Power through Simulation

Greg Snow

February 8, 2018

Motivation

Power

Two common questions in study design:

- How big of a sample do I need?
- What are my chances of a successful study?

Other Questions

Questions that should be asked more:

- What does *No Effect* look like
- What does *low power* look like
- Type I, Type II, Type S, and Type M errors
- Robustness to assumptions and their effect on power

Power Training

- Simple cases
  - One sample z-test with known variance
  - One sample test of proportions
- Pre Built tools

Pre Built Tools

- Simple problems only
- Complex/confusing Effect Size
- Hidden Assumptions
- Not Flexible Enough

Power

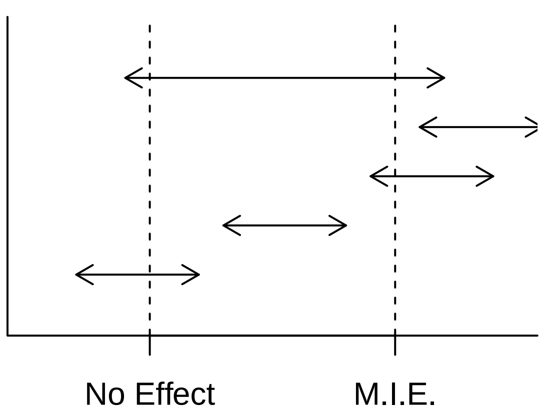
What is needed for Power

- Sample Size
  - Balanced/Unbalanced
  - Numbers of Nested Units
- Measure(s) of Variability
- Expected or Minimally Interesting Difference (Effect Size)
- Other assumptions, covariates, etc.

What is needed for Sample Size

- Desired Power(s)
- Measure(s) of Variability
- Expected or Minimally Interesting Difference (Effect Size)
- Other assumptions, covariates, etc.

Confidence Interval Approach



Examples

Examples

The following are problems where existing tools do not apply or were not satisfactory.

Non-normal Data

The Central Limit Theorem lets us use the t-test and regression when the population is not normal, but the sample size is large enough. But what is large enough and what effect does the non-normality have on the power?

Fishers Exact Test

Small sample contingency table where χ^2 analysis is unlikely to be accurate. Plan to use Fishers Exact test.

Binomial CI

Study was to show that a rare event had a rate under 4% (previous data suggested about 2%). Planned to find 95% CI using Binomial with a uniform prior and show that entire CI is below 4%

Regression

Full-Reduced Regression model. Do x_1-x_3 have a significant effect after accounting for x_4-x_8? With a few different assumed covariance structures.

Survival Analysis

Cox Proportional Hazards model with censored and truncated data. Compare different assumptions on amount of censoring.

Mixed Effects Model

Testing main effect and interaction on students who are nested in schools (Knowledge and Fitness are both outcomes).

Non-Inferiority

Non-inferiority studies declare success as long as the confidence interval on the parameter of interest is greater than an equivalence value.

Multivariate Response

When you have multiple outcome variables then tools like Wilks Lambda can be appropriate. But existing tools are only for univariate outcomes.

Power Through Simulation

4 Steps

1. Decide what your data is going to look like
2. Decide how you will analyze the data
3. Simulate data from 1, then analyze it using 2
4. Repeat 3 a bunch of times

The power is the proportion of times that the null is rejected.

(You may iterate a few times between 1 and 2)

Data

What will your data look like?

- What things are you likely to change?
  - Sample Size
  - Effect Size (difference between means)
  - Regression Coefficients
- Normal or other distribution(s)?
- What will be constant? (or does not matter)
  - Mean of group 1
  - Intercept
  - Variance/Standard Deviation
- Shape/structure of data

Analysis

- How will you analyze the data?
- What test will you use?
- What is your cut-off for significance?

Simulation

Write a function that will generate the data and analyze it.

Things that you will change should be function arguments.

Run the simulation a bunch of times

- Parallel processing if available.
- Run with the Null true to verify Type I error rate.
- Start with 100 runs until you find the parameter values of interest.
- Run 10,000 to 1,000,000 runs for the conditions of interest.

