Documentation for the 'aov.contrast' Function

This documentation has four sections. Section I presents documentation in the standard R-Help format. It is probably more useful for people who are experienced users of R. Section II has a more discursive explanation of the aov.contrast function that is probably more useful to more inexperienced users of R. Section III has examples that use 'aov.contrast' in the context of an analysis of covariance. Section IV shows the R code for the 'aov.contrast' function.

Section I: R documentation for the 'aov.contrast' function

Description

Computes t-tests, confidence intervals, and other statistics for user-defined contrasts (linear combinations of cell means) in a between-subjects analysis of variance (completely randomized design).

Usage

aov.contrast <- function(lm.input, margin=1, contrast.list, strata=NA,
                         Confidence.Level = .95, var.equal=TRUE, show.cellcode=TRUE, p.digits=6)

Arguments

lm.input A 'lm' or 'aov' object, i.e., the output of a call to 'lm' or 'aov'.
margin A vector of positive integers that indicate the factor or factors over whose levels the contrast is defined. The factors are listed in the same order as in the 'formula' argument of the call to 'aov'. For example, if the 'Y ~ A1*A2*A3' is the formula in the call to 'aov', then 'margin=3' indicates that the contrast coefficients apply to the levels of the 'A3' factor.
contrast.list A list of numeric vectors. Each vector indicates the contrast coefficients for the marginal means specified by 'margin'. All vectors must have the same length as the number of marginal means determined by 'margin'.
strata If 'strata' is non-missing, it must be a vector of integers indicating the factor or factors that play the role of strata. Each contrast is computed within every fixed level of the stratifying factors.
Confidence.Level Level of confidence for the confidence intervals.
var.equal If var.equal = TRUE if and only if homogeneity of variance is assumed in the test of the contrast. The default is var.equal = TRUE.
show.cellcode When margin specifies more than one factor, e.g., 'margin = c(2, 3)', the contrast coefficients define a linear combination with respect to all combinations of factors 2 and 3. If factor 2 has k levels and factor 3 has m levels, there are k*m combinations of levels. If 'show.cellcode = TRUE', then the output displays the correspondence between the sequence of coefficients specified through 'contrast.list' and the k*m levels of factors 2 and 3.
p.digits

The number of digits to the right of the decimal place on the p-value. Set 'p.digits = NA' to request scientific notation.

Details

The 'aov' object that is specified by 'lm.input' determines the error term from which all contrast standard errors are calculated. The 'call' component of 'lm.input' identifies the dependent variable and factors in terms of which the contrast is computed. The 'aov.contrast' function can identify these variables under either of two conditions: (a) the variable is contained in a dataframe that was specified in the 'data' argument of the call to 'aov', or (b) the variable is on the search path. If neither (a) nor (b) is satisfied, then 'aov.contrast' will fail to find the variables in terms of which the contrasts are defined. Each component of 'contrast.list' should be a numeric vector of the same length as the number of marginal means specified by 'margin'. If var.equal = TRUE, the standard error of a contrast is based on the error term from lm.input (deviance(lm.input)) and the degrees of freedom (df's) are the same as the df's of deviance(lm.input). If var.equal = FALSE, the standard error of a contrast is based on the within cell error of the cells that have non-zero coefficients in the contrast, and the df's are computed by the Welch-Satterthwaite formula (Brownlee, 1965).

Value

The output is a matrix with one row for each contrast that is specified in the 'contrast.list' argument. The output includes the estimate of the contrast, confidence bounds, t statistic, p-value of the t statistic, the standard error of the contrast, and the degrees of freedom for error.

References


Welch, B. L. (1947). The generalization of 'Student's' problem when several different population variances are involved. Biometrika, 34, 23-35.

Examples

# First we create a data set with 3 factors (2 x 3 x 4 design, 2 replications per cell).

x1 <- rep(sort(rep(1:4, 2)), 6)
x2 <- rep(sort(rep(1:3,8)), 2)
x3 <- sort(rep(1:2, 24))
set.seed(10)
y <- x1 + x2 + x3 + x1*x3 + x2*x3 + rnorm(length(x1), 0, 3)
data.set <- data.frame(y = y,
                        x1 = factor(x1), x2 = factor(x2), x3 = factor(x3))
aov.out <- aov(y ~ x1*x2*x3, data = data.set)

# The next command computes 2 contrasts with respect to the x1 factor.
# The first contrast is the difference between the first and second marginal mean.
# The second contrast is the difference between the average of the first and
# second marginal mean and the third marginal mean.

aov.contrast(aov.out, margin=1,
              contrast.list=list(c( 1, -1, 0, 0), c(.5, .5, -1, 0)))
# The next set of commands checks that the preceding aov.contrast commands computed
# the correct estimates for the two contrasts.

cell.means <- tapply(data.set$y, data.set[,c("x1","x2","x3")], mean)
x1.means <- apply(cell.means, 1, mean)
c("contrast 1" = x1.means[1] - x1.means[2],
   "contrast 2" = .5*x1.means[1] + .5*x1.means[2] - x1.means[3])

# For more precision on the p-values in the output, use either
aov.contrast(aov.out, margin=1,
   contrast.list=list(c( 1, -1, 0, 0), c( .5, .5, -1, 0)),
   p.digits=9 )
aov.contrast(aov.out, margin=1,
   contrast.list=list(c( 1, -1, 0, 0), c( .5, .5, -1, 0)),
   p.digits=NA )

# The next command uses the Welch-Satterthwaite procedure for computing a
# test of the contrast without assuming homogeneity of variance.
aov.contrast(aov.out, margin=1,
   contrast.list=list(c( 1, -1, 0, 0), c( .5, .5, -1, 0)),
   var.equal = FALSE )

# The next command computes the same two contrasts as above, except now the
# contrasts are computed separately for each level of x3.
aov.contrast(aov.out, margin=1,
   contrast.list=list(c( 1, -1, 0, 0), c( .5, .5, -1, 0)),
   strata = 3 )

# The next set of commands checks that the preceding aov.contrast command computed
# the correct estimates for the two contrasts within the levels of x3.
x1.3.means <- apply(cell.means, c(1,3), mean)
x1.3.means[1,] - x1.3.means[2,]
.5*x1.3.means[1,] + .5*x1.3.means[2,] - x1.3.means[3,]

# The next command repeats this analysis without assuming homogeneity of variance.
aov.contrast(aov.out, margin=1,
   contrast.list=list(c( 1, -1, 0, 0), c( .5, .5, -1, 0)),
   strata = 3, var.equal = FALSE )

# The next aov.contrast command computes an interaction contrast between
# levels 2 and 3 of the x1 factor and levels 1 and 2 of the x2 factor.
aov.contrast(aov.out, margin=c(1,2), contrast.list=list( 
   c( 0, 1, -1, 0, -1, 1, 0, 0, 0, 0, 0, 0) ))

# The next set of commands checks that the preceding aov.contrast command computed
# the correct estimate for the interaction contrast.
x1.x2.means <- apply(cell.means, c(1, 2), mean)
x1.x2.means[1,2] - x1.x2.means[1,3] - x1.x2.means[2,2] + x1.x2.means[2,3]

# The next set of commands shows the computation of contrasts between adjusted means
# in a oneway analysis of covariance (ancova).
Section II: How to use 'aov.contrast' to compute contrasts between marginal means in a between-subjects analysis of variance (completely randomized design)

The 'aov.contrast' function computes tests and confidence intervals for linear combinations of means in a between-subjects analysis of variance. The tests can apply to main effects of factors, to interaction contrasts, or to contrasts computed within fixed levels of other factors.

The following R commands create a hypothetical data set that will be used in all subsequent examples.

