The Stata Journal (2002)
2, Number 2, pp. 107–124

Power by simulation

A. H. Feiveson
NASA Johnson Space Center
alan.h.feiveson1@jsc.nasa.gov

Abstract. This paper describes how to write Stata programs to estimate the power of virtually any statistical test that Stata can perform. Examples given include the *t* test, Poisson regression, Cox regression, and the nonparametric rank-sum test.

Keywords: st0010, power, simulation, random number generation, postfile, copula, sample size

1 Introduction

Statisticians know that in order to properly design a study producing experimental data, one needs to have some idea of whether the scope of the study is sufficient to give a reasonable expectation that hypothesized effects will be detectable over experimental error. Most of us have at some time used published procedures or canned software to obtain sample sizes for studies intended to be analyzed by *t* tests, or even analysis of variance. However, with more complex methods for describing and analyzing both continuous and discrete data (for example, generalized linear models, survival models, selection models), possibly with provisions for random effects and robust variance estimation, the closed-form expressions for power or for sample sizes needed to achieve a certain power do not exist. Nevertheless, Stata users with moderate programming ability can write their own routines to estimate the power of virtually any statistical test that Stata can perform.

2 General approach to estimating power

2.1 Statistical inference

Before describing methodology for estimating power, we first define the term "power" and illustrate the quantities that can affect it. In this article, we restrict ourselves to classical (as opposed to Bayesian) methodology for performing statistical inference on the effect of experimental variable(s) X on a response Y. This is accomplished by calculating a p-value that attempts to probabilistically describe the extent to which the data are consistent with a null hypothesis, H_0, that X has no effect on Y. We would then reject H_0 if the p-value is less than a predesignated threshold α. It should be pointed out that considerable criticism of the use of p-values for statistical inference has appeared in the literature; for example, Berger and Sellke (1987), Loftus (1993), and Schervish (1996). However, for this article, we accept this methodology as the modus operandi.

2.2 Defining power

In simple terms, the power of the above test on the effect of X is defined as the probability that H_0 is rejected; however, it makes no sense to talk about a quantitative "probability" without reference to some stochastic model. By defining power as the probability of some event, we imply that the event, namely rejection of H_0, is random. More specifically, we contemplate a hypothetical scenario in which the identically sized experiment could be run over and over, each time collecting new data and doing a new hypothesis test. If this scenario can be adequately modeled, we may thus estimate power by simulating data from multiple replications of the experiment and simply calculate the proportion of rejections as an estimate of the power. What does it mean to model the experimental scenario? For our purposes, it means that the experiment design, the values of X, and the distribution of Y must be sufficiently specified such that the resulting power as a function of α is uniquely determined. Let us examine each of these scenario components:

1. **Experiment design.** The scope of the experiment must be specified in terms of definition of experimental units, overall sample size, crossed, hierarchical or multilevel structure (if applicable) with specified sample sizes at each level, and so forth. This allows one to generate simulated data that could be matched to each observation of Y in an actual experiment.

2. **Values of X.** Some components of X may be determined by the design; for example, dummy variables in ANOVA. Values of others, such as independent variables in regression models, must be specified, since these in general affect power.

3. **Distribution of Y.** The distribution of Y must be completely specified for all values of the covariates in the dataset. In fully parametric models, this means specifying the distributional family and all parameter values that correspond to an assumed departure from H_0. There may also be other parameters (for example, variances and covariances) that affect power. Values of these *nuisance parameters* must also be specified. For semiparametric models such as Stata's `xtgee` or `stcox`, or for nonparametric analysis, a specific model is still needed to simulate the data, even though that model is not necessarily assumed in the analysis (see the example in Section 3.3). These requirements may give the impression of being overbearing or not feasible, but must nevertheless be satisfied to calculate power, whether by simulation or pure mathematical analysis. For example, in a two-sample t test, one assumes two normal distributions (distributional family) with the same specified variance (nuisance parameter) and a given hypothetical difference between the means (departure from H_0).

2.3 Bias

In the ideal situation, the p-value should be calculated such that the probability of rejection is exactly α when H_0 is true. Since α is arbitrary, this implies that the distribution of the p-value over many replications of the experiment should be uniform on [0,1].

