, this system is *overparametrized*, meaning there are more coefficients than can be estimated. We can, however, replace the three variables $\hat{\alpha}_A$, $\hat{\alpha}_B$, and $\hat{\alpha}_C$ with a pair of variables $\hat{\beta}_1$ and $\hat{\beta}_2$ that are functionally independent linear combinations of the original variables, and also independent of $\hat{\mu}$. Such a replacement can be done in more than one way. For unordered factors such as `scores.treat`, the default choice in S-PLUS is the set of *Helmert contrasts* :

$$\hat{\beta}_1 = -\hat{\alpha}_A + \hat{\alpha}_B$$
$$\hat{\beta}_2 = 2\hat{\alpha}_C - (\hat{\alpha}_A + \hat{\alpha}_B) .$$

These contrasts, in effect, contrast the *i*th level with the average of the preceding levels.

More generally, if you have variables α_i, i=1, …, k, you can reparametrize with the k - 1 variables β_j as follows:

$$\beta_j = j\alpha_{j+1} + \sum_{i=1}^{j} \alpha_i . \qquad (14.1)$$

The transpose of the matrix of coefficients for equation 14.1 is the following

$k \times (k-1)$ *contrast matrix*:

$$A = \begin{bmatrix} -1 & -1 & -1 & \dots & -1 \\ 1 & -1 & -1 & \dots & -1 \\ 0 & 2 & -1 & \dots & -1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & k-1 \end{bmatrix}. \qquad (14.2)$$

You can recover the original treatment effects $\boldsymbol{\alpha}$ from the reparametrized variables $\boldsymbol{\beta}$ by matrix multiplication as follows:

$$A\boldsymbol{\beta} = \boldsymbol{\alpha}.$$

Returning to our simple example, we can rewrite our original variables in terms of $\hat{\beta}_1$ and $\hat{\beta}_2$ as follows:

$$\begin{aligned} \hat{\alpha}_A &= -\hat{\beta}_1 - \hat{\beta}_2 \\ \hat{\alpha}_B &= \hat{\beta}_1 - \hat{\beta}_2 \quad . \\ \hat{\alpha}_C &= 2\hat{\beta}_2. \end{aligned}$$

S-PLUS now solves the following system of equations:

$$\begin{aligned} 4.4 &= \hat{\mu} - \hat{\beta}_1 - \hat{\beta}_2 \\ 7.6 &= \hat{\mu} + \hat{\beta}_1 - \hat{\beta}_2. \\ 7.0 &= \hat{\mu} + 2\hat{\beta}_2. \end{aligned}$$

If we use `aov` as usual to create the `aov` object `scores.aov`, we can use the `coef` function to look at the solved values $\hat{\mu}$, $\hat{\beta}_1$, and $\hat{\beta}_2$:

```
> scores.aov <- aov(scores ~ scores.treat)
> coef(scores.aov)
(Intercept) scores.treat1 scores.treat2
   6.333333           1.6     0.3333333
```

In our example, the contrast matrix is as follows:

$$\begin{pmatrix} -1 & -1 \\ 1 & -1 \\ 0 & 2 \end{pmatrix}.$$

You can obtain the contrast matrix for any factor object using the `contrasts` function. For unordered factors such as `scores.treat`, `contrasts` returns the Helmert contrast matrix of the appropriate size:

```
> contrasts(scores.treat)
  [,1] [,2]
A   -1   -1
B    1   -1
C    0    2
```

The contrast matrix, together with the treatment coefficients returned by `coef`, provides an alternative to using `model.tables` to calculate effects:

```
> contrasts(scores.treat) %*% coef(scores.aov)[-1]
        [,1]
A -1.9333333
B  1.2666667
C  0.6666667
```

For *ordered* factors, the Helmert contrasts are replaced, by default, with *polynomial contrasts* that model the response as a polynomial through equally spaced points. For example, suppose we define an ordered factor `water.temp` as follows:

```
> water.temp <- ordered(c(65, 95, 120))
> water.temp
[1] 65  95  120
 65 < 95 < 120
```

The contrast matrix for `water.temp` uses polynomial contrasts:

```
> contrasts(water.temp)
            .L         .Q
 65 -0.7071068  0.4082483
 95  0.0000000 -0.8164966
120  0.7071068  0.4082483
```

For the polynomial contrasts, $\hat{\beta}_1$ represents the linear component of the response, $\hat{\beta}_2$ represents the quadratic component, and so on. When examining ANOVA summaries, you can split a factor's effects into contrast terms to examine each component's contribution to the model. See the section Splitting Treatment Sums of Squares into Contrast Terms, page 422, for complete details.

At times it is desirable to give particular contrasts to some of the coefficients. In our example, you might be interested in a contrast that has A equal to a weighted average of B and C. This might occur, for instance, if the treatments were really doses. You can add a contrast attribute to the factor using the assignment form of the `contrasts` function:

```
> contrasts(scores.treat) <- c(4,-1,-3)
> contrasts(scores.treat)
  [,1]       [,2]
A    4  0.2264554
B   -1 -0.7925939
C   -3  0.5661385
```

Note that a second contrast was automatically added.

