MacAnova Version 4.07

This file consists of Chapter 10 of **MacAnova User's Guide** by Gary W. Oehlert and Christopher Bingham, issued as Technical Report Number 617, School of Statistics, University of Minnesota, revised August 1998, describing Version 4.07 of MacAnova.

Fonts used in this manual are Palatino, Courier, and Symbol.

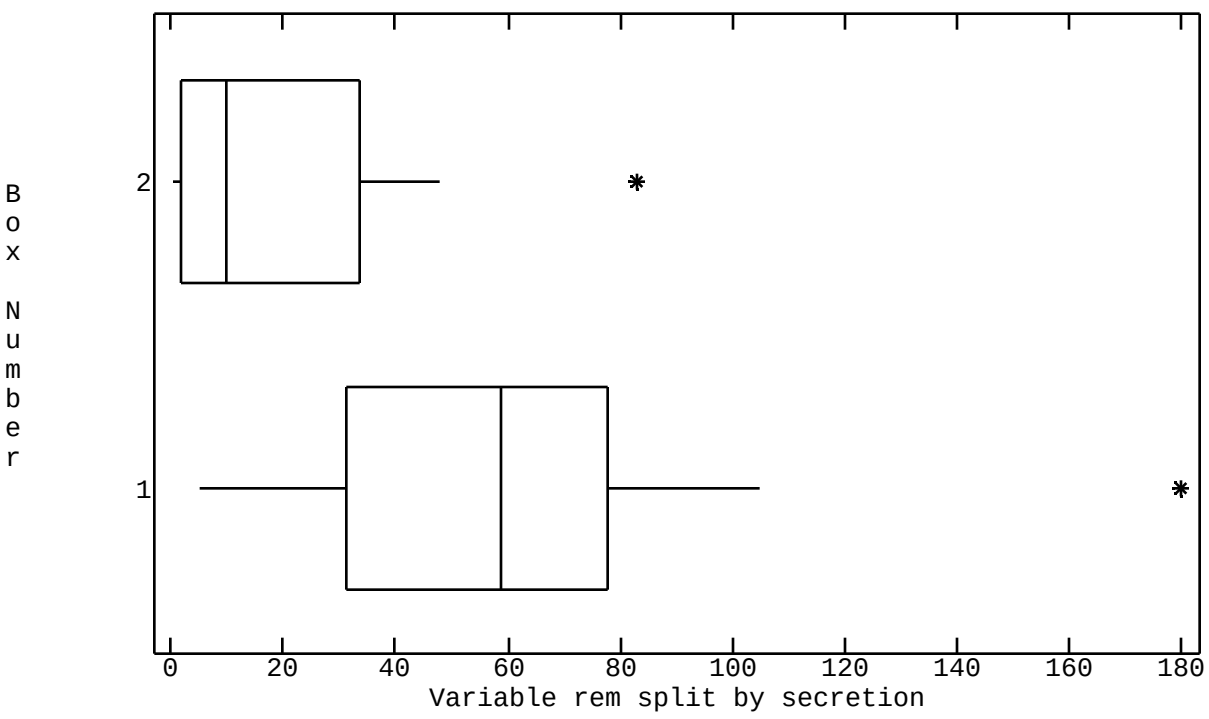
For information concerning MacAnova, write University of Minnesota, Department of Applied Statistics, 352 Classroom Office Building, 1994 Buford Avenue, St. Paul, MN 55108-6042.

10. Examples

10.1 **Introduction** This section illustrates a number of MacAnova commands, primarily related to linear models, and illustrates the specification of models for different situations. It is *not* a complete inventory of models and the examples should *not* be construed as complete or recommended analyses. Many procedures (residual plots, for example) should be used in nearly every example, but are only used once or twice in this section to save space. The sample data sets are taken from Montgomery (1984) or Devore and Peck (1993).

10.2 **Simple descriptive statistics** This is Example 19, page 95 of Devore and Peck (1993). The `rem` variable is the REM period latency and the levels of factor `secretion` indicate normal (1) or hyper (2) cortisol secretion.

```
Cmd> rem <- vector(.5,1,2.4,5,15,19,48,83)

Cmd> rem <- vector(rem,5,5.5,6.7,13.5,31,40,47,47,59,62,68,72,\
78,84,89,105,180)

Cmd> secretion <- factor(vector(rep(2,8),rep(1,17)))

Cmd> boxplot(split(rem,secretion),\
xlab:"Variable rem split by secretion",\
title:"Data from Example 3.19, page 100 of Devore & Peck, Ed. 2")
```

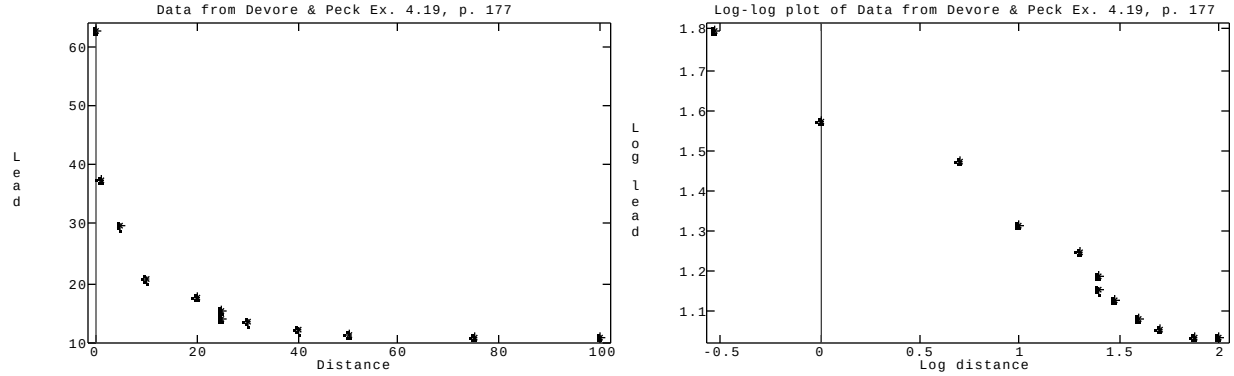


```
Cmd> stemleaf(rem[secretion==1], title:\
"Data from Example 3.19 Devore & Peck 2nd Ed., normal secretion")
Data from Example 3.19 Devore & Peck 2nd Ed., normal secretion
    3    +0|556
    4     1|3
    4     2|
    5     3|1
    8     4|077
  ( 1)    5|9
    8     6|28
    6     7|28
    4     8|49
    2     9|
    2    10|5
 High  180
          1|1 represents 11  Leaf digit unit = 1

Cmd> stemleaf(rem[secretion==2],title:\
"Data from Example 3.19 Devore & Peck 2nd Ed., hyper secretion")
Data from Example 3.19 Devore & Peck 2nd Ed., hyper secretion
    4    +0|0125
    4     1|59
    2     2|
    2     3|
    2     4|8
 High  83
          1|1 represents 11  Leaf digit unit = 1

Cmd> describe(rem[secretion==1]) # stats for hyper secretion
component: n
(1)           17
component: min
(1)            5
component: q1
(1)           31
component: median
(1)           59
component: q3
(1)           78
component: max
(1)          180
component: mean
(1)       58.394
component: var
(1)         1932

Cmd> tt <- t2val(rem[secretion==1],rem[secretion==2])#2 sample t

Cmd> vector(tt,1 - cumstu(tt,8+17-2),2*(1 - cumstu(abs(tt),8+17-2)))
(1)       2.1333     0.021897     0.043793 t-stat, 1,2 tail Pvals
```

Note: You cannot use t instead of `tt` since `t()` is a MacAnova function.

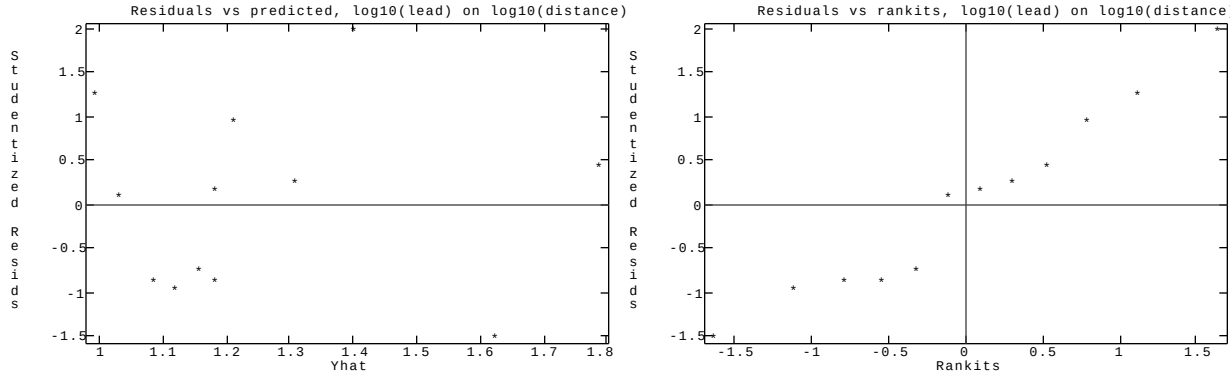
10.3 **Simple regression** This is example 4.19 from page 177 of Devore and Peck (1993). Variable x is the distance in meters from a highway and y is the lead content of the soil in ppm.

```
Cmd> x <- vector(.3,1,5,10,20,25,25,30,40,50,75,100) # distance

Cmd> y <- vector(62.75,37.51,29.70,20.71,17.65,15.41,14.15,\
13.50,12.11,11.40, 10.85,10.85) # lead content

Cmd> plot(Distance:x,Lead:y,title:\
"Data from Devore & Peck Ex. 18, p. 153") # It's pretty curved

Cmd> plot(log10(x),log10(y),\
xlab:"Log distance",ylab:"Log lead",\
title:"Log-log plot of Data from Devore & Peck Ex. 4.19, p. 177")
```




```
Cmd> # log vs log plot is much straighter; regress log(y) on log(x)

Cmd> logx <- log10(x); logy <- log10(y)

Cmd> regress("logy=logx") # Do simple linear regression of logs
Model used is logy=logx
                Coef       StdErr            t
CONSTANT       1.6225     0.020599       78.767 Intercept
logx         -0.31472     0.015039      -20.928 Slope

N: 12,  MSE: 0.0014366, DF: 10,  R^2: 0.97768
Regression F(1,10): 437.97, Durbin-Watson: 1.8778
To see the ANOVA table type 'anova()'

Cmd> anova() # gives ANOVA table for preceding regression
Model used is logy=logx
WARNING: summaries are sequential
               DF           SS           MS
CONSTANT        1       18.969       18.969
logx            1      0.62921      0.62921
ERROR1         10     0.014366    0.0014366
```

```
Cmd> #Now use macros resvshyat & resvsrankits to look at residuals

Cmd> resvsyhat(title:\
"Residuals vs predicted, log10(lead) on log10(distance)")
Cmd> resvsrankits(title:\
"Residuals vs rankits, log10(lead) on log10(distance)")
```



`plot(logy-RESIDUALS,RESIDUALS)` and `plot(rankits(RESIDUALS),RESIDUALS)` would give almost the same plots except that `resvsyhat` and `resvsrankits` use studentized residuals

By default, `regress()` and `anova()` do not print *P* values and/or *F*-statistics. You can, of course, compute them "by hand".

```
Cmd> tstats <- vector(coefs()/secoefs(coefs:F))

Cmd> twotailt(tstats,DF[3]) # Sec. 2.
(1)   2.6645e-15    1.377e-09

Cmd> fstat <- (SS[2]/DF[2])/(SS[3]/DF[3])

Cmd> vector(fstat,1 - cumF(fstat,DF[2],DF[3]))
(1)       437.97    1.377e-09  F-statistic and P-value
```

When you use keyword phrases `pvals:T` and `fstats:T` as arguments to `anova()` and `regress()` (and other GLM commands) they will compute and print *P* values, and, where appropriate, *F*-statistics. The following illustrates this and also the transforming of data "on the fly" using `{...}` (see Sec. 3.4.1).

```
Cmd> regress("{log10(y)}={log10(x)}",pvals:T)
Model used is {log10(y)}={log10(x)}
                 Coef       StdErr            t      P-Value
CONSTANT       1.6225     0.020599       78.767   2.6645e-15
log10(x)    -0.31472     0.015039      -20.928    1.377e-09

N: 12,  MSE: 0.0014366, DF: 10,  R^2: 0.97768
Regression F(1,10): 437.97, P-value: 1.377e-09, Durbin-Watson: 1.8778
To see the ANOVA table type 'anova()'

Cmd> anova(,fstat:T) # to suppress P values, use pvals:F
Model used is {log10(y)}={log10(x)}
WARNING: summaries are sequential
                DF          SS          MS             F      P-value
CONSTANT         1      18.969      18.969  13203.46834            0
{log10(x)}       1     0.62921     0.62921    437.96893    1.377e-09
ERROR1          10    0.014366   0.0014366
```

```
Cmd> setoptions(pvals:T,fstats:T)
```

This changes the default behavior of `regress()`, `anova()`, and the other GLM commands, so that *P* values and *F*-statistics will always be printed (see Sec. 8.1.3).