However, in many complex analyses, p-values are based on asymptotic approximations, and the resulting probability of rejection under H_0 may not be exactly α. In this case, the test is said to be biased. In the process of estimating power, it is often desirable to include a case when H_0 holds, to check for bias. This process is illustrated in the examples of Section 3.

2.4 Outline of the method

A step-by-step process for carrying out power estimation in Stata would then resemble the following:

1. Use the underlying model to generate random data with (a) specified experimental design and sample size(s), (b) values of X, (c) parameter values that express the distribution of Y under the alternative hypothesis one is trying to detect, and (d) values of nuisance parameters such as variances.

2. Run the Stata estimation program (`regress`, `glm`, etc.) on these randomly generated data.

3. Retrieve the p-value directly, or calculate it yourself. For some Stata commands such as `tabi`, the p-value is directly retrievable as a "return" quantity (you can type `return list` after running the command to see if this is the case). In many estimation commands, the p-value is obtained by a z test; that is, assuming the parameter estimate divided by its standard error has a standard normal distribution. In this case, you can retrieve the parameter estimates and standard errors using `e(b)` and `e(V)`, and then calculate the p-value (see Step 4). In other situations, a retrievable statistic such as a log-likelihood ratio is assumed to have a chi-squared distribution under H_0.

4. Calculate the test statistic and p-value, if necessary. For example, if it is a z test, divide the parameter estimate of interest by its standard error and use the `normprob` function to get the p-value. Specifically, suppose `b` is the coefficient estimate with standard error `sb`. The two-sided p-value to test the hypothesis of a zero coefficient is then equal to $2[1-\Phi(Z)]$, where $Z = \texttt{b}/\texttt{sb}$ and Φ is the standard normal cumulative distribution function, returned by `normprob`.

5. Do Steps 1–4 a large number of times, say, NIT, and save the p-values in a file with NIT observations.

6. The estimated power for a level-α test is simply the proportion of observations (out of NIT) for which the p-value is less than α.

It is important that design-related values of X such as group identity, drug doses, temperature settings, etc. remain fixed throughout all NIT iterations of the simulation. This is because power may well be a function of X. For random covariates such as ages of subjects, whether or not to regenerate them for each iteration depends on whether

a) the user wishes to calculate power conditionally on fixed values of these covariates (such as those observed in the actual data), or b) one wishes the random event of rejection to reflect the joint distribution of Y and X over hypothetical reenactments of the experiment. In (a), values of X may be randomly generated for the first simulation, but must then remain the same for all subsequent simulations. In (b), one must assume some model for the joint distribution of Y and X and use it to generate both X and Y for each iteration.

As a check, always run the simulation for the null case to make sure the estimated power comes out close to α (to within the sampling error of the number of iterations). More generally, in the null case, the distribution of the p-values should be uniform. This may be easily tested in Stata using the `ksmirnov` command. If this check fails, then the test could be biased and/or the simulation could be programmed incorrectly.

Much of the bookkeeping involved in saving the simulated p-values (and any other desired quantities) can be facilitated with the help of `postfile` commands. This will be illustrated in the following examples.

3 Examples

3.1 Example 1: [two-sample t test]

The first example we give is one for which theoretical results abound in the literature: the two-sample t test. In fact, Stata has the built-in command `sampsi` to estimate power for this analysis. Let's see how good `sampsi` really is for estimating the power of the test of equal means when they differ by an amount Δ. The model for the two-sample t test may be written as

$$y_{ij} = \mu_i + e_{ij} \tag{1}$$

where the μ_i are population means, the e_{ij}'s are iid $\mathrm{N}(0, \sigma^2)$, and y_{ij} is the jth observation in the ith group, where $i = 1, 2$, $j = 1, \ldots, N_i$.

Although not explicitly appearing in the above formulation of the model, here X consists of dummy variables X_1 and X_2, indicating membership in groups 1 and 2, respectively. Thus, we could have expressed the first part of the model as $y_{ij} = \mu_1 X_1 + \mu_2 X_2 + e_{ij}$.

