```
################## matpow.R #################################
#               source("matpow.R")                          #
#   Then use the functions matpow1 & 2 interactively.       #
#   Notice that the matpow functions depend on fpow2 below. #
#############################################################

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

matpow1 <- function(C,eff,f,wantpow=0.80,alpha=0.05)
# H0: C Mu = 0
# Mu is r x 1
# C     is q x r contrast matrix
# eff   is vector of effects (non-zero 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
```

```
matpow2 <- function(C,mu,f,wantpow=0.80,alpha=0.05)
# H0: C Mu = 0
# Mu is r x 1, in SD units
# C     is q x r contrast matrix
# eff   is vector of effects (non-zero 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.")
    eff <- C%*%mu
    kore <- solve(C%*%diag(1/f)%*%t(C))
    effsize <- t(eff)%*%kore%*%eff
    q <- dim(C)[1] ; r <- dim(C)[2]
    matpow1 <- fpow2(r,q,effsize,wantpow,alpha)
    matpow1
    } # End of function matpow2
```

First try the two-sample t-test. Want sample size to detect a diff of 1/2 SD, with probability 0.80, with sample sizes equal. Answer should be n=128, or 64 per group. First I pasted the 3 functions in, and then:

```
>
> cmat <- rbind(c(1,-1))
> diff <- 0.5
> relsampsizes <- c(1,1)
> matpow1(cmat,diff, relsampsizes) # Use defaults of 0.80 power and the 0.05 level
[1] 128
>


>
> # What would happen if the first group had twice as many subjects as the second?
> relsampsizes <- c(2,1)
> matpow1(cmat,diff, relsampsizes)
[1] 144
> 144/3
[1] 48
> # Okay, n1 = 96 and n2 = 48
>
```

It might be more convenient to specify population means, but they must be in SD units.

```
>
> realmeans <- c(0,.5) ; cmat <- rbind(c(1,-1)) ; relsampsizes <- c(1,1)
> matpow2(cmat,realmeans, relsampsizes)
[1] 128
>
```

Now suppose we have an A by B (2 by 3) factorial design, and we are focusing on the main effect for B. Say the difference between marginal means 1 and 2 is one SD unit, and there is no difference between marginal means 2 and 3. We want to detect this with probability 0.90 at the 0.05 level. As usual, all sample sizes will be equal.

```
                          BACTERIA TYPE
                TEMP      1          2          3
                   1     mu11       mu12       mu13
                   2     mu21       mu22       mu23
```

It is easiest to write the contrast matrix as

```
     1 -1  0   1 -1  0
     0  1 -1   0  1 -1
```

But watch out. The alternative we want to detect is

```
½ (mu11+mu21) – ½ (mu12+mu22) = 1, therefore mu11 + mu21 - mu12 - mu22 = 2
```

```
# Recall   matpow1 <- function(C,eff,f,wantpow=0.80,alpha=0.05)

>
> cmat <- rbind( c(1, -1,  0,   1, -1,  0),
+                c(0,  1, -1,   0,  1, -1) )
> truth <- c(2,0) # Note the two
> relsampsizes <- c(1,1,1,1,1,1) # Equal
> matpow1(cmat,truth,relsamplesizes,0.90)
Error in matpow1(cmat, truth, relsamplesizes, 0.9) :
         object "relsamplesizes" not found
> matpow1(cmat,truth,relsampsizes,0.90)
[1] 61
> # About n=10 per group. Could make it n=11 to be safe.
>
```

Would this be easier?

```
>
> trumean <- c(1,0,0,1,0,0)
> # Use same cmat and relsampsizes
> matpow2(C=cmat,mu=trumean,f= relsampsizes,wantpow=0.90)
[1] 61
>
```