10.4 **Multiple linear regression** This data is Example 14.6, page 675 of Devore and Peck (1993). The two X variables are extractable iron and extractable aluminum; the Y variable is a phosphate adsorption index.

```
Cmd> iron <- vector(61,175,111,124,130,173,169,\
169,160,244,257,333,199)

Cmd> aluminum <- vector(13,21,24,23,64,38,33,61,39,71,112,88,54)

Cmd> adsorption <- vector(4,18,14,18,26,26,21,30,28,36,65,62,40)

Cmd> regress("adsorption = iron + aluminum")
Model used is adsorption = iron + aluminum
                Coef       StdErr            t
CONSTANT      -7.3507       3.4847      -2.1094
iron          0.11273     0.029691       3.7969
aluminum        0.349     0.071306       4.8944

N: 13,  MSE: 19.179, DF: 10,  R^2: 0.94847
Regression F(2,10): 92.026, Durbin-Watson: 2.6339
To see the ANOVA table type 'anova()'

Cmd> anova() # display ANOVA table, too
Model used is adsorption = iron + aluminum
WARNING: summaries are sequential
               DF           SS           MS
CONSTANT        1        11580        11580
iron            1       3070.5       3070.5
aluminum        1       459.43       459.43
ERROR1         10       191.79       19.179

Cmd> print(DF,SS,ms:SS/DF)#print side effect variables; see Sec. 3.6
DF:
(1)            1            1            1           10
SS:
(1)        11580       3070.5       459.43       191.79
ms:
(1)        11580       3070.5       459.43       19.179

Cmd> COEF # side effect variable containing coefficients
     CONSTANT         iron     aluminum
      -7.3507      0.11273        0.349

Cmd> regcoefs() # coefficients in readable format; see Sec. 3.13.1
                Coef       StdErr            t
CONSTANT      -7.3507       3.4847      -2.1094
iron          0.11273     0.029691       3.7969
aluminum        0.349     0.071306       4.8944

Cmd> dfe <- DF[4] ; mse <- SS[4]/dfe;mse # compute error mean square
(1)       19.179
```

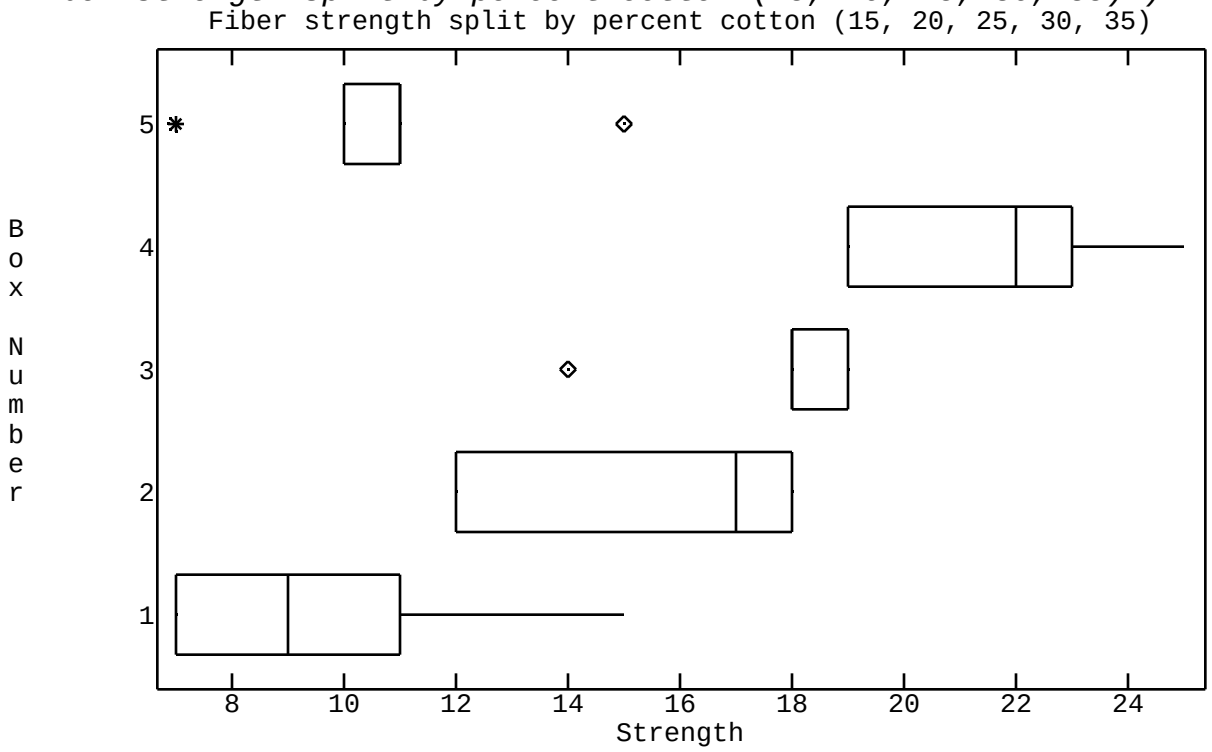
```
Cmd> # Predict absorption when Iron==200,Al==60; see Sec. 3..18

Cmd> regpred(vector(200,60))
component: estimate
(1)       36.136
component: SEest
(1)       1.3017
component: SEpred
(1)       4.5687

Cmd> HII # Look at leverages; see Sec. 3.6
 (1)      0.30754      0.27815      0.15449      0.14506      0.35893
 (6)      0.10313      0.12684      0.13338      0.08757      0.15238
(11)      0.53456      0.53144     0.086537

Cmd> studres <- RESIDUALS/sqrt(mse*(1 - HII))

Cmd> studres # internally studentized residuals
 (1)    -0.017302     -0.45867      0.11455      0.82601      -1.0383
 (6)      0.14126     -0.54206     -0.73346      0.88505      -2.2161
(11)       1.4359      0.36646       1.4504

Cmd> # Now compute externally studentized residuals = t-statistics

Cmd> studres*(sqrt((dfe-1)/(dfe-studres*studres)))
 (1)    -0.016414     -0.43978      0.10875      0.81181      -1.0428
 (6)      0.13414     -0.52196     -0.71533      0.87459      -2.9471
(11)        1.529      0.35002       1.5484

Cmd> studres*studres*HII/(1-HII)/3 # Cook's distance;3 = DF in model
 (1)   4.4318e-05     0.027022   0.00079928     0.038588      0.20119
 (6)   0.00076483     0.014227     0.027598      0.02506       0.2943
(11)      0.78938     0.050772      0.06643

Cmd> # Macro resid computes all these with one command; see Sec.3.17

Cmd> resid()
           Depvar    StdResids          HII     Cook's D      t-stats
(1)             4    -0.017302      0.30754   4.4318e-05    -0.016414
(2)            18     -0.45867      0.27815     0.027022     -0.43978
(3)            14      0.11455      0.15449   0.00079928      0.10875
(4)            18      0.82601      0.14506     0.038588      0.81181
(5)            26      -1.0383      0.35893      0.20119      -1.0428
(6)            26      0.14126      0.10313   0.00076483      0.13414
(7)            21     -0.54206      0.12684     0.014227     -0.52196
(8)            30     -0.73346      0.13338     0.027598     -0.71533
(9)            28      0.88505      0.08757      0.02506      0.87459
(10)           36      -2.2161      0.15238       0.2943      -2.9471
(11)           65       1.4359      0.53456      0.78938        1.529
(12)           62      0.36646      0.53144     0.050772      0.35002
(13)           40       1.4504     0.086537      0.06643       1.5484
```

Note that Case 10 does not appear to fit model well and that deletion of case 11 would most influence the analysis. See Weisberg (1985) for a discussion of residuals and Cook's distance.

10.5 **One way ANOVA** This is example 3-1 of Montgomery. The response is the tensile strength of a synthetic fiber; the experimental treatment is the percent of cotton: 15, 20,

25, 30 or 35. There are five observations for each treatment.

```
Cmd> strength <- vector(7,7,15,11,9, 12,17,12,18,18,\
 14,18,18,19,19, 19,25,22,19,23, 7,10,11,15,11)

Cmd> percent <- rep(run(15,35,5),rep(5,5))

Cmd> grp <- makefactor(percent) # use macro to create factor

Cmd> print(format:"2.0f",percent,grp) # print using changed format
percent:
 (1) 15 15 15 15 15 20 20 20 20 20 25 25 25 25 25 30 30 30 30 30 35
(22) 35 35 35 35
grp:
 (1)  1  1  1  1  1  2  2  2  2  2  3  3  3  3  3  4  4  4  4  4  5
(22)  5  5  5  5

Cmd> boxplot(split(strength,grp),xlab:"Strength",title:\
"Fiber strength split by percent cotton (15, 20, 25, 30, 35)")
```




```
Cmd> # Clear differences in strength as percent cotton varies.

Cmd> # See Sec. 2.12.2 for boxplot()

Cmd> anova("strength=grp") # One-way ANOVA with 5 groups
Model used is strength=grp
                DF          SS          MS
CONSTANT         1        5655        5655
grp              4      475.76      118.94
ERROR1          20       161.2        8.06
```

```
Cmd> contrast("grp",vector(-2,-1,0,1,2)) # linear effect of percent
component: estimate
(1)          8.2
component: ss
(1)        33.62
component: se
(1)        4.015

Cmd> # See Sec. 3.16 for contrast()

Cmd> # Compare with the polynomial regression model in Sec. 10.6

Cmd> tabs(strength, grp) # Cell by cell statistics; see Sec. 3.12
component: mean
(1)          9.8         15.4         17.6         21.6         10.8
component: var
(1)         11.2          9.8          4.3          6.8          8.2
component: count
(1)            5            5            5            5            5
```

10.6 **Polynomial regression** We continue with the same data, modeling strength as a fourth order polynomial in percent.

```
Cmd> percent2 <- percent*percent  # quadratic term

Cmd> percent3 <- percent2*percent # cubic term

Cmd> percent4 <- percent3*percent # quartic term

Cmd> anova("strength=percent+percent2+percent3+percent4",fstat:T)
Model used is strength=percent+percent2+percent3+percent4
WARNING: summaries are sequential
               DF           SS           MS            F      P-value
CONSTANT        1         5655         5655    701.61787            0
percent         1        33.62        33.62      4.17122     0.054524
percent2        1       343.21       343.21     42.58242   2.3255e-06
percent3        1        64.98        64.98      8.06203     0.010133
percent4        1       33.946       33.946      4.21163     0.053469
ERROR1         20        161.2         8.06

Cmd> regcoefs() # Coefficients, etc. see Sec. 3.13.1
                Coef       StdErr            t
CONSTANT       -406.4       231.52      -1.7554
percent        73.777        40.63       1.8158
percent2      -4.8077       2.5851      -1.8598
percent3      0.13773     0.070868       1.9435
percent4   -0.0014533   0.00070817      -2.0522

Cmd> # Notice SS for 'percent' is linear contrast SS from above

Cmd> # Compute F-statistics and P-values "by hand"

Cmd> f <- (SS[run(2,5)]/DF[run(2,5)])/(SS[6]/DF[6]) # F-stats

Cmd> f ; 1-cumF(f,DF[run(2,5)],DF[6]); invF(1-.05,1,DF[6])
(1)       4.1712       42.582        8.062       4.2116 F-stats
(1)     0.054524   2.3255e-06     0.010133     0.053469 P-Values
(1)       4.3512   5% critical value
```

Strict believers in 5% significance (as Montgomery apparently is) would conclude the

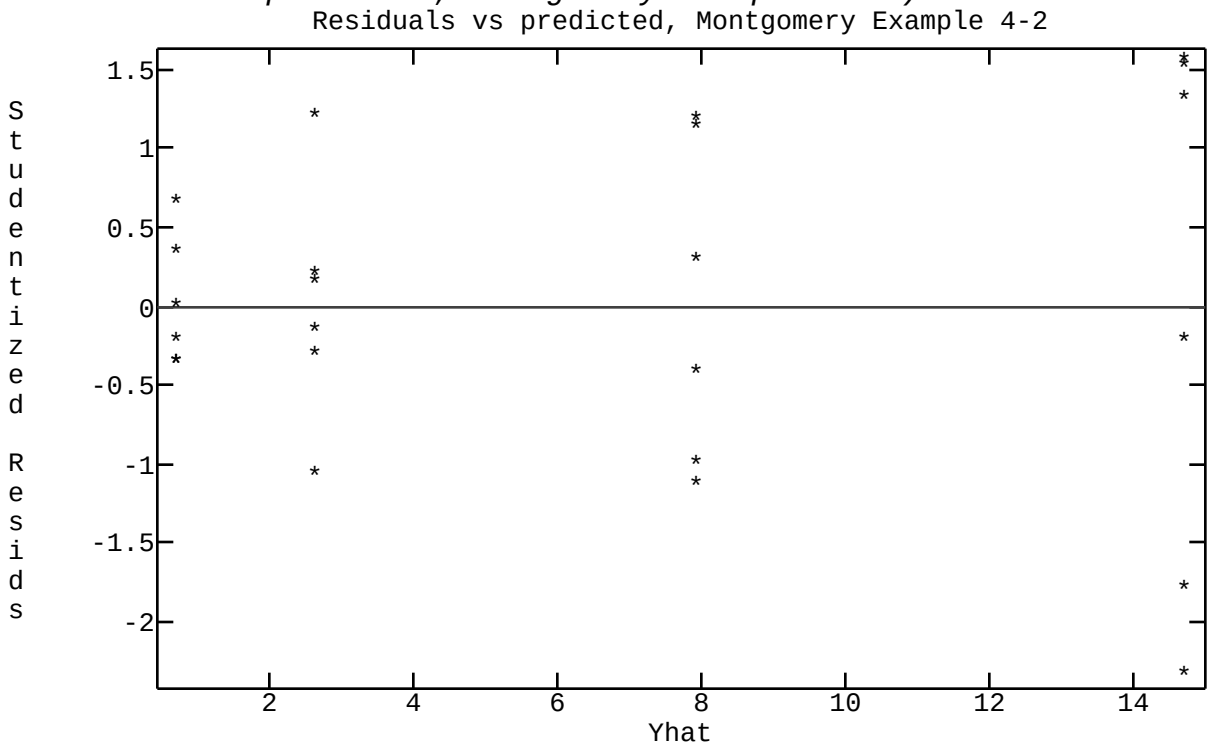
cubic term is significant and the quartic term is not. Let's do a cubic regression using shortcut P3( ) described in Sec. 3.4.3.

```
Cmd> regress("strength=P3(percent)",pvals:T)
Model used is strength=P3(percent)
                   Coef       StdErr            t      P-Value
CONSTANT         62.611       39.757       1.5748      0.13024
percent         -9.0114       5.1966      -1.7341     0.097558
(percent)^2      0.48143      0.21605       2.2284     0.036915
(percent)^3      -0.0076     0.002874      -2.6444     0.015164

N: 25,  MSE: 9.2927, DF: 21,  R^2: 0.69363
Regression F(3,21): 15.848, P-value: 1.2953e-05, Durbin-Watson: 2.2203
To see the ANOVA table type 'anova()'

Cmd> anova(,fstat:T)
Model used is strength=P3(percent)
WARNING: summaries are sequential
               DF           SS           MS            F      P-value
CONSTANT        1         5655         5655    608.54957            0
{percent}       1        33.62        33.62      3.61791     0.070966
{(percent)^2}   1       343.21       343.21     36.93394   4.9628e-06
{(percent)^3}   1        64.98        64.98      6.99262     0.015164
ERROR1         21       195.15       9.2927
```

10.7 **Variance stabilizing transformations** This is Example 4-2, page 94 from Montgomery. The responses are estimates of peak discharge during flood flow when using four different estimation techniques.

```
Cmd> discharge <- vector(.34,.12,1.23,.7,1.75,.12,.91,2.94,2.14,\
2.36,2.86,4.55,6.31,8.37,9.75,6.09,9.82,7.24,17.15,11.82,\
10.95,17.2,14.35,16.82)

Cmd> method <- factor(rep(run(4),rep(6,4)))#6 1's,6 2's,6 3's,6 4's

Cmd> anova("discharge=method")
Model used is discharge=method
                DF           SS           MS
CONSTANT         1       1012.6       1012.6
method           3       708.35       236.12
ERROR1          20       62.081       3.1041

Cmd> tabs(discharge,method) # Group statistics; see Sec. 3.12
component: mean
(1)          0.71       2.6267         7.93       14.715
component: var
(1)       0.43704       1.4213       2.7128        7.845
component: count
(1)             6            6            6            6

Cmd> # Let's look at residuals; see Sec.3.17
```

```
Cmd> resvsyhat(title:\
"Residuals vs predicted, Montgomery Example 4-2")
```