Refitting the model, we now get different coefficients (but the fit remains the same).

```
> scores.aov2 <- aov(scores ~ scores.treat)
> coef(scores.aov2)
 (Intercept) scores.treat1 scores.treat2
    6.333333    -0.4230769      -1.06434
```

More details on working with contrasts can be found in the section Contrasts: The Coding of Factors, in chapter 2.

14.2 SUMMARIZING ANOVA RESULTS

Results from an analysis of variance are typically displayed in an *analysis of variance table*, which shows a decomposition of the variation in the response: the total sum of squares of the response is split into sums of squares for each treatment and interaction and a residual sum of squares. You can obtain the ANOVA table, as we have throughout this chapter, by using `summary` on the result of a call to `aov`, such as this overly simple model for the `wafer` data:

```
> attach(wafer)
> wafer.aov <- aov( pre.mean ~ visc.tem + devtime
+                   + etchtime)
> summary(wafer.aov)
          Df Sum of Sq   Mean Sq  F Value     Pr(F)
visc.tem   2  1.343361 0.6716807 3.678485 0.0598073
devtime    2  0.280239 0.1401194 0.767369 0.4875574
etchtime   2  0.103323 0.0516617 0.282927 0.7588959
Residuals 11  2.008568 0.1825971
```

Splitting Treatment Sums of Squares into Contrast Terms

Each treatment sum of squares in the ANOVA table can be further split into terms corresponding to the treatment contrasts. By default, the Helmert contrasts are used for unordered factors and polynomial contrasts for ordered factors. For instance, with ordered factors you can assess whether the response is fairly linear in the factor by listing the polynomial contrasts separately. In the dataset `wafer`, you can examine the linear and quadratic contrasts of `devtime` and `etchtime` by using the `split` argument to the `summary` function:

```
> summary(wafer.aov, split = list(etchtime =
+     list(L = 1, Q = 2),
+     devtime = list(L = 1, Q = 2)))
              Df Sum of Sq   Mean Sq  F Value     Pr(F)
visc.tem       2  1.343361 0.6716807 3.678485 0.0598073
devtime        2  0.280239 0.1401194 0.767369 0.4875574
  devtime: L   1  0.220865 0.2208653 1.209577 0.2949025
  devtime: Q   1  0.059373 0.0593734 0.325161 0.5799830
etchtime       2  0.103323 0.0516617 0.282927 0.7588959
  etchtime: L  1  0.094519 0.0945188 0.517636 0.4868567
  etchtime: Q  1  0.008805 0.0088047 0.048219 0.8302131
Residuals     11  2.008568 0.1825971
```

Each of the (indented) split terms sum to their overall sum of squares.

The `split` argument can evaluate only the effects of the contrasts used to specify the ANOVA model: if you wish to test a specific contrast, you need to set it explicitly before fitting the model. Thus, if you want to test a polynomial contrast for an unordered factor, you must specify polynomial contrasts for the factor before fitting the model. The same is true for other non-default contrasts. For instance, the variable `visc.tem` in the `wafer` dataset is a three-level factor constructed by combining two levels of viscosity (204 and 206) with two levels of temperature (90 and 105).

```
> levels(visc.tem)
[1] "204,90"  "206,90"  "204,105"
```

To assess viscosity, supposing temperature has no effect, we define a contrast that takes the difference of the middle and the sum of the first and third levels of `visc.tem`; the contrast matrix is automatically completed:

```
> contrasts(visc.tem) <- c(-1,2,-1)
> contrasts(visc.tem)
        [,1]          [,2]
204,90    -1 -7.071068e-01
206,90     2 -1.110223e-16
204,105   -1  7.071068e-01
```

```
> wafer.aov <- aov( pre.mean ~ visc.tem + devtime
+                 + etchtime)
```

In this fitted model, the first contrast for `visc.aov` reflects the effect of viscosity:

```
> summary(wafer.aov, split = list(visc.tem = list(visc = 1)))
                 Df Sum of Sq  Mean Sq  F Value     Pr(F)
visc.tem          2  1.343361 0.671681 3.678485 0.0598073
  visc.tem: visc  1  1.326336 1.326336 7.263730 0.0208372
devtime           2  0.280239 0.140119 0.767369 0.4875574
etchtime          2  0.103323 0.051662 0.282927 0.7588959
Residuals        11  2.008568 0.182597
```

Treatment Means and Standard Errors

Commonly the ANOVA model is written in the form “grand mean plus treatment effects,”

$$y_{ijk} = \mu + \alpha_i + \beta_j + (\alpha\beta)_{ij} + \varepsilon_{ijk}$$

The treatment effects, α_i, β_j, $(\alpha\beta)_{ij}$ reflect changes in the response due to that combination of treatments. In this parameterization, the effects are constrained, usually to sum to zero.

Unfortunately, the use of the term “effect” in ANOVA is not standardized: in factorial experiments an effect is the difference between treatment levels, in balanced designs it is the difference from the grand mean, and in unbalanced designs there are (at least) two different standardizations that make sense.

The coefficients of an `aov` object returned by `coef(aov.object)` are coefficients for the contrast variables derived by the `aov` function, rather than the grand-mean-plus-effects decomposition. The functions `dummy.coef` and `model.tables` translate the internal coefficients into the more natural treatment effects.

