

Random Independent Variables

Example: $e_1, e_2, e_3 \sim \text{Poisson}(5)$
 $X_1 = e_1 + e_3$ $X_2 = e_2 + e_3$ $X_3 \sim N(10,10)$ ind. of e_1, e_2, e_3
 $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \varepsilon$ $\varepsilon \sim N(0, \sigma^2)$
 $\beta_0 = 0$ $\beta_1 = 1$ $\beta_2 = 0.1$ $\beta_3 = 0.2$ $\sigma^2 = 25$

First a crude Monte Carlo: Simulate \mathbf{X} and $\boldsymbol{\varepsilon}$, generate \mathbf{Y} , do the test, repeat.

```
#####
# randpow0.R
#           A brutal simulation to estimate Power for a
#           normal linear model with 3 random IVs.
#           H0: beta2=beta3=0
#
# Run with   source("randpow0.R")
#
#####

merror <- function(phat,m,alpha=0.01)
# (1-alpha)*100% margin of error for a proportion
{
  z <- qnorm(1-alpha/2)
  merror <- z * sqrt(phat*(1-phat)/m) # m is (Monte Carlo) sample size
  merror
} # End function merror
#set.seed(12345)
n <- 100 ; m <- 1000 ; signif <- numeric(m)
b0 <- 0 ; b1 <- 1 ; b2 <- .1 ; b3 <- .2
sigma <- 5 ; confid <- .99

for(i in 1:m)
{
  e1 <- rpois(n,5) ; e2 <- rpois(n,5) ; e3 <- rpois(n,5)
  x1 <- e1+e3 ; x2 <- e2+e3 ; x3 <- rnorm(n,10,sqrt(10))
  epsilon <- rnorm(n,0,sigma)
  y <- b0 + b1*x1 + b2*x2 + b3*x3 + epsilon
  mod1 <- lm(y~x1) ; mod2 <- lm(y~x1+x2+x3)
  signif[i] <- anova(mod1,mod2)[2,6] < 0.05
}
Phat <- mean(signif)
cat("\n")
cat("      A brutal simulation to estimate Power for a \n")
cat("      normal linear model with 3 random IVs.      \n")
cat("\n")
cat(" n,m = ",n,m,"\n")
cat(" b0, b1, b2, b3, sigma = ",b0, b1, b2, b3, sigma, "\n")
cat("\n")
cat(" Estimated power = ",Phat,", with ",confid*100,"% margin of error \n")
cat(merror(Phat,m,1-confid),".\n"); cat("\n")
```

```
#####
# randpow1.R
#           Power for the normal linear model, with random
#           independent variables. Run with source("randpow1.R")
#
#####

#set.seed(12345)
n <- 100           # Sample size
m <- 50           # Monte Carlo sample size
beta <- c(0,1,.1,.2) # Parameters
sigma <- 5
C <- rbind( c(0,0,1,0), # H0: C Beta = 0
            c(0,0,0,1) )
alpha <- 0.05     # Significance level of test
confid <- .99     # Want 100*confid percent margin of
                  # error for MC estimate
#####
# One-time-only calculations
Y <- numeric(m)
eff <- C%*%beta
q <- dim(C)[1] ; r <- dim(C)[2]
##### Monte Carlo Loop #####
for(i in 1:m)
  {
  # Generate Random IVs and bind into X matrix. This will
  # vary from problem to problem.
    e1 <- rpois(n,5) ; e2 <- rpois(n,5) ; e3 <- rpois(n,5)
    x1 <- e1+e3 ; x2 <- e2+e3 ; x3 <- rnorm(n,10,sqrt(10))
    X <- cbind(matrix(1,n),x1,x2,x3)
  # Calculate non-centrality parameter
    xpxinv <- solve(t(X)%*%X)
    kore <- solve(C%*%xpxinv%*%t(C))
    ncp <- t(eff)%*%kore%*%eff/sigma^2
    Y[i] <- 1 - pf(qf(1-alpha,q,n-r),q,n-r,ncp)
  }
##### End of Monte Carlo Loop #####

Phat <- mean(Y) ; SDhat <- sqrt(var(Y))
margin <- qnorm(1-(1-confid)/2)*SDhat/sqrt(m)
cat("\n")
cat("      Monte Carlo integration to estimate Power for a \n")
cat("      normal linear model.      \n")
cat("\n")
cat(" n,m = ",n,m,"\n")
cat(" Beta = ",beta, "\n")
cat(" Sigma = ",sigma, "\n")
cat("\n")
cat(" Estimated power = ",Phat,", with ",confid*100,"% margin of error \n")
cat(margin,".\n")
cat("\n")
```

