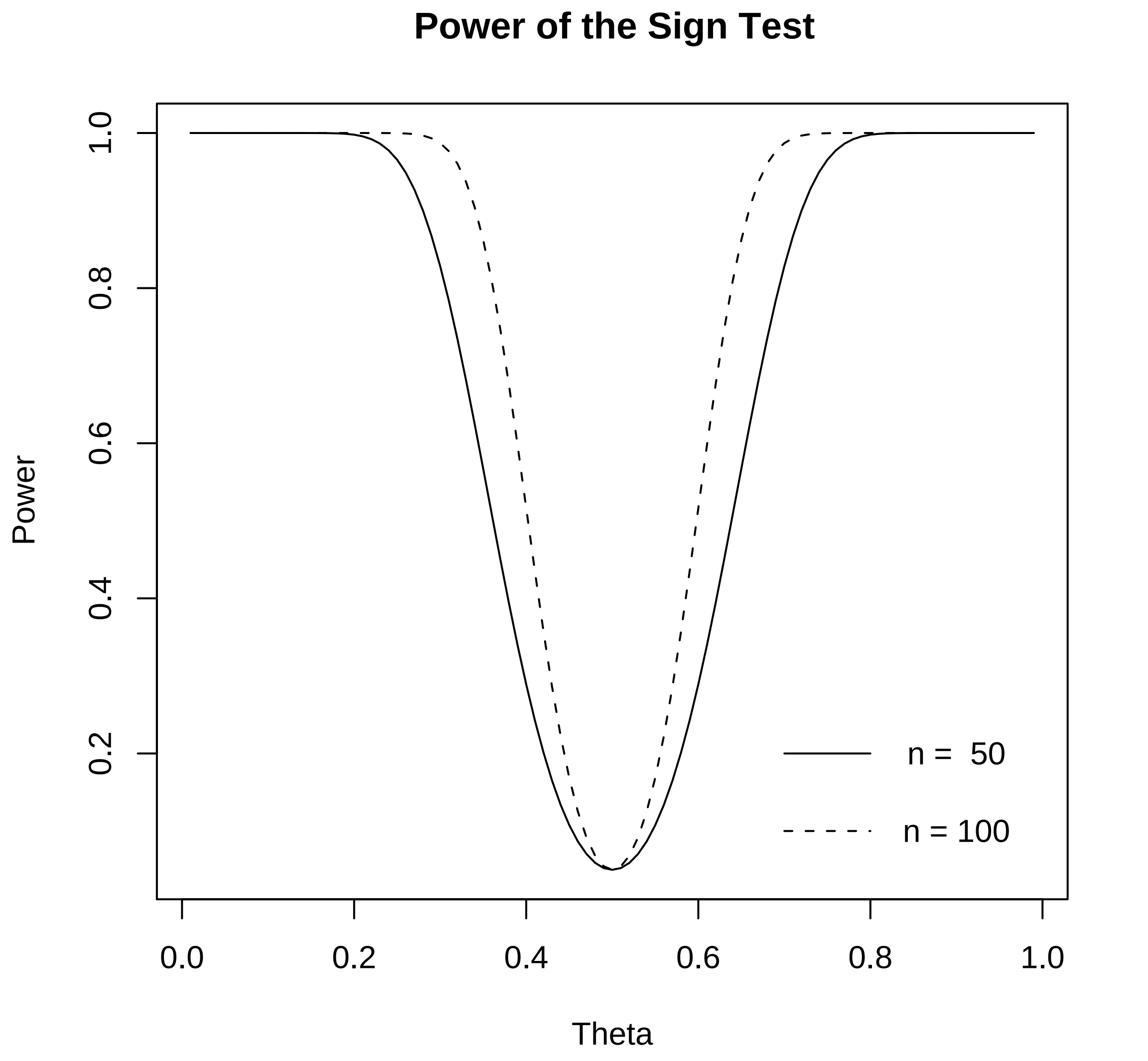
Power Calculations 1

```
signpow <- function(theta,n) # Power of the sign test
    {
    L <- sqrt(n)*(.5-theta)/sqrt(theta*(1-theta))
    R <- 1.96/(2*sqrt(theta*(1-theta)))
    signpow <- 1 - pnorm(L+R) + pnorm(L-R)
    signpow
    } # End of function signpow


> signpow(.5,100)
[1] 0.04999579
> signpow(.5,1000)
[1] 0.04999579
> signpow(.51,1000)
[1] 0.09687793
> signpow(.51,10000)
[1] 0.515994
> signpow(.51,20000)
[1] 0.8074681

######################################################################
# powplot.R  -  Plot Power of sign test as a function of true Theta  #
#                                                                    #
# R --vanilla < powplot.R                                            #
#####################################################################


signpow <- function(theta,n)  # Power of the sign test
    {
    L <- sqrt(n)*(.5-theta)/sqrt(theta*(1-theta))
    R <- 1.96/(2*sqrt(theta*(1-theta)))
    signpow <- 1 - pnorm(L+R) + pnorm(L-R)
    signpow
    } # End of function signpow

Theta <- seq(from=.01,to=.99,by=.01)
Power <- signpow(Theta,50)
p100 <- signpow(Theta,100)
system("rm powplot.pdf")
pdf("powplot.pdf")

plot(Theta,Power,type="l")
lines(Theta,p100,lty=2)
title("Power of the Sign Test")
x1 <- c(.70,.80) ; y1 <- c(.2,.2) ; lines(x1,y1,lty=1)
text(.9,.2,"n =  50")
x2 <- c(.70,.80) ; y2 <- c(.1,.1) ; lines(x2,y2,lty=2)
text(.9,.1,"n = 100")
```