Balanced Designs

In a balanced design, both computing and interpreting effects are straightforward:

```
> gun.aov <- aov(Rounds ~ Method + Physique/Team, gun)
> coef(gun.aov)
 (Intercept)    Method Physique.L  Physique.Q
    19.33333 -4.255556  -1.154941 -0.06123724
 PhysiqueSTeam1 PhysiqueATeam1 PhysiqueHTeam1
         1.9375           0.45          -0.45
 PhysiqueSTeam2 PhysiqueATeam2 PhysiqueHTeam2
        -0.4875    0.008333333     -0.1083333
```

The `dummy.coef` function translates the coefficients into the more natural effects:

```
> dummy.coef(gun.aov)
$"(Intercept)":
 (Intercept)
    19.33333

$Method:
       M1        M2
 4.255556 -4.255556

$Physique:
[1]  0.7916667  0.0500000 -0.8416667

$"Team %in% Physique":
   1T1        2T1       3T1   1T2       2T2
 -1.45 -0.4583333 0.5583333 2.425 0.4416667
        3T2    1T3        2T3        3T3
 -0.3416667 -0.975 0.01666667 -0.2166667
```

For the default contrasts, these effects always sum to zero.

The same information is returned in a tabulated form by `model.tables` Note that `model.tables` calls `proj`, hence it is helpful to use `qr = T` in the call to `aov`.

```
> model.tables(gun.aov, se = T)
Refitting model to allow projection
Tables of effects


 Method
    M1     M2
 4.256 -4.256

 Physique
      S    A       H
 0.7917 0.05 -0.8417

 Team %in% Physique
Dim 1 : Physique
Dim 2 : Team
```

```
     T1     T2     T3
S -1.450  2.425 -0.975
A -0.458  0.442  0.017
H  0.558 -0.342 -0.217

Standard errors of effects
     Method Physique Team %in% Physique
     0.3381   0.4141             0.7172
rep 18.0000  12.0000             4.0000
```

Using the first method, the gunners fired on average 4.26 more rounds than the overall mean. The standard errors for the effects are simply the residual standard error scaled by the replication factor, `rep`, the number of observations at each level of the treatment. For instance, the standard error for the `Method` effect is:

$$\mathrm{se(Method)} = \frac{\mathrm{se(Residual)}}{\sqrt{\mathrm{replication(Method)}}} = \frac{1.434}{\sqrt{18}} = 0.3381$$

The `model.tables` function also computes cell means for each of the treatments. This provides a useful summary of the analysis that is more easily related to the original data.

```
> model.tables(gun.aov, type = "means", se = T)
Tables of means

Grand mean
 19.33

 Method
    M1    M2
 23.59 15.08

 Physique
     S     A     H
 20.13 19.38 18.49

 Team %in% Physique
Dim 1 : Physique
Dim 2 : Team
     T1    T2    T3
S 18.68 22.55 19.15
A 18.93 19.83 19.40
H 19.05 18.15 18.28
```

```
Standard errors for differences of means
     Method Physique Team %in% Physique
     0.4782   0.5856              1.014
rep 18.0000  12.0000              4.000
```

The first method had an average firing rate of 23.6 rounds. For the tables of means, standard errors of *differences* between means are given, as these are usually of most interest to the experimenter. For instance the standard error of differences for `Team %in% Physique` is

$$\mathrm{SED} = \sqrt{2\times\frac{2.0576}{4}} = 1.014$$

To gauge the statistical significance of the difference between the first and second small physique teams, we can compute the “least significant difference (LSD)” for the `Team %in% Physique` interaction. The validity of the statistical significance is based on the assumption that the model is correct and the residuals are Gaussian. The plots of the residuals indicate these are not unreasonable assumptions for this dataset—you can verify this by creating a histogram and normal QQ-plot of the residuals as follows:

```
> hist(resid(gun.aov))
> qqnorm(resid(gun.aov))
```

The LSD at the 95% level is

$$t(0.975, 26)\times\mathrm{SED}(\texttt{Team \%*\% Physique}).$$

We use the t-distribution with 26 degrees of freedom because the residual sum of squares has 26 degrees of freedom. In S-PLUS, we type the following:

```
> qt(0.975, 26) * 1.014
[1] 2.084307
```

Since the means of the two teams differ by more than 2.08, the teams are different at the 95% level of significance. From an interaction plot it is clear that the results for teams of small physique are unusually high.

2^k Factorial Designs

In factorial experiments, where each experimental treatment has only two levels, a treatment *effect* is, by convention, the difference between the high and low levels. Interaction effects are half the average difference between paired levels of an interaction. These *factorial effects* are computed when `type = "feffects"` is used in the `model.tables` function:

```
> catalyst.aov <- aov( Yield ~ ., catalyst, qr = T)
> model.tables(catalyst.aov, type = "feffects", se = T)
```

```
Table of factorial effects
     Effects    se
Temp    23.0 5.062
Conc    -5.0 5.062
Cat      1.5 5.062
```

Unbalanced Designs

When designs are unbalanced (there are unequal numbers of observations in some cells of the experiment), the effects associated with different treatment levels can be standardized in different ways. For instance, suppose we use only the first 35 observations of the `gun` data set:

```
> gunsmall.aov <- aov(Rounds ~ Method
+               + Physique/Team, gun,
+               + subset=1:35, qr = T)
```

