# Calculating Power with the Matrix Approach

```
################## 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(L,eff,f,wantpow=0.80,alpha=0.05,printES=F)
# H0: L Beta = 0
# Beta is p x 1
# L     is r x p contrast matrix
# eff   is r x 1 vector of effects (L beta - h) in sd units
# f     is vector of p RELATIVE sample sizes, all non-negative
#
    {
    f <- f/sum(f)
    if(min(f)<=0) stop("Cell sample sizes must all be positive.")
    kore <- solve(L%*%diag(1/f)%*%t(L))
    effsize <- t(eff)%*%kore%*%eff
    if(printES) print("Effect Size = ",effsize)
    r <- dim(L)[1] ; p <- dim(L)[2]
#    cat("p,r,effsize,wantpow,alpha = ",p,r,effsize,wantpow,alpha,"\n")
    matpow1 <- fpow2(p,r,effsize,wantpow,alpha)
    matpow1
    } # End of function matpow1


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

>
> source("http://www.utstat.toronto.edu/~brunner/appliedf12/data/fpow.txt")
> conmat <- rbind(c(1, -1, -1,  1,  0,  0),
+                 c(0,  0,  1, -1, -1,  1)  )
> effect <- c(0,-.5)
> ssizes <- numeric(6) + 1 # Equal sample sizes
>
>
> matpow1(conmat,effect,ssizes) # Using default wantpower of 0.80 and alpha=0.05
[1] 697
>
```

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

Here is another way to organize the input. Assume $H_0$ is $L\beta=0$. The user gives a value for the vector $\beta/\sigma$, bearing in mind that only differences between $\beta/\sigma$ values (cell means, in standard deviation units) are going to matter. Consider 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.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(L,beta,f,wantpow=0.80,alpha=0.05)
# H0: L Beta = 0
# Beta is p x 1
# L     is r x p 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 <- L%*%beta
    kore <- solve(L%*%diag(1/f)%*%t(L))
    effsize <- t(eff)%*%kore%*%eff
    r <- dim(L)[1] ; p <- dim(L)[2]
#    cat("p,r,effsize,wantpow,alpha = ",p,r,effsize,wantpow,alpha,"\n")
    matpow2 <- fpow2(p,r,effsize,wantpow,alpha)
    matpow2
    } # End of function matpow2
```

```
> source("http://www.utstat.toronto.edu/~brunner/appliedf12/data/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(conmat,cellmeans,ssizes) # Using default wantpower of 0.80 and alpha=0.05
[1] 697
```

# Suppose you wanted to play with relative sample sizes.

Consider 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?

```
> # Defining needed functions
> source("http://www.utstat.toronto.edu/~brunner/appliedf12/data/powerfun.R")
> 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.

Perhaps most estimation and inference should be frequentist, but most design and power analysis should be Bayesian.