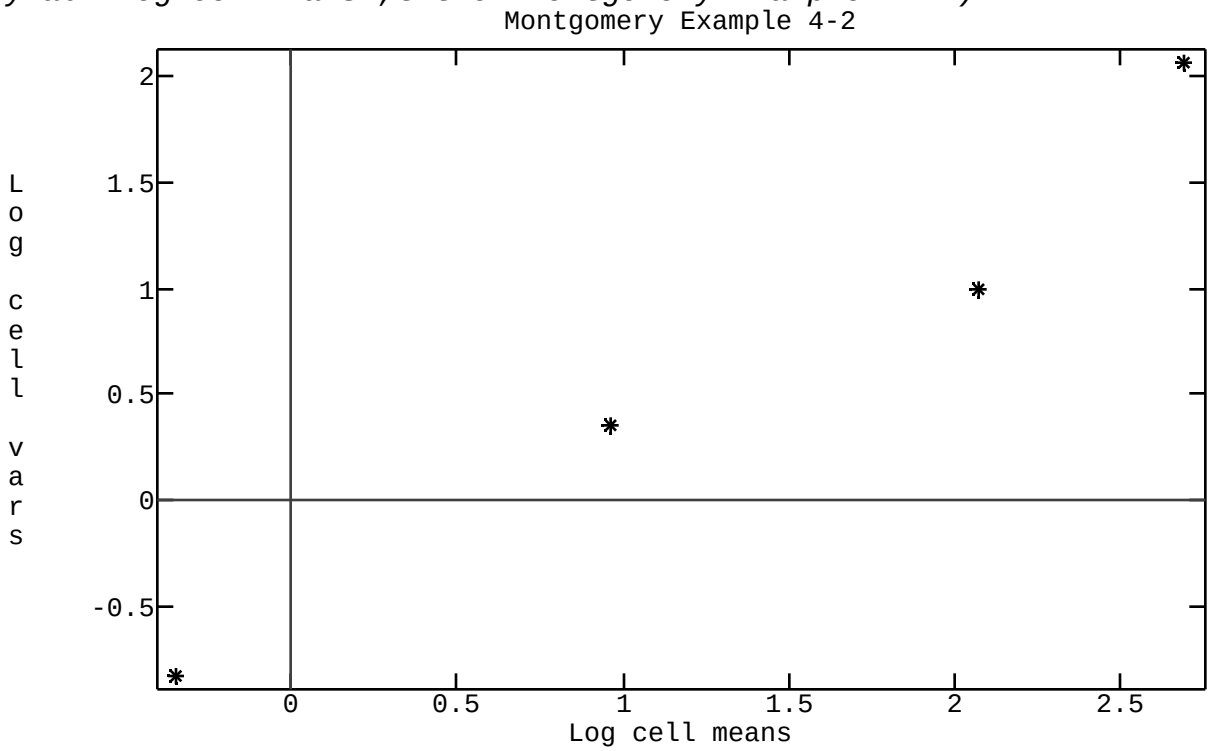


There appears to be pretty clear evidence that the variance increases with mean response. This often indicates the response variable should be transformed.

```
Cmd> @tmp <- tabs(discharge,method);means <- @tmp$mean;\
          vars <- @tmp$var
```

Note the use of the temporary variable `@tmp` which disappears before the next prompt.

```
Cmd> plot(log(means),log(vars),xlab:"Log cell means",\
ylab:"Log cell vars",title:"Montgomery Example 4-2")
```




```
Cmd> regress("{log10(sqrt(vars))}={log10(means)}",pvals:T)
Model used is {log10(sqrt(vars))}={log10(means)}
                    Coef       StdErr            t      P-Value
CONSTANT         -0.12077     0.043562      -2.7724       0.1092
log10(means)      0.44647     0.056589       7.8897     0.015688

N: 4,  MSE: 0.0032157, DF: 2,  R^2: 0.96887
Regression F(1,2): 62.248, P-value: 0.015688, Durbin-Watson: 2.7556
To see the ANOVA table type 'anova()'
```

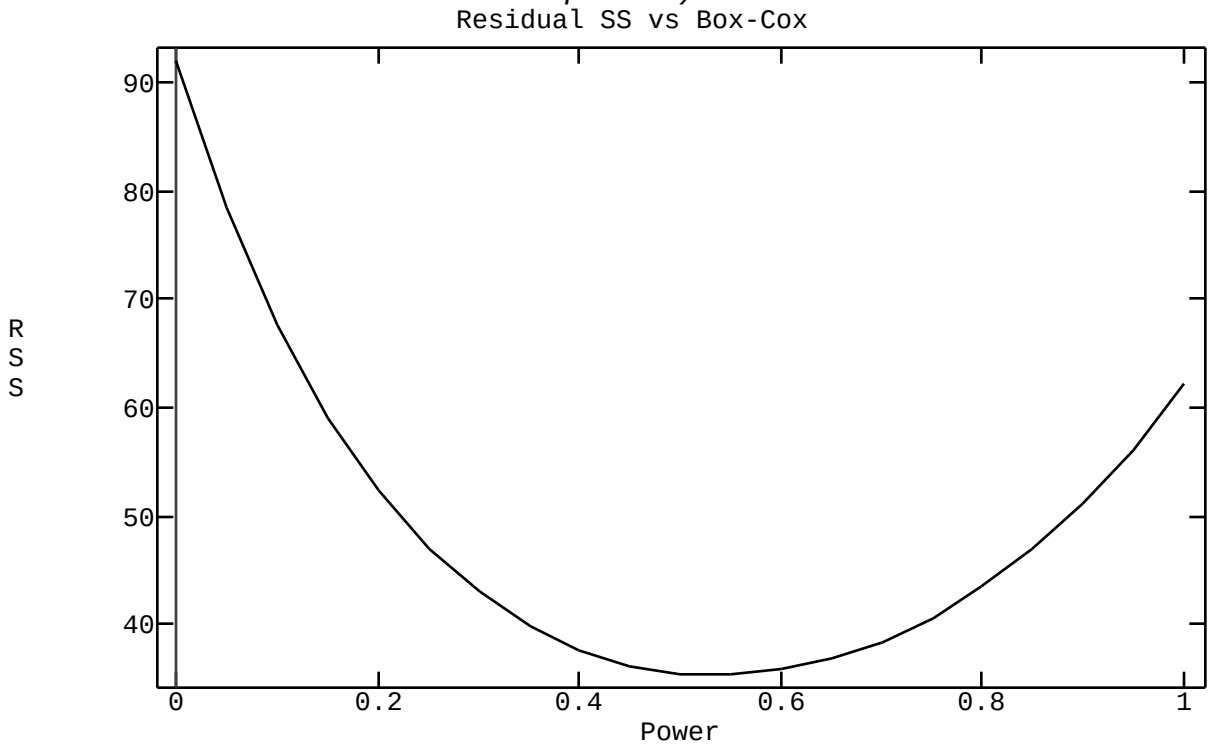
One minus the regression coefficient, namely $y^{1-0.446}$, is the suggested power transformation, approximately a square root.

```
Cmd> anova("{discharge^.5}=method")
Model used is {discharge^.5}=method
                DF           SS           MS
CONSTANT         1       120.52       120.52
method           3       32.684       10.895
ERROR1          20       2.6884      0.13442

Cmd> tabs(discharge^.5,method) # Much better on this scale
component: mean
(1)      0.75742        1.582       2.8033       3.8208
component: var
(1)      0.16358      0.14879     0.085844      0.13947
component: count
(1)            6            6            6            6
```

As an alternative, lets try Box-Cox transformations, seeking the power with minimum residual sum of squares. We use pre-defined macro boxcox and compute residual

sums of squares in a `for` loop. `boxcox(x,0)` is a scaled log transformof `x` and when `p` ≠ 0 `boxcox(x,p)` is a scaled power transform (see Sec. 2.10.1).

```
Cmd> powers <- run(0,1,.05); ss <- 0*powers

Cmd> for(@i,run(length(powers))){
    @tmp <- boxcox(discharge,powers[@i])
    anova("@tmp=method",silent:T) # silent:T suppresses anova() output
    ss[@i] <- SS[3];;}

Cmd> lineplot(Power:powers,RSS:ss,\
title:"Residual SS vs Box-Cox power")
```



The power at the minimum of the curve is often taken as indicating the "optimal" power transformation. Because this is near Power = .5, this method also leads to choosing a square root transformation.

Macro `boxcoxvec` in file `design.mac` distributed with MacAnova provides a short cut method for doing the preceding.

```
Cmd> getmacros(boxcoxvec,quiet:T)

Cmd> help(boxcoxvec, file:"design.hlp") # see Sec. 8.6.1
boxcoxvec(rhs_model,y,powers:pow) computes the error SS for y
transformed to the boxcox powers given in pow and modeled using the
explanatory variables in rhs_model.  The returned value is a
structure with named components power and SS giving the boxcox powers
and the error SS.

rhs_model is the right hand side of the anova model (the part
following '=') as a character scalar, for example "x + a + b".  If
powers:pow is omitted, the default powers are run(-1,2,.25).

*****  Interrupt  *****
.
```

```
Cmd> boxcoxvec("method",discharge,powers:powers)
component: power
 (1)            0         0.05          0.1         0.15          0.2
 (6)         0.25          0.3         0.35          0.4         0.45
(11)          0.5         0.55          0.6         0.65          0.7
(16)         0.75          0.8         0.85          0.9         0.95
(21)            1
component: SS
 (1)       91.958       78.439       67.658       59.084       52.303
 (6)       46.989       42.891       39.817       37.617       36.179
(11)       35.424       35.294       35.756       36.792       38.404
(16)       40.607       43.432       46.923       51.142       56.163
(21)       62.081

Cmd> ss[run(5)] # some of values computed previously
(1)       91.958       78.439       67.658       59.084       52.303
```

10.8 **Randomized complete blocks** This is example 5-1 on page 129 of Montgomery. The response is the depth of a depression made when a tip is pressed with a standard force into a piece of metal. We wish to see if there are any differences in readings between four types of tips (treatment); we use a tip of each type once with each of four metal specimens (block).

```
Cmd> tiptype <-factor(rep(run(4),rep(4,4)))

Cmd> # tiptype is vector(1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4)

Cmd> specimen <- factor(rep(run(4),4))# vector(1,2,3,4,1,2,3,4,....)

Cmd> depth <- vector(9.3,9.4,9.6,10,9.4,9.3,9.8,9.9,9.2,9.4,\
9.5,9.7,9.7,9.6,10,10.2)

Cmd> anova("depth=specimen+tiptype")
Model used is depth=specimen+tiptype
                DF           SS           MS
CONSTANT         1       1482.2       1482.2
specimen         3        0.825        0.275
tiptype          3        0.385      0.12833
ERROR1           9         0.08    0.0088889

Cmd> anova("depth=tiptype+specimen") # different factor order
Model used is depth=tiptype+specimen
                DF           SS           MS
CONSTANT         1       1482.2       1482.2
tiptype          3        0.385      0.12833
specimen         3        0.825        0.275
ERROR1           9         0.08    0.0088889
```

Because the data are balanced, the sums of squares for `tiptype` and `specimen` are the same for both orderings of these factors (see Sec. 3.9). We can tell MacAnova recognized the balance because the message `WARNING: summaries are sequential` is not printed.

```
Cmd> coefs("tiptype") # determine the treatment effects; Sec. 3.13
(1)        -0.05       -0.025       -0.175         0.25

Cmd> # These do not depend on order of factors in model
(1)        -0.05       -0.025       -0.175         0.25

Cmd> # Compute a contrast in tiptype.  See Sec. 3.16

Cmd> contrast("tiptype",vector(-2,3,1,-2))
component: estimate
(1)        -0.65
component: ss
(1)     0.093889
component: se
(1)          0.2

Cmd> sum(vector(-2,3,1,-2)*coefs("tiptype")) # confirmation of value
(1)        -0.65   Same as component estimate

Cmd> contrast("tiptype",vector(-2,3,1,-2),"specimen")# Note by-var.
component: estimate
(1)         -0.6         -0.7         -0.3           -1
component: ss
(1)         0.02     0.027222        0.005     0.055556
component: se
(1)          0.4          0.4          0.4          0.4
```

This last computes the contrast separately for each block (level of specimen).

10.8.1 **Multiple comparisons** A particularly important type of contrast is one with all coefficients 0 except for one +1 and one –1, that is a direct comparison of two factor levels. These can, of course be computed using contrast(). However, a common task is to carry out all $k(k-1)/2$ comparisons of different factor levels, where k is the number of levels, and determine which of them are significantly different from zero, using a *multiple comparison* procedure such as Bonferronized *t*-tests (BSD or Bonferroni Significant Difference), the studentized range (HSD = Honestly Significant Difference or Tukey method) or multiple ordinary two-sample *t*-tests (LSD or Least Significant Difference). The BSD uses a two-tail Student's t critical value corresponding to $\frac{\alpha}{\frac{1}{2}k(k-1)}$ instead of α. See Chapter 8 of Steel and Torrie (1980) for details of the HSD and LSD procedures.

Macro pairedcomp in file design.mac distributed with MacAnova is to be run immediately after an anova() command. It computes all the pairwise differences in the effects for a factor and displays them in a form that indicates which are significantly different.

```
Cmd> getmacros(pairedcomp)
) pairedcomp(termname,lev [,method:T] ) does paired
) comparisons of the means of the levels of the factor given in
) termname at level of significance lev.  If only these two
) arguments are used, then the paired comparisons are done using
) the Bonferroni method.  Other techniques can be requested by
) using a method keyword.  Currently supported methods are: lsd
) (least significant difference), bsd (Bonferroni), snk
```

```
) (Student-Newman-Keuls), hsd (Tukey's honest significant
) difference, also called the studentized range procedure), regwb
) (regw with Bonferroni test), and regwr (regw with studentized
) range). Thus, for example, pairedcomp("trt",.01,hsd:T) does
) paired comparisons between the levels of trt at signficance .01
) using the hsd method.
)
) termname should be a single factor, not an interaction.
) lev should be a number between 0 and 1
)
) The printed output is one row for each level of the term, sorted
) from smallest to largest effect, giving the "underlines", term
) number, and effect.
)
) An alternative form is pairedcomp(termname,critval:val)  In this
) form, all paired t-test statistics are compared to the critical
) value given in val.  The same output is printed as before.
```

Here we apply it to the above data set. The default behavior is to compute the BSD at the specified significance level.

```
Cmd> pairedcomp("tiptype",.05) # BSD at .05 level
  |    3   -0.175
  |    1    -0.05
  |    2   -0.025
       4     0.25
```

The display indicates that levels 1, 2 and 3 do not differ significantly, but level 4 differs significantly from all of them. The numbers are just the elements of `coefs("tiptype")` in increasing order.

You can specify other multiple comparison methods using a keyword phrase. The choices are `bsd:T` (the default), `lsd:T`, `snk:T`, `hsd:T`, `regwb:T` and `regwr:T` (see comments after `getmacros(pairedcomp)` above).

```
Cmd> pairedcomp("tiptype",.01,hsd:T) #HSD at .01 level
  |      3   -0.175
  |      1    -0.05
  | |    2   -0.025
    |    4     0.25
```

Again, levels 1, 2, and 3 do not differ significantly, but now the display indicates that, although level 4 differs significantly from levels 1 and 3 it does not differ significantly from level 2. `pairedcomp` uses `invstudrng()` (Sec. 2.12.8) to compute the required critical values.

The HSD is fully appropriate only when all the $s_{\hat{\alpha}_i-\hat{\alpha}_j}$ are equal, as is the case when the design is balanced. However, it is a good approximation even when the standard errors differ.