The `dummy.coef` function standardizes treatment effects to sum to zero:

```
> dummy.coef(gunsmall.aov)
$"(Intercept)":
 (Intercept)
    19.29177

$Method:
       M1        M2
 4.297115 -4.297115

$Physique:
[1]  0.83322650  0.09155983 -0.92478632

$"Team %in% Physique":
   1T1        2T1       3T1   1T2       2T2
 -1.45 -0.4583333 0.6830128 2.425 0.4416667

        3T2    1T3        2T3        3T3
 -0.2169872 -0.975 0.01666667  -0.466025
```

The `model.tables` function computes effects that are standardized so the weighted effects sum to zero: $\sum_{i=1}^{T} n_i \tau_i = 0$, where n_i is the replication of level i and τ_i the effect. The `model.tables` effects are identical to the values of the projection vectors computed by `proj(gunsmall.aov)`:

```
> model.tables(gunsmall.aov)
```

```
Tables of effects

 Method
        M1      M2
     4.135  -4.378
rep 18.000  17.000

 Physique
          S        A        H
     0.7923  0.05065  -0.9196
rep 12.0000 12.00000  11.0000

 Team %in% Physique
Dim 1 : Physique
Dim 2 : Team
        T1     T2     T3
S   -1.450  2.425 -0.975
rep  4.000  4.000  4.000
A   -0.458  0.442  0.017
rep  4.000  4.000  4.000
H    0.639 -0.261 -0.505
rep  4.000  4.000  3.000
```

With this standardization, treatment effects are orthogonal: consequently cell means can be computed by simply adding effects to the grand mean; standard errors are also more readily computed.

```
> model.tables(gunsmall.aov, type="means", se=T)
Standard error information not returned as design
   is unbalanced.
Standard errors can be obtained through se.contrast.
Tables of means
Grand mean

 19.45

 Method
       M1    M2
    23.59 15.08
rep 18.00 17.00
```

```
 Physique
        S    A     H
    20.25 19.5 18.53
rep 12.00 12.0 11.00

 Team %in% Physique
Dim 1 : Physique
Dim 2 : Team
       T1    T2    T3
S   18.80 22.67 19.27
rep  4.00  4.00  4.00
A   19.05 19.95 19.52
rep  4.00  4.00  4.00
H   19.17 18.27 18.04
rep  4.00  4.00  3.00
```

Note that the `(Intercept)` value returned by `dummy.coef` is not the grand mean of the data, and the coefficients returned are not a decomposition of the cell means. This is a difference that occurs only with unbalanced designs: in balanced designs the functions `dummy.coef` and `model.tables` return identical values for the effects.

In the unbalanced case, the standard errors for comparing two means depend on the replication factors, hence it could be very complex to tabulate all combinations. Instead, they can be computed directly with `se.contrast`. For instance, to compare the first and third teams of heavy physique:

```
> se.contrast(gunsmall.aov, contrast = list(
+         Physique=="S"  Team == "T1",
+         Physique=="S"  Team == "T3"),
+         data = gun[1:35,])
[1] 1.018648
```

By default, the standard error of the difference of the means specified by `contrast` is computed. Other contrasts are specified by the argument `coef`. For instance, to compute the standard error of the contrast tested in the section Splitting Treatment Sums of Squares into Contrast Terms, page 422, for the variable `visc.tem`:

```
> attach(wafer)
> se.contrast(wafer.aov, contrast = list(
+         visc.tem ==levels(visc.tem)[1],
+         visc.tem == levels(visc.tem)[2],
+         visc.tem == levels(visc.tem)[3]),
+         coef = c(-1,2,-1), data = wafer)
```

```
Refitting model to allow projection
[1] 0.07793052
```

The value of the contrast can be computed from `model.tables(wafer.aov)`. The effects for `visc.tem` are:

```
visc.tem
204,90  206,90 204,105
0.1543 -0.3839  0.2296
```

The contrast is `-0.3839 - mean(c(0.1543, 0.2296)) = -0.5758`. The standard error for testing whether the contrast is zero is 0.0779; clearly the contrast is nonzero.

14.3 MULTIVARIATE ANALYSIS OF VARIANCE

Multivariate analysis of variance, known as MANOVA, is the extension of analysis of variance techniques to multiple responses. The responses for an observation are considered as one multivariate observation, rather than as a collection of univariate responses.

If the responses are independent, then it is sensible to just perform univariate analyses. However, if the responses are correlated, then MANOVA can be more informative than the univariate analyses as well as less repetitive.