Power of the Sign Test
Power
1.0
0.8
0.6
0.4
0.2
0.0
0.2
0.4
0.6
0.8
1.0
Theta
n = 50
n = 100

```
######################################################################
# signpower.R  -  Find sample size for sign test to get required     #
#                 power. Do   source("signpower.R")   and then use   #
#                 the function size1 interactively.                  #
######################################################################

signpow <- function(theta,n)  # Power of the sign test
    {
    L <- sqrt(n)*(.5-theta)/sqrt(theta*(1-theta))
    R <- 1.96/(2*sqrt(theta*(1-theta)))
    signpow <- 1 - pnorm(L+R) + pnorm(L-R)
    signpow
    } # End of function signpow


size1 <- function(truet,needpow=0.80,nstart=1,nend=1000000)
    {
    nn <- nstart ; pow <- 0
    while(pow<needpow)
        {
        pow <- signpow(truet,nn)
        nn <- nn+1
        if(nn>nend) stop("Too many iterations!")
        }
    cat("\n")
    cat("For true Theta of ",truet,", sign test requires a sample \n")
    cat("        size of ",nn-1," to have power of ",pow,"\n")
    cat("\n")
    } # End function size1
```

```
> source("signpower.R")
> size1(.75)

For true Theta of  0.75 , sign test requires a sample
        size of  29  to have power of  0.8011995

> signpow(.75,28) # Just checking
[1] 0.7857723
>
> # Want power of 0.99 when theta = .51
> size1(0.51,0.99)

For true Theta of  0.51 , sign test requires a sample
        size of  45922  to have power of  0.99
```

```
>
> # How about 60% chance of detecting gaze?
> size1(0.60,0.99)

For true Theta of  0.6 , sign test requires a sample
        size of  450  to have power of  0.9900893

> # 75% chance of detecting gaze?
> size1(0.75,0.99)

For true Theta of  0.75 , sign test requires a sample
        size of  64  to have power of  0.9907533
```

$$\Sigma = \begin{bmatrix} 1.42 & 0.42 & 0.42 & 0.00 & 0.00 \\ 0.42 & 1.42 & 0.42 & 0.00 & 0.00 \\ 0.42 & 0.42 & 1.42 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 1.18 & 0.18 \\ 0.00 & 0.00 & 0.00 & 0.18 & 1.18 \end{bmatrix}$$

Denoting the sample variance-covariance matrix by $\mathbf{S}$, and the jth sample variance (diagonal element of $\mathbf{S}$) by s_j^2, the large-sample likelihood ratio test statistic for k variables may be written

$$G = n(\sum_{j=1}^{k} \log s_j^2 - \log |\mathbf{S}|),$$

where $|\mathbf{S}|$ refers to the determinant of $\mathbf{S}$. Under the null hypothesis that $\mathbf{\Sigma}$ is diagonal, G has a chisquare distribution with $\frac{1}{2}k(k-1)$ degrees of freedom (the number of unique off-diagonal elements). Here, the degrees of freedom equal 10.

$$\widehat{P} \pm z_{1-\frac{\alpha}{2}} \sqrt{\frac{\widehat{P}(1-\widehat{P})}{M}}$$

This formula is implemented in the S function `merror` for "margin of error."

```
merror <- function(phat,m,alpha) # (1-alpha)*100% merror for a proportion
    {
    z <- qnorm(1-alpha/2)
    merror <- z * sqrt(phat*(1-phat)/m)  # m is (Monte Carlo) sample size
    merror
    }
```

Table 1: Monte Carlo Sample Size Required to Estimate Power with a Specified 99% Margin of Error