When the same critical value is used for testing all subsets of factor levels for homogeneity, as is the case for the LSD and HSD, you can provide an explicit critical value for a statistic of the form $t = \left(\hat{\alpha}_i - \hat{\alpha}_j\right) / s_{\hat{\alpha}_i-\hat{\alpha}_j}$, where $\hat{\alpha}_i$ is an estimated factor effect and $s_{\hat{\alpha}_i-\hat{\alpha}_j}$ is the estimated standard error of $\hat{\alpha}_i - \hat{\alpha}_j$. We illustrated it with redoing the

multiple comparison based on the HSD. Since critical values for the HSD are usually tabulated for a statistic $Q = \sqrt{2} \times t$, you need to divide the tabled value q_α by $\sqrt{2}$. One of many sources for the critical values is Table A.8 of Steel and Torrie (1980).

```
Cmd> # From Steel and Torrie, the 1% critical value for the
Cmd> # Studentized range with 4 groups and 9 error d.f. is 5.96
Cmd> Cmd> pairedcomp("tiptype",critval:5.96) # HSD @ 1% level
  |      3   -0.175
  |      1    -0.05
  | |    2   -0.025
    |    4     0.25
```

Alternatively, you can compute the critical value within MacAnova using `invstudrng()`.

```
Cmd> usage(invstudrng,orig:T)
invstudrng(P, ngroup, errorDf [,epsilon]), elements of P between 0
  and 1, elements of ngroup integers >= 2, elements of errorDf >= 1,
  epsilon > 0 small
Cmd> Q <- invstudrng(1-.01, 4,DF[4]); Q
(1)       5.9576       Rounds to 5.96
Cmd> pairedcomp("tiptype",critval:5.96/sqrt(2))
  |      3   -0.175
  |      1    -0.05
  | |    2   -0.025
    |    4     0.25
```

A more restricted multiple comparisons situation occurs when one treatments, say the first, is a control and you want to compare all the other treatments to it. Assume for the sake of an example that this is the case with these data and we want to compare level 1 of `tiptype` with levels 2, 3 and 4.

One possibility is to Bonferronize the $k-1$ t-statistics, that is, use one- or two-tail Student's t critical value corresponding to $\alpha/(k-1) = \alpha/3$.

```
Cmd> effects <- coefs(tiptype);effects # effects of tiptype
(1)        -0.05       -0.025       -0.175         0.25
Cmd> stderr <- sqrt(2*(SS[4]/DF[4])/4)
Cmd> (effects[-1] - effects[1])/stderr # t-statistics
(1)       0.375       -1.875         4.5
Cmd> invstu(1 - .025/3, DF[4]) # 2 tail critical value
(1)       2.9333
```

Bonferronized critical values are conservative, in the sense that the correct critical values are always smaller. Accurate critical values for this situation were given by C. W. Dunnett (see Steele and Torie (1980), Sec. 8.9). These can be computed using `invdunnett()` (see Sec. 2.12.8).

```
Cmd> invdunnett(1-.05, 4, DF[4])# 4 groups, DF[4] error d.f.
(1)       2.8116
```

Both critical values give the same conclusion: Only the t-statistic comparing level 4 of

tiptype with level 1 is significant. No conclusions are drawn concerning differences among levels 2, 3 and 4.

10.8.2 **Randomized block with data missing** To illustrate an analysis when balance has been lost because of missing data, we set one case (tiptype=1 and specimen=1) to MISSING. In a randomized block design, the order of block followed by treatment (specimen + tiptype) is appropriate.

```
Cmd> depth[1] <- ?; anova("depth=specimen+tiptype")
Model used is depth=specimen+tiptype
WARNING: cases with missing values deleted
WARNING: summaries are sequential
                DF           SS           MS
CONSTANT         1       1395.9       1395.9
specimen         3      0.72567      0.24189
tiptype          3      0.37611      0.12537
ERROR1           8     0.075556    0.0094444
```

Contrasts and their sums of squares change when there are missing data.

```
Cmd> contrast("tiptype",vector(-2,3,1,-2))
component: estimate
(1)     -0.69444
component: ss
(1)     0.097534
component: se
(1)       0.2161
Cmd> contrast("tiptype",vector(-2,3,1,-2),"specimen") #by-variable
component: estimate
(1)      MISSING         -0.7         -0.3           -1
component: ss
(1)      MISSING     0.027222        0.005     0.055556
component: se
(1)      MISSING      0.41231      0.41231      0.41231
```

The contrast output is MISSING for block 1 because that is where the missing value is.

You can use pairedcomp in this unbalanced case, too.

```
Cmd> pairedcomp("tiptype",.05) # BSD at .05 level
  |    3   -0.181
  |    1  -0.0333
  |    2  -0.0306
       4    0.244
```

The effects are different but the conclusions are the same as in the complete data case.

Because the data are no longer balanced (Sec. 3.9), an ANOVA with the terms in the other order yields different sums of squares for tiptype and specimen. It is an *incorrect* analysis, since inference about treatments must be made adjusted for blocks.

```
Cmd> anova("depth=tiptype+specimen")
Model used is depth=tiptype+specimen
WARNING: cases with missing values deleted
WARNING: summaries are sequential
                DF           SS           MS
CONSTANT         1       1395.9       1395.9
tiptype          3      0.37317      0.12439
specimen         3      0.72861      0.24287
ERROR1           8     0.075556    0.0094444
```

10.9 **Latin squares** The data are from Table 5-9, page 146 of Montgomery. There are two blocking factors, `batches` (the rows in the Latin square) and `operators` (columns). The treatment is the mixing formulation of dynamite (letters), and the response is a measure of explosive force. The Latin square used is

A	B	C	D	E
B	C	D	E	A
C	D	E	A	B
D	E	A	B	C
E	A	B	C	D

```
Cmd> force <- vector(24,20,19,24,24, 17,24,30,27,36, 18,38,26,27,21,\
    26,31,26,23,22, 22,30,20,29,31)

Cmd> operators <- factor(rep(run(5),5)) # 1,2,3,4,5,1,2,3,4,5,...

Cmd> batches <- factor(rep(run(5),rep(5,5))) # 1,1,1,1,1,2,2,2,...

Cmd> mix <- factor(vector(1,2,3,4,5, 2,3,4,5,1, 3,4,5,1,2,\
    4,5,1,2,3, 5,1,2,3,4))

Cmd> anova("force=batches+operators+mix",fstat:T)
Model used is force=batches+operators+mix
                DF           SS           MS            F      P-value
CONSTANT         1        16129        16129   1512.09375   5.3957e-14
batches          4           68           17      1.59375      0.23906
operators        4          150         37.5      3.51562     0.040373
mix              4          330         82.5      7.73438    0.0025365
ERROR1          12          128       10.667
```

For a complete data Latin square, the order of the terms in the model does not matter. Thus `anova("force=mix+batches+operator")` would yield the same sums of squares in a different order. MacAnova recognizes this as can be inferred from the absence of the "`WARNING: summaries are sequential`" message. However, if there were missing values, it would be an error if the term for treatment (`mix`) were not last.

10.10 **Balanced incomplete blocks** The data are from Table 6-1 of Montgomery. The response is the elapsed time for a reaction, the blocks are batches of raw material, and the treatments are different types of catalyst. In this example we do only the intrablock analysis. Because balanced incomplete block data are not "balanced" as understood by MacAnova, it is again essential to have blocks precede treatments in the model. The catalyst types used for batches 1, 2, 3, and 4 were (1,3,4), (1,2,3), (2,3,4), and (1,2,4),

respectively. This is a balanced incomplete block design because each treatment appears equally often (2 times) with each other treatment.

```
Cmd> time <- vector(73,73,75,74,75,75,67,68,72,71,72,75)

Cmd> batches <- factor(rep(run(4),rep(3,4)))#vector(1,1,1,2,2,2,...)

Cmd> catalyst <- factor(vector(1,3,4,1,2,3,2,3,4,1,2,4))

Cmd> anova("time=batches+catalyst",fstat:T)
Model used is time=batches+catalyst
WARNING: summaries are sequential
               DF           SS           MS            F      P-value
CONSTANT        1        63075        63075  97038.46154   6.4698e-12
batches         3           55       18.333     28.20513    0.0014678
catalyst        3        22.75       7.5833     11.66667     0.010739
ERROR1          5         3.25         0.65

Cmd> secoefs("catalyst") # adjusted treatment effects
component: coefs
(1)       -1.125       -0.875         -0.5          2.5
component: se
(1)      0.42757      0.42757      0.42757      0.42757

Cmd> contrast("catalyst",vector(1,1,1,-3))#compare last with 1st 3
component: estimate
(1)          -10
component: ss
(1)       22.222   Almost the whole treatment SS
component: se
(1)       1.7103

Cmd> temp <- (catalyst==1) + (catalyst==2) + (catalyst==3) - \
3*(catalyst==4) # create vector equivalent to contrast

Cmd> print(format:"3.0f",temp)
temp:
 (1)   1   1  -3   1   1   1   1   1  -3   1   1  -3

Cmd> anova("time=batches+temp+catalyst",fstat:T)
Model used is time=batches+temp+catalyst
WARNING: summaries are sequential
               DF           SS           MS            F      P-value
CONSTANT        1        63075        63075  97038.46154   6.4698e-12
batches         3           55       18.333     28.20513    0.0014678
temp            1       22.222       22.222     34.18803    0.0020716
catalyst        2      0.52778      0.26389      0.40598      0.68646
```

The small sum of square for `catalyst` after including the contrast vector indicates that there is no evidence catalysts 1, 2, and 3 differ, but strong evidence they differ from catalyst 4.

10.11 **Analysis of covariance** This is example 16-1, page 480 of Montgomery. The response is breaking strength of a fiber, the covariate is diameter of the fiber, and there are three different machines (treatments) making the fibers that we would like to compare. First we do an analysis of variance on breaking strength, ignoring fiber diameter.

```
Cmd> strength <- vector(36,41,39,42,49,40,48,39,45,44,35,37,42,34,32)
Cmd> diameter <- vector(20,25,24,25,32,22,28,22,30,28,21,23,26,21,15)
Cmd> machine <-factor(rep(run(3),rep(5,3)))#vector(1,1,1,1,1,2,2,...,)
Cmd> anova("strength=machine",fstat:T)# analysis ignoring diameter
Model used is strength=machine
               DF           SS           MS            F      P-value
CONSTANT        1        24241        24241   1412.07379   8.1046e-14
machine         2        140.4         70.2      4.08932     0.044232
ERROR1         12          206       17.167

Cmd> predtable(seest:T) # unadjusted treatment means & std errors
component: estimate
(1)         41.4         43.2           36
component: SEest
(1)       1.8529       1.8529       1.8529

Cmd> secoefs("machine") # unadjusted machine effects
component: coefs
(1)          1.2            3         -4.2
component: se
(1)       1.5129       1.5129       1.5129
Cmd> contrast("machine",vector(1,-1,0)) # unadjusted contrast
component: estimate
(1)         -1.8
component: ss
(1)          8.1
component: se
(1)       2.6204
Cmd> contrast("machine",vector(1,1,-2)) # unadjusted contrast
component: estimate
(1)         12.6
component: ss
(1)        132.3
component: se
(1)       4.5387
```

Now we do an *analysis of covariance* with covariate diameter. This assumes that the slope of the regression of strength on diameter is the same for each machine (parallel line model). Notice that the machine effect is much smaller after allowing for the covariate, as is the error mean square.

```
Cmd> anova("strength=diameter+machine",fstat:T)
Model used is strength=diameter+machine
WARNING: summaries are sequential
               F           SS           MS            F      P-value
CONSTANT        1        24241        24241   9527.89438            0
diameter        1       305.13       305.13    119.93304   2.9601e-07
machine         2       13.284       6.6419      2.61064      0.11808
ERROR1         11       27.986       2.5442
```

```
Cmd> predtable(seest:T) # adjusted treatment means and SE's
component: estimate
(1)       40.382       41.419       38.798
component: SEest
(1)      0.72363      0.74442      0.78788

Cmd> secoefs("machine"); secoefs("diameter")
component: coefs     Machine effects adjusted for diameter
(1)      0.18241       1.2192      -1.4016
component: se
(1)        0.595      0.62012      0.67167
component: coefs     Slope of diameter in model
(1)      0.95399
component: se        Its standard error
(1)      0.11405

Cmd> contrast("machine", vector(1,-1,0))
component: estimate  Adjusted contrast
(1)      -1.0368
component: ss
(1)       2.6656
component: se
(1)       1.0129
```

The usual analysis of covariance assumes that the lines are parallel. It is always a good idea to check this by fitting a model the effectively fits separate lines to each machine.

```
Cmd> anova("strength=machine+diameter+diameter.machine",fstat:T)
Model used is strength=machine+diameter+diameter.machine
WARNING: summaries are sequential
                  DF          SS          MS           F     P-value
CONSTANT           1       24241       24241  8640.65457           0
machine            2       140.4        70.2    25.02306  0.00021073
diameter           1      178.01      178.01    63.45381  2.2905e-05
machine.diameter   2      2.7372      1.3686     0.48784     0.62929
ERROR1             9      25.249      2.8054
```

The machine by diameter term is not significant (P = .63) so there is no evidence the slopes differ.

The previous model implicitly parametrizes the dependence of `strength` on `diameter` in terms of an overall slope (term `diamater`) and three deviations from that slope (`machine.diameter`).

```
Cmd> coefs(diameter)# different from after strength=diameter+machine
(1)      0.94187

Cmd> coefs("machine.diameter") # deviations from .94187
(1,1)      0.16241
(2,1)     -0.08473
(3,1)    -0.077675
```

By omitting the `diameter` term, you can fit the same model parametrized by separate slopes for each machine.

```
Cmd> anova("strength=machine+diameter.machine")
Model used is strength=machine+diameter.machine
WARNING: summaries are sequential
                   DF           SS           MS
CONSTANT            1        24241        24241
machine             2        140.4         70.2
machine.diameter    3       180.75        60.25
ERROR1              9       25.249       2.8054

Cmd> coefs("machine.diameter")
(1,1)       1.1043
(2,1)      0.85714
(3,1)       0.8642

Cmd> coefs("machine.diameter") - 0.94187 #deviations from ave slope
(1,1)      0.16241
(2,1)    -0.084727
(3,1)    -0.077672
```

10.12 **Factorial models** The data are from Example 7-4, page 225 of Montgomery. The response is a transformation of the volume of a packaged beverage; treatments are `percent` carbonation, `pressure`, and `speed`. All treatments are quantitative and have evenly spaced levels.

```
Cmd> volume <- vector(-3,-1,0,1,5,4,-1,0,2,1,7,6,-1,0,2,3,7,9,1,1,\
6,5,10,11)

Cmd> percent <- factor(rep(rep(run(3),rep(2,3)),4))

Cmd> pressure <- factor(rep(run(2),rep(12,2)))

Cmd> speed <- factor(rep(rep(run(2),rep(6,2)),2))
```

Study these last three lines carefully to understand how `rep()` is used to create factor levels. Here is what the factors actually look like.

```
Cmd> print(format:"2.0f",percent,pressure,speed)
percent:
 (1)  1  1  2  2  3  3  1  1  2  2  3  3  1  1  2  2  3  3  1  1  2
(22)  2  3  3
pressure:
 (1)  1  1  1  1  1  1  1  1  1  1  1  1  2  2  2  2  2  2  2  2  2
(22)  2  2  2
speed:
 (1)  1  1  1  1  1  1  2  2  2  2  2  2  1  1  1  1  1  1  2  2  2
(22)  2  2  2
```

```
Cmd> anova("volume=percent*pressure*speed") # Full factorial model
Model used is volume=percent*pressure*speed
                          DF           SS           MS
CONSTANT                   1       234.38       234.38
percent                    2       252.75       126.38
pressure                   1       45.375       45.375
percent.pressure           2         5.25        2.625
speed                      1       22.042       22.042
percent.speed              2      0.58333      0.29167
pressure.speed             1       1.0417       1.0417
percent.pressure.speed     2       1.0833      0.54167
ERROR1                    12          8.5      0.70833
```

Redo it, pooling the two degrees of freedom for three-way interaction into error by omitting `percent.pressure.speed` from the model.