---

<table>
<thead>
<tr>
<th>R Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data.set &lt;- data.frame(</td>
<td></td>
</tr>
<tr>
<td>score = c(43, 55, 51, 36, 52, 61, 58, 62, 55, 79, 63),</td>
<td></td>
</tr>
<tr>
<td>method = factor(c(&quot;A&quot;,&quot;A&quot;,&quot;A&quot;,&quot;B&quot;,&quot;B&quot;,&quot;B&quot;,&quot;C&quot;,&quot;C&quot;,&quot;C&quot;)),</td>
<td></td>
</tr>
<tr>
<td>age = c(72, 74, 71, 69, 70, 73, 73, 75, 70, 75, 73))</td>
<td></td>
</tr>
<tr>
<td>lm.results &lt;- lm(score ~ method + age, data = data.set)</td>
<td></td>
</tr>
<tr>
<td>aov.contrast(lm.results,</td>
<td></td>
</tr>
<tr>
<td>contrast.list = list(c(1, -1, 0), c(1, 0, -1), c(0, 1, -1)))</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Section II: How to use 'aov.contrast' to compute contrasts between marginal means in a between-subjects analysis of variance (completely randomized design)</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>The 'aov.contrast' function computes tests and confidence intervals for linear combinations of means in a between-subjects analysis of variance. The tests can apply to main effects of factors, to interaction contrasts, or to contrasts computed within fixed levels of other factors.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>The following R commands create a hypothetical data set that will be used in all subsequent examples.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>x1 &lt;- rep(sort(rep(1:4, 2)), 6)</td>
<td>These commands create the codes for three factors.</td>
</tr>
<tr>
<td>x2 &lt;- rep(sort(rep(1:3,8)), 2)</td>
<td></td>
</tr>
<tr>
<td>x3 &lt;- sort(rep(1:2, 24))</td>
<td></td>
</tr>
<tr>
<td>cbind(x1, x2, x3)</td>
<td>This command lets you see the values of x1, x2 and x3.</td>
</tr>
<tr>
<td>set.seed(10)</td>
<td>'set.seed' sets the initial state of the random number generator to '10'.</td>
</tr>
<tr>
<td></td>
<td>The only reason to do this is that it causes all users to draw the same</td>
</tr>
<tr>
<td></td>
<td>random sample.</td>
</tr>
<tr>
<td>y &lt;- x1 + x2 + x3 + x1<em>x3 + x2</em>x3 + rnorm(48, 0, 8)</td>
<td>This command creates a variable, y, that results from the additive</td>
</tr>
<tr>
<td></td>
<td>effects of x1, x2, and x3, and the interaction of x3 with x1 and with x2.</td>
</tr>
<tr>
<td></td>
<td>It also has a normal random error component.</td>
</tr>
<tr>
<td>data.set &lt;- data.frame(y = y,</td>
<td>This command creates a dataframe called 'data.set' with variables, y,</td>
</tr>
<tr>
<td>x1 = factor(x1),</td>
<td>x1, x2, and x3. The command also converts x1, x2, and x3 from numeric vectors to factors.</td>
</tr>
<tr>
<td>x2 = factor(x2),</td>
<td></td>
</tr>
<tr>
<td>x3 = factor(x3))</td>
<td></td>
</tr>
</tbody>
</table>
The experimental design is a 2 x 3 x 4 design with 2 replications per cell. Notation for the population cell means for this design are shown in Table 1.

The marginal means shown in Table 1 average the cell means along a row or column within a level of x3. For example, $\mu_{111}$ is the mean of $\mu_{111}$, $\mu_{112}$, and $\mu_{113}$. When averaging over all other factors, the marginal means are $\mu_{1••}$, $\mu_{2••}$, $\mu_{3•}$, and $\mu_{4•}$ for the x1 factor; $\mu_{•11}$, $\mu_{•12}$, and $\mu_{•13}$ for the x2 factor; and $\mu_{••1}$ and $\mu_{••2}$ for the x3 factor.

Contrasts can be defined in terms of linear combinations of cell means or linear combinations of marginal means. For example, $\mu_{1••} - \mu_{2••}$ is the difference between levels 1 and 2 of x1 averaged over all other factors; $\mu_{1•1} - \mu_{2•1}$ is the difference between levels 1 and 2 of x1 averaged over x2 with x3 held fixed at x3 = 1; $\mu_{111} - \mu_{211}$ is the difference between levels 1 and 2 of x1 with x2 and x3 held fixed at x2 = 1 and x3 = 1, respectively. Interaction contrasts involve more than one level of several factors. For example, $\mu_{12•} - \mu_{13•} + \mu_{23•}$ examines whether the difference between levels 2 and 3 of x2 is different at x1 = 1 and x1 = 2, after averaging over x3. Examples will be given that show how to test all of these different types of contrasts.

The R functions, 'lm' and 'aov', are used to compute analyses of variance. This documentation assumes that you are already familiar with the use of 'lm' and 'aov'. All examples in this documentation are based on 'aov' but nothing in these examples would change if 'lm' were to replace the use of 'aov' in the examples. For example,

```r
aov.out <- aov(y ~ x1*x2*x3, data = data.set)
```

This command computes an analysis of variance with dependent variable y and independent variables x1, x2 and x3. All interactions are included in the model. The results of the analysis of variance are stored under the name, 'aov.out'.

The preceding code stores the output of the call to 'aov' in an object called 'aov.out' (any other name could have been chosen, e.g., 'Nancy'); 'aov' is called an 'aov' object because it was produced by a call to the 'aov' function. The 'data' argument is not strictly necessary because the variables, y, x1, x2, and x3, exist on the search path, but it is generally good practice to use an explicit 'data' argument as a way to

---

1 'lm' is used to fit regression and ancova models as well as anova models.
maintain a clear record of where the variables are to be found. There are many ways to extract the information that is stored in 'aov.out'. The following code displays a few relevant aspects of the information in 'aov.out'.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>anova(aov.out)</code></td>
<td>This command shows the standard anova table for this analysis.</td>
</tr>
<tr>
<td><code>aov.out$call</code></td>
<td>This command shows the 'call' component of the 'aov.out' object. The 'aov.contrast' function uses the 'call' component to identify the data set and variables with respect to which the contrasts are defined.</td>
</tr>
<tr>
<td><code>deviance(aov.out)</code></td>
<td>These commands shows the value of the error term and degrees of freedom for error.</td>
</tr>
<tr>
<td><code>aov.out$df.resid</code></td>
<td>These commands shows the value of the error term and degrees of freedom for error.</td>
</tr>
</tbody>
</table>

**Example 1:** The simplest version of the 'aov.contrast' function is as follows.

```r
aov.contrast(lm.input = aov.out, margin = 1, contrast = c(1, -.5, -.5, 0))
```

Specifying 'lm.input = aov.out' tells 'aov.contrast' what data set to use (data.set) and the names of the dependent and independent variables. Specifying 'margin = 1' indicates that the contrasts will be defined with respect to the first independent variable in the call to 'aov' - in this case, this would be the x1 factor. Contrasts on the marginals of the other factors would be specified with 'margin = 2' or 'margin = 3'. The 'contrast.list' argument (abbreviated to 'contrast') indicates that we want to compute statistics for the difference between the first marginal mean of x1 and the average of the second and third marginal mean.

The output from this call to 'aov.contrast' is shown below:

```
Data:  data.set
Model: y ~ x1 * x2 * x3
MS Error = 36.3309808320505, df.error = 24, N.valid = 48, N.missing = 0
Confidence level = 0.95

Contrasts for means of factor x1
Contrast coefficients:
C1  1.000 -0.500 -0.500  0.000
Contrast tests assume homogeneity of variance.
1 -3.5927904   4.3982703 2.1310496 24 -1.6892100 0.104131
```

The output tells us that the data for the analysis is in a dataframe called 'data.set'; that the original call to 'aov' fitted the model: 'y ~ x1 * x2 * x3'; and that the error term that was found for this model was .32.76 with 24 degrees of freedom (df's). The output then shows the correspondence between the levels of the x1 factor and the contrast coefficients. One could fit a different model, e.g., 'y ~ x1 + x2 + x3', and the error term would naturally change. The model formula in the 'aov.contrast' output reminds us that the tests are based on the error term from the fit of a specific anova model, in this case, 'y ~ x1 * x2 * x3'.