Simulation Examples

Starting with what we know

```
> power.t.test(n=300, delta=0.8, sd=3)
```

```

     Two-sample t test power calculation

              n = 300
          delta = 0.8
             sd = 3
      sig.level = 0.05
          power = 0.9033319
    alternative = two.sided

NOTE: n is number in *each* group
```

Simulating in R

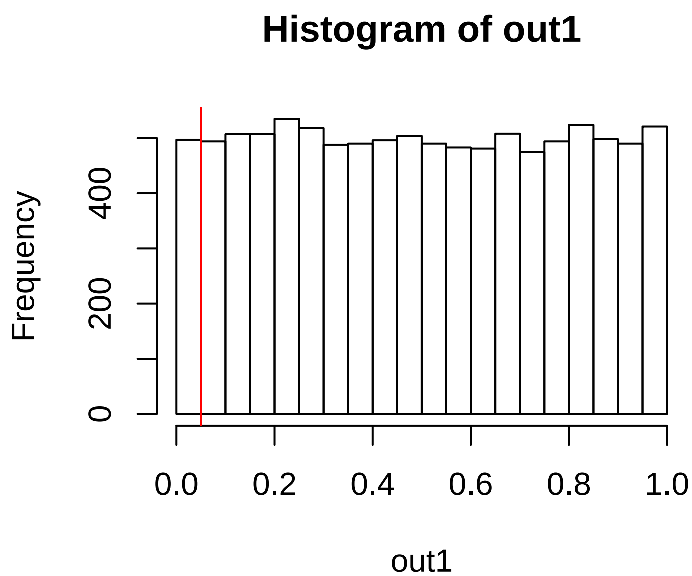
```
> simfun <- function(n=100, diff=0, sd=1) {
+   x1 <- rnorm(n, 0,    sd)
+   x2 <- rnorm(n, diff, sd)
+   t.test(x1, x2)$p.value
+ }
```

Simulating under the Null

```
> out1 <- replicate(10000, simfun(n=300, diff=0.0, sd=3))
> mean(out1 <= 0.05)
```

```
[1] 0.0497
```

```
> hist(out1)
> abline(v=0.05, col='red')
```

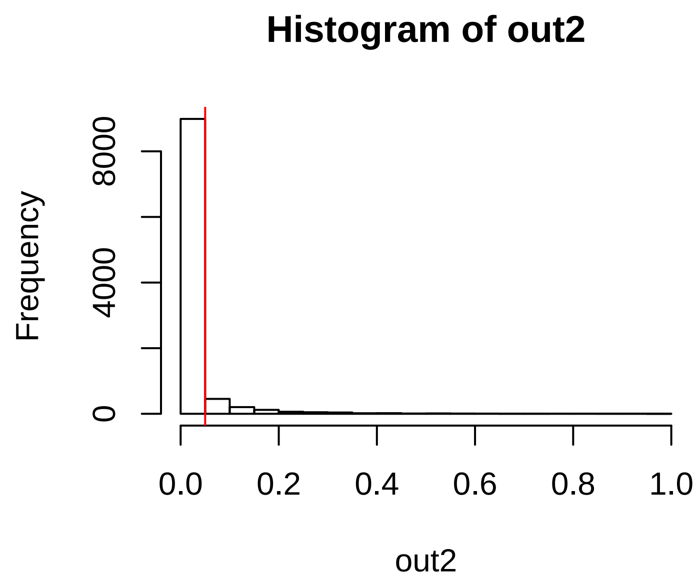


Simulating for Power

```
> out2 <- replicate(10000, simfun(n=300, diff=0.8, sd=3))
> mean(out2 <= 0.05)
```

```
[1] 0.8987
```

```
> hist(out2)
> abline(v=0.05, col='red')
```