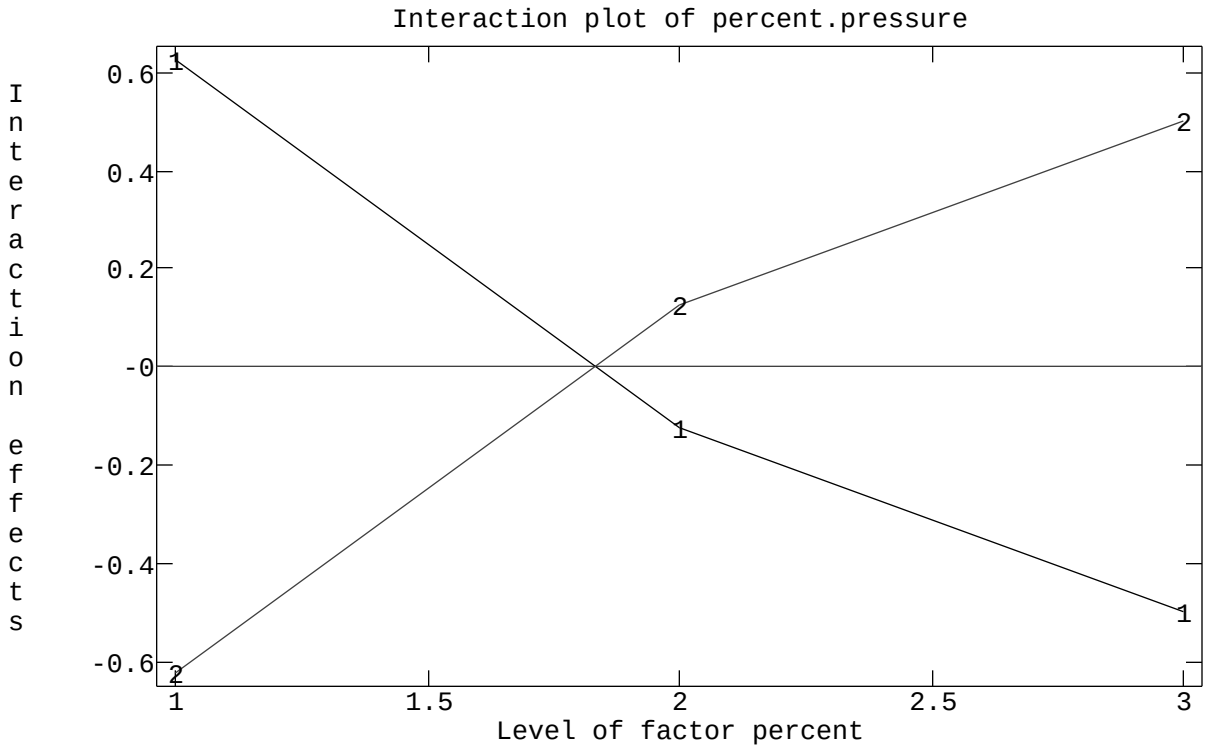
```
Cmd> anova("volume=percent*pressure*speed
    -percent.pressure.speed",fstat:T)
Model used is volume=percent*pressure*speed-percent.pressure.speed
                 DF           SS           MS            F      P-value
CONSTANT          1       234.38       234.38    342.39130    3.077e-11
percent           2       252.75       126.38    184.61739    8.6824e-11
pressure          1       45.375       45.375     66.28696    1.1157e-06
percent.pressure  2         5.25        2.625      3.83478     0.046983
speed             1       22.042       22.042     32.20000    5.7379e-05
percent.speed     2      0.58333      0.29167      0.42609      0.66125
pressure.speed    1       1.0417       1.0417      1.52174      0.23767
ERROR1            4       9.5833      0.68452
```

There is a hint of a `pressure` by `percent` interaction. Let's look at the `pressure` effect separately for each level of `percent`.

```
Cmd> contrast("pressure",vector(-1,1),"percent") # percent is byvar
component: estimate
(1)          1.5            3         3.75
component: ss
(1)          4.5           18       28.125
component: se
(1)      0.58503      0.58503      0.58503
```

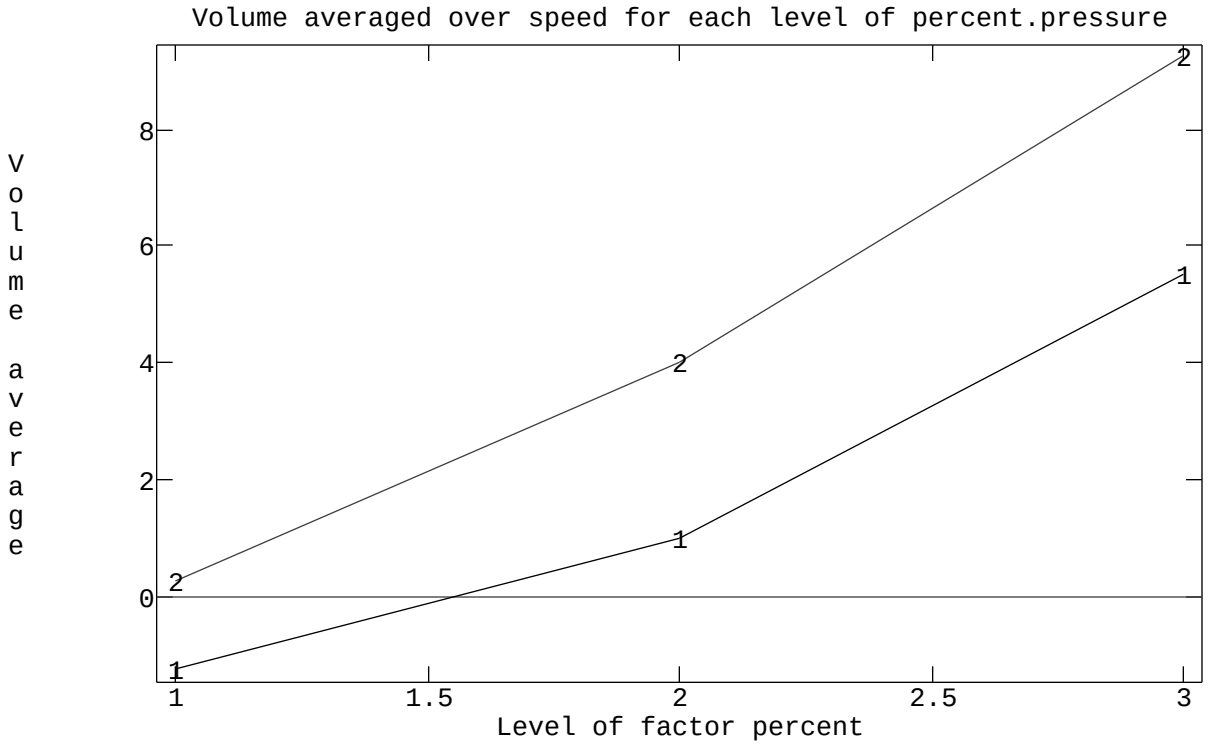
(See Sec. 3.16 for the use of a by-variable with `contrast()`.) Use macro `colplot` to look at the interaction effects.

```
Cmd> colplot(coefs("percent.pressure"),\
title:"Interaction plot of percent.pressure",\
xlab:"Level of factor percent",ylab:"Interaction effects")
```



This interaction plot certainly shows interaction! It is harder to see this if we just plot the average responses:

```
Cmd> colplot(tabs(volume,percent,pressure)$mean,title:\
"Volume averaged over speed for each level of percent.pressure",\
xlab:"Level of factor percent",ylab:"Volume average")
```



We see volume increases with percent, but with slightly different slopes for the two levels of pressure.

```
Cmd> coefs()[vector(2,3,4)] # get coefficients for 3 terms
percent
(1)       -3.625       -0.625         4.25
pressure
(1)        -1.375         1.375
percent.pressure
(1,1)        0.625       -0.625
(2,1)       -0.125        0.125
(3,1)         -0.5          0.5
```

It appears as though a linear by linear interaction might explain what is going on. Let's see if it does. Here we use outer() (Sec. 3.16) to compute the necessary contrast coefficients for the linear by linear contrast.

```
Cmd> concoefs <- outer(vector(-1,0,1),vector(-1,1)); concoefs
(1,1)           1          -1
(2,1)          -0           0
(3,1)          -1           1
```

```
Cmd> contrast("percent.pressure",concoefs)
component: estimate
(1)         2.25
component: ss
(1)       5.0625
component: se
(1)      0.82736
```

Since the entire `pressure` by `percent` interaction sum of squares is 5.25, we see this is almost entirely accounted for by this single degree of freedom for linear by linear interaction.

10.13 **Factorial designs with confounding** A basic rule for computing ANOVA tables for confounded factorial designs is to include a term for blocks in the model *preceding* the factorial terms. This works for both complete or partial confounding. The degrees of freedom between blocks may be broken up into replicates and blocks within replicates if so desired.

Our example is Example 10-3 page 319 of Montgomery. It is a partially confounded 2^3 factorial in two replicates, where ABC is confounded in replicate 1 and AB is confounded in replicate 2. *Warning* Don't try to use `rep` as a factor name, since `rep()` is a function.

```
Cmd> y <- vector(-3,2,2,1, 0,-1,-1,6, -1,0,3,5, 1,0,1,1)
Cmd> a <- factor(vector(1,2,2,1, 2,1,1,2, 1,1,2,2, 2,1,2,1))
Cmd> b <- factor(vector(1,2,1,2, 1,2,1,2, 1,1,2,2, 1,2,1,2))
Cmd> c <- factor(vector(1,1,2,2, 1,1,2,2, 1,2,1,2, 1,1,2,2))
Cmd> repl <- factor(rep(run(2),rep(8,2)))
Cmd> blk <- factor(rep(vector(1,2,1,2),rep(4,4)))
Cmd> print(format:"3.0f",hconcat(a,b,c,repl,blk,y)')
MATRIX: Rows are a, b, c, repl, blk, y
 (1,1)  1   2   2   1   2   1   1   2   1   1   2   2   2   1   2   1
 (2,1)  1   2   1   2   1   2   1   2   1   1   2   2   1   2   1   2
 (3,1)  1   1   2   2   1   1   2   2   1   2   1   2   1   1   2   2
 (4,1)  1   1   1   1   1   1   1   1   2   2   2   2   2   2   2   2
 (5,1)  1   1   1   1   2   2   2   2   1   1   1   1   2   2   2   2
 (6,1) -3   2   2   1   0  -1  -1   6  -1   0   3   5   1   0   1   1
```

```
Cmd> anova("y=repl/blk+a*b*c# blocks nested within replicates",\
    fstats:T)
Model used is y=repl/blk+a*b*c# blocks nested within replicates
WARNING: summaries are sequential
               DF           SS           MS            F      P-value
CONSTANT        1           16           16     21.33333    0.0057416
repl            1            1            1      1.33333       0.3004
repl.blk        2          2.5         1.25      1.66667      0.27885
a               1           36           36     48.00000   0.00096125
b               1        20.25        20.25     27.00000    0.0034782
a.b             1          0.5          0.5      0.66667      0.45135
c               1        12.25        12.25     16.33333    0.0099085
a.c             1         0.25         0.25      0.33333      0.58872
b.c             1            1            1      1.33333       0.3004
a.b.c           1          0.5          0.5      0.66667      0.45135
ERROR1          5         3.75         0.75
```

Here we have blocks nested within replicates plus the factorial terms. Of course, we only have half information on the partially confounded terms `a.b` and `a.b.c`. Thus the estimated variance of `a.b` and `a.b.c` effects is twice the estimated variance of the other effects. Here we compute the estimated variances of the `a`, `b`, and `a.b` effects (see Sec. 3.13).

```
Cmd> secoefs(coefs:F)[vector(4,5,6)]^2 # variances of some effects
component: a
(1)     0.046875     0.046875
component: b
(1)     0.046875     0.046875
component: a.b
(1,1)      0.09375      0.09375
(2,1)      0.09375      0.09375

Cmd> 0.09375/0.046875 # var[a.b effects] = 2*var[a or b effects]
(1)            2
```

10.14 **Fractional factorial designs** Fractional factorial designs may be analyzed just like an ordinary factorial. Any effects that are aliased to preceding effects have zero degrees of freedom.

Our example is from Table 11-3 of Montgomery. The data are from a 2^{4-1} fractional factorial with defining relation I=ABCD.

```
Cmd> a <- factor(rep(run(2),4))

Cmd> b <- factor(rep(rep(run(2),rep(2,2)),2))

Cmd> c <- factor(rep(run(2),rep(4,2)))

Cmd> d <- factor(vector(1,2,2,1,2,1,1,2))

Cmd> y <- vector(45,100,45,65,75,60,80,96)
```

```
Cmd> print(format:"3.0f",hconcat(a,b,c,d,y)')
MATRIX: Rows are a, b, c, d and y
(1,1)   1   2   1   2   1   2   1   2
(2,1)   1   1   2   2   1   1   2   2
(3,1)   1   1   1   1   2   2   2   2
(4,1)   1   2   2   1   2   1   1   2
(5,1)  45 100  45  65  75  60  80  96

Cmd> anova("y=a+b+c+d # Main effects model",fstat:T)
Model used is y=a+b+c+d # Main effects model
                DF          SS          MS           F      P-value
CONSTANT         1       40044       40044    85.29180    0.0026858
a                1         722         722     1.53781       0.3031
b                1         4.5         4.5     0.00958      0.92819
c                1         392         392     0.83493      0.42823
d                1       544.5       544.5     1.15974       0.3604
ERROR1           3      1408.5       469.5
```

The absence of a "`WARNING: summaries are sequential`" message reflects the fact that this is recognized by MacAnova to be a balanced main effects design. See Sec. 3.9.

```
Cmd> anova("y=d+a*b*c") # all main effects + a.b + a.c +b.c + a.b.c
Model used is y=d+a*b*c
WARNING: summaries are sequential
                DF          SS          MS
CONSTANT         1       40044       40044
d                1       544.5       544.5
a                1         722         722
b                1         4.5         4.5
a.b              1           2           2
c                1         392         392
a.c              1       684.5       684.5
b.c              1         722         722
a.b.c            0           0   undefined
ERROR1           0           0   undefined
```

We see that `a.b.c` is completely confounded and that the error term in the first analysis is made up of `a.b`, `a.c`, and `b.c` (and confounded effects).