The output gives us the estimate of the contrast, the 95% confidence bounds for the contrast, the t-statistic, p-value, and standard error of the contrast, and the degrees of freedom for error. If these results were based on real data, one could report the results as follows: "The test for the difference between $\mu_1$••
and the mean of $\mu_{2\bullet\bullet}$ and $\mu_{3\bullet\bullet}$ was highly significant ($t(24) = -1.69$, MS error = 36.33, $p = .10$). The 95% confidence interval for $\mu_{1\bullet\bullet} - (\mu_{2\bullet\bullet} + \mu_{3\bullet\bullet})/2$ was $-3.60 \pm 4.40$.

Example 2: The 'aov.contrast' function allows us to compute statistics several contrasts at once, as long as all the contrast pertain to the same 'aov' object, and the same factor(s). For example:

```r
aov.contrast(lm.input = aov.out, margin = 1, contrast = list(  
c(1, -1, 0, 0),
c(1, -.5, -.5, 0),
c(1, -1/3, -1/3, -1/3) ) )
```

yields statistics for three contrasts. Note that the 'contrast.list' argument must now be a list of numeric vectors, each having the same length as the levels of the first factor.

```
| Data: data.set |
| Model: y ~ x1 * x2 * x3 |
| MS Error = 36.3309808320505, df.error = 24, N.valid = 48, N.missing = 0 |
| Confidence level = 0.95 |
| Contrasts for means of factor x1 |
| Contrast coefficients: |
| C1 | 1.000 | -1.000 | 0.000 | 0.000 |
| C2 | 1.000 | -0.500 | -0.500 | 0.000 |
| C3 | 1.00000000 | -0.33333333 | -0.33333333 | -0.33333333 |
| Contrast tests assume homogeneity of variance. |
| C1 | -2.6908761 | 5.0786851 | 2.4607242 | 24 | -1.0935302 | 0.285016 |
| C2 | -3.5997904 | 4.3982703 | 2.1310496 | 24 | -1.6892100 | 0.104131 |
| C3 | -4.1995700 | 4.1467290 | 2.0091729 | 24 | -2.0901984 | 0.047370 |
```

The names for the contrasts, 'C1', 'C2', and 'C3', were automatically generated, but the user can specify names that override these default names, e.g.,

```r
aov.contrast(lm.input = aov.out, margin = 1, contrast = list(  
"Dif 1 v 2" = c(1, -1, 0, 0),
"Dif 1 v 23" = c(1, -.5, -.5, 0),
"Dif 1 v 234" = c(1, -1/3, -1/3, -1/3) ) )
```

Example 3: The 'strata' argument of 'aov.contrast' allows one to test a contrast within each level of the stratifying factor(s). For example,

```r
aov.contrast(lm.input = aov.out, margin = 1, contrast = c(1, -1, 0, 0), strata=3)
```

computes a test of the difference between the first and second levels of x1 within each of the levels of x3. It also computes a test of this contrast averaged over the levels of x3. Here is the output.
Data: data.set
Model: y ~ x1 * x2 * x3
MS Error = 36.3309808320505, df.error = 24, N.valid = 48, N.missing = 0
Confidence level = 0.95

Contrasts for means of factor x1 within levels of factor x3
Notation: 'C1' alone refers to the 'C1' contrast averaged over the other factors.
          'C1 | i' refers to the 'C1' contrast computed within x3 = i

Contrast coefficients:
C1  1.000 -1.000  0.000  0.000

Contrast tests assume homogeneity of variance.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>-2.6908761</td>
<td>5.0786851 2.4607242</td>
<td>24</td>
<td>-1.09353016</td>
<td>0.285016</td>
<td></td>
</tr>
<tr>
<td>C1</td>
<td>1</td>
<td>-7.9924190</td>
<td>7.1823453 3.4799895</td>
<td>24</td>
<td>-2.29667904</td>
<td>0.030668</td>
</tr>
<tr>
<td>C1</td>
<td>2</td>
<td>2.6106668</td>
<td>7.1823453 3.4799895</td>
<td>24</td>
<td>0.75019386</td>
<td>0.460430</td>
</tr>
</tbody>
</table>

The output that is labeled 'C1' shows the results for the contrast averaged over the levels of x2 and x3, i.e.,
this output shows that the 95% confidence interval for $\mu_{1\bullet\bullet} - \mu_{2\bullet\bullet}$ is $-2.28 \pm 4.82$. (See Table 1 for the
notation for population means.) The output that is labeled 'C1 | 1' shows the same contrast computed
within x3 = 1 (still averaged over x2) and the output that is labeled 'C1 | 2' shows this contrast computed
within x3 = 2 (again averaged over x2). Thus the 95% confidence interval for $\mu_{1\bullet 1} - \mu_{2\bullet 1}$ is $-2.19 \pm
6.82$ and the 95% confidence interval for $\mu_{1\bullet 2} - \mu_{2\bullet 2}$ is $-2.37 \pm 6.82$.

The 'strata' argument can also be a vector representing several factors, in which case the specified
contrast is computed within fixed levels of every combination of the stratifying factors. For example:

```r
aov.contrast(lm.input = aov.out, margin = 1,
             contrast = c(1, -1, 0, 0), strata = c(2, 3))
```

specifies the contrast between the first and second levels of x1 computed within each combination of
levels of x2 and x3. Here is the output.
Data:  data.set  
Model:  y ~ x1 * x2 * x3  
MS Error = 36.3309808320505, df.error = 24, N.valid = 48, N.missing = 0  
Confidence level = 0.95  
Contrasts for means of factor x1 within levels of factors x2 * x3  
Notation: 'C1' alone refers to the 'C1' contrast averaged over the other factors.  
'C1 | i*j' refers to the 'C1' contrast computed within x2 = i & x3 = j  

Contrast coefficients:  
C1  1.000 -1.000  0.000  0.000  

Contrast tests assume homogeneity of variance.  

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>-2.6908761</td>
<td>5.0786851</td>
<td>2.4607242</td>
<td>24</td>
<td>-1.09353016</td>
<td>0.285016</td>
</tr>
<tr>
<td>C1</td>
<td>1*1</td>
<td>5.2199676</td>
<td>12.4401870</td>
<td>6.0275186</td>
<td>24</td>
<td>0.86602264</td>
</tr>
<tr>
<td>C1</td>
<td>1*2</td>
<td>-3.3165813</td>
<td>12.4401870</td>
<td>6.0275186</td>
<td>24</td>
<td>-0.55023991</td>
</tr>
<tr>
<td>C1</td>
<td>2*1</td>
<td>-16.9628483</td>
<td>12.4401870</td>
<td>6.0275186</td>
<td>24</td>
<td>-2.81423408</td>
</tr>
<tr>
<td>C1</td>
<td>2*2</td>
<td>12.8758010</td>
<td>12.4401870</td>
<td>6.0275186</td>
<td>24</td>
<td>2.13616942</td>
</tr>
<tr>
<td>C1</td>
<td>3*1</td>
<td>-12.2343761</td>
<td>12.4401870</td>
<td>6.0275186</td>
<td>24</td>
<td>-2.02975335</td>
</tr>
<tr>
<td>C1</td>
<td>3*2</td>
<td>-1.7272194</td>
<td>12.4401870</td>
<td>6.0275186</td>
<td>24</td>
<td>-0.28655563</td>
</tr>
</tbody>
</table>

In addition to showing that the 95% confidence interval for \( \mu_{1••} - \mu_{2••} \) is \(-2.69 \pm 5.08\), the output shows, e.g., that the 95% confidence interval for \( \mu_{111} - \mu_{211} \) is \(5.21 \pm 12.44\), and the 95% confidence interval for \( \mu_{112} - \mu_{212} \) is \(-3.32 \pm 12.44\), etc.