	Power Being Estimated					
Margin of Error	0.70	0.75	0.80	0.85	0.90	0.99
0.10	140	125	107	85	60	7
0.05	558	498	425	339	239	27
0.01	13,934	12,441	10,616	8,460	5,972	657
0.005	55,734	49,762	42,464	33,838	23,886	2,628
0.001	1,393,329	1,244,044	1,061,584	845,950	59,7141	65,686

```
# cvm.R
# Power for test of zero correlation for an entire matrix
# (Large Sample LR Test)
# Execute with   source("cvm.R")
#
M <- 10000
sim <- numeric(M)
set.seed(32448)
n <- 50 ; v1 <- .42 ;       v2 <- .18
          s1 <- sqrt(v1) ; s2 <- sqrt(v2)
G <- function(datamat)
   {
   nn <- dim(datamat)[1] ; kk <- dim(datamat)[2] ; df <- kk*(kk-1)/2
   G <- numeric(3)
   names(G) <- c("Chisq","df","P-value")
   S <- var(datamat)
   G[1] <- nn * ( sum(log(diag(S))) - sum(log(eigen(S)$values)) ) #$
   G[2] <- df
   G[3] <- 1 - pchisq(G[1],df)
   G
   } # End function G
merror <- function(phat,m,alpha=0.01) # (1-alpha)*100% merror for a proportion
    {
    z <- qnorm(1-alpha/2)
    merror <- z * sqrt(phat*(1-phat)/m)  # m is (Monte Carlo) sample size
    merror
    }

for(j in 1:M)
   {

e1 <- rnorm(n,0,s1) ; e2 <- rnorm(n,0,s2)
x1 <- rnorm(n)+e1 ; x2 <- rnorm(n)+e1 ;  x3 <- rnorm(n)+e1 ;
x4 <- rnorm(n)+e2 ;  x5 <- rnorm(n)+e2
dat <- cbind(x1,x2,x3,x4,x5)
# print(G(dat))
print(j)
sim[j] <- G(dat)[3] < .05

    }
poww <- length(sim[sim==1])/M
cat("Power = ", poww ,"\n")
cat("Plus or Minus 99% Margin of error: ",merror(poww,M),"\n")
```

Here's the output.

```
Power =  0.6987
Plus or Minus 99% Margin of error:  0.01181849
```

Here is the model.

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon},$$

where $\mathbf{X}$ is an $n \times r$ matrix of known constants, $\boldsymbol{\beta}$ is a $r \times 1$ vector of unknown constants, and $\boldsymbol{\epsilon}$ is multivariate normal with mean zero and covariance matrix $\sigma^2\mathbf{I}_n$, and $\sigma^2 > 0$ is an unknown constant.

The null hypothesis is $H_0 : \mathbf{C}\boldsymbol{\beta} = \mathbf{h}$ The F statistic for testing this null hypothesis is

$$F^* = \frac{(\mathbf{C}\widehat{\boldsymbol{\beta}} - \mathbf{h})'(\mathbf{C}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{C}')^{-1}(\mathbf{C}\widehat{\boldsymbol{\beta}} - \mathbf{h})}{q\,MSE}$$

When H_0 is false, F^* has a *noncentral* F distribution with parameters q, $n-r$ and ϕ. A useful formula for ϕ is

$$\phi = \frac{(\mathbf{C}\boldsymbol{\beta} - \mathbf{h})'(\mathbf{C}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{C}')^{-1}(\mathbf{C}\boldsymbol{\beta} - \mathbf{h})}{\sigma^2}$$

Comparing two means

```
> n <- seq(from=120,to=140,by=2) ; phi <- n/16 ; ddf <- n-2
> cbind(n,pf(qf(.95,1,ddf),1,ddf,phi,FALSE))
        n
 [1,] 120 0.7752659
 [2,] 122 0.7820745
 [3,] 124 0.7887077
 [4,] 126 0.7951683
 [5,] 128 0.8014596
 [6,] 130 0.8075844
 [7,] 132 0.8135460
 [8,] 134 0.8193475
 [9,] 136 0.8249920
[10,] 138 0.8304825
[11,] 140 0.8358223
```

Comparing *r* means

```
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

> 3 * var(c(0,.25,.5,.75)) / 4
[1] 0.078125

> source("fpow2.R")
> fpow2(r=4,q=3,effsize=0.078125)
[1] 144
```

```
> # Signal to noise ratio
> source("fpow2.R")
> fpow2(r=6,q=5,effsize=0.10)
[1] 134
```

Interaction example: Effect size is 0.01388889

```
> source("fpow2.R")
> fpow2(r=6,q=2,effsize=0.01388889,wantpow=0.80,alpha=0.05)
[1] 697
```