In S-PLUS the `manova` function is used to estimate the model. The formula needs to have a matrix as the response:

```
> wafer.manova <- manova(cbind(pre.mean, post.mean) ~ .,
+                        wafer[,c(1:9, 11)])
```

The `manova` function creates an object of class `"manova"`. This class of object has methods specific to it for a few generic functions. The most important function is the `"manova"` method for `summary`, which produces a MANOVA table:

```
> summary(wafer.manova)
          Df Pillai Trace approx. F num df   den df   P-value
  maskdim 1  0.9863       36.00761  2        1        0.11703
 visc.tem 2  1.00879      1.01773   4        4        0.49341
   spinsp 2  1.30002      1.85724   4        4        0.28173
 baketime 2  0.80133      0.66851   4        4        0.64704
 aperture 2  0.96765      0.93733   4        4        0.52425
  exptime 2  1.63457      4.47305   4        4        0.08795
  devtime 2  0.99023      0.98065   4        4        0.50733
 etchtime 2  1.26094      1.70614   4        4        0.30874
Residuals 2
```

There are four common types of test in MANOVA. The example above shows the Pillai-Bartlett trace test, which is the default test in S-PLUS. The last four columns show an approximate *F* test (since the distributions of the four test statistics are not implemented). The other available tests are Wilks' Lambda, Hotelling-Lawley trace, and Roy's maximum eigenvalue. (By the way, a model with this few residual degrees of freedom is not likely to produce informative tests.)

You can view the results of another test by using the `test` argument. The following command shows you Wilks' lambda test:

```
> summary(wafer.manova, test="wilk")
```

Below is an example of how to see the results of all four of the multivariate tests:

```
> wafer.manova2 <- manova(cbind(pre.mean, post.mean,
+     log(pre.dev), log(post.dev)) ~ maskdim + visc.tem
+     + spinsp, wafer)
> wafer.ms2 <- summary(wafer.manova2)
> for(i in c("p","w","h","r")) print(wafer.ms2, test=i)
```

You can also look at the univariate ANOVA tables for each response with a command like:

```
> summary(wafer.manova, univariate=T)
```

Hand and Taylor (1987) provide a nice introduction to MANOVA. Many books on multivariate statistics contain a chapter on MANOVA. Examples include Mardia, Kent and Bibby (1979), and Seber (1984).

14.4 SPLIT-PLOT DESIGNS

A split-plot design contains more than one source of error. This can arise because factors are applied at different scales, as in the guayule example below.

Split-plots are also encountered because of restrictions on the randomization. For example, an experiment involving oven temperature and baking time will probably not randomize the oven temperature totally, but rather only change the temperature after all of the runs for that temperature have been made. This type of design is often mistakenly analyzed as if there were no restrictions on the randomization (an indication of this can be *p*-values that are close to 1). See Hicks (1973) and Daniel (1976).

S-PLUS includes the `guayule` data frame which is also discussed in Chambers and Hastie (1992). This experiment was on eight varieties of guayule (a rubber producing shrub) and four treatments on the seeds. Since a flat (a shallow box for starting seedlings) was not large enough to contain all 32 combinations of variety and treatment, the design was to use only a single variety in each flat and to apply each treatment within each flat. Thus the flats each consist of four sub-plots. This is a split-plot design since flats are the experimental unit for varieties, but the sub-plots are the experimental unit for the treatments. The response is the number of plants that germinated in each sub-plot.

To analyze a split-plot design like this, put the variable that corresponds to the whole plot in an `Error` term in the formula of the `aov` call:

```
> gua.aov1 <- aov(plants ~ variety*treatment
+                 + Error(flats), guayule)
```

As usual, you can get an ANOVA table with `summary`:

```
> summary(gua.aov1)
Error: flats
          Df Sum of Sq  Mean Sq  F Value     Pr(F)
variety    7   763.156 109.0223 1.232036 0.3420697
Residuals 16  1415.833  88.4896

Error: Within
                  Df Sum of Sq  Mean Sq  F Value       Pr(F)
treatment          3  30774.28 10258.09 423.4386 0.00000e+00
variety:treatment 21   2620.14   124.77   5.1502 1.32674e-06
Residuals         48   1162.83    24.23
```

This shows varieties tested with the error from flats, while treatment and its interaction with variety are tested with the within-flat error (which is substantially smaller).

The `guayule` data actually represent an experiment in which the flats were grouped into replicates—making three sources of error, or a split-split-plot design. To model this we put more than one term inside the `Error` term:

```
> gau.aov2 <- aov(plants ~ variety*treatment
+                    + Error(reps/flats), guayule)
> summary(gau.aov2)
Error: reps
          Df Sum of Sq  Mean Sq F Value Pr(F)
Residuals  2  38.58333 19.29167

Error: flats %in% reps
          Df Sum of Sq  Mean Sq  F Value     Pr(F)
variety    7   763.156 109.0223 1.108232 0.4099625
Residuals 14  1377.250  98.3750
```

```
Error: Within
                  Df Sum of Sq Mean Sq  F Value       Pr(F)
treatment          3  30774.28 10258.09 423.4386 0.00000e+00
variety:treatment 21   2620.14   124.77   5.1502 1.32674e-06
Residuals         48   1162.83    24.23
```

The `Error` term could also have been specified as `Error(reps + flats)`. However, the specification `Error(flats + reps)` would not give the desired result (the sequence within the `Error` term is significant); explicitly stating the nesting is preferred. Note that only one `Error` term is allowed.

14.5 REPEATED-MEASURES DESIGNS

Repeated-measures designs are those that contain a sequence of observations on each subject—for example, a medical experiment in which each patient is given a drug, and observations are taken at zero, one, two and three weeks after taking the drug. (The above description is a little too simplistic to encompass all repeated-measures designs, but it captures the spirit.)