You can get the terms in a possibly more natural order using the "pseudo-power" model construction (Sec. 3.4). Because of balance, the sums of squares are the same.

```
Cmd> anova("y=d+(a+b+c)^3") # same model n a different order
Model used is y=d+(a+b+c)^3
WARNING: summaries are sequential
                DF          SS          MS
CONSTANT         1       40044       40044
d                1       544.5       544.5
a                1         722         722
b                1         4.5         4.5
c                1         392         392
a.b              1           2           2
a.c              1       684.5       684.5
b.c              1         722         722
a.b.c            0           0   undefined
ERROR1           0           0   undefined
```

10.15 **Split plot designs** Split plots are factorial experiments with a randomization restriction wherein the levels of one factor are assigned at random to *groups* of units, and the levels of the second factor are assigned at random to units *within* groups. The groups of units and the units with a group are often called *whole plots* and *subplots*, respectively, and the factors are referred to as *whole plot factors* and *subplot factors*. This reflects the use of this design in agriculture, where often it is inconvenient or impossible to apply the treatment associated with the whole plot factor to units as small as subplot units.

There are at least two slightly different flavors of split plot designs. The levels of the whole plot factor may be assigned to the whole plots completely randomly, or, when the whole plots are further grouped into blocks or replicates, the whole plot factor may be assigned to whole plots as in a randomized complete (or even incomplete) block design. In the first case, the usual error term for inference about the effects of the whole plot factor is the replicate-within-factor effect, while in the second case it is the factor by replicate interaction. It is conventional to label this term in the ANOVA table as an error term, in addition to the final error term which is used in inference on subplot factor effects and interactions involving subplot factors.

The example data are a subset of Table 13-7 of Montgomery. The whole plot factor has three levels and the subplot factor has four. The whole plot factor levels were assigned to whole plots as in a randomized block design with two replicates.

```
Cmd> repl <- factor(rep(run(2),rep(12,2))) # replicate

Cmd> whole <- factor(rep(rep(run(3),rep(4,3)),2))# whole plot factor

Cmd> sub <- factor(rep(run(4),6)) # subplot factor

Cmd> y <- vector(30,35,37,36,34,41,38,42,29,26,33,36,\
28,32,40,41,31,36,42,40,31,30,32,40)

Cmd> anova("y=repl+whole+E(repl.whole)+sub+whole.sub",fstat:T)
Model used is y=repl+whole+E(repl.whole) + sub + whole.sub
               DF           SS           MS            F      P-value
CONSTANT        1        29400        29400   4126.31579   0.00024226
repl            1          1.5          1.5      0.21053      0.69139
whole           2       138.25       69.125      9.70175     0.093443
ERROR1          2        14.25        7.125      1.20423      0.34401
sub             3       266.33       88.778     15.00469   0.00075634
whole.sub       6       58.417       9.7361      1.64554      0.24071
ERROR2          9        53.25       5.9167
```

ERROR1 and ERROR2 are the whole plot and split plot error terms respectively.

The standard errors of the whole plot effects and contrasts among them should be computed using the whole plot error mean square, while the standard errors of the subplot effects and the interaction effects should be computed using the subplot error mean square. Unless instructed otherwise, secoefs() always uses the last error term. You can use keyword errorterm with both secoefs() and contrast() to specify a the error mean square to be used in computing standard errors. See Sec. 3.13 and 3.16.

```
Cmd> secoefs("whole",errorterm:"ERROR1") # use whole plot error
component: coefs
(1)       -0.125            3       -2.875
component: se
(1)      0.77055      0.77055      0.77055
```

Without `errorterm:"ERROR1"`, you get different, erroneous standard errors:

```
Cmd> secoefs("whole",coefs:F) # by default is uses subplot error
(1)      0.70218      0.70218      0.70218    Not correct
```

```
Cmd> contrast("whole",vector(0,1,-1),errorterm:"ERROR1")
component: estimate
(1)        5.875
component: ss
(1)       138.06
component: se
(1)       1.3346
```

```
Cmd> secoefs("sub") # uses subplot error by default which is o.k.
component: coefs
(1)         -4.5      -1.6667            2       4.1667
component: se
(1)      0.85999      0.85999      0.85999      0.85999
```

10.16 **Multivariate analysis of variance** When you have several response variables or measurement on individuals in different groups or undergoing different treatments, it is often of interest to compare the the entire mean vectors for the groups. One approach to this is the one-way MANOVA (multivariate analysis of variance). We illustrate it with the famous Fisher iris data which we read from file `macanova.dat` using function `matread()`.

```
Cmd> y <- matread("macanova.dat","irisdata")
irisdata       150     5 FORMAT
) Data from Johnson & Wichern, Ed. 3, Table 11.5, p. 566
) Measurements on petals of 4 varieties of Iris. Originally published
) in R. A. Fisher, The use of multiple measurements in taxonomic
) problems, Annals of Eugenics, 7 (1936) 179-198
) Col. 1: variety number (1=I.setosa, 2=I.versiColor, 3=I. virginica)
) Col. 2: x1 = sepal length
) Col. 3: x2 = sepal width
) Col. 4: x3 = petal length
) Col. 5: x4 = petal width
) Rows 1-50:    group 1 = Iris setosa
) Rows 51-100:  group 2 = Iris versicolor
) Rows 101-150: group 3 = Iris virginica
```

```
Cmd> varieties <- factor(y[,1]); y <- y[,-1]
```

```
Cmd> y <- matrix(y,labels:structure("@",vector("Sepal Len",\
"Sepal Wid","Petal Len", "Petal Wid"))) # add labels, Sec. 8.4.1

Cmd> manova("y=varieties")
Model used is y=varieties
WARNING: summaries are sequential
                          SS and SP Matrices
               DF
CONSTANT        1
             Sepal Len    Sepal Wid    Petal Len    Petal Wid
Sepal Len       5121.7       2679.8       3293.9       1051.2
Sepal Wid       2679.8       1402.1       1723.4       550.01
Petal Len       3293.9       1723.4       2118.4       676.06
Petal Wid       1051.2       550.01       676.06       215.76
varieties       2
             Sepal Len    Sepal Wid    Petal Len    Petal Wid
Sepal Len       63.212      -19.953       165.25       71.279
Sepal Wid      -19.953       11.345       -57.24      -22.933
Petal Len       165.25       -57.24        437.1       186.77
Petal Wid       71.279      -22.933       186.77       80.413
ERROR1         147
             Sepal Len    Sepal Wid    Petal Len    Petal Wid
Sepal Len       38.956        13.63       24.625        5.645
Sepal Wid        13.63       16.962       8.1208       4.8084
Petal Len       24.625       8.1208       27.223       6.2718
Petal Wid        5.645       4.8084       6.2718       6.1566
```

The diagonals of these matrices are the ANOVA sums of squares for each variable. These matrices are saved in array SS. SS[2,,] is the hypothesis matrix *H* and SS[3,,] is the error matrix *E*. All the usual test statistics for equality of means can be computed from these *H* and *E*. In particular we can compute univariate *F*-statistics.

```
Cmd> h <- SS[2,,];e <- SS[3,,]#hypothesis (between) & error (within)

Cmd> f <- (diag(h)/DF[2])/(diag(e)/DF[3])

Cmd> f # f-statistics
(1)       119.26        49.16       1180.2       960.01

Cmd> 1 - cumF(f,DF[2],DF[3]) # P values
(1)            0            0            0            0
```

All the *P* values are extremely small, indicating there is strong evidence against the null hypothesis that all three varieties have the same means.

You can also request output in other forms. If you don't want to see the matrices, use sssp:F as an argument.

```
Cmd> manova(,sssp:F) # uses most recent model
Model used is y=varieties
WARNING: summaries are sequential
                          SS and SP Matrices
                DF
CONSTANT         1
                     Type 'SS[1,,]' to see SS/SP matrix
varieties        2
                     Type 'SS[2,,]' to see SS/SP matrix
ERROR1         147
                     Type 'SS[3,,]' to see SS/SP matrix
Cmd> SS[2,,] # varieties matrix
                       Sepal Len   Sepal Wid   Petal Len   Petal Wid
varieties Sepal Len       63.212     -19.953      165.25      71.279
          Sepal Wid      -19.953      11.345      -57.24     -22.933
          Petal Len       165.25      -57.24       437.1      186.77
          Petal Wid       71.279     -22.933      186.77      80.413
```

This option becomes the default when each row of a matrix would require more than one line. In that case, you can force the printing of the matrices by `sssp:T`.

Alternatively, you can request univariate analysis of each variable by using keyword phrases `pvals:T` or `fstat:T`.

```
Cmd> manova(,fstat:T)# to suppress P values, use pvals:F
Model used is y=varieties
WARNING: summaries are sequential
                DF           SS           MS            F      P-value
CONSTANT         1
 Sepal Len               5121.7       5121.7  19326.50528            0
 Sepal Wid               1402.1       1402.1  12151.14260            0
 Petal Len               2118.4       2118.4  11439.11809            0
 Petal Wid               215.76       215.76   5151.66322            0
varieties        2
 Sepal Len               63.212       31.606    119.26450            0
 Sepal Wid               11.345       5.6725     49.16004            0
 Petal Len                437.1       218.55   1180.16118            0
 Petal Wid               80.413       40.207    960.00715            0
ERROR1         147
 Sepal Len               38.956      0.26501
 Sepal Wid               16.962      0.11539
 Petal Len               27.223      0.18519
 Petal Wid               6.1566     0.041882
```

This arranges the output by terms. To get a complete ANOVA table for each variable separately, you can use `byvar:T`, possibly together with `fstat:T` or `pvals:T`.

```
Cmd> manova("y=varieties",byvar:T,fstat:T)
Model used is y=varieties
WARNING: summaries are sequential
                             Sepal Len
                DF           SS           MS            F      P-value
CONSTANT         1       5121.7       5121.7  19326.50528            0
varieties        2       63.212       31.606    119.26450            0
ERROR1         147       38.956      0.26501
```

```
                              Sepal Wid
               DF          SS           MS            F      P-value
CONSTANT        1      1402.1       1402.1  12151.14260            0
varieties       2      11.345       5.6725     49.16004            0
ERROR1        147      16.962      0.11539
                              Petal Len
               DF          SS           MS            F      P-value
CONSTANT        1      2118.4       2118.4  11439.11809            0
varieties       2       437.1       218.55   1180.16118            0
ERROR1        147      27.223      0.18519
                              Petal Wid
               DF          SS           MS            F      P-value
CONSTANT        1      215.76       215.76   5151.66322            0
varieties       2      80.413       40.207    960.00715            0
ERROR1        147      6.1566     0.041882
```

10.16.1 **Multivariate test statistics** There are several test statistics that are commonly used for testing MANOVA hypotheses. All are based on some form of comparison of a hypothesis matrix H with an error matrix E. On of the most important is the modified log likelihood ratio statistic, which has a large sample chi-squared distribution under the null hypothesis that the groups mean vectors are the same, assuming multivariate normality with common covariance matrix. Variables `fh` and `fe` defined in the following lines are the hypothesis and error degrees of freedom and `p` is the number of variables. `m1` is a scaling factor designed to make the chi-squared approximation as good as possible.

```
Cmd> lambda <- det(e)/det(e+h); lambda# Wilk's lambda
(1)     0.023439

Cmd> fe <- DF[3]; fh <- DF[2]; p <- ncols(y); vector(fe, fh, p)
(1)          147            2            4

Cmd> m1 <- fe - (p - fh + 1)/2

Cmd> w <- -m1 * log(lambda); vector(m1,w)
(1)        145.5       546.12

Cmd> df <- p*fh

Cmd> vector(df,1-cumchi(w,df))#-m1*log(lambda)~chisq(p*fh)
(1)            8            0
```

Because the value 546.12 for the statistic is so large, the P value computed using the χ^3_8 approximation is 0. Other test statistics in use are $T = m_2 \mathrm{tr} E^{-1}H$ and $V = m_3 \mathrm{tr}(E + H)^{-1}H$, where $m_2 = f_e - p + 1$ and $m_3 = f_h + f_e$, both of which also are approximately distributed as chi-squared with $p \times f_h$ degrees of freedom under the null hypothesis.

```
Cmd> m2 <- fe - p + 1; m3 <- fe + fh; vector(m2, m3)
(1)          144          149

Cmd> m2*trace(solve(e, h)) # T
(1)       4676.7

Cmd> m3*trace(solve(h+e,h)) # V
(1)       177.59
```

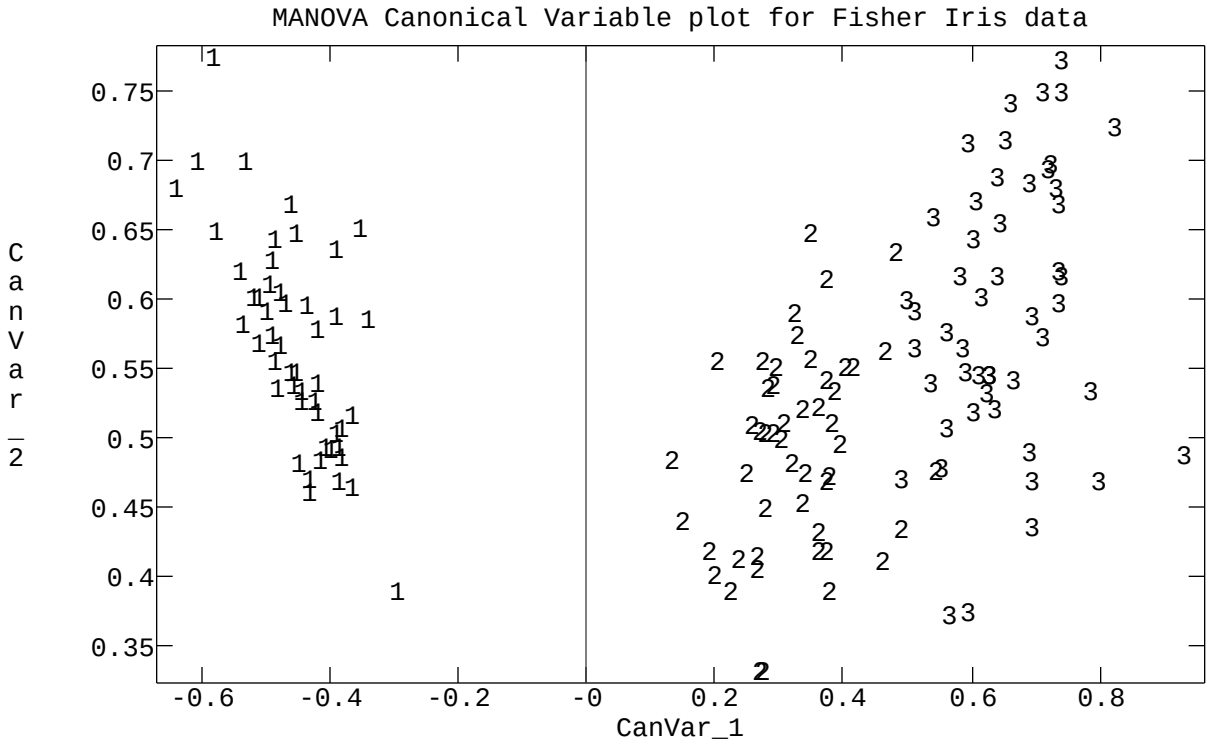
These test statistics can also be computed from what are sometimes called the *relative eigenvalues* and *relative eigenvectors* of H relative to E (see Sec. 6.2.3). If vector u satisfies $Hu = \lambda Eu$, then u is a relative eigenvector with relative eigenvalue λ. Alternatively, u and λ are an ordinary eigenvector and eigenvalue pair of the *non-symmetric* matrix $E^{-1}H$. If H and E are p by p symmetric matrices with E positive definite, there are p real eigenvalues $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_p$, with associated eigenvectors $u_1, u_2, ..., u_p$, computable by function `releigen()` (`releigenvals()` computes just the eigenvalues). For example $\log(\det(E)/\det(H+E)) = -\Sigma_i \log(1 + \lambda_i)$, $\mathrm{tr}E^{-1}H = \Sigma_i \lambda_i$, and $\mathrm{tr}(E + H)^{-1}H = \Sigma_i \{\lambda_i / (1 + \lambda_i)\}$.

```
Cmd> eigs <- releigen(h,e); eigs # relative eigen things
component: values
(1)       32.192      0.28539   7.6558e-15   1.8202e-16
component: vectors
                 (1)          (2)          (3)          (4)
Sepal Len   -0.068406    0.0019879      0.13207      0.22724
Sepal Wid    -0.12656      0.17853     -0.20587    -0.080981
Petal Len     0.18155    -0.076864     -0.23642    -0.076453
Petal Wid      0.2318      0.23417      0.37333     -0.04695

Cmd> m1 * sum(log(1 + eigs$values)) # log likelihood ratio
(1)       546.12   Same as computed above

Cmd> m2 * sum(eigs$values) # T
(1)       4676.7   Same as computed above

Cmd> m3 * sum(eigs$values/(1+eigs$values)) # V
(1)       177.59   Same as computed above
```