Now compare output from the randpow0 and randpow1

```
> howlong0 <- system.time(source("randpow0.R"))
```

```
      A brutal simulation to estimate Power for a  
      normal linear model with 3 random IVs.
```

```
n,m = 100 1000  
b0, b1, b2, b3, sigma = 0 1 0.1 0.2 5
```

```
Estimated power = 0.209 , with 99 % margin of error  
0.03311910 .
```

```
> howlong0
```

```
[1] 40.18  8.52 50.72  0.00  0.00
```

```
> help(system.time)
```

Value:

```
      A numeric vector of length 5 containing the user cpu, system cpu,  
      elapsed, subproc1, subproc2 times. The subproc times are the user  
      and system cpu time used by child processes (and so are usually  
      zero).
```

```
> sum(howlong0[1:2])
```

```
[1] 48.7
```

```
> howlong1 <- system.time(source("randpow1.R"))
```

```
      Monte Carlo integration to estimate Power for a  
      normal linear model.
```

```
n,m = 100 50  
Beta = 0 1 0.1 0.2  
Sigma = 5
```

```
Estimated power = 0.2051531 , with 99 % margin of error  
0.009545068 .
```

```
> sum(howlong1[1:2])
```

```
[1] 0.28
```

```
> # Trying to get precision from randpow0; set m = 10,000
```

```
> howlong0b <- system.time(source("randpow0.R"))
```

```
Estimated power = 0.2112 , with 99 % margin of error  
0.01051350 .
```

```
> sum(howlong0b[1:2]) # About 8 CPU minutes
```

```
[1] 489.5
```

```
#####
# randpow2.R
#
#           Approximate Power for the normal linear model,
#           with random independent variables. X-prime-X/n
#           goes a.s. to a constant. Put the constant in.
#
#           Run with source("randpow2.R")
#
#####

n <- 100           # Sample size
beta <- c(0,1,.1,.2) # Parameters
sigma <- 5
C <- rbind( c(0,0,1,0), # H0: C Beta = 0
            c(0,0,0,1) )
alpha <- 0.05      # Significance level of test
# Call this the "MOMent MATrix"
# X-prime-X divided by n goes to this a.s.

mommat <- rbind( c( 1, 10, 10, 10),
                 c(10,110,105,100),
                 c(10,105,110,100),
                 c(10,100,100,110)
                 )

#
# Calculate non-centrality parameter
xpxinv <- solve(mommat)
kore <- solve(C%*%xpxinv%*%t(C))
eff <- C%*%beta
ncp <- n * t(eff)%*%kore%*%eff/sigma^2
q <- dim(C)[1] ; r <- dim(C)[2]
# Estimated Power is still called P-Hat
Phat <- 1 - pf(qf(1-alpha,q,n-r),q,n-r,ncp)
cat("\n")
cat("      Rough Power guess for a normal linear model.\n")
cat("\n")
cat(" n = ",n," \n")
cat(" Beta = ",beta, " \n")
cat(" Sigma = ",sigma, " \n")
cat("\n")
cat(" Estimated power = ",Phat," \n")
cat("\n")
```

Compare output of randpow1 and randpow2

```
> # Want a bit more precision. Go into randpow1.R and set m = 200.  
>  
> howlong1b <- system.time(source("randpow1.R"))
```

```
Monte Carlo integration to estimate Power for a  
normal linear model.
```

```
n,m = 100 200  
Beta = 0 1 0.1 0.2  
Sigma = 5
```

```
Estimated power = 0.2082554 , with 99 % margin of error  
0.004648202 .
```

```
> sum(howlong1b[1:2])  
[1] 1.09
```

```
> source("randpow2.R")
```

```
Rough Power guess for a normal linear model.
```

```
n = 100  
Beta = 0 1 0.1 0.2  
Sigma = 5
```

```
Estimated power = 0.2104835
```