Confidence Interval on the Power

```
> binom.test(sum(out2<=0.05), length(out2))


	Exact binomial test

data:  sum(out2 <= 0.05) and length(out2)
number of successes = 8987, number of trials
= 10000, p-value < 2.2e-16
alternative hypothesis: true probability of success is not equal to 0.5
95 percent confidence interval:
 0.8926197 0.9045470
sample estimates:
probability of success
                0.8987
```

Another Approach

```
> diff <- 0.8
> n <- 300
> sd <- 3
> nsim <- 10000
```

```
> x1 <- matrix(rnorm(n*nsim,0,sd), nrow=nsim)
> x2 <- matrix(rnorm(n*nsim,diff,sd), nrow=nsim)
> x <- cbind(x1,x2)
> out <- apply(x, 1, function(xx)
+   t.test(xx ~ rep(1:2, each=n))$p.value)
> mean(out <= 0.05)
```

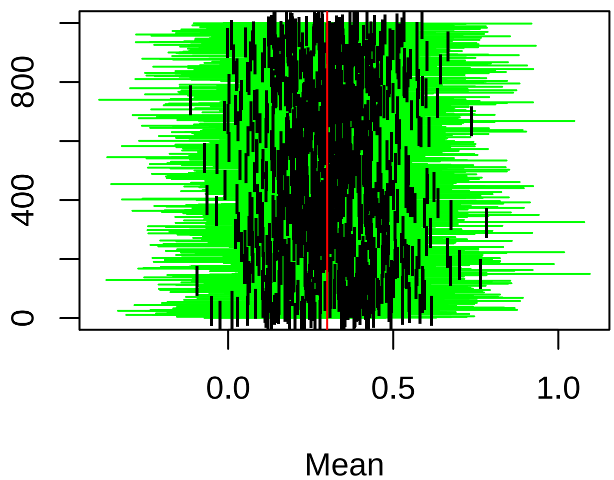
```
[1] 0.9027
```

Low Power

```
> simfun <- function(n=100, diff=0, sd=1) {
+   x <- rnorm(n, diff, sd)
+   out <- t.test(x)
+   c(p=out$p.value, mean=out$estimate, ci=out$conf.int)
+ }
>
> out <- replicate(1000, simfun(n=50, diff=0.3))
> mean(out[1,] < 0.05)
```

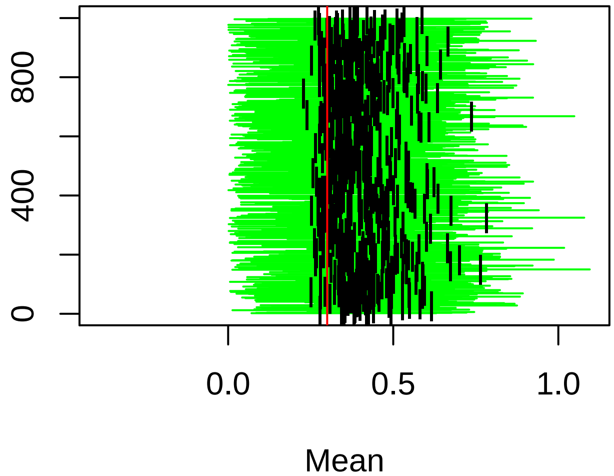
```
[1] 0.553
```

```
> plot(out[2,], seq_len(ncol(out)),
+      xlim=range(out[3:4,]), type='n',
+      ylab='', xlab='Mean')
> segments(x0=out[3,], x1=out[4,], y0=seq_len(ncol(out)),
+          col='green')
> points(out[2,], seq_len(ncol(out)), pch='|')
> abline(v=0.3, col='red')
```



Low Power Conditional

```
> w <- out[1,] < 0.05
> plot(out[2,], seq_len(ncol(out)),
+      xlim=range(out[3:4,]), type='n',
+      ylab='', xlab='Mean')
> segments(x0=out[3,], x1=out[4,], y0=seq_len(ncol(out)),
+          col=ifelse(w, 'green',NA))
> points(out[2,], seq_len(ncol(out)), pch=ifelse(w,'|',''))
> abline(v=0.3, col='red')
```