10.16.2 **MANOVA canonical variables** The linear combinations $z_j = u_j'y$ of all p variables $y_1, y_2, ..., y_p$ with coefficients are taken from the j^{th} relative eigenvector are the MANOVA *canonical variables*. z_1 is the linear combination that has the largest possible univariate F statistic for testing the null hypothesis and z_2 is the linear combination that has the next largest F and whose residuals are uncorrelated with the residuals of z_1. Often a scatter plot of the first two canonical variables is informative.

```
Cmd> z <- y %*% eigs$vectors # compute canonical variables
```

```
Cmd> chplot(CanVar_1:z[,1],CanVar_2:z[,2],varieties,\
title:"MANOVA Canonical Variable plot for Fisher Iris data")
```



Among other things this plot shows *I. setosa* is quite distinct, both in mean and covariance matrix, from *I. versicolor* and *I. virginica*.

10.17 **Repeated measures designs** Repeated measures designs may be analyzed using multivariate, and in some cases, univariate techniques. This section describes a univariate technique using an adjusted analysis of variance.

One of the simplest repeated measure designs is similar to a split plot design (see Sec. 10.15), except that the subplot treatment is not fully randomized. The individual cases correspond to whole plots and the factor defining the different repeated measurements corresponds to the subplot factor. This is the case, for example, when subjects are randomly assigned to one of several treatment groups, and then a certain response, say systolic blood pressure, is measured at a number different times, the same times for each subject. Ordinarily you would be interested in the effect of the treatment, and the effect of elapsed time since treatment, as well as the interaction of these two factors. The subjects are analogous to whole plots, and the different measurement times to subplots, but clearly it is impossible to randomly assign levels of the "subplot" or *within subjects* factor. Thus there is a fundamental violation of the assumptions underlying the usual ANOVA.

Sometimes the "subplot" treatment has a factorial structure of its own. For example, in the blood pressure study situation, at each time point both systolic and diastolic blood pressure might be measured, so that each measurement on a subject is categorized in two ways, by time and by type of blood pressure. In the univariate analysis of a repeated measures design, we assume that each subplot factor (and

interaction) itself has a random interaction with whole plots. This induces a separate error term for each repeated measure effect. In the language of repeated measures, whole plot treatments are "grouping" factors, and subplot treatments are "trial" factors.

As an example, consider the repeated measures design described in Winer (1971, p. 546). Six subjects were randomly assigned to one of two noise levels (the grouping factor analogous to a whole plot factor). Each was asked to calibrate three dials during each of three time periods (`dial` and `period` are crossed trial factors and are analogous to subplot factors). The response is a measure of the `accuracy` of calibration. Here is a fairly standard univariate ANOVA of these repeated measures data, with a separate error term for each main effect and interaction.

```
Cmd> accuracy <- vector(45,35,60,50,42,56, 53,41,65,48,45,60,\
60,50,75,61,55,77, 40,30,58,25,30,40, 52,37,54,34,37,39,\
57,47,70,51,43,57, 28,25,40,16,22,31, 37,32,47,23,27,29,\
46,41,50,35,37,46) # accuracy[subjects,period,dial]

Cmd> noise <- factor(rep(rep(run(2),rep(3,2)),9))

Cmd> period <- factor(rep(run(3),rep(18,3)))

Cmd> dial <- factor(rep(rep(run(3),rep(6,3)),3))

Cmd> subjinnoise <- factor(rep(run(3),18))

Cmd> anova("accuracy=noise + E(subjinnoise.noise) +
 period*noise + E(subjinnoise.period.noise) +
 dial*noise + E(subjinnoise.dial.noise) +
 dial*period*noise",fstat:T)
Model used is accuracy=noise + E(subjinnoise.noise) +
 period*noise + E(subjinnoise.period.noise) +
 dial*noise + E(subjinnoise.dial.noise) +
 dial*period*noise"
                  DF          SS           MS            F      P-value
CONSTANT           1  1.0587e+05   1.0587e+05    169.99349   0.00019973
noise              1      468.17       468.17      0.75174      0.43484
ERROR1             4      2491.1       622.78     21.21097   0.00025662
period             2      3722.3       1861.2     63.38884   1.2413e-05
noise.period       2         333        166.5      5.67077     0.029268
ERROR2             8      234.89       29.361      2.22526      0.13944
dial               2      2370.3       1185.2     89.82316   3.3037e-06
noise.dial         2      50.333       25.167      1.90737      0.21022
ERROR3             8      105.56       13.194      1.66084      0.18446
period.dial        4      10.667       2.6667      0.33566      0.84992
noise.period.dial  4      11.333       2.8333      0.35664      0.83567
ERROR4            16      127.11       7.9444
```

The `ERROR4` SS is precisely the `subjinnoise.dial.noise.period` SS.

This output illustrates one feature of the `fstats:T` option. Each *F*-statistic is the ratio of the `MS` value on its line and the `MS` value on the next `ERROR` line. Thus the *F* for `dial` is 1185.2 / 13.194 = 89.829.

If you did not use `fstat:T`, the *F* statistic for testing `noise` by `period` interaction might be computed as follows.

```
Cmd> ms <- SS/DF # compute mean squares
```

```
Cmd> df <- DF;termnames <- TERMNAMES
Cmd> J1 <- termnames=="noise.period"
Cmd> J2 <- termnames=="ERROR2"
Cmd> f <- ms[J1]/ms[J2];f
(1)       5.6708   F-statistic
Cmd> 1 - cumF(f,df[J1],df[J2])
(1)     0.029268   Its P value
```

This univariate analysis of repeated measures assumes that certain covariance assumptions are met. In particular, it assumes that the within subject correlations between the responses for each combination of within subject factors are the same. If these assumptions fail, the computed *P* value will tend to be inappropriately small. Although there is a purely multivariate approach to compute test statistics for the various hypotheses, there are conceptual advantages to be able to use the ANOVA computations, adjusting the tests so that the *P* values are accurate. Moreover, when the number of subjects is smaller than the number of measurements, as in this case, the usual multivariate statistics cannot be computed. Geisser and Greenhouse (1958) showed how to modify the degrees of freedom used for computing *P* values to improve accuracy. For each *F*-statistic, a matrix *M* is computed, and then the numerator and denominator degrees of freedom are multiplied by the factor $(\mathrm{tr}\, M)^2 / (f\, \mathrm{tr}(M^2))$, where *f* is the degrees of freedom associated with a trial factor in the numerator. Fundamental to the computations is a matrix *S* estimating the within-subject variances and covariances of the responses. After transforming the response *vector* to a *matrix* with each row containing the data for a single subject and each column corresponding to a `dial` by `time` period combination, we use `manova()` to compute *S*.

```
Cmd> accuracy1 <- matrix(accuracy,6);noise1 <- factor(noise[run(6)])
Cmd> manova("accuracy1=noise1") # accuracy1 is 6 by 9
Model used is accuracy1=noise1
WARNING: summaries are sequential
NOTE: SS/SP matrices suppressed because of size; use 'manova(,sssp:T)'
                          SS and SP Matrices
                DF
CONSTANT         1
                    Type 'SS[1,,]' to see SS/SP matrix
noise1           1
                    Type 'SS[2,,]' to see SS/SP matrix
ERROR1           4
                    Type 'SS[3,,]' to see SS/SP matrix
Cmd> list(SS,DF,RESIDUALS) # see shapes of side-effect variables
DF              REAL   3
RESIDUALS       REAL   6     9        Matrix
SS              REAL   3     9     9  Array
```

All output from `manova()` could be suppressed by using keyword phrase `silent:T`.

Each `SS[i,,]` for `i` = 1, 2, 3 is a 9 by 9 matrix, generalizing the usual sum of squares in an analysis of variance. We compute the estimated covariance matrix as

SS[3,,]/DF[3], save it as s and then use it to compute adjustments to the degrees of freedom for the the various ANOVA *F*-tests.

```
Cmd> s <- matrix(SS[3,,],9)/DF[3] # estimated covariance matrix
```

The first three responses for each subject (first 3 columns of accuracy1) are for period 1, followed by those for period 2, etc. Thus matrix vtot computed below is the estimated covariance matrix of the period totals. From it we can compute an adjustment factor to adjust the degrees of freedom in the *F*-tests for period and noise.period.

```
Cmd> period1 <- rep(run(3),rep(3,3));tmpx <- 1*(period1==run(3)')
Cmd> hconcat(period1,tmpx) # tmpx are dummy variables for period
(1,1)           1           1           0           0
(2,1)           1           1           0           0
(3,1)           1           1           0           0
(4,1)           2           0           1           0
(5,1)           2           0           1           0
(6,1)           2           0           1           0
(7,1)           3           0           0           1
(8,1)           3           0           0           1
(9,1)           3           0           0           1
Cmd> vtot <- tmpx' %*% s %*% tmpx; e <- (1/3)*rep(1,3)%*%rep(1,3)'
Cmd> print(vtot,e)
vtot: est covar matrix of period totals
(1,1)      1025.8      812.33      524.83
(2,1)      812.33      690.83      443.08
(3,1)      524.83      443.08      327.83
e:
(1,1)     0.33333     0.33333     0.33333
(2,1)     0.33333     0.33333     0.33333
(3,1)     0.33333     0.33333     0.33333
Cmd> m <- vtot - e %*% vtot # m is 3 by 3
Cmd> adjustment <- trace(m)^2/(2*trace(m %*% m)); adjustment
(1)       0.6476
```

This is really a short cut method for the following computation, involving the computation of a 9 by 9 matrix tmpx1 which projects onto the period main effects space.

```
Cmd> tmpx1 <- (1/3) * tmpx %*% tmpx' - (1/9)*rep(1,9)%*%rep(1,9)'
Cmd> m1 <- tmpx1 %*% s # (projection matrix) %*% (covariance matrix)
Cmd> trace(m1)^2/(2*trace(m1%*%m1))
(1)       0.6476
```

The Greenhouse-Geisser correction is 0.6476 and it applies to both the period main effect and the noise.period interaction. No matter what adjustment were made, the former is huge so we use the adjustment in testing the latter.

```
Cmd> dfad1 <- adjustment*df[J1] # adjust DF
Cmd> dfad2 <- adjustment*df[J2]
```

```
Cmd> 1 - cumF(f,dfad1,dfad2) # adjusted P value
(1)      0.05694
```

We multiplied both the numerator and denominator degrees of freedom in the *F*-test by .6476 to get an improved *P* value. We would use the same adjustment factor to adjust the degrees of freedom in the `period` main effect *F*-test.

Now we repeat the process to obtain the Greenhouse-Geisser adjustment for the `dial` main effect `noise.dial` interaction. Now `vtot` is the estimated covariance matrix of the dial totals for each individual.

```
Cmd> dial1 <- rep(run(3),3);tmpy <- 1*(dial1==run(3)')
Cmd> hconcat(dial1,tmpy)
(1,1)            1            1            0            0
(2,1)            2            0            1            0
(3,1)            3            0            0            1
(4,1)            1            1            0            0
(5,1)            2            0            1            0
(6,1)            3            0            0            1
(7,1)            1            1            0            0
(8,1)            2            0            1            0
(9,1)            3            0            0            1
Cmd> vtot <- tmpy %c% s %*% tmpy
Cmd> m <- vtot - e %*% vtot # same e == (1/3)*rep(1,3) %*% rep(1,3)'
Cmd> adjustment <- trace(m)^2/(2*trace(m %*% m)); adjustment
(1)      0.91707
```

Alternatively,

```
Cmd> tmpy1 <- (1/3) * tmpy %*% tmpy' - (1/9)*rep(1,9)%*%rep(1,9)'
Cmd> m1 <- tmpy1 %*% s
Cmd> trace(m1)^2/(2*trace(m1%*%m1))
(1)      0.91707
```

There is only a slight correction to the degrees of freedom for the `dial` and `noise.dial` *F*-tests. To get `dial` by `period` interaction, we need to remove everything that looks like main effects of dials or periods. We do this by multiplying through by the projection onto the interaction space of dials and periods. `I9` is the 9 by 9 identity matrix. These terms have 4 degrees of freedom.

```
Cmd> I9 <- dmat(9,1) # 9 by 9 diagonal matrix with 1's on diagonal
Cmd> tmpxy <- (I9 - tmpx%*%tmpx'/3) %*% (I9 - tmpy%*%tmpy'/3)
Cmd> # or tmpxy <- I9-tmpx1-tmpy1-(1/9)*rep(1,9) %*% rep(1,9)'
Cmd> m1 <- tmpxy %*% s
Cmd> adjustment <- trace(m1)^2/(4*trace(m1%*%m1)); adjustment
(1)      0.51342
```

The correction to `period.dial` is to approximately halve the degrees of freedom. The same correction would apply to the `noise.period.dial` interaction.