**Example 4:** It is permissible to specify more than one factor in the 'margin' argument. This is useful for testing interaction contrasts. For example, suppose that the following contrast is specified:

```r
aov.contrast(lm.input = aov.out, margin = c(1, 2), contrast = c( 0, 1, -1, 
0, -1, 1, 
0, 0, 0, 
0, 0, 0 ))
```

To understand this call to 'aov.contrast', we need to see how 'aov.contrast' enumerates the levels of factors x1 and x2. Table 2 shows the correspondence between levels of x1 and x2 and the sequence of contrast coefficients. From Table 2, we can see that the contrast specified above is:  
(\( \mu_{12•} - \mu_{13•} \)) - (\( \mu_{22•} - \mu_{23•} \)). Essentially, when the 'margin' argument specifies multiple factors, the factor levels are incremented starting with the last (right-most) factor. Here is the output.

<table>
<thead>
<tr>
<th>x2</th>
<th>x1</th>
<th>Level 1</th>
<th>Level 2</th>
<th>Level 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level 1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Level 2</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Level 3</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>Level 4</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td></td>
</tr>
</tbody>
</table>
Data: data.set
Model: y ~ x1 * x2 * x3
MS Error = 36.3309808320505, df.error = 24, N.valid = 48, N.missing = 0
Confidence level = 0.95

Contrasts for means of factors x1 * x2
Correspondence between the contrast coefficients and the levels of [1] x1 * x2
x2
x1   1  2  3
  1  1  2  3
  2  4  5  6
  3  7  8  9
  4 10 11 12

Contrast coefficients:
C1  0.000  1.000 -1.000  0.000 -1.000  1.000  0.000  0.000  0.000  0.000  0.000  0.000

Contrast tests assume homogeneity of variance.
1  4.9372741   12.440187 6.0275186 24 0.81912216 0.420777

The output shows that we cannot reject \((\mu_{12} - \mu_{13}) - (\mu_{22} - \mu_{23}) = 0\) at the .05 level \((t(24) = .82, \text{MS error} = 36.33, p = .42)\).

Section III. Analysis of Covariance (Ancova) Examples
The computation of contrasts in an ancova is essentially the same as in an anova, except that the contrasts are linear combinations of adjusted cell means rather than cell means. Adjusted cell means are the predicted values for cells where the predictions are computed with respect to the mean values of the covariates. The following examples show how to compute adjusted cell means and contrasts among the adjusted cell means, but the anova contrasts that were computed in previous examples can also be computed in the context of an ancova.

Oneway Ancova Data Set:
Suppose that children are taught a skill by one of three methods, A, B, or C. Their proficiency with the skill is measured on a test, and stored in a variable called 'score'. Their age in months at the beginning of training is recorded in a variable, 'age'.

\[
data.set <- \text{data.frame(}
\text{score = c(43, 55, 51, 36, 52, 61, 58, 62, 55, 79, 63),}
\text{method = factor(c("A","A","A","A","B","B","B","C","C","C")),}
\text{age = c(72, 74, 71, 69, 70, 73, 73, 75, 70, 75, 73))}
\]

Example 5. Adjusted Cell means
\[
\text{lm.results <- lm(score ~ method + age, data = data.set)}
\]
The adjusted cell means are 48.64, 57.84, and 61.72 for levels A, B, and C of the 'method' factor. These means are the predicted values of the A, B, and C methods at the mean value of 'age'. A separate computation shows that the 'mean(age)' is 72.27 (months).

**Example 6. Contrasts between levels in a oneway ancova.**

\[
\begin{align*}
\text{aov.contrast(lm.results,} & \\
\text{contrast.list = list(c(1, -1, 0), c(1, 0, -1), c(0, 1, -1))} & \\
\end{align*}
\]

The difference in mean adjusted 'score' for A and B is \(-9.20 \pm 10.24\); for A and C it is \(-13.08 \pm 10.20\); for B and C it is \(-3.88 \pm 10.55\). Only the second of these differences is significant at the .05 level.
Twoway Ancova Data Set:

This data set is like the preceding one except that in this case there are two factors and two covariates. Suppose that children are taught a skill by one of three methods, A, B, or C. The gender of the children is recorded in a factor variable 'sex'. Their proficiency with the skill is measured on a test before and after training. Proficiency before training is stored in a variable 'score0' and proficiency after training is stored in a variable 'score1'. Their age in months at the beginning of training is recorded in a variable, 'age'. In this analysis, the dependent variable is 'score1', the factor variables are 'method' and 'sex', and the covariates are 'score0' and 'age'.

data.set <- data.frame(
    score0 = c(38, 54, 52, 48, 30, 44, 56, 56, 50, 54, 43, 65, 67, 53),
    score1 = c(43, 55, 53, 51, 36, 52, 61, 64, 58, 62, 55, 79, 73, 63),
    method = factor(c('A','A','A','A','A','B','B','B','B','C','C','C','C','C')),
    sex = factor(c('M','M','F','M','F','F','F','F','M','F','F','M','F','M')),
    age = c(72, 74, 68, 71, 69, 70, 73, 69, 73, 75, 70, 75, 74, 73))

Example 7.

lm.results <- lm(score1 ~ score0 + age + method*sex, data = data.set)
aov.contrast(lm.input = lm.results, margin = c(1,2),
contrast.list = list(
    c(1, 0, 0, 0, 0, 0), c(0, 1, 0, 0, 0, 0), c(0, 0, 1, 0, 0, 0),
    c(0, 0, 0, 1, 0, 0), c(0, 0, 0, 0, 1, 0), c(0, 0, 0, 0, 0, 1))))
Data: data.set
Model: score1 ~ score0 + age + method * sex
MS Error = 4.27192271900576, df.error = 6, N.valid = 14, N.missing = 0
Confidence level = 0.95

Contrasts for means of factors method * sex
Correspondence between the contrast coefficients and the levels of [1] method * sex
sex
method 1 2
 1 1 2
 2 3 4
 3 5 6

Contrast coefficients:
C1 1.000 0.000 0.000 0.000 0.000 0.000
C2 0.000 1.000 0.000 0.000 0.000 0.000
C3 0.000 0.000 1.000 0.000 0.000 0.000
C4 0.000 0.000 0.000 1.000 0.000 0.000
C5 0.000 0.000 0.000 0.000 1.000 0.000
C6 0.000 0.000 0.000 0.000 0.000 1.000

Contrast tests assume homogeneity of variance.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>51.819632</td>
<td>4.7547179</td>
<td>1.9431505</td>
<td>6 26.667843</td>
<td>0.000000</td>
</tr>
<tr>
<td>C2</td>
<td>53.215635</td>
<td>3.1183318</td>
<td>1.2743948</td>
<td>6 41.757574</td>
<td>0.000000</td>
</tr>
<tr>
<td>C3</td>
<td>57.596043</td>
<td>3.2176075</td>
<td>1.3149667</td>
<td>6 43.800383</td>
<td>0.000000</td>
</tr>
<tr>
<td>C4</td>
<td>58.908146</td>
<td>5.2076366</td>
<td>2.1282486</td>
<td>6 27.679166</td>
<td>0.000000</td>
</tr>
<tr>
<td>C5</td>
<td>60.295875</td>
<td>3.1163319</td>
<td>1.2737001</td>
<td>6 47.339146</td>
<td>0.000000</td>
</tr>
<tr>
<td>C6</td>
<td>64.564965</td>
<td>4.1423765</td>
<td>1.6928998</td>
<td>6 38.138681</td>
<td>0.000000</td>
</tr>
</tbody>
</table>

To determine the correspondence between the contrast estimates and the combinations of 'method' and 'sex', consider the table that is shown in the output:

sex
method 1 2
 1 1 2
 2 3 4
 3 5 6

Bear in mind that 'F' precedes 'M' in the alphabetical order, so 'sex' 1 is female and 'sex' 2 is male. Similarly, 'A' precedes 'B' precedes 'C' in the alphabetical order, so 'method' 1 corresponds to A, and similarly for 'method' 2 and 3. Therefore the adjusted means for the factor combinations of 'method' and 'gender' are:

<table>
<thead>
<tr>
<th>gender</th>
<th>F</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>51.819632</td>
<td>53.215635</td>
</tr>
<tr>
<td>B</td>
<td>57.596043</td>
<td>58.908146</td>
</tr>
</tbody>
</table>
Example 8. Gender effect within each method.

```r
lm.results <- lm(score1 ~ score0 + age + method*sex, data = data.set)
aov.contrast(lm.input = lm.results, margin = 2, contrast.list=c(1, -1),
              strata = 1)
```

### Data: data.set
### Model: score1 ~ score0 + age + method * sex
### MS Error = 4.27192271900576, df.error = 6, N.valid = 14, N.missing = 0
### Confidence level = 0.95

Contrasts for means of factor sex within levels of factor method
Notation: 'C1' alone refers to the 'C1' contrast averaged over the other factors.
           'C1 | i' refers to the 'C1' contrast computed within method = i

Contrast coefficients:

```
[,1]   [,2]
C1  1.000 -1.000
```

Contrast tests assume homogeneity of variance.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>-2.3257318</td>
<td>3.7037437</td>
<td>1.5136400</td>
<td>6</td>
<td>-1.53651580</td>
<td>0.175322</td>
</tr>
<tr>
<td>C1</td>
<td>-1.3960030</td>
<td>5.8550411</td>
<td>2.3928288</td>
<td>6</td>
<td>-0.58341114</td>
<td>0.580868</td>
</tr>
<tr>
<td>C1</td>
<td>-1.3121027</td>
<td>6.3892034</td>
<td>2.6111294</td>
<td>6</td>
<td>-0.50250391</td>
<td>0.633220</td>
</tr>
<tr>
<td>C1</td>
<td>-4.2690898</td>
<td>4.7255343</td>
<td>1.9312238</td>
<td>6</td>
<td>-2.21056193</td>
<td>0.069090</td>
</tr>
</tbody>
</table>

Averaged over methods, the difference between females and males was $-2.33 \pm 3.70$ with 95% confidence. Within method A, the difference was $-1.40 \pm 5.86$; within method B, it was $-1.31 \pm 6.39$; within method C, it was $-4.27 \pm 4.73$.

As noted, examples that were presented in the context of an anova can also be implemented in the context of an ancova. The only exception is the Welch-Satterthwaite method for testing contrasts without assuming homogeneity of variance. This method is only available with respect to anovas, not with respect to ancovas.

---

Other Arguments to the Function

- **Confidence.Level** specifies the confidence level of the confidence intervals. The default is 'Confidence.Level = .95' but one could change this, e.g., to 'Confidence.Level = .90'.
- **var.equal** If var.equal = TRUE, then the standard error of a contrast is calculated by the standard formula from the pooled within cell error variance (mean squared within groups). If var.equal = FALSE, then the standard error is calculated by the formula \( \sqrt{\sum (c_i^2 s_i^2 / n_i)} \), where \( c_i \) is the contrast coefficient for the i-th cell, \( s_i^2 \) is the variance of the i-th cell, and \( n_i \) is the sample size of i-th cell. Furthermore, if var.equal = FALSE, the degrees of freedom for error are computed by the Welch-
Satterthwaite formula.

show.cellcode specifies whether the table should be displayed that shows the correspondence between contrast coefficients and the combinations of factors specified in the 'margin' argument. The 'show.cellcode' argument has no effect (no table is displayed) when 'margin' specifies only one factor.

p.digits specifies the number of digits to the right of the decimal place to be shown in p-value of the test. Set 'p.digits = NA' if scientific notation is preferred.

Conclusion

This is my first attempt at writing a R function for user-defined contrasts in a between-subjects anova. A number of improvements would be desirable and still need to be written. Specifically, one would like at least the following:

- Scheffé, Tukey, and Bonferroni multiple comparison procedures.

Let me be the first to say that I am a very crude R programmer, and the code for the 'aov.contrast' function could use a lot of improvement. If anyone notices errors in the computations of this function, please describe the error in an email to jmiyamot@u.washington.edu. It would be helpful if the message included a small data set and some R code that illustrates the error. In suggesting improvements to the code for 'aov.contrast', please bear in mind that I may have difficulty understanding more sophisticated programming techniques.

Section IV. R Code

The 'aov.contrast' function calls another function, 'contrast.coefs', to compute an array of contrast coefficients given only the marginal contrast coefficients (the contrast coefficients as defined in terms of the marginal means of a factor). It also call a function, 'cm.matrix', to compute a nonsingular square matrix that transforms treatment contrasts to cell means. Both functions must be available on the search path if 'aov.contrast' is to work properly. The code for both of these functions is given below.

# The following is the definition of the 'contrast.coefs' function.

contrast.coefs <- function(margin=marg, contrast=numeric(0), levs = f.lev ) {
  if ( length(contrast) != prod(levs[margin]) )
    stop(paste( "WARNING: Number of contrast coeffs does not match",
                "number of levels of factor." ) )

  # permute.index is used to compute permutations and inverse permutations
  # of array indices that can be used to temporarily convert an array into
  # a matrix. This is useful when computing the contrast coefficient array.
  permute.index <- function(n.factors = n.fac, last = marg) {
    o1 <- 1:n.factors
    n.marg <- length(last)
    o2 <- rbind( cbind(o1[-last], order(o1[-last])),
                cbind( o1[last], (n.factors+1) - (n.marg:1))
    return(o2)
  }
}
o2 <- matrix(o2, ncol=2)
dimnames(o2) <- list(o1, c("initial", "permuted"))
matrix(o2[order(o2[,1]), ], ncol=2)
}  #end of permute.index definition

p.index <- permute.index(n.fac = length(levs), last = rev(margin) )
input.ord <- (1:length(levs))[order(p.index[,2] )]
output.ord <- p.index[, 2]
con.levs <- prod(levs[margin])
avg.over <- prod(levs[-margin])

arr.1 <- array(1, dim=levs) / avg.over
mat.1 <- matrix(aperm(arr.1, input.ord), ncol = con.levs)
mat.2 <- apply( rbind(contrast, mat.1), 2, function(x) {x[1]*x[-1]})
arr.2 <- array(mat.2, levs[input.ord])
arr.3 <- aperm(arr.2, output.ord)
arr.3

### Documentation below this line ###########################

# 'margin' is the margin over which the input coefficients are defined.
# 'contrast' is a vector of contrast coefficients of the same length as the
# number of levels of 'margin'.
# 'levs' is a vector that specifies the number of levels of each factor in the design.
# The output is an array of contrast coefficients whose dimensions are determined
# by 'levs'.
}  #end of contrast.coefs definition

#-----------------------------------------------------------
# The following is the definition of the 'cm.matrix' function.

cm.matrix <- function(lm.input, Output = c("sparse", "full")[1]) {
  # The 'cm.matrix' function assumes that the input from the 'lm' call
  # 'computed so-called 'treatment contrasts'. This assumption was violated
  # by the 'lm.input' argument of the current call to 'cm.matrix'. The
  # contrasts for 'lm.input' are:

  if (any(lm.input$contrast != "contr.treatment")) {
    tmp <- paste(paste(names(lm.input$contrast), "", lm.input$contrast, sep=""), collapse="\n")
    stop("The 'cm.matrix' function assumes that the input from the 'lm' call"
         "computed so-called 'treatment contrasts'. This assumption was violated"
         "by the 'lm.input' argument of the current call to 'cm.matrix'. The"
         "contrasts for 'lm.input' are:

      if (!(Output %in% c("sparse", "full"))) stop("'n'Output' argument must be set to 'sparse' or 'full'.")