The experiment design is defined by the X_i and the sample sizes N_i, while the distribution of $Y = \{y_{ij}\}$ is specified through the normal assumption and given values of μ_1, μ_2 and σ^2. Since H_0 is defined by $\mu_1 = \mu_2$, any departure from H_0 may be expressed in terms of $\Delta = \mu_1 - \mu_2$. Here, σ^2 is a nuisance parameter because the power of the test of $\mathrm{H}_0 : \mu_1 = \mu_2$ depends on σ^2 through the quantity Δ/σ. As a result, the value of σ must also be specified. Suppose we know from previous studies that $\sigma^2 = 1.0$, and we wish to estimate the power when $\alpha = 0.05$, $N_1 = 8$, $N_2 = 12$, and $\Delta = 1.2$ using 1,000 simulated experiments. Following is a program to do so:

```
/* t2sim.do: calculates power of two-sample t test by simulation
             arguments are: n1 = sample size of 1st group
                            n2 = sample size of 2nd group
                            del = actual difference between the two means
                            sig = common standard deviation of error term
                            al = test level, "alpha"
                            NIT = no. of simulations */
version 7.0
args n1 n2 del sig al NIT
local df=‘n1’+‘n2’-2 /* degrees of freedom */
drop _all
set obs ‘NIT’
gen s2=‘sig’*‘sig’*invchi2(‘df’,uniform() )/‘df’
gen delm=‘del’ + ‘sig’*sqrt((1/‘n1’+1/‘n2’) )*invnorm(uniform())
gen sed = sqrt(s2*(1/‘n1’+1/‘n2’) )
gen tobs=delm/sed
gen pv=tprob(‘df’,tobs)
count if pv < ‘al’
scalar pow=r(N)/‘NIT’
noi di "est power = ",scalar(pow)," for al = ‘al’ "
exit
```

This program is efficient because it avoids having to simulate each observation for each replicated experiment. Instead it uses the fact that the sample statistics needed to perform the t test: (1) D, the difference between the sample means, and (2) S^2, the pooled variance estimate, have known distributions and are statistically independent. Specifically, $D \sim N\{\Delta, \sigma^2(1/N_1 + 1/N_2)\}$ and $S^2 \sim \sigma^2\chi^2(\nu)/\nu$, where ν is the degrees of freedom $\nu = N_1 + N_2 - 2$. In the program, d and S^2 are generated directly, being named `delm` and `s2`. The t-value for the test $\Delta = 0$ is the ratio $t_{\text{obs}} = D/\text{s.e.}(D)$, where $\text{s.e.}(D) = S\sqrt{1/N_1 + 1/N_2}$. The corresponding two-sided p-value is the area under the t-distribution density for $|t| > t_{\text{obs}}$, which can be evaluated using Stata's `tprob` function with ν degrees of freedom. We now run our program:

```
. run t2sim 8 12 1.20 1.0 .05 1000
est power =  .702  for al = .05
```

There were 702 rejections in 1,000 simulations, giving an estimated power of 0.702. A confidence interval for the power may be obtained by running `cii`. For example,

```
. cii 1000 702
                                                      -- Binomial Exact --
    Variable |        Obs        Mean    Std. Err.       [95% Conf. Interval]
-------------+---------------------------------------------------------------
             |       1000        .702    .0144636        .6725858    .7302194
```

gives a 95% confidence interval of (0.673, 0.730) for the power. (Of course, the width of the interval can be made smaller by increasing NIT.) Running `sampsi` gives

```
. sampsi 0.0 1.2, n1(8) n2(12) a(.05) sd(1)
Estimated power for two-sample comparison of means
Test Ho: m1 = m2, where m1 is the mean in population 1
                    and m2 is the mean in population 2
Assumptions:
         alpha =   0.0500  (two-sided)
            m1 =        0
            m2 =      1.2
           sd1 =        1
           sd2 =        1
sample size n1 =        8
            n2 =       12
         n2/n1 =     1.50
Estimated power:
         power =   0.7483
```

This value is considerably beyond the upper 95% confidence limit for power based on our simulation. These results suggest that at least for this case, `sampsi` is overestimating the true power.

