# Calculating Power for F-tests with the Matrix Approach

These functions are available for tests of $H_0$: $\mathbf{L\beta} = \mathbf{0}$. They all return the minimum sample size necessary for a desired power. Get them all with

```
source("http://www.utstat.utoronto.ca/~brunner/Rfunctions/Powerfun.txt")
```

`fpow3`: Applies to the general multiple regression model. Input is

*p* = number of regression parameters
*r* = number of linear constraints and
Effect size = ncp/n, which must be calculated separately.

`matpow1`: Factorial ANOVA with no covariates, cell means coding. Input is

L = r x p contrast matrix
effect = r x 1 vector of effects (L beta - h) in sd units
f = vector of p relative sample sizes, all non-negative

`matpow2`: Factorial ANOVA with no covariates, cell means coding. Input is

Beta = a *p x 1* $\boldsymbol{\beta}/\sigma$, true cell means in standard deviation units
L = an *r x p* contrast matrix
f = a vector of relatve sample sizes, all non-negative

```
fpow3 <- 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                   #
#############################################################################
    {
    pow <- 0 ; nn <- p+1 ; oneminus <- 1 - alpha
    while(pow < wantpow)
        {
        nn <- nn+1
        lambda <- nn * effsize
        ddf <- nn-p
        pow <- 1 - pf(qf(oneminus,r,ddf),r,ddf,lambda)
        }#End while
    return(nn)
    }      # End of function fpow3

matpow1 <- function(L,eff,f,wantpow=0.80,alpha=0.05,printES=F)
#################################################################
#                                                               #
# Testing H0: L beta = 0, cell means coding and no covariates.  #
# Beta is p x 1 Input                                           #
#   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]
    needn <- fpow3(p,r,effsize,wantpow,alpha)
    return(needn)
    } # End of function matpow1
```

| | 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 |

```
> source("http://www.utstat.utoronto.ca/~brunner/Rfunctions/Powerfun.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 vector of true cell means, in standard deviation units. That's $\boldsymbol{\beta}/\sigma$. It works because only differences between $\boldsymbol{\beta}/\sigma$ values 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 <- function(L,beta,f,wantpow=0.80,alpha=0.05)
##############################################################################
# H0: L Beta = 0                                                             #
# Beta is p x 1 beta/sigma, true cell means in standard deviation units      #
# 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]
    needn <- fpow3(p,r,effsize,wantpow,alpha)
    return(needn)
    } # End of function matpow2
```

```
> conmat <- rbind(c(1, -1,  -1,  1,   0,  0),
+                 c(0,  0,   1, -1,  -1,  1)  )
> cellmeans <- c(0,.25,0,.25,0,-.25)
> sizes <- numeric(6) + 1 # Equal sample sizes
> 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 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?

```
> 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.