This is promising. randpow1b gave us Estimated power = 0.2082554. As the sample size increases, my rough method should get even better. Let's find the sample size needed for power of 0.80, both ways. First the rough way because it's faster. Just the end of the search:

```
> source("randpow2.R")
```

```
Rough Power guess for a normal linear model.
```

```
n = 510  
Beta = 0 1 0.1 0.2  
Sigma = 5
```

```
Estimated power = 0.799915
```

```
> source("randpow2.R")
```

```
Rough Power guess for a normal linear model.
```

```
n = 511  
Beta = 0 1 0.1 0.2  
Sigma = 5
```

```
Estimated power = 0.800743
```

Now with randpow1 -- true (though still estimated) power

```
> source("randpow1.R")
```

```
Monte Carlo integration to estimate Power for a  
normal linear model.
```

```
n,m = 511 200  
Beta = 0 1 0.1 0.2  
Sigma = 5
```

```
Estimated power = 0.795004 , with 99 % margin of error  
0.005473328 .
```

```
> source("randpow1.R")
```

```
Monte Carlo integration to estimate Power for a  
normal linear model.
```

```
n,m = 515 200  
Beta = 0 1 0.1 0.2  
Sigma = 5
```

```
Estimated power = 0.8000344 , with 99 % margin of error  
0.004817262 .
```

Compared to $n = 511$ for the rough method.

Logistic Regression

Give me some probabilities and I will give you logistic regression coefficients: Example for 2 IVs. $H_0: \beta_2=0$

Say to the client,

If x_1 and x_2 both equal 70, what's $P(Y=1)$?

If $x_1=90$ and $x_2=70$, what's $P(Y=1)$?

If x_1 and x_2 both equal 90, what's $P(Y=1)$?

Probably you should ask more questions than the number of regression coefficients. That's why `lsfit` is used below instead of `solve`.

```
> x1 <- c(70,90,90) ; x2 <- c(70,70,90) ; pi <- c(.2,.7,.8)
> lodds <- log(pi/(1-pi))
> cbind(x1,x2,pi,lodds)
      [,1] [,2] [,3] [,4]
[1,]    70    70  0.2 -1.3862944
[2,]    90    70  0.7  0.8472979
[3,]    90    90  0.8  1.3862944
> tellme <- lsfit(cbind(x1,x2),lodds)

> tellme$coef
      Intercept          X1          X2
-11.09035489   0.11167961   0.02694983
> # Checking ...
> xb <- sum(tellme$coef*c(1,70,70))
> exp(xb)/(1+exp(xb))
[1] 0.2
> xb <- sum(tellme$coef*c(1,90,70))
> exp(xb)/(1+exp(xb))
[1] 0.7
> xb <- sum(tellme$coef*c(1,90,90))
> exp(xb)/(1+exp(xb))
[1] 0.8
```

Do just one simulation, showing details

```
>
> # Function merror is already defined.
> n <- 500 ; m <- 100 ; signif <- numeric(m)
> beta <- c(-11.09035489,0.11167961,0.02694983)
> confid <- .99
> ##### One-time-only calculations #####
> critval <- qchisq(.95,1)
> # Calculate square root matrix A. A%*%Z will have var-cov Sigma
> sigma <- rbind( c(50,sqrt(2000)/2),
+               c(sqrt(2000)/2,40) )
>
> sigma
      [,1]      [,2]
[1,] 50.00000 22.36068
[2,] 22.36068 40.00000
>
> spec <- eigen(sigma)
>
> spec
$values
[1] 67.91288 22.08712

$vectors
      [,1]      [,2]
[1,] -0.7804543  0.6252128
[2,] -0.6252128 -0.7804543

> A <- spec$vectors %*% diag(sqrt(spec$values))

>
> A%*%t(A)
      [,1]      [,2]
[1,] 50.00000 22.36068
[2,] 22.36068 40.00000
>
```



```

> #
> # This will be inside Monte Carlo loop -- do it once to check
> #
>     Z <- rbind(rnorm(n),rnorm(n))
>     X <- cbind(matrix(1,n),t(A%%Z))
>     X[,2] <- X[,2]+70 ; X[,3] <- X[,3]+80
>     # Okay that's X. Now simulate Y
>     xb <- X %% beta
>     pi <- exp(xb)/(1+exp(xb))
>     Y <- rbinom(n,1,pi)
>
>     fullmod <- glm(Y ~ X[,2:3], family=binomial ) # Full model
> summary(fullmod)

```

```

Call:
glm(formula = Y ~ X[, 2:3], family = binomial)

```

```

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.5067  -0.7739  -0.5760   0.7411   2.6692