Repeated-measures designs are similar to split-plot designs in that there is more than one source of error (between subjects and within subjects), but there is correlation in the within-subjects observations. In the example we expect that the observations in week three will be more similar to week two observations than to week zero observations. Because of this, the split-plot analysis (referred to as the univariate approach) is valid only under certain restrictive conditions.

We will use the artificial dataset `drug.mult`, which has the following form:

```
> drug.mult
  subject gender  Y.1  Y.2  Y.3  Y.4
1      S1      F 75.9 74.3 80.0 78.9
2      S2      F 78.3 75.5 79.6 79.2
3      S3      F 80.3 78.2 80.4 76.2
4      S4      M 80.7 77.2 82.0 83.8
5      S5      M 80.3 78.6 81.4 81.5
6      S6      M 80.1 81.1 81.9 86.4
```

The dataset consists of the two factors `subject` and `gender`, and the matrix `Y` which contains 4 columns. The first thing to do is stretch this out into a form suitable for the univariate analysis:

```
> drug.uni <- drug.mult[rep(1:6, rep(4,6)), 1:2]
> ymat <- data.matrix(drug.mult[,paste("Y.",1:4, sep="")])
> drug.uni <- cbind(drug.uni, time=ordered(rep(paste("Week",
+         0:3,sep=""),6)), y=as.vector(t(ymat)))
```

The univariate analysis treats the data as a split-plot design:

```
> summary(aov(y ~ gender*time + Error(subject), drug.uni))
Error: subject
          Df Sum of Sq  Mean Sq  F Value   Pr(F)
gender     1  60.80167 60.80167 19.32256 0.01173
Residuals  4  12.58667  3.14667

Error: Within
            Df Sum of Sq  Mean Sq  F Value     Pr(F)
time         3  49.10833 16.36944 6.316184 0.0081378
gender:time  3  14.80167  4.93389 1.903751 0.1828514
Residuals   12  31.10000  2.59167
```

Tests in the “Within” stratum are valid only if the data satisfy the “circularity” property, in addition to the usual conditions. Circularity means that the variance of the difference of measures at different times is constant; for example, the variance of the difference between the measures at week 0 and week 3 should be the same as the variance of the difference between week 2 and week 3. We also need the assumption that actual contrasts are used; for example, the `contr.treatment` function should not be used. When circularity does not hold, then the *p*-values for the tests will be too small.

One approach is to perform tests which are as conservative as possible. Conservative tests are formed by dividing the degrees of freedom in both the numerator and denominator of the *F* test by the number of repeated measures minus one. In our example there are four repeated measures on each subject, so we divide by 3. The split-plot and the conservative tests are:

```
> 1 - pf(6.316184, 3, 12) # usual univariate test
[1] 0.008137789
> 1 - pf(6.316184, 1, 4) # conservative test
[1] 0.06583211
```

These two tests are telling fairly different tales, so the data analyst would probably move on to one of two alternatives. A Huynh-Feldt adjustment of the degrees of freedom provides a middle ground between the tests above—see Winer, Brown and Michels (1991), for instance. The multivariate approach, discussed below, substantially relaxes the assumptions.

The univariate test for “time” was really a test on three contrasts. In the multivariate setting we want to do the same thing, so we need to use contrasts in the response:

```
> drug.man <- manova(ymat %*% contr.poly(4) ~ gender,
+                    drug.mult)
> summary(drug.man, intercept=T)
```

```
           Df Pillai Trace approx. F num df   den df    P-value
(Intercept) 1  0.832005     3.301706  3        2
0.241092
     gender 1  0.694097     1.512671  3        2
0.421731
  Residuals 4
```

The line marked "(Intercept)" corresponds to "time" in the univariate approach, and similarly the "gender" line here corresponds to "gender:time". The *p*-value of .24 is larger than either of the univariate tests—the price of the multivariate analysis being more generally valid is that quite a lot of power is lost. Although the multivariate approach is preferred when the data do not conform to the required conditions, the univariate approach is preferred when they do (the trick, of course, is knowing which is which).

Let's look at the univariate summaries that this MANOVA produces:

```
> summary(drug.man, intercept=T, univar=T)
Response: .L
            Df Sum of Sq Mean Sq  F Value     Pr(F)
(Intercept)  1    22.188 22.1880 4.327255 0.1059983
gender       1     6.912  6.9120 1.348025 0.3101900
Residuals    4    20.510  5.1275

Response: .Q
            Df Sum of Sq  Mean Sq F Value     Pr(F)
(Intercept)  1  5.415000 5.415000 5.30449 0.0826524
gender       1  4.001667 4.001667 3.92000 0.1188153
Residuals    4  4.083333 1.020833

Response: .C
            Df Sum of Sq  Mean Sq  F Value     Pr(F)
(Intercept)  1  21.50533 21.50533 13.22049 0.0220425
gender       1   3.88800  3.88800  2.39016 0.1969986
Residuals    4   6.50667  1.62667
```

If you add up the respective degrees of freedom and sums of squares, you will find that the result is the same as the univariate "Within" stratum. For this reason, the univariate test is sometimes referred to as the "average *F* test".

The above discussion has focused on classical inference, which should not be done before graphical exploration of the data.