      # Set some variables that are useful later:
      ### 'VN' = dataframe with variable names from 'lm' call + name of the data set;
      ### 'n.coef' = the number of estimated coefficients;
      ### 'Fac.names' = character vector of names for the factor variables;
      ### 'n.Facs' = number of factors in the 'lm' call;
      ### 'levs' = list of factor levels (1 component for each factor);
      ### 'L.levs' = a numeric vector; each element is the number of levels of a factor;
      ### 'q.names' = character vector of names for the quantitative variables;
      ### 'exist.qvars' is boolean variable, TRUE if there are quantitative (non-factor)
### variables, FALSE otherwise.

```r
VN <- v.names(lm.input)
n.coef <- length(lm.input$coef)
Fac.names <- VN[VN[,"Type"] == 'factor', 1]
n.Facs <- length(Fac.names)
levs <- lapply(Get.Vars(vn.out = VN, v.type = 'factor'), levels)
f.levs <- sapply(levs, length)
q.names <- VN[VN[,"Type"] == 'quant', 1]
exist.qvars <- length(q.names) > 0

# The 'trt.coef.names' function takes a dataframe of factors ('Facs') as input and
# creates a grid of factor levels, sorted by factors from left to right. Each row of
# the output includes the numeric codes for a combination of factor levels,
# the alphanumeric names for the factor levels, and the alphanumeric name that
# R gives to the corresponding regression coefficient. Note that 'trt.coef.names'
# does not include any rows for the covariates (quantitative variables) in the 'lm'
# call. These rows are added after running the 'trt.coef.names' function.

trt.coef.names <- function(Facs) {
  Fac.names <- names(Facs)
n.Facs <- length(names(Facs))
levs <- lapply(Facs, levels)
f.levs <- sapply(levs, length)
LL <- NULL
for (i in 1:n.Facs)
  LL <- c(LL, list(1:f.levs[[i]]))
names(LL) <- paste(names(Facs), '.n', sep = '')
data1 <- data.frame(expand.grid(LL), expand.grid(levs))
coef.name <- rep(NA, length(data1[,1]))
for (i in 1:length(data1[,1])) {
  n.curr <- ""
  for (k in 1:n.Facs) {
    if (n.curr != "" & data1[i,k] > 1)
      n.curr <- paste(n.curr, ':', Fac.names[k],
        (levs[[k]])[data1[i,k]], sep = '')
    if (n.curr == "" & data1[i,k] > 1)
      n.curr <- paste(Fac.names[k],
        (levs[[k]])[data1[i,k]], sep = '')
  }
  # end 'for (k in 1:n.Facs)'
  if (n.curr == "") n.curr <- '(Intercept)'
  coef.name[i] <- n.curr
}
# end 'for (i in 1:length(data1[,1])'
coef.temp1 <- data.frame(cf.ord.ini=1:n.coef,
  coef.name=names(lm.input$coef), coef=lm.input$coef)
```
dimnames(coef.temp1) <- list(1:n.coef, names(coef.temp1))

if (exist.qvars) {
  tm.mns <- sapply(Get.Vars(VN, v.type = "quant"), mean)
  coef.temp2 <- merge(x = coef.temp1,
    y = data.frame(coef.name = names(tm.mns), cov.means = tm.mns),
    all.x = TRUE)
  rm(coef.temp1)
} else { # end 'if (exists.qvars)'
  coef.temp2 <- coef.temp1
  rm(coef.temp1)
} # end 'else'

# 'coef.fm' is a dataframe (look at it). Note that the variable ‘cf.ord.ini’ orders the rows
# in the same order as 'lm.input$coef'. The variable 'indx.ord.ini' orders the rows such
# that the factor levels are sorted from left to right (see columns of factor variables).
# The rows for the covariates are last in the 'indx.ord.ini' ordering. Having the rows
# sorted by 'cf.ord.ini' is useful when computing the matrix that transforms 'lm.input$coef'
# to a vector of cell means. The 'indx.ord.ini' ordering is used to determine how to reorder
# the vector of cell means in an order that conforms to the sorted order of the factors.
coef.fm <- merge(x=t2.n, y=coef.temp2, all.y = TRUE)
coef.fm <- coef.fm[order(coef.fm$cf.ord.ini),]
if (exist.qvars)
    coef.fm[is.na(coef.fm$indx.ord.ini),]$indx.ord.ini <-
    (max(nm(coef.fm$indx.ord.ini)) + 1):n.coef

# 'trt.to.cm' is a square matrix that transforms the vector of coefficients for treatment contrasts
# to a vector of cell means. I.e., 'lm.input$coef %*% trt.to.cm' equals a row vector
# of cell means.
trt.to.cm <- matrix(0, nrow=n.coef, ncol=n.coef)

for (i in 1:n.coef) for (k in 1:n.coef) {
  if (coef.fm$coef.name[i] %in% q.names & k == i)
    trt.to.cm[k,i] <- 1
  if (!(coef.fm$coef.name[i] %in% q.names)) {
    if (coef.fm$coef.name[k] %in% q.names)
      trt.to.cm[k, i] <- coef.fm$cov.means[k]
    if (!(coef.fm$coef.name[k] %in% q.names))
      if (k <= i &
          (all( coef.fm[k, 2:(n.Facs+1)] == 1 |
            coef.fm[k, 2:(n.Facs+1)] == coef.fm[i, 2:(n.Facs+1)])))
        trt.to.cm[k, i] <- 1
  } #end 'if (!(coef.fm$coef.name[i] %in% q.names))'
} #end 'for (i in 1:n.coef) for (k in 1:i)'

sort.by.effect <- diag(n.coef)
sort.by.effect <- sort.by.effect[, coef.fm$cf.ord.ini[order(coef.fm$indx.ord.ini)] ]
trt.to.sorted.cm <- trt.to.cm %*% sort.by.effect
  # This is the transforming matrix

if (n.Facs > 1) { effect.names <- apply(apply(coef.fm[
    order(coef.fm$indx.ord.ini), 2:(n.Facs + 1)],
    2, paste, sep=":"), 1, paste, collapse=":")
} else
effect.names <- as.character(coef.fm[order(coef.fm$indx.ord.ini), 2])

if (exist.qvars)
effect.names[(n.coef - length(q.names)+1):n.coef] <- q.names
dimnames(trt.to.sorted.cm) <- list( names(lm.input$coef),
  effect.names)

factor.frame <- coef.fm[order(coef.fm$indx.ord.ini), 2:(2*n.Facs + 1)]
coef.type <- rep("(adj) cell mean", length(factor.frame[,1]))
if (exist.qvars)
  coef.type[(n.coef-length(q.names)+1):n.coef] <-
  paste(q.names, "coef")
factor.frame <- data.frame(factor.frame, coef.type = coef.type)

if (Output == "sparse") output <- trt.to.sorted.cm
if (Output == "full") output <-
  list(matrix = trt.to.sorted.cm, anova.cells = factor.frame)
output

### Documentation below this line ###################
# 'cm.matrix' creates a matrix that transforms contrast estimates for anova factors
# to estimates of cell means for the corresponding factorial design. The 'lm.input'
# argument is created by a call to 'lm' or 'aov'. If 'lm.input' is the 'lm' output that
# serves as input to 'cm.matrix', then 'lm.input$coef %*% cm.matrix(lm.input)'
# is a vector of adjusted cell means for the anova design (plus the regression
# coefficients for the covariates)
# If 'Output = "sparse"', then the only output is the transforming matrix.
# If 'Output = "full"', then the output is a list. Component 1 is the transforming
# matrix and Component 2 is a dataframe that shows the relationship between the
# adjusted cell means and the factor levels.
# If 'Output = "full"', then 'lm.input$coef %*% cm.matrix(lm.input)[1]' is
# the vector of adjusted cell means (plus the regression coefficients for the
# covariates) and 'cm.matrix(lm.input)[2]' is the dataframe that shows the
# correspondence between 'lm.input$coef %*% cm.matrix(lm.input)[1]' and
# the factor levels.
} #end 'cm.matrix' definition

# The following is the definition of the 'aov.contrast' function.
aov.contrast <- function(lm.input, margin=1, contrast.list, strata=NA,
  Confidence.Level = .95, var.equal=TRUE, show.cellcode=TRUE, p.digits=6)
{
  # The code in this paragraph sets the values of a number of parameters that are
  # determined by 'lm.input'.
VN <- v.names(lm.input) # 'VN' = dataframe of information regarding var names;
name.in <- VN$Names # 'name.in' = vector of data & variable names
n.coef <- length(lm.input$coef) # 'n.coef' = number of regression coeffs
Y <- as.vector(Get.Vars(VN, "dv")[[1]]) # 'Y' = dv
n.total <- length(Y) # n.total = sample size including cases with missing values
df.error <- df.residual(lm.input) # df error

msw <- NA; vcov.mns <- NA; var.arr <- NA # The variables will be given values later. They are set to 'NA' here to avoid confusion with other
# variables that have the same name in the calling environment.

