Calculating Power with the Matrix Approach

```
/*********************************  matpow1.sas *****************************/
title 'Power analysis for fixed-effects factorial ANOVA: Matrix approach';
options linesize = 79 pagesize = 100 noovp formdlim='-';
proc iml;
/********* Edit this input: Rows of matrices are separated by commas  ********/
     alpha = 0.05;
     wantpow = .80;
     f = {1,1,1,1,1,1};          /* Relative sample sizes       */
     C = { 1 -1 -1  1  0  0,     /* Contrast matrix             */
           0  0  1 -1 -1  1};
     eff = {0,-.5};              /* In standard deviation units */
/*****************************************************************************/
     r = nrow(f) ; q = nrow(eff); f = f/sum(f);

/*** Echoing input ***/
     print " Significance level: " alpha;
     print " Desired power: " wantpow;
     print " Relative sample sizes:" f;
     print " Contrast matrix "; print C;
     print " Effect matrix (in SD units): " eff;
     print "  ";
/*** Checking input ***/
     print " Checking input ...  ";
     inerr = 0; /* Input error = no. This may be changed below */
     if nrow(C) ^= q then do;
        print "Error: Length of eff must equal number of rows in C";
        inerr=1;
     end;
     if ncol(C) ^= r then do;
        print "Error: Length of f must equal number of cols in C";
        inerr=1;
     end;
     if eff`*eff = 0 then do;
        print "Error: eff can't be zero; this would cause an infinite loop!";
        inerr=1;
     end;
     aa = min(f##2);
     if aa = 0 then do;
        print "Error: No sample size may be zero.";
        inerr=1;
     end;

/*** Proceed if no input errors ***/
     if inerr = 1 then abort;
        else do;
        print "Input appears to be okay.  ";
             core = inv(C*inv(diag(f))*C`);
             effsize = eff`*core*eff;  print " Effect size = " effsize;
             power = 0; n = r+1; oneminus = 1-alpha;  /* Initializing ...*/
             do until (power >= wantpow);
                n = n+1 ;
                ncp = n * effsize;
                df2 = n-r;
                power = 1-probf(finv(oneminus,q,df2),q,df2,ncp);
             end; /*  End Loop */
             print " ";
             print " Required sample size is " n;
             print " ";
             print " ";
        end; /* End computation (Conditional on no input errors) */
```

```
--------------------------------------------------------------------------------

       Power analysis for fixed-effects factorial ANOVA: Matrix approach      1
                                                 15:22 Sunday, January 18, 2004

                                                  ALPHA

                            Significance level:      0.05


                                             WANTPOW

                              Desired power:       0.8


                                                      F

                           Relative sample sizes: 0.1666667
                                                  0.1666667
                                                  0.1666667
                                                  0.1666667
                                                  0.1666667
                                                  0.1666667


                                     Contrast matrix


                                            C

                     1         -1         -1          1          0          0
                     0          0          1         -1         -1          1


                                                                  EFF

                        Effect matrix (in SD units):                0
                                                                 -0.5




                                   Checking input ...


                               Input appears to be okay.


                                             EFFSIZE

                               Effect size =  0.0138889




                                                                   N

                          Required sample size is        697
```

Since 697/6 = 116.1667, make it n = 117*6 = 702.

```
################## matpow1.R #######################
#              source("matpow1.R")                 #
#   Then use the function matpow1 interactively.    #
#   Notice that function matpow1 depends on fpow2,  #
#   also given in the second part of this file.     #
#####################################################
matpow1 <- function(C,eff,f,wantpow=0.80,alpha=0.05)
# H0: C Beta = 0
# Beta is r x 1
# C     is q x r contrast matrix
# eff   is vector of effects (C beta - h) in sd units, length r
# f     is vector of RELATIVE sample sizes, all non-negative
#
    {
    f <- f/sum(f)
    if(min(f)<=0) stop("Cell sample sizes must all be positive.")
    kore <- solve(C%*%diag(1/f)%*%t(C))
    effsize <- t(eff)%*%kore%*%eff
    q <- dim(C)[1] ; r <- dim(C)[2]
#    cat("r,q,effsize,wantpow,alpha = ",r,q,effsize,wantpow,alpha,"\n")
    matpow1 <- fpow2(r,q,effsize,wantpow,alpha)
    matpow1
    } # End of function matpow1


fpow2 <- function(r,q,effsize,wantpow=0.80,alpha=0.05)
#############################################################################
# Power for the general multiple regression model, testing H0: C Beta = h   #
#       r       is the number of beta parameters                            #
#       q       Number rows in the C matrix = numerator df                  #
#       effsize is ncp/n, a squared distance between C Beta and h           #
#       wantpow is the desired power, default = 0.80                        #
#       alpha   is the significance level, default = 0.05                   #
#############################################################################
    {
    pow <- 0 ; nn <- r+1 ; oneminus <- 1 - alpha
    while(pow < wantpow)
        {
        nn <- nn+1
        phi <- nn * effsize
        ddf <- nn-r
        pow <- 1 - pf(qf(oneminus,q,ddf),q,ddf,phi)
        }#End while
    fpow2 <- nn
    fpow2  # Returns needed n
    }      # End of function fpow2
```

```
>
> source("matpow1.R")
> conmat <- rbind(c(1, -1, -1,  1,  0,  0),
+                 c(0,  0,  1, -1, -1,  1)  )
> effect <- c(0,-.5)
> ssizes <- numeric(6) + 1
>
>
> matpow1(conmat,effect,ssizes) # Using default wantpower of 0.80 and alpha=0.05
[1] 697
>
```

Again, since 697/6 = 116.1667, make it n = 117*6 = 702.