Many books discuss repeated measures. Some examples are Hand and Taylor (1987), Milliken and Johnson (1984), Crowder and Hand (1990), Winer, Brown and Michels (1991).

14.6 RANK TESTS FOR ONE-WAY AND TWO-WAY LAYOUTS

This section briefly describes how to use two non-parametric rank tests for ANOVA: the *Kruskal-Wallis* rank sum test for a one-way layout and the *Friedman* test for unreplicated two-way layout with (randomized) blocks.

Since these tests are based on ranks, they are *robust* with regard to the presence of outliers in the data; that is, they are not affected very much by outliers. This is not the case for the classical *F* tests.

You can find detailed discussions of the Kruskal-Wallis and Friedman rank-based tests in a number of books on nonparametric tests; for example, Lehmann (1975) and Hettmansperger (1984).

The Kruskal-Wallis Rank Sum Test

When you have a one-way layout, as in the section Experiments with One Factor in the previous chapter, you can use the *Kruskal-Wallis rank sum* test `kruskal.test` to test the null hypothesis that all group means are equal.

We illustrate how to use `kruskal.test` for the blood coagulation data of 13.1. First you set up your data as for a one-factor experiment (or one-way layout). You create a vector object `coag`, arranged by factor level (or treatment), and you create a factor object `diet` whose levels correspond to the factor levels of vector object `coag`. Then use `kruskal.test`:

```
> kruskal.test(coag,diet)

         Kruskal-Wallis rank sum test

data:  coag and diet
Kruskal-Wallis chi-square = 17.0154, df = 3,
 p-value = 7e-04
alternative hypothesis: two.sided
```

The *p*-value of $p = .0007$ is highly significant. This *p*-value is computed using an asymptotic chi-squared approximation. See the help file for more details.

You may find it helpful to note that `kruskal.test` and `friedman.test` return the results of its computations, and associated information, in the same style as the functions in chapter 3, Statistical Inference for One and Two Sample Problems.

The Friedman Rank Sum Test

When you have a two-way layout with one blocking variable and one treatment variable, you can use the *Friedman rank sum* test `friedman.test` to test the null hypothesis that there is no treatment effect.

We illustrate how you use `friedman.test` for the penicillin yield data described in 13.2 of the previous chapter. The general form of the usage is

```
friedman.test(y,groups,blocks)
```

where `y` is a numeric vector, `groups` contains the levels of the treatment factor and `block` contains the levels of the blocking factor. Thus you can do:

```
> attach(pen.df) # make treatment and blend available
> friedman.test(yield, treatment, blend)

         Friedman rank sum test

data:  yield and pen.design[, 2] and pen.design[, 1]
Friedman chi-square = 3.4898, df = 3, p-value = 0.3221
alternative hypothesis: two.sided
```

The *p*-value is *p*=.32, which is not significant. This *p*-value is computed using an asymptotic chi-squared approximation. For further details on `friedman.test`, see the help file.

14.7 VARIANCE COMPONENTS MODELS

Variance components models are used when there is interest in the variability of one or more variables other than the residual error. For example, manufacturers often run experiments to see which parts of the manufacturing process contribute most to the variability of the final product. In this situation variability is undesirable, and attention is focused on improving those parts of the process that are most variable. Animal breeding is another area in which variance components models are routinely used. Some data, from surveys for example, that have traditionally been analyzed using regression can more profitably be analyzed using variance component models.

Estimating the Model

To estimate a variance component model, you first need to use `is.random` to state which factors in your data are random. A variable that is marked as being random will have a variance component in any models that contain it. Only variables that inherit from class `"factor"` can be declared random. Although `is.random` works on individual factors, it is often more practical to use it on the columns of a data frame. You can see if variables are declared random by using `is.random` on the data frame:

```
> is.random(pigment)
Batch Sample Test
    F      F    F
```

Declare variables to be random by using the assignment form of `is.random`:

```
> is.random(pigment) <- c(T, T, T)
> is.random(pigment)
Batch Sample Test
    T      T    T
```

Because we want all of the factors to be random, we could have simply done the following:

```
> is.random(pigment) <- T
```

The value on the right is replicated to be the length of the number of factors in the data frame.

Once you have declared your random variables, you are ready to estimate the model using the `varcomp` function. This function takes a formula and other arguments very much like `lm` or `aov`. Because the pigment data are from a nested design, the call has the following form:

```
> pigment.vc <- varcomp(Moisture ~ Batch/Sample, pigment)
> pigment.vc
Variances:
    Batch Sample %in% Batch Residuals
 7.127976          28.53333 0.9166667
Call:
varcomp(formula = Moisture ~ Batch/Sample, data = pigment)
```

The result of `varcomp` is an object of class `"varcomp"`. You can use `summary` on `"varcomp"` objects to get more details about the fit, and you can use `plot` to get QQ-plots for the normal distribution on the estimated effects for each random term in the model.

Estimation Methods

The `method` argument to `varcomp` allows you to choose the type of variance component estimator. Maximum likelihood and REML (restricted maximum likelihood) are two of the choices. REML is very similar to maximum likelihood but takes the number of fixed effects into account (the usual unbiased estimate of variance in the one-sample model is a REML estimate). See Harville (1977) for more details on these estimators.

