Estimating Power

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

$$\lambda = \frac{(\mathbf{L}\boldsymbol{\beta} - \mathbf{h})^\top(\mathbf{L}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{L}^\top)^{-1}(\mathbf{L}\boldsymbol{\beta} - \mathbf{h})}{\sigma^2}$$

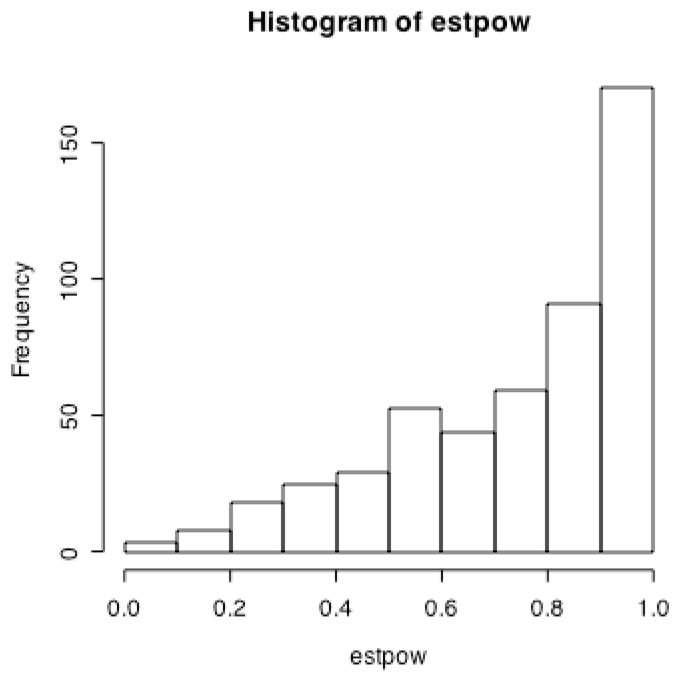
```
Idea: Estimate phi with r F. Use estimated phi to

      Estimate power for this sample size.

      Estimate effect size and find the sample size necessary to obtain desired
      power.
```

```
> # With n=64 per group and means 1/2 sd apart, have true power of 0.80
> # True phi = 128 * 1/2(1-1/2) * (1/2)^2 = 8
> set.seed(9999)
> critval = qf(0.95,1,126)
> exper = rnorm(64,mean=1,sd=2); contr = rnorm(64,mean=0,sd=2)
> ttest = t.test(exper,contr,paired=F,var.equal=T)
> # r=1 so phihat = F = t^2
> phihat = ttest$statistic^2; phihat
       t
10.12523
> powhat = 1-pf(critval,1,126,phihat); powhat
[1] 0.8844847
>

> exper = rnorm(64,mean=1,sd=2); contr = rnorm(64,mean=0,sd=2)
> ttest = t.test(exper,contr,paired=F,var.equal=T)
> # r=1 so phihat = F = t^2
> phihat = ttest$statistic^2; phihat
       t
7.118774
> powhat = 1-pf(critval,1,126,phihat); powhat
[1] 0.7541977
```