10.18 **Logistic regression** Let's look at the example data in Table 12.3 of Weisberg. This data gives the number of responses of cows to various shocks, with 70 trials at each of 6 currents (0 to 5 milliamperes). Following Weisberg, we ignore the fact that only seven cows were used and that therefore not all responses to shocks can be considered to be independent. We choose a sequential analysis of deviance (Sec. 4.2.1).

```
Cmd> y <- vector(0,9,21,47,60,63) # counts

Cmd> n <- 70 # or n <- rep(70,6)

Cmd> current <- run(0,5)

Cmd> logistic("y=current",n,inc:T,pvals:T)
Model used is y=current
WARNING: summaries are sequential
                DF     Deviance         MDev      P-value
CONSTANT         1      0.95274      0.95274      0.32902
current          1       241.13       241.13            0
ERROR1           4       9.3526       2.3382     0.052865

Cmd> regcoefs() # see Sec. 3.13.1
NOTE: standard errors assume scale parameter is 1
                 Coef       StdErr            t
CONSTANT       -3.301       0.3238      -10.195
current        1.2459      0.11193       11.132
```

You can use macro `regcoefs` here because all the variables on the right hand side of the model are variates. The column labelled `t` is usually compared to normal probability points (for example, `invnor(1-.05/2)`) to test whether the coefficient is 0. A better test statistic is based on the signed square root of the incremental deviance. Provided `inc:T` is an argument to `logistic()`, this deviance is the next to the last element of `SS`.

```
Cmd> sqrt(SS[2])*coefs(2)/abs(coefs(2))
(1)       15.528

Cmd> logistic("y=current-1+1",n,inc:T,silent:T) # make CONSTANT last

Cmd> sqrt(SS[2])*coefs(2)/abs(coefs(2))
(1)      -13.736
```

Variate `current` is highly significant, as can be seen from either the *t*-statistic for its coefficient or the square root deviance increment statistic. The almost significant `ERROR1` deviance ($P = .053$) suggests the possibility the dependence on `current` may not be linear or that the binomial model is not appropriate, perhaps because the lack of independence.

```
Cmd> logistic("y=P2(current)",n,inc:T,pvals:T) # quadratic model
Model used is y=P2(current)
WARNING: summaries are sequential
                DF     Deviance         MDev      P-value
CONSTANT         1      0.95274      0.95274      0.32902
{current}        1       241.13       241.13            0
{(current)^2}    1       5.5426       5.5426     0.018559
ERROR1           3         3.81         1.27      0.28273
```

```
Cmd> regcoefs()
NOTE: standard errors assume scale parameter is 1
                     Coef       StdErr            t
CONSTANT          -4.4043      0.65897      -6.6836
{current}          2.2404      0.47723       4.6946
{(current)^2     -0.18078     0.079678      -2.2689
```

The quadratic term is quite significant and there is now no evidence of lack of fit.

10.19 **Poisson regression** Our example uses the data in Problem 16.17 on p. 830 of Devore and Peck (1993). This cross classifies 445 college students according to their drug use (never, occasional, regular) and their parents' alcohol and drug use (neither, one, both). We wish to know if the two classifications are independent.

```
Cmd> counts <- vector(141,68,17,54,44,11,40,51,19)

Cmd> parentuse <- factor(rep(run(3),rep(3,3))) # 1,1,1,2,2,2,3,3,3

Cmd> studentuse <- factor(rep(run(3),3)) # 1,2,3,1,2,3,1,2,3

Cmd> poisson("counts=studentuse + parentuse",inc:T,pvals:T)
Model used is counts=studentuse + parentuse
WARNING: summaries are sequential
                DF     Deviance         MDev      P-value
CONSTANT         1       2599.8       2599.8            0
studentuse       2       138.96        69.48            0
parentuse        2        57.38        28.69   3.4683e-13
ERROR1           4       22.254       5.5634   0.00017841

Cmd> secoefs("parentuse") # or secoefs(3)
NOTE: standard errors assume scale parameter is 1
component: coefs
(1)      0.48308     -0.24611     -0.23697
component: se
(1)     0.063214     0.074694     0.074508

Cmd> HII # "leverages", see Sec. 3.6
(1)      0.76776      0.68813      0.55984      0.64368      0.52151
(6)      0.32469      0.64474      0.52294       0.3267
```

The model `counts=studentuse + parentuse"` is additive in the log means which corresponds to the model of independence.

The small *P* value for `ERROR1` indicates significant lack of fit. Either the data are more variable than one would expect for Poisson data or the model that assumes independence of the two factors is incorrect. Lets look at the residuals and fitted values.

```
Cmd> matrix(WTDRESIDUALS,3) # weighted residuals in the log scale
(1,1)       1.9819     -0.46946      -2.3735
(2,1)      -1.6247      0.64478       1.6869
(3,1)      -1.4061     -0.15101       2.1658

Cmd> predtable() # table of fitted values
(1,1)       119.35       57.562        58.09
(2,1)       82.782       39.926       40.292
(3,1)        23.87       11.512       11.618
```

We see that alternate corners of the residual table are high, suggesting a single degree of

freedom interaction which is proportional to the product of main effect coefficients similar to the Tukey one degree of freedom for non-additivity. Let's try it out.

```
Cmd> onedof <- outer(coefs("parentuse"),coefs("studentuse"))

Cmd> #See Sec 3.16 for another use of outer()

Cmd> print(onedof) # display as table
onedof:
(1,1)      0.31807      0.14134     -0.45942
(2,1)     -0.16204    -0.072008      0.23405
(3,1)     -0.15603    -0.069336      0.22537

Cmd> onedof <- vector(onedof) # make a vector

Cmd> poisson("counts=studentuse+parentuse+onedof",inc:T,pvals:T)
Model used is counts=studentuse+parentuse+onedof
WARNING: summaries are sequential
                DF     Deviance         MDev      P-value
CONSTANT         1       2599.8       2599.8            0
studentuse       2       138.96        69.48            0
parentuse        2        57.38        28.69   3.4683e-13
onedof           1       19.238       19.238   1.1539e-05
ERROR1           3       3.0155       1.0052      0.38923
```

The single degree of freedom for a product interaction has accounted for almost all the interaction in the model leaving non-significant lack-of-fit.

Alternatively, for the first analysis, but not the second, we could use *ipf()* which uses iterative proportional fitting, but does not provide standard errors, and HII is incorrect. Since ipf() cannot analyze models with variates the analysis involving onedof cannot be done.

```
Cmd> ipf("counts=studentuse+parentuse",inc:T)
Model used is counts=studentuse+parentuse
                DF     Deviance         MDev
CONSTANT         1       2599.8       2599.8
studentuse       2       138.96        69.48
parentuse        2        57.38        28.69
ERROR1           4       22.254       5.5634

Cmd> secoefs("parentuse")
ERROR: standard errors not available after ipf()

Cmd> coefs("parentuse")
(1)      0.48308     -0.24611     -0.23697

Cmd> HII  # this is not correct; it sums to the right total, though
(1)      0.55556      0.55556      0.55556      0.55556      0.55556
(6)      0.55556      0.55556      0.55556      0.55556
```

10.20 **Robust regression** Now redo the multiple regression example we did in Sec. 10.4, but pretending that we had miscoded the eighth adsorption observation as 300 instead of 30. Look at how the ordinary regression changes and see how the robust regression obtains results similar to the ordinary regression on the original data.

```
Cmd> adsorption[8] <- 300
```

```
Cmd> regress("adsorption=iron+aluminum",pvals:T)
Model used is adsorption=iron+aluminum
                Coef       StdErr            t      P-Value
CONSTANT       29.819       62.529      0.47689      0.64369
iron         -0.26721      0.53277     -0.50155      0.62684
aluminum       1.3827       1.2795       1.0806      0.30525

N: 13,  MSE: 6175.4, DF: 10,  R^2: 0.13141
Regression F(2,10): 0.75648, P-value: 0.49439, Durbin-Watson: 2.0567
To see the ANOVA table type 'anova()'
```

Neither variable is significant and the overall regression F has fallen from 92.026 to 0.756. Moreover, the regression coefficients have changed a lot from -7.3507, 0.11273, and 0.349. In other words, we have an analysis that really tells us nothing. What do we get with robust regression? Something close to what we had before.

```
Cmd> robust("adsorption=iron+aluminum",fstats:T)
Model used is adsorption=iron+aluminum
WARNING: summaries are sequential
              DF          SS*          MS*           F*     P-value*
CONSTANT       1        12644        12644    688.97178   1.4839e-10
iron           1       3172.8       3172.8    172.88387   1.2313e-07
aluminum       1       626.13       626.13     34.11716   0.00016359
ERROR1        10       183.52       18.352
* ANOVA is approximate and should be interpreted with caution

Robust estimate of sigma: 4.5468
```

The approximate ANOVA table can be used much as an ordinary ANOVA table. For instance we can test the significance of `aluminum` by an approximate F-test. If we had not used `fstats:T`, we would compute this as follows:

```
Cmd> f <- (SS[3]/DF[3])/(SS[4]/DF[4]) # F-statistic

Cmd> vector(f,1-cumF(f,DF[3],DF[4]))# F-statistic & P value
(1)       34.117   0.00016359  Highly significant
```

If the errors were independent normal with variance σ^2, $\hat{\sigma}^2$. where $\hat{\sigma}$ is the estimated scale, is an approximately unbiased estimator of σ^2. Here $\hat{\sigma}^2 = 4.5468^2 = 20.67$, a little larger than the error mean square in the ANOVA.

```
Cmd> temp <- secoefs(byterm:F) # get coefs and their stderrs

Cmd> compnames(temp) # coefs(byterm:F) has 2 components
(1) "coefs"
(2) "se"

Cmd> coef <- vector(temp[1]);se <- vector(temp[2]);tstats <- coef/se

Cmd> hconcat(coef,se,tstats) # table of coeffs, std errors, t-stats
(1,1)      -6.1326       3.4086      -1.7992  CONSTANT
(2,1)     0.097178     0.029043        3.346  iron
(3,1)      0.40743      0.06975       5.8413  aluminum
```

Note the use of `byterm:F` with `secoefs()`. Normally `secoefs()` has one component for each term, with each component of `secoefs()` itself having two-components, `coefs` and `se`. With `byterm:F`, `secoefs()` has two components, `coefs` and `se`, each of which is a structure with one component for each term. See Sec. 3.13. Also note the

use of `vector()` with a structure argument, combining all the elements in a structure into a vector. You could also compute, say, `coef` by `vector(temp$coefs)` or `vector(temp$coefs$CONSTANT,temp$coefs$iron,temp$coefs$aluminum)`. If we had not used `byterm:F`, we would have to use either `vector(temp[1][1], temp[2][1], temp[3][1])` or `vector(temp$CONSTANT$coefs, temp$iron$coefs, temp$aluminum$coefs)`.

After `robust()`, the side effect vector `WTDRESIDUALS` contains modified residuals computed at the final stage of the iteration. When `abs(RESIDUALS[i])` $< c\hat{\sigma}$, `WTDRESIDUALS[i] = RESIDUALS[i]`; otherwise `WTDRESIDUALS[i]` $= \pm c\hat{\sigma}$, where $\hat{\sigma}$ is the robust estimate of scale and c is the truncation point used in the algorithm (see Sec. 4.3) whose default value is .75. Thus `WTDRESIDUALS/RESIDUALS` should be 1 for all "non-truncated" residuals.

```
Cmd> WTDRESIDUALS/RESIDUALS
 (1)            1            1            1            1      0.51868
 (6)            1            1      0.012877           1      0.32462
(11)            1            1       0.71154

Cmd> WTDRESIDUALS
 (1)      -1.0916      -1.4297      -0.43231       2.7117      -3.4105
 (6)     -0.16143      -2.7356        3.4105       2.6945      -3.4105
(11)      0.52653    -0.081364        3.4105
```

The values for truncated cases have been underlined. The value 3.4105 is not exactly equal to $c\hat{\sigma} = .75\hat{\sigma} = .75 \times 4.5468 = 3.4101$ because of incomplete convergence of the iterative algorithm. The ratios `WTDRESIDUALS/RESIDUALS` are not really weights but do indicate the importance given to each case in the final fit. Note that observation 8, the one that was contaminated with the enormous value, has a ratio near zero; three other points have ratios considerably less than 1.

As with `anova()` the sums of squares are computed sequentially so that only the significance of `aluminum` (the last term) can be tested by an F-test based on the table. To test the effect of `iron`, you would redo the analysis with model "`adsorption= aluminum+iron`" or use keyword phrase `marginal:T` with the original model (see Sec. 3.11).

```
Cmd> robust("adsorption=iron+aluminum",fstats:T,marginal:T)
Model used is adsorption=iron+aluminum
WARNING: SS are Type III sums of squares
               DF          SS*          MS*           F*     P-value*
CONSTANT        1       59.402       59.402      3.23674       0.1022
iron            1       205.46       205.46     11.19513    0.0074162
aluminum        1       626.13       626.13     34.11716   0.00016359
ERROR1         10       183.52       18.352
* ANOVA is approximate and should be interpreted with caution

Robust estimate of sigma: 4.5468
```

If this had been done first, a second iterative fit would have been unnecessary. Even without the use of `marginal:T`, you can compute approximate ANOVA tables for equivalent models specified in a different order with only one use of `robust()`. The key is knowing that the ANOVA table is computed by doing an ordinary ANOVA computation on a certain vector $\tilde{y}$ of "pseudo data". Each element of $\tilde{y}$ is the sum of the fitted value for the case and K times the modified residuals in `WTDRESIDUALS` where the value of K is $\frac{1}{\mu}\left(1+\frac{p}{n}\left(\frac{1-\mu}{\mu}\right)\right)$. Here n is the sample size, p = the degrees of freedom in the model fit, and $\mu = m/n$, where m = number of non-truncated values. You can determine m from the number of cases for which `WTDRESIDUALS/RESIDUALS` = 1, allowing for possible rounding error.

```
Cmd> m <- sum(round(WTDRESIDUALS/RESIDUALS,12) == 1 )

Cmd> n <- length(adsorption); p <- sum(DF[-4]); vector(n,m,p)
(1)          13           9           3

Cmd> mu <- m/n; K <- (1 + (p/n)*(1-mu)/mu)/mu; K
(1)      1.5926

Cmd> fit <- adsorption - RESIDUALS; y1 <- fit + K * WTDRESIDUALS

Cmd> anova("y1=iron+aluminum",marginal:T) # same result as robust()
Model used is y1=iron+aluminum
WARNING: SS are Type III sums of squares
                DF          SS          MS
CONSTANT         1      59.402      59.402
iron             1      205.46      205.46
aluminum         1      626.13      626.13
ERROR1          10      183.52      18.352
```

The results of `secoefs()` would be the same as well.

It is also of interest to compare the robust analysis with the results of least squares regression omitting case 8 entirely as would be reasonable once you had determined that case 8 was deviant.

```
Cmd> adsorption[8] <- ? # set case 8 to MISSING

Cmd> regress("adsorption=iron+aluminum",pvals:T)
Model used is adsorption=iron+aluminum
WARNING: cases with missing values deleted
                 Coef       StdErr            t      P-Value
CONSTANT      -6.8757       3.6342      -1.8919     0.091053
iron          0.10788     0.031191       3.4586    0.0071765
aluminum      0.36221      0.07541       4.8032   0.00096936

N: 13,  MSE: 20.164, DF: 9,  R^2: 0.95124
Regression F(2,9): 87.787, P-value: 1.2483e-06, Durbin-Watson: 2.6746
To see the ANOVA table type 'anova()'
```

```
Cmd> anova(,marginal:T,fstat:T)
Model used is adsorption=iron+aluminum
WARNING: cases with missing values deleted
WARNING: SS are Type III sums of squares
                DF           SS           MS            F      P-value
CONSTANT         1       72.174       72.174      3.57944     0.091053
iron             1        241.2        241.2     11.96200    0.0071765
aluminum         1       465.19       465.19     23.07090   0.00096936
ERROR1           9       181.47       20.164
```

The results, of course, differ from the robust results, but they are roughly similar.