Here is another way to organize the input. Assume H_0 is $C\beta=0$. The user gives a value for the entire vector $C\beta/\sigma$, bearing in mind that only differences between β/σ values (cell means, in standard deviation units) are going to matter. Remember the table:

	Level of B		
Level of A	1	2	Average
1	0.000	0.250	0.125
2	0.000	0.250	0.125
3	0.000	-0.250	-0.125
Average	0.000	0.083	0.042

```
/******************************  matpow2.sas ******************************/
title 'Power analysis for fixed-effects factorial ANOVA: Matrix approach';
title2 'Specify a beta vector instead of just an effect';
options linesize = 79 pagesize = 100 noovp formdlim='-';
proc iml;
/********* Edit this input: Rows of matrices are separated by commas  ********/
     alpha = 0.05;
     wantpow = .80;
     f = {1,1,1,1,1,1};          /* Relative sample sizes       */
     C = { 1 -1 -1  1  0  0,     /* Contrast matrix             */
           0  0  1 -1 -1  1};
     beta = {0,.25,0,.25,0,-.25};

              /* In standard deviation units */
/*************************************************************************/
     eff = C*beta;
     r = nrow(f) ; q = nrow(eff); f = f/sum(f);
/*** Echoing input ***/
     print " Significance level: " alpha;
     print " Desired power: " wantpow;
     print " Relative sample sizes:" f;
     print " Contrast matrix "; print C;
     print " Beta matrix (in SD units): " beta;
     print " Effect (h) matrix in SD units: " eff;
     print "  ";
/*** Checking input ***/
```

Skip this part. It's the same as before.

```
/*** Proceed if no input errors ***/
     if inerr = 1 then abort;
        else do;
        print "Input appears to be okay.  ";
             core = inv(C*inv(diag(f))*C`);
             effsize = eff`*core*eff;  print " Effect size = " effsize;
             power = 0; n = r+1; oneminus = 1-alpha;  /* Initializing ...*/
             do until (power >= wantpow);
                n = n+1 ;
                ncp = n * effsize;
                df2 = n-r;
                power = 1-probf(finv(oneminus,q,df2),q,df2,ncp);
             end; /*  End Loop */
             print " ";
             print " Required sample size is " n;
             print " ";
             print " ";
        end; /* End computation (Conditional on no input errors) */
```

```
################## matpow2.R #######################
#                  source("matpow2.R")                  #
#    Then use the function matpow2 interactively.     #
#    Notice that function matpow1 depends on fpow2,   #
#    also given in the second part of this file.      #
#######################################################
matpow2 <- function(C,beta,f,wantpow=0.80,alpha=0.05)
# H0: C Beta = 0
# Beta is r x 1
# C     is q x r contrast matrix
# f     is vector of RELATIVE sample sizes, all non-negative
#
    {
    f <- f/sum(f)
    if(min(f)<=0) stop("Cell sample sizes must all be positive.")
    eff <- C%*%beta
    kore <- solve(C%*%diag(1/f)%*%t(C))
    effsize <- t(eff)%*%kore%*%eff
    q <- dim(C)[1] ; r <- dim(C)[2]
#    cat("r,q,effsize,wantpow,alpha = ",r,q,effsize,wantpow,alpha,"\n")
    matpow1 <- fpow2(r,q,effsize,wantpow,alpha)
    matpow1
    } # End of function matpow2


fpow2 <- function(r,q,effsize,wantpow=0.80,alpha=0.05)
#############################################################################
# Power for the general multiple regression model, testing H0: C Beta = h   #
#       r       is the number of beta parameters                            #
#       q       Number rows in the C matrix = numerator df                  #
#       effsize is ncp/n, a squared distance between C Beta and h           #
#       wantpow is the desired power, default = 0.80                        #
#       alpha   is the significance level, default = 0.05                   #
#############################################################################
    {
    pow <- 0 ; nn <- r+1 ; oneminus <- 1 - alpha
    while(pow < wantpow)
        {
        nn <- nn+1
        phi <- nn * effsize
        ddf <- nn-r
        pow <- 1 - pf(qf(oneminus,q,ddf),q,ddf,phi)
        }#End while
    fpow2 <- nn
    fpow2  # Returns needed n
    }      # End of function fpow2



> source("matpow2.R")
> conmat <- rbind(c(1, -1, -1,  1,  0,  0),
+                 c(0,  0,  1, -1, -1,  1)  )
> cellmeans <- c(0,.25,0,.25,0,-.25)
> ssizes <- numeric(6) + 1
>
> matpow2(effmat,cellmeans,ssizes) # Using default wantpower of 0.80 and alpha=0.05
[1] 697
```

Suppose you wanted to play with relative sample sizes.

Consider again the case of four equally spaced population means, all a quarter σ apart. Using `fpow2.sas` and the R function `fpow2`, we found that a total sample size of *n* = 144 was required to obtain a power of 0.80 against this alternative when the sample sizes were all equal.

The optimal allocation is to split all the observations equally between treatments One and Four. This is unreasonable. But what if one went part of the way there -- say, by giving two-thirds of the sample to those two treatments?

```
> source("powerfun.R") # Defining needed functions
> beta <- c(0,.25,.5,.75) # Really beta/sigma
> Cmat <- rbind( c(1,-1, 0, 0),
+                c(0, 1,-1, 0),
+                c(0, 0, 1,-1) )
> f <- c(2,1,1,2)
> matpow2(Cmat,beta,f)
[1] 115
```

Get the same power with 26 fewer subjects, or 18%.

Say, based on the cautious hunch that treatments 1 and 4 would be the most different.

A lot of this is strongly tied to subjective judgement.

A Bayesian approach is possible.

Put a prior distribution on $\boldsymbol{\beta}/\sigma$, and choose sample sizes to maximize *expected* power.