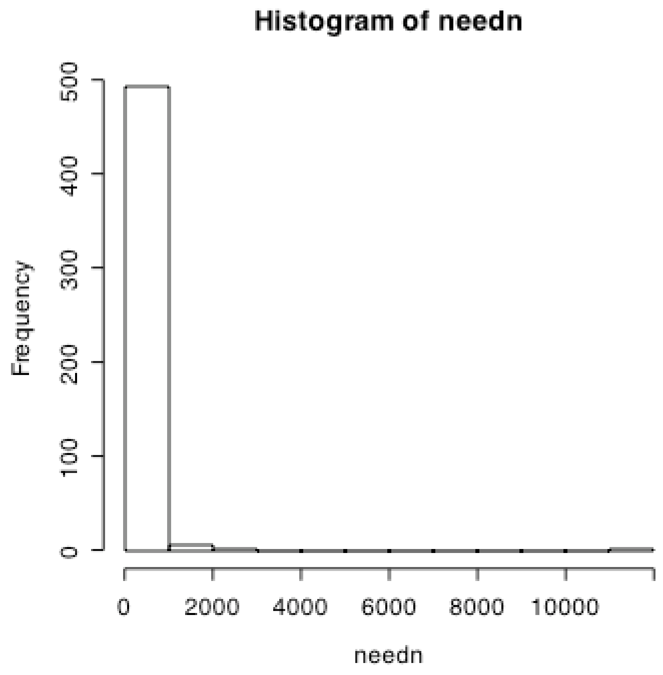
```
> exper = rnorm(64,mean=1,sd=2); contr = rnorm(64,mean=0,sd=2)
> ttest = t.test(exper,contr,paired=F,var.equal=T)
> # r=1 so phihat = F = t^2
> phihat = ttest$statistic^2; phihat
       t
7.719538
> powhat = 1-pf(critval,1,126,phihat); powhat
[1] 0.7873356
> exper = rnorm(64,mean=1,sd=2); contr = rnorm(64,mean=0,sd=2)
> ttest = t.test(exper,contr,paired=F,var.equal=T)
> # r=1 so phihat = F = t^2
> phihat = ttest$statistic^2; phihat
       t
1.288968
> powhat = 1-pf(critval,1,126,phihat); powhat
[1] 0.2033561
> exper = rnorm(64,mean=1,sd=2); contr = rnorm(64,mean=0,sd=2)
> ttest = t.test(exper,contr,paired=F,var.equal=T)
> # r=1 so phihat = F = t^2
> phihat = ttest$statistic^2; phihat
       t
9.238939
> powhat = 1-pf(critval,1,126,phihat); powhat
[1] 0.8545965
> exper = rnorm(64,mean=1,sd=2); contr = rnorm(64,mean=0,sd=2)
> ttest = t.test(exper,contr,paired=F,var.equal=T)
> # r=1 so phihat = F = t^2
> phihat = ttest$statistic^2; phihat
        t
0.4823292
> powhat = 1-pf(critval,1,126,phihat); powhat
[1] 0.1059439
> exper = rnorm(64,mean=1,sd=2); contr = rnorm(64,mean=0,sd=2)
> ttest = t.test(exper,contr,paired=F,var.equal=T)
> # r=1 so phihat = F = t^2
> phihat = ttest$statistic^2; phihat
       t
10.61639
> powhat = 1-pf(critval,1,126,phihat); powhat
[1] 0.898559
```

```
> # Estimate the ncp, like SPSS
> # Two-sample simulation: Version 3
> # ncp = n * q(1-q) * delta^2
> # For detecting delta = |mu1-mu2|/sigma = 1/2 with prob .8, need
> # n1 = n2 = 64 for n of 128, so true phi = 128 * 1/2(1-1/2) * (1/2)^2 = 8
>
> findn <- function(wantpow=.8,esize,nstart=3,q=.5)
+        # Specific to the 2-sample problem
+        # ncp = n * q(1-q) * delta^2
+     {
+     pow <- 0 ; nn <- nstart
+     while(pow < wantpow)
+         {
+         nn <- nn+1
+         ddf <- nn-2
+         ncp <- nn * q*(1-q) * esize^2
+         pow <- pf(qf(.95,1,ddf),1,ddf,ncp,FALSE)
+         # cat("n=",nn," pow=",pow,"\n")
+         }# End while
+     findn <- nn
+     if(q==.5 && 2*trunc(findn/2)!=findn) findn <- findn+1
+     # Make sure findn is an even number if q = 1/2
+     findn
+     } # End function findn
>
>
> # equal sample sizes n, half a standard dev. difference bet means
> n <- 128 ; ddf <- n-2 ; vect <- numeric(n/2)+1
> x <- c(vect,numeric(n/2)) ; dif <- 1 ; sigma = 2; eff <- x*dif
>
> M <- 50 ; simp <- numeric(M) ; effsize <- numeric(M) ; phihat = numeric(M)
> estpow <- numeric(M) ; needn <- numeric(M)
> set.seed(4444)
>
>
```

```
> for(i in 1:M)
+     {
+     y <- eff+rnorm(n,0,sigma)
+     amat <- anova(lm(y~x))
+     simp[i] <- amat[1,5]
+     F <- amat[1,4]
+     phihat[i] = F
+     effsize[i] <- sqrt(4*F/n) # Based on F = t^2
+     estpow[i] <- 1-pf(qf(.95,1,ddf),1,ddf,F)
+     needn[i] <- findn(esize=effsize[i])
+     }# Next simulation
>
> sort(estpow)
 [1] 0.1432342 0.2696208 0.3150827 0.3817362 0.3873463 0.4794296 0.4835506 0.5207429
 [9] 0.5250893 0.5377409 0.5727125 0.5851788 0.5868119 0.6697578 0.6765977 0.7032535
[17] 0.7128237 0.7150001 0.7197754 0.7439522 0.7501208 0.7614079 0.7616633 0.8253608
[25] 0.8264381 0.8272442 0.8636311 0.8755966 0.8795982 0.8908869 0.8993721 0.9005881
[33] 0.9140435 0.9218954 0.9230968 0.9337530 0.9443918 0.9454180 0.9471108 0.9544974
[41] 0.9654680 0.9727600 0.9735055 0.9741944 0.9765474 0.9794381 0.9805759 0.9821376
[49] 0.9856131 0.9932533

> sort(needn) # With this effect size, n needed for power of 0.80
 [1]   54   60   62   64   64   66   68   68   68   72   78   80   80   82   86   90
[17]   90   92   96   98  100  104  106  108  120  120  120  142  142  146  148  156
[33]  158  158  162  172  174  212  212  218  238  244  248  272  274  356  362  456
[49]  550 1268

# Do it again with M = 500 simulations, and

> hist(estpow)
```



```
> hist(needn)  # Unfortunate bec of outlier(s)
```



```
> sort(needn)[1:20]
 [1] 30 32 38 42 42 42 42 42 42 44 44 44 46 48 48 48 48 48 48 48

> sort(needn)[480:500]
 [1]   550   562   572   616   650   730   740   742   758   776   840   860   864
[14]  1058  1108  1200  1268  1458  2634 11504 11654
```