```

```

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -11.06211    1.72105  -6.428 1.30e-10 ***
X[, 2:3]1     0.09372     0.01821   5.147 2.65e-07 ***
X[, 2:3]2     0.04106     0.02119   1.938  0.0527 .
---

```

```

Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

```

(Dispersion parameter for binomial family taken to be 1)

```

```

Null deviance: 564.52  on 499  degrees of freedom
Residual deviance: 512.30  on 497  degrees of freedom
AIC: 518.3

```

```

Number of Fisher Scoring iterations: 3

```

```

>     redmod <- glm(Y ~ X[,2], family=binomial ) # Reduced model
>     anova(redmod,fullmod)

```

```

Analysis of Deviance Table

```

```

Model 1: Y ~ X[, 2]
Model 2: Y ~ X[, 2:3]
  Resid. Df Resid. Dev  Df Deviance
1       498     516.07
2       497     512.30   1     3.77

```

```

[1] 3.841459

```

```

> anova(redmod,fullmod)[2,4]>critval

```

```

[1] FALSE

```

```

>

```

```

#####
# logpow0.R
#           A brutal simulation to estimate Power for a
#           logistic regression model with 2 random IVs.
#
# Run with   source("logpow0.R")
#
#####

merror <- function(phat,m,alpha=0.01)
# (1-alpha)*100% margin of error for a proportion
{
  z <- qnorm(1-alpha/2)
  merror <- z * sqrt(phat*(1-phat)/m) # m is (Monte Carlo) sample size
  merror
} # End function merror
#set.seed(12345)
n <- 100 ; m <- 200 ; signif <- numeric(m)
beta <- c(-11.09035489,0.11167961,0.02694983)
confid <- .99
##### One-time-only calculations #####
critval <- qchisq(.95,1)
# Calculate square root matrix A. A%*%Z will have var-cov Sigma
sigma <- rbind( c(50,sqrt(2000)/2),
               c(sqrt(2000)/2,40) )
spec <- eigen(sigma)
A <- spec$vectors %*% diag(sqrt(spec$values))
##### Monte Carlo Loop #####
for(i in 1:m)
{
  # Simulate X
  Z <- rbind(rnorm(n),rnorm(n))
  X <- cbind(matrix(1,n),t(A%*%Z))
  X[,2] <- X[,2]+70 ; X[,3] <- X[,3]+80
  # Okay that's X. Now simulate Y
  xb <- X %*% beta
  pi <- exp(xb)/(1+exp(xb))
  Y <- rbinom(n,1,pi)
  fullmod <- glm(Y ~ X[,2:3], family=binomial ) # Full model
  redmod <- glm(Y ~ X[,2], family=binomial ) # Reduced model
  signif[i] <- anova(redmod,fullmod)[2,4]>critval
}
##### End Monte Carlo Loop #####
Phat <- mean(signif)
cat("\n")
cat("      Simulation to estimate Power for a \n")
cat("      logistic regression model with 2 random IVs. \n")
cat("\n")
cat(" n,m = ",n,m,"\n")
cat(" Beta = ",beta, "\n")
cat(" Sigma = \n")
print(sigma)
cat("\n")
cat(" Estimated power = ",Phat,", with ",confid*100,"% margin of error \n")
cat(merror(Phat,m,1-confid),".\n")
cat("\n")
##### End of logpow0.R #####

```

```
> howlong <- system.time(source("logpow0.R")) ; sum(howlong[1:2])
```

Simulation to estimate Power for a
logistic regression model with 2 random IVs.

```
n,m = 100 200
Beta = -11.09035 0.1116796 0.02694983
Sigma =
      [,1]      [,2]
[1,] 50.00000 22.36068
[2,] 22.36068 40.00000
```

```
Estimated power = 0.11 , with 99 % margin of error
0.05698931 .
```

```
[1] 24.81
```

Step	Sample size n	MC sample size m	Estimated power	99% Margin of error	CPU Time (Sec.)
1	100	200	0.11	0.06	24.81
2	500	200	0.29	0.08	45.81
3	700	200	0.345	0.087	56.62
4	1500	200	0.665	0.086	104.84
5	1800	200	0.710	0.083	124.83
6	2000	200	0.735	0.080	*
7	2500	200	0.855	0.064	172.58
8	2250	500	0.832	0.043	384.5
9	2168	500	0.816	0.045	365.69
10	2170	2000	0.817	0.02	1466.73
11	2135	2000			