Low Power Conditional

```
> mean( out[2,w] > 0.3 )
```

```
[1] 0.8951175
```

```
> mean( out[2,w] > 0.6 )
```

```
[1] 0.03435805
```

```
> mean( out[3,w] > 0.3 )
```

```
[1] 0.05605787
```

Mixture of Normals

```
> simfun <- function(n=100, diff=0, sd=2, p=0.1) {
+   x <- rnorm(n, diff,
+               ifelse(rbinom(n,1,p), sd, 1))
+   c(t.test(x)$p.value,wilcox.test(x)$p.value)
+ }
> out1 <- replicate(10000, simfun(n=30, diff=0))
> rowMeans(out1 <= 0.05)
```

```
[1] 0.0477 0.0500
```

```
> out2 <- replicate(10000, simfun(n=30, diff=0.5))
> rowMeans(out2 <= 0.05)
```

```
[1] 0.6549 0.6732
```

Fishers Exact Test

```
> simfun <- function(n=50, p=c(0.5, 0.5, 0.5)) {
+   x1 <- sample(1:3, n, replace=TRUE)
+   x2 <- rbinom(n, 1, p[x1])
+   fisher.test( table(x1,x2) )$p.value
+ }
> out1 <- replicate(10000,
+                    simfun(n=50, p=c(0.5, 0.5, 0.5)))
> out2 <- replicate(10000,
+                    simfun(n=50, p=c(0.5, 0.3, 0.7)))
> c(mean(out1<=0.05), mean(out2<=0.05))
```

```
[1] 0.0473 0.5188
```

Beta Binomial CI

```
> simfun <- function(n=100, true.p=0.02, test.p=0.04,
+                     alpha=0.05) {
+   x <- rbinom(1, n, true.p)
+   qbeta(1-alpha/2, x+1, n-x+1) < test.p
+ }
> out1 <- replicate(10000, simfun(n=1000, true.p=0.04))
> mean(out1)
```

```
[1] 0.0171
```

```
> out2 <- replicate(10000, simfun(n=1000))
> mean(out2)
```

```
[1] 0.9469
```

Regression

```
> library(MASS)
> simfun <- function(n=100, beta=rep(0,9),
+                    mu.x=rep(0,8), var.x=diag(8),
+                    sig=1) {
+   x <- mvrnorm(n, mu.x, var.x)
+   y <- cbind(1,x) %*% beta + rnorm(n,0,sig)
+   mydat <- cbind( as.data.frame(x), y)
+   names(mydat) <- c(paste0('x',1:8), 'y')
+   fit1 <- lm(y ~ x4 + x5 + x6 + x7 + x8, data=mydat)
+   fit2 <- lm(y ~ ., data=mydat)
+   anova(fit1, fit2)[2,6]
+                    }
```

Logistic Regression

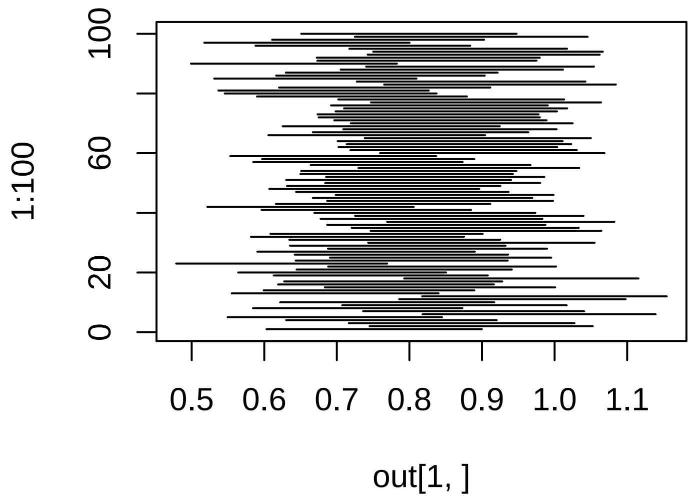
```
> simfun <- function(n=1000, b0=0, b1=0) {
+   x <- rnorm(n)
+   eta <- b0 + b1*x
+   p <- exp(eta)/(1+exp(eta))
+   y <- rbinom(n,1,p)
+   fit <- glm(y~x, family=binomial)
+   coef(summary(fit))[2,4]
+ }
```