3.2 Example 2: [Poisson regression]

We wish to test the effect of a drug dose on a Poisson distributed response y in rats. For a sample of n rats, we have the Poisson regression model $y \sim \text{Poisson}(\mu)$, where

$$\log \mu_i = \beta_0 + \beta_i x_i$$

for $i = 1, \ldots, n$, where x_i is the dose in mg given to the ith rat. Suppose we are trying to decide what sample size would give reasonable power for the test of the null hypothesis $\beta_1 = 0$ if the true value of β_1 were 0.65 (the alternative hypothesis), under an experimental design with three levels of dosage (0.2, 0.5, and 1.0 mg) repeated r times on $n = 3r$ different rats. The following program estimates the power for a fixed value of r. One can then run it for several values to arrive at an r of choice.

```
/* poispow.do - does power estimation for poisson regression model  */
version 7.0
args NIT rlow rhigh rstep d1 d2 d3 be1 alpha
/* model: log(mu) = be0 + be1*x
       y ~ Poisson(mu)
Specifically, power is estimated for testing the hypothesis be1 = 0, against
a user-specified alternative for a sample size r = no. rats per dose level,
ranging from rlow to rhigh in steps of rstep. Without loss of generality, we
can assume the true value of be0 is 0.
Note: This simulation uses Joe Hilbe's ‘rnd’ random number generator
      obtained from STB-41.
```

```
In the ‘args’ statement:
        NIT is number of iterations in the simulation
        rlow is the smallest value of r to be considered
        rhigh is the largest value of r to be considered
        rstep is the step size for incrementing r
        d1, d2 and d3 are the fixed dosages
        be1 is the "true" value of beta1 (the alternative hypothesis).
        alpha is the level of the test (e.g. alpha = .05)
*/
forv r = ‘rlow’(‘rstep’)‘rhigh’ {
        drop _all
        set obs 3
        gen x=‘d1’ in 1
        replace x=‘d2’ in 2
        replace x=‘d3’ in 3
        expand ‘r’
        sort x
        gen mu=exp(0 + ‘be1’*x)   /* (the "zero" is to show be0 = 0) */
/* Note: Here, I generated my "x" and mu-values and now store them in a
         dataset -tempx- so that the same values could be used throughout
         the simulation  */
        save tempx,replace
/* Initialize the posting process */
/* The simulated p-values will be save in temppow.dta */
        tempname buff
        postfile ‘buff’ pval using temppow,replace
/* run NIT simulated Poisson regressions for a fixed value of r */
        forv it = 1/‘NIT’ {
                qui use tempx,clear
                /*generates Poisson(mu) random variable in variable -xp- */
                qui rndpoix mu
                qui poisson xp x  /*performs the Poisson regression */
                matrix V=e(V)
                matrix b=e(b)
                scalar zrat=b[1,1]/sqrt(V[1,1])  /* the "z"-ratio */
                scalar pval = 2*(1-normprob(abs(zrat)))  /* the "p"-value  */
                post ‘buff’ (pval)   /*post the result to temppow.dta */
        }
        postclose ‘buff’   /* close the buffer */
        use temppow,clear
/*The dataset in memory now contains NIT simulated p-values. To get an
estimate of the power, just count the proportion of pval’s that are less
than alpha:  */
        count if pval<‘alpha’
        scalar power=r(N)/‘NIT’
        noi di "Power is ",%8.4f scalar(power),  /*
        */ " for ‘r’ rats per dose and alpha = ‘alpha’"
}
```

Note the use of the postfile commands to update the relevant simulation results to a new file without worrying about destroying the current dataset. In this case, the file temppow.dta is overwritten for each value of r, but we just as easily could have saved it under a different name each time through the loop over values of r. We now run this program for various values of r (10, 20, 30, 40, 50, 60, and 70), with 100 simulations per run, and with $\alpha = 0.05$.

```
. run poispow 100 10 70 10 .2 .5 1.0 .64 .05
Power is    0.3000  for 10 rats per dose and alpha = .05
Power is    0.4400  for 20 rats per dose and alpha = .05
Power is    0.7000  for 30 rats per dose and alpha = .05
Power is    0.8100  for 40 rats per dose and alpha = .05
Power is    0.9000  for 50 rats per dose and alpha = .05
Power is    0.9100  for 60 rats per dose and alpha = .05
Power is    0.9700  for 70 rats per dose and alpha = .05
```

Although each individual estimate is not that accurate, being based on only 100 trials, one could fit a logistic or other regression curve through this data to get a good estimate of which value of r would give a prescribed power, say, of 0.80. Of course, one could get better accuracy by increasing NIT at the expense of more computer time.