The default method is a MINQUE (minimum norm quadratic unbiased estimate); this class of estimator is locally best at a particular spot in the parameter space. The MINQUE option in S-Plus is locally best if all of the variance components (except that for the residuals) are zero. The MINQUE estimate agrees with REML for balanced data. See Rao (1971) for details. This method was made the default because it is less computationally intense than the other methods, however, it can do significantly worse for severely unbalanced data (Swallow and Monahan (1984)).

You can get robust estimates by using `"method=winsor"`. This method creates new data by moving outlying points or groups of points toward the rest of the data. One of the standard estimators is then applied to this possibly revised data. Burns (1992) gives details of the algorithm along with simulation results. This method uses much larger amounts of memory than the other methods if there are a large number of random levels, such as in a deeply nested design.

Random Slope Example

We now produce a more complicated example in which there are random slopes and intercepts. The data consist of several pairs of observations on each of several individuals in the study. An example might be that the *y* values represent the score on a test and the *x* values are the time at which the test was taken.

Let's start by creating simulated data of this form. We create data for 30 subjects and 10 observations per subject:

```
> subject <- factor(rep(1:30, rep(10,30)))
> set.seed(357) # makes these numbers reproducible
> trueslope <- rnorm(30, mean=1)
> trueint <- rnorm(30, sd=.5)
> times <- rchisq(300, 3)
> scores <- rep(trueint, rep(10,30)) + times *
+     rep(trueslope, rep(10,30)) + rnorm(300)
> test.df <- data.frame(subject, times, scores)
> is.random(test.df) <- T
> is.random(test.df) subject T
```

Even though we want to estimate random slopes and random intercepts, the only variable that is declared random is `subject`. Our model for the data has two coefficients: the mean slope (averaged over subjects) and the mean intercept. It also has three variances: the variance for the slope, the variance for the intercept, and the residual variance.

The following command estimates this model using Maximum Likelihood (the default MINQUE is not recommended for this type of model):

```
> test.vc <- varcomp(scores ~ times * subject,
+     data=test.df, method="ml")
```

This seems very simple. We can see how it works by looking at how the formula get expanded. The right side of the formula is expanded into four terms:

```
scores ~ 1 + times + subject + times:subject
```

The intercept term in the formula, represented by 1, gives the mean intercept. The variable `times` is fixed and produces the mean slope. The `subject` variable is random and produces the variance component for the random intercept. Since any interaction containing a random variable is considered random, the last term, `times:subject`, is also random; this term gives the variance component for the random slope. Finally, there is always a residual variance.

Now we can look at the estimates:

```
> test.vc
Variances:
   subject times:subject Residuals
 0.3162704      1.161243 0.8801149
Message:
[1] "RELATIVE FUNCTION CONVERGENCE"
Call:
varcomp(formula = scores ~ times*subject, data=test.df,
  method = "ml")
```

This shows the three variance components. The variance of the intercept, which has true value .25, is estimated as .32. Next, labeled `times:subject` is the variance of the slope, and finally the residual variance. We can also view the estimates for the coefficients of the model, which have true values of 0 and 1.

```
> coef(test.vc)
 (Intercept)   times
   0.1447211 1.02713
```

14.8 REFERENCES

Burns, P. J. (1992). *Winsorized REML estimates of variance components*. Submitted.

Chambers, J.M. and Hastie, T.J. (1992). *Statistical Models in S*. Wadsworth and Brooks Cole Advanced Books and Software, Pacific Grove, CA.

Crowder, M. J. and Hand, D. J. (1990). *Analysis of Repeated Measures*. Chapman and Hall, London.

Daniel, C. (1976). *Applications of Statistics to Industrial Experimentation*. Wiley, New York.

Hand, D. J. and Taylor, C. C. (1987). *Multivariate Analysis of Variance and Repeated Measures*. Chapman and Hall, London.

Harville, D. A. (1977). *Maximum likelihood approaches to variance component estimation and to related problems (with discussion).* Journal of the American Statistical Association, 72:320–340.

Hettmansperger, T.P. (1984). *Statistical Inference Based on Ranks*. John Wiley, New York.

Hicks, C. R. (1973). *Fundamental Concepts in the Design of Experiments*. Holt, Rinehart and Winston, New York.

Lehmann, E.L. (1975). *Nonparametrics: Statistical Methods Based on Ranks*.

Holden-Day, San Francisco.

Mardia, K. V. and Kent, J. T. and Bibby, J. M. (1979). *Multivariate Analysis*. Academic Press, London.

Milliken, G. A. and Johnson, D. E. (1984). *Analysis of Messy Data Volume 1: Designed Experiments*. Van Nostrand Reinhold, New York.

Rao, C. R. (1971). *Estimation of variance and covariance components—MINQUE theory.* Journal of Multivariate Analysis, 1:257–275.

Seber, G. A. F. (1984). *Multivariate Observations*. Wiley, New York.

Swallow, W. H. and Monahan, J. F. (1984). *Monte Carlo comparison of ANOVA, MIVQUE, REML, and ML estimators of variance components*. Technometrics, 26:47–57.

Winer, B. J. and Brown, D. R. and Michels, K. M. (1991). *Statistical Principles in Experimental Design*. McGraw-Hill, New York.