Simulating Power

Matthew Wheeler

Up till now

We have dealt with power in idealized settings:

1. Known distributions i.e. t-test
2. Assumed normality
3. Other toy problems where the math is easy (relatively speaking).

In the real world

You will not always have such idealized settings. As we learned last week, many tests are likelihood ratio tests. Such tests:

1. Are based upon asymptotic approximations.
2. May not behave as you would expect in small sample situations.
3. May have competing tests, and you want to figure out which is “more powerful.”

Simulation gives us a solution

Simulation puts us a step closer to reality:

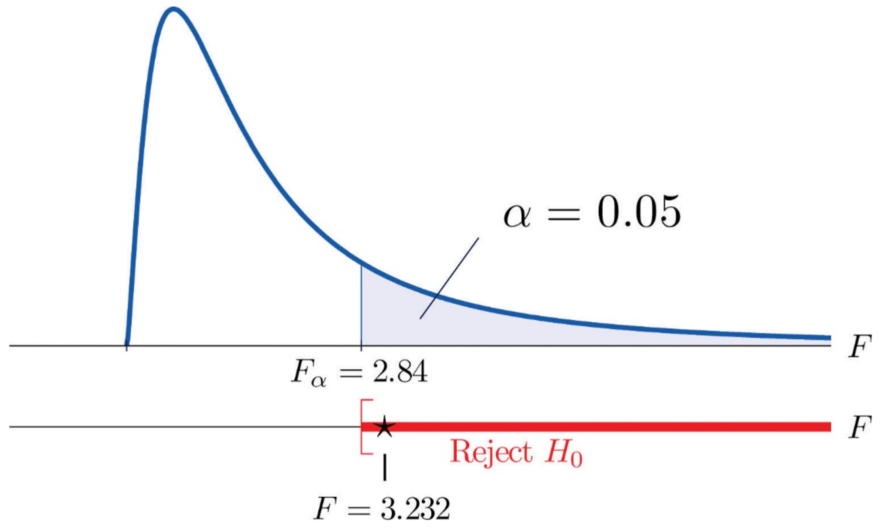
1. We can make our data distribution following some non-normal distribution.
2. We choose any test we like after the fit of the data.
3. We can see what happens when our assumptions are violated.

Steps to a simulating power:

1. Determine the hypothetical data generating mechanism (normal, binomial, Poisson etc.)
2. Determine the ‘true parameters’ this includes:
    - Means
    - Variances
3. Determine the tests of interest (e.g. Likelihood Ratio, F-test)
4. Code it up!

Remember

Power is the probability I reject the null, so what I do is randomly simulate an experiment a large number of times and count the number of times I reject the null. From this count, we can estimate power. The more simulations = the more accurate estimate of power.



Example 1

Logistic Regression : A toxin is associated with increased mortality in zebrafish. We are interested in estimating the dose-response curve related to the toxin.

Hypothesis: Is there a positive relationship between the toxin and zebrafish mortality?

Assume:

1) We have 30 broods of zebrafish per tank.
2) The number of deaths are binomially distributed with probability p.
3) Background death rate is 10%, so the log odds are log(.1/.9) = -2.197.
4) Each brood is dosed a uniform random amount from 0 ug to 2ug
5) Assume logistic regression, so the dose response is

   logit(p(dose)/[1-p(dose)]) = b0 + b1 dose

Generating Binomial Random Variables

rbinom(number,size,p)

number – number of independent binomial experiments

size – The size of the binomial experiment (brood size of 30)

p - the probability of a ‘success’

Fitting the model in R

glm(cbind(y,n) ~ x,family=“binomial)

cbind(y,n) <- like SAS y is the number of successes. n size of the experiment

family = “binomial” by default logistic regression is used for the “binomial family”

Extracting Tests

summary(fit)$coefficients	- this will extract the coefficients
vcov(fit)	- this will extract the variance covariance of the coefficients. Needed in certain situations
Anova(fit,type=3)	- from the package ‘Car’ will provide an analysis of deviance test

One simulation in R

```
x <- runif(n,0,2) #my x values fall raondomly on the interval [0,1]
beta0 <- -2.197225     # background death rate

logit <- beta0 + betas1[jj]*x #select the betas
p <- 1/(1+exp(-logit))
y <- rbinom(n,N,p)       # our 'model' is a binomial model with logistic regression

fit.glm <- glm(cbind(y,N)~x,family="binomial") # defaults to logistic regression
test <- Anova(fit.glm,test="LR",type=3) #Analysis of Deviance

pval <- as.numeric(test[1,3]) #extract the p-value for the Analysis of Deviance for WALD

tests[ii,jj] = (pval < alpha)      #test
```

Example 2

The number of speeding tickets in 10 years is thought to relate to the number of miles per over (on average) some one goes.

ASSUME: Count data, log rate, Poisson

Log-rate = b0 + b1 speed

Fitting a poison model

In R, fitting a poison log-linear model is essentially the same using glm as it is in genmod

```
glm(y~x,family="poisson")
```

All you have to do differently is say the data are “poisson.”

One Iteration of the simulation

```
x <- runif(n,0,10) #my x values fall raondomly on the interval [0,10]
                   # so we have average miles over between 0 and 10mph
beta0 <- 1         #my betas - we need to specify them just like proc glmpower

logit <- beta0 + betas1[jj]*x #select the betas
lambda <- exp(logit)
y <- rpois(n,lambda)      # our 'model' is a poisson model with log-linear regression

fit.glm <- glm(y~x,family="poisson") # defaults to log-linear regression
test <- Anova(fit.glm,test="LR",type=3) #Analysis of Deviance use "LR"

pval <- as.numeric(test[1,3]) #extract the p-value for the Analysis of Deviance
tests[ii,jj] = (pval < alpha) # significant if the p-value is less than alpha
```

What about our project?

```
x <- rep(1:4,n/4) #my x values fall raondomly on the interval [0,1]
beta0 <-  -3.59512     #my betas - we need to specify them just like proc glmpower


logit <- beta0 + betas1[x] #select the betas
p <- 1/(1+exp(-logit))
y <- rbinom(n,N,p)       # our 'model' is a binomial model with logistic regression


 fit.glm <- glm(cbind(y,N)~as.factor(x),family="binomial") # defaults to logistic regression
test2 <- summary(fit.glm)$coefficients #extract estimated coefficients from the table
test <- Anova(fit.glm,test="LR",type=3) #Analysis of Deviance
pval <- as.numeric(test[1,3]) #extract the p-value for the Analysis of Deviance for WALD


tests[ii,1] = (pval < alpha) # significant if the p-value is less than alpha - overall test
tests[ii,2] = (test2[2,4] < alpha) #  significant if the p-value is less than alpha - first parameter
tests[ii,3] = (test2[3,4] < alpha) #  significant if the p-value is less than alpha - second parameter
tests[ii,4] = (test2[4,4] < alpha) #  significant if the p-value is less than alpha - third parameter
```