Finally, (perhaps this should have been done first), we run the program with $\beta_1 = 0$ and a large value of NIT (say, 1,000) to see if the level of the test is "close" to 0.05. If not, the normal approximation is not adequate, or we may have programmed the procedure incorrectly. To get the most provocative case (that is, the smallest sample size considered), we run this for $r = 10$.

```
. run poispow 1000 10 10 10 .2 .5 1.0 0.0 .05
Power is    0.0450  for 10 rats per dose level and alpha = .05
```

There were 45 simulated p-values less than 0.05, which for 1,000 binomial trials gives a 95% confidence interval of about (0.033, 0.060) for the level of the test (see `cii` command results below). With the value 0.05 being in this confidence interval, one might conclude that the normal approximation z test is adequate, even for 30 observations ($r = 10$).

```
. cii 1000 45
                                                         ── Binomial Exact ──
    Variable |        Obs        Mean    Std. Err.       [95% Conf. Interval]
-------------+---------------------------------------------------------------
             |       1000        .045    .0065555        .0330098    .0597525
```

Finally, we check the entire distribution of simulated p-values for a uniform distribution:

```
. ksmirnov pval=pval
One-sample Kolmogorov-Smirnov test against theoretical distribution
           pval
 Smaller group       D       P-value  Corrected
 ----------------------------------------------
 pval:              0.0135    0.696
 Cumulative:       -0.0237    0.324
 Combined K-S:      0.0237    0.627      0.611
```

It passes with flying colors. Figure 1 gives the histogram of `pval` with 32 bins.

```
. graph pv, xlab ylab bin(32)
```

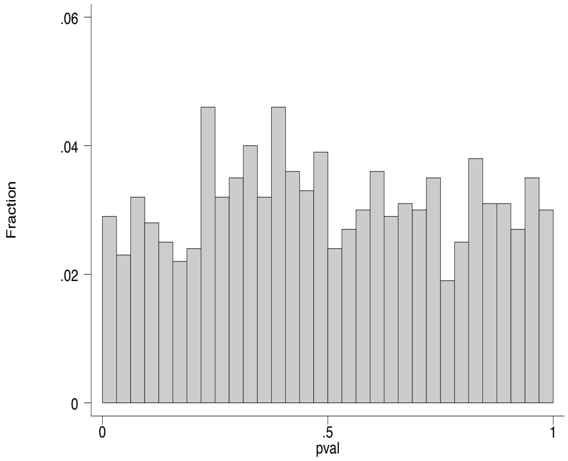

Figure 1: Empirical distribution of p-values when H_0 is true: Poisson regression example.

3.3 Example 3: [stcox with clustering]

Suppose you plan to test the effect of a new manufacturing process on the strength of fiber-wrapped pressure vessels. You plan to compare a number of vessels made with the standard and new process using survival analysis of failure or censoring times, measured in months. Let X be a binary indicator of the new process and let $h_0(t)$ and $h(t)$ be the hazard functions associated with the failure-time distribution under the standard process and the new process, respectively. You would like to apply the proportional hazards model

$$h(t) = h_0(t)\exp(\beta X)$$

to the results and the test the hypothesis H_0 that the new process has no effect on reliability; that is, that $\beta = 0$.

You also know that vessels are manufactured in batches, and that all vessels from a given batch must be manufactured by the same procedure. Furthermore, there is an overhead cost of procuring vessels from each batch, as well as an experimental cost per vessel of actually running the experiment. In particular, for an experiment with B batches and n vessels per batch, the cost is approximately proportional to $2B + nB$. Based on this cost function, you would like to choose one of these three designs, which have about the same cost to run:

(*Continued on next page*)

Table 1: Three possible experimental designs

Design	B	n	nB	2B	Total Cost Index
1	12	6	72	24	96
2	16	4	64	32	96
3	24	2	48	48	96

Clearly, if there is no intra-batch correlation, you should choose the design that tests the most vessels; namely, the first design. On the other hand, if the failure times of vessels from the same batch are highly correlated, you should choose the design with the most batches; that is, the third design. For intermediate situations, one must have some idea of the correlation to identify the design with the best power for testing H_0. Suppose you know from a previous study of standard vessels that the intra-batch correlation of log failure times ranges from about 0.40 to 0.60. How can you use this information in a simulation to find the design with the best power for testing $\beta = 0$, assuming that $\beta = -1.25$?

Although `stcox` assumes a proportional hazards model