Confidence Interval Width

```
> library(MASS)
> simfun <- function(n=1000, b0=0, b1=0) {
+   x <- rnorm(n)
+   eta <- b0 + b1*x
+   p <- exp(eta)/(1+exp(eta))
+   y <- rbinom(n,1,p)
+   fit <- glm(y~x, family=binomial)
+   ci <- confint(fit,2)
+   c(ci, ci[2]-ci[1])
+ }
>
> out <- replicate(100, simfun(b1=0.8))
> apply(out, 1, range)
```

```
         2.5 %    97.5 %    97.5 %
[1,] 0.4785973 0.7693356 0.2787030
[2,] 0.8181014 1.1545843 0.3370907
```

```
> plot(out[1,], 1:100, type='n', xlim=range(out[1:2,]))
> segments(out[1,],1:100, out[2,])
```



Cox Model Regression

```
> library(survival)
> simfun <- function(n=100, beta = 0, cens.scale=1) {
+   x <- rnorm(n, 10, 2)
+   y <- rexp(n, 1/(1+beta*x))
+   cens <- rexp(n, 1/cens.scale)
+   time <- pmin(y,cens)
+   status <- ifelse(y<cens, 1, 0)
+ #print(table(status))
+   fit <- coxph(Surv(time,status) ~ x)
+   coef(summary(fit))[1,5]
+ }
```

Mixed Effects Model

```
> library(lme4)
> simfun <- function(n.student=100, n.school=20,
+                    sig.student=1, sig.school=2,
```

```
+                     b0=0, b1=0, b2=0, b12=0) {
+   x1 <- rnorm(n.student*n.school, 0, 1)
+   x2 <- rbinom(n.student*n.school, 1, 0.5)
+   re.school <- rnorm(n.school,0,sig.school)
+   school.id <- rep(1:n.school, each=n.student)
+   y <- b0 + b1*x1 + b2*x2 + b12*x1*x2 +
+     re.school[school.id] +
+     rnorm(n.student*n.school,0,sig.student)
+   fit1 <- lmer( y ~ x1 + (1|school.id))
+   fit2 <- lmer( y ~ x1*x2 + (1|school.id))
+   anova(fit1,fit2)[2,8]
+ }
```

Multiple Sample Sizes

```
> simfun <- function(n=c(10,20,30,50,75,100), diff=0.4) {
+   x <- rnorm(max(n), diff, 1)
+   sapply(n, function(nn) t.test(x[seq_len(nn)])$p.value)
+ }
>
> out <- replicate(1000, simfun())
> rbind( c(10,20,30,50,75,100),
+   apply(out, 1, function(x) mean(x<=0.05)))
```

```
       [,1]   [,2]   [,3]   [,4]   [,5]    [,6]
[1,] 10.000 20.000 30.000 50.000 75.000 100.000
[2,]  0.237  0.403  0.571  0.791  0.926   0.977
```

Paralell Processing

```
> library(parallel)
> cl <- makeCluster(4)
> clusterSetRNGStream(cl, 20160405)
> clusterExport(cl, "simfun")
> simfun2 <- function(i,...) simfun(...)
> out <- parSapply(cl, 1:1000, FUN=simfun2, diff=0.6)
> rbind( c(10,20,30,50,75,100),
+   apply(out, 1, function(x) mean(x<=0.05)))
```

```
       [,1]   [,2]   [,3]   [,4] [,5] [,6]
[1,] 10.000 20.000 30.000 50.000   75  100
[2,]  0.435  0.753  0.908  0.988    1    1
```

Questions?