F.vars <- Get.Vars(VN, "factor") # 'F.vars' = dataframe of anova factors
F.names <- VN[VN[, "Type"] == 'factor', 1] # 'F.names' = names of factor vars
n.Facs <- length(F.names) # 'n.Facs' = number of factor vars
levs <- lapply(Get.Vars(vn.out = VN, v.type = 'factor'), levels) # 'levs' = list. Each component is a vector of level names for a factor. There are 'n.Facs' components.
f.levs <- sapply(levs, length) # 'f.levs' is a vector showing the number of levels for each factor.

q.names <- VN[VN[, "Type"] == 'quant', 1] # Vector of names of quant vars
exist.qvars <- length(q.names) > 0 # 'exist.qvars' = TRUE iff there were quantitative variables in the 'lm' call.
if (exist.qvars) Q.vars <- Get.Vars(VN, "quant")

# The following code creates a variable 'data.description' that describes where is the data for this analysis.
if (VN[1,1] == "All vars on search path")
data.description <- paste("All variables were on the search path.")
if (!VN[1,1] == "All vars on search path") & all(nm(VN[, "On.Dataframe"]))
data.description <- VN[1,1]
if (!VN[1,1] == "All vars on search path") & !all(nm(VN[, "On.Dataframe"]))
data.description <- paste("Some variables were in ",
VN[1,1], ". Other variables were on the search path.", sep="")

# Following code checks for input errors and defines variable names.
if ("subset" %in% names(lm.input$call) ) stop(paste("The 'lm' or 'aov' call that produced the input to this call to\n",
"'aov.c2' made use of a 'subset' clause. The 'aov.c2' function was not \n",
"designed to take 'subset' clauses into account in the computation of\n",
"contrast statistics.'\n",
"Recommendation: It may be possible to produce the desired analysis by\n",
"using the 'strata' argument of the aov.c2 function. If this is not possible, then\n",
"create a new data set that contains only the desired subset of data, redo the\n",
"lm or aov with respect to this new data set, then apply aov.c2 to the\n",
"output of this revised call to lm or aov.\n\n", sep="") ) # end of 'stop' and 'paste' clauses
# Following code checks that the 'lm' contrasts set to 'contr.treatment'.
if (!all(lm.input$contrasts == 'contr.treatment')) stop(paste("\nThe 'lm' or 'aov' call that produced the input to this call to\n",
"'aov.c2' did not compute 'treatment' contrasts. The 'aov.c2' doesn't\n",
"function assumes that the contrasts for the 'lm' or 'aov' call\n",
"are 'treatment' contrasts. Check the 'contrasts' argument of\n",
"the call to 'lm' or 'aov.\n\n", sep="") ) # end of 'stop' and 'paste' clauses

# If the 'contrast.list' argument is a vector, the next code changes it to a list.
if (!is.list(contrast.list)) contrast.list <- list(contrast.list)
# If any components of 'contrast.list' lack names, the next code gives the component a name of the form "Ck".
if (is.null(names(contrast.list)))
  names(contrast.list) <-
paste("C", 1:length(contrast.list), sep="") else
for (i in 1:length(contrast.list))
  if (nchar(names(contrast.list)[i]) == 0)
    names(contrast.list)[i] <- paste("C", i, sep="")

# The following are checks that the contrast.list is well defined.
if (mode(contrast.list) != "list")
  stop(paste("\n The 'contrast.list' argument must be a list object.\n",
    "The current type of 'contrast.list' is a", mode(contrast.list),
    "object.\n",
    "Try writing: contrast.list <- list(c(...)) \n\n")
if (!all(prod(f.levs[margin]) == sapply(contrast.list, length)) ) {
  marg.desc <- names(F.vars)[margin[1]]
  if (length(margin) > 1)
    for (i in 2:length(margin))
      marg.desc <- paste(marg.desc, " * ", names(F.vars)[margin[i]])
  stop(paste("Number of coefficients in contrast does not match ",
    "number of levels in f.levs[margin] \n\n")
  } # end of if
} # end of if (!all(prod(f.levs[margin]) == sapply(contrast.list, length))
if (!var.equal & exist.qvars) stop(
  "\nThis function requires the homogeneity of variance assumption in the",
  "computation of contrast statistics for analysis of covariance. The",
  "Welch/Satterthwaite approximation for degrees of freedom in an",
  "inhomogeneous case is not available with respect to an analysis of",
  "covariance."
)

# Need to insert code that deals with the case where some of the levels
# or combinations of levels are empty. Basically, check for empty cells,
# decide whether they are "serious" (definition?), and 'stop' the computation
# if necessary.
# It may be useful to expand all contrasts on 'contrast.list' to contrasts
# that are defined over every cell in the factorial anova. (You have to do
# this anyway farther down in this function. Once you do this, you can check
# whether any cells with nonzero coefficients in the expanded contrasts are
# associated with empty cells in the anova. If so, the contrast cannot be
# computed. Previous work (see 'e:\sa\exp1\FunctionDefs.doc') defined computations
# that allowed one to revise the contrast to take into account the fact that
# some cells were missing.
# ________________________________________________________________

# Restrict the data set to nonmissing cases (assumes that na.action = na.omit)
if (exist.qvars) {
  valid.case <- apply(!is.na(data.frame(Y, F.vars, Q.vars)), 1, all )
  Q.vars <- data.frame(Q.vars[valid.case,])
  names(Q.vars) <- q.names
} else  #end 'if (exist.qvars')
  valid.case <- apply(!is.na(data.frame(Y, F.vars)), 1, all )
Y <- Y[valid.case]
F.vars <- data.frame(F.vars[valid.case,])
names(F.vars) <- F.names

# Compute (adjusted) cell means and error estimates for the estimated means.
# The reason for doing this computation here rather than within the 'cn' function (the
# function that computes the contrast statistics) is that these numbers are needed in
# the computation of every contrast and standard errors, so it doesn't make sense
# to repeat these computations for each contrast.
# There are different approaches to computing these estimates depending on whether
# we are computing an anova, an ancova or an anova without the homogeneity of
# variance assumption.

n.arr <- tapply(!is.na(Y), INDEX = F.vars, sum)  # n.arr = array of cell sample sizes
m.arr <- tapply(Y, INDEX = F.vars, mean)  # 'm.arr' = array of cell means. Note that 'm.arr' gets changed in the next 'if' if there are covariates in the analysis.

if (var.equal) {
    msw <- deviance(lm.input)/df.error  # mean squared error
    if (exist.qvars) {
        trt.to.means <- cm.matrix(lm.input)
        m.arr <- array((lm.input$coef %*% trt.to.means)[1:(n.coef - length(q.names))],
                        dim=f.levs)
        # m.arr = array of adjusted cell means. We drop the coefficients for the covariates
        vcov.mns <- (t(trt.to.means) %*% (vcov(lm.input) %*% trt.to.means))[1:(n.coef - length(q.names)),
                         1:(n.coef - length(q.names))]
    }  #end 'if (exist.qvars)'
} #end 'if (var.equal)'

if (!var.equal) var.arr <- tapply(Y, INDEX = F.vars, var)
# var.arr is the array of cell mean squares

# The 'cn' function computes the statistics for an individual contrast.
# It is called by the 'sapply' function below to compute statistics for each
# contrast in 'contrast.list'.
cn <- function(x, margin.cn=margin) {
        cntr.arr <- contrast.coefs(margin = margin.cn, contrast = x, levs = f.levs)
        cntr.arr.vec <- as.vector(cntr.arr)
        est <- sum( cntr.arr * m.arr)
        alpha <- 1 - Confidence.Level

        if (var.equal) {
            # compute results assuming variances are equal.
            if (exist.qvars)
                var.est <- t(cntr.arr.vec) %*% vcov.mns %*% cntr.arr.vec else
                var.est <- msw*sum((cntr.arr*cntr.arr)/n.arr)
            s.e <- sqrt(var.est)
            bnds <- qt(1 - (alpha/2), df.error)*s.e
            t.stat <- est / s.e
            pval <- 2*(1 - pt(abs(t.stat), df.error) )
            output <- c(est, bnds, s.e, df.error, t.stat, pval)
            names(output) <- c("Estimate","Conf.Bounds","Std.Error", "Df",
                              "T.Stat","Signif")
        } #end 'if (var.equal)'
        if (!var.equal) {
            a.arr <- (cntr.arr*cntr.arr)/n.arr  # See 'e:\quant\Welch's Approx.doc'
            # for the explanation of the 'a' array.
            s.e <- sqrt(sum(a.arr*var.arr))
            df.adj <- (s.e^4) / sum(((a.arr*var.arr)^2)/(n.arr - 1))
            t.stat <- est / s.e
            bnds <- qt(1 - (alpha/2), df.adj)*s.e
        }
}
pval <- 2*(1 - pt(abs(t.stat), df.adj) )
output <- c(est, bnds, s.e, df.adj, t.stat, pval)
) #end of 'else' for case where variances not assumed to be equal
output ) #end of 'cn' definition
contrast.factors <- as.vector(paste(F.names[margin], collapse=" * "))
dim.margin <- NULL
for (i in 1:length(margin)) dim.margin <- c(dim.margin, list(1:f.levs[margin[i]]))
names(dim.margin) <- names(F.vars)[margin]
cell.nums <- aperm(array(1:prod(f.levs[margin]), dim=rev(f.levs[margin]),
           dimnames=rev(dim.margin) ), length(margin):1)

# Produce output that describes the results of the initial lm or aov.
cat(paste('Data:  ', data.description,
    '
    Model: ', lm.input$call[2],
    '
    MS Error = ', msw, ', df.error = ', df.error,
    ', N.valid = ', n.valid, ', N.missing = ', n.total - length(Y),
    '
    Confidence level = ', Confidence.Level,
    '
    Contrasts for means of factor',
    if (length(margin) > 1) 's  ' else '  ',
    contrast.factors, sep="" ) )

# Prepare output that describes the stratifying factors ('strata' argument).
if (!is.na(any(strata))) {
    strata.factors <- paste(names(F.vars)[strata], collapse=" * ")
cat(paste(" within levels of factor",
          if (length(strata) > 1) "s  " else "  ",
          strata.factors, "\n", sep="")
    n.str <- prod(f.levs[strata])
    str.index <- matrix(NA, nrow=n.str, ncol=length(strata) )
    for (i in 1:length(strata)) {
        if (i == 1) tm.ini <- 1 else tm.ini <- prod(f.levs[ strata[1:(i-1)] ])
        if (i == length(strata))
            tm.fnl <- 1 else tm.fnl <-
            prod(f.levs[strata[(i+1):length(strata)]])
        vec.inner <- NULL
        for (j in 1:f.levs[strata[i]])
            vec.inner <- c(vec.inner, rep(j, tm.fnl))
        str.index[ ,i] <- rep(vec.inner, tm.ini)
    } #end of for (i in 1:length(strata))
    str.suffix <- apply(str.index, 1,
        function(x) { paste(x, collapse="*") } )
    str.desc <- NULL
    index.char <- c("i","j","k","m","n","p","q",

for (i in 1:length(strata)) {
  if (i == 1) { str.desc <-
    paste(names(F.vars)[strata[1]], "=", index.char[1], sep=" ")
  sample.suffix <- index.char[1] } else {
    str.desc <- paste(str.desc, paste(names(F.vars)[strata[i]], ",", index.char[i %% 16], sep=" ", sep=" & ")
    sample.suffix <- paste(sample.suffix, ",", index.char[i], sep="")
  } #end of if ... else ...
} #end of for (i in 1:length(strata))

if (!is.na(any(strata))) {
  n.clevs <- prod(f.levs[margin])
  stratified.clist <- NULL
  for (i in 1:length(contrast.list)) {
    temp.clist <- NULL
    # The next for loop creates a matrix of indices for the levels of the stratum variables.
    for (j in 1:n.str) {
      tm.con <- list(c(rep(0, (j-1)*n.clevs), contrast.list[[i]],
        "r", "s", "t", "u", "v", "w", "x", "y", "z")
      }
    }
  }

  # The next command computes the unstratified contrast statistics.
  output.nostrata <- t(sapply(contrast.list, cn))
}

# Create output for case where stratifying variables are specified
if (!is.na(any(strata))) {
  n.clevs <- prod(f.levs[margin])
  stratified.clist <- NULL
  for (i in 1:length(contrast.list)) {
    temp.clist <- NULL
    # The next for loop creates a matrix of indices for the levels of the stratum variables.
    for (j in 1:n.str) {
      tm.con <- list(c(rep(0, (j-1)*n.clevs), contrast.list[[i]],
        "r", "s", "t", "u", "v", "w", "x", "y", "z")
      }
    }
  }

  # The next command computes the unstratified contrast statistics.
  output.nostrata <- t(sapply(contrast.list, cn))
rep(0, (n.str - j)*n.clevs))
names(tm.con) <- paste(names(contrast.list)[i], ".", j, sep=""

temp.clist <- c(temp.clist, tm.con)
} # end of for (j in 1:n.str)
stratified.clist <- c(stratified.clist, temp.clist)
} # end of 'for (i in 1:length(contrast.list))'

# Compute contrasts within each stratum.
output.temp <- t(sapply(stratified.clist, function(x) {
  cn(x, margin = c(strata, margin))
}))
output.strata <- NULL
for (i in 1:length(contrast.list)) {
  output.strata <- rbind(output.strata, output.nostrata[i,])
  output.strata <- rbind(output.strata,
    output.temp[(i-1)*n.str + 1:n.str,])
} # end of 'for (i in 1:length(contrast.list))'

rnames <- NULL
for (i in 1:length(contrast.list)) {
  rnames <- c(rnames, paste(names(contrast.list)[i], " "))
  for (j in 1:n.str)
    rnames <- c(rnames, paste(names(contrast.list)[i], "|",
      str.suffix[j], " ")
  } # end of 'for (i in 1:length(contrast.list))'

dimnames(output.strata) <- list(rnames, dimnames(output.nostrata)[[2]])
} # end of if (!is.na(any(strata)))

if (is.na(any(strata))) output <- output.nostrata else
output <- output.strata
if (is.na(p.digits)) output.fm <- output else { # begin reformatting p-values
  output.fm <- data.frame(matrix(output[, 1:5], ncol=5),
    pval=formatC(output[, "Signif"], format="f", digits = p.digits))
  colnames(output.fm) <- colnames(output)
} # end of 'else' for reformatting p-values
if (var.equal)
cat("Contrast tests assume homogeneity of variance.\n") else {
cat("These tests do not assume homogeneity of variance. ")
cat("Welch-Satterthwaite procedure used.\n")
}

output.fm
} ### end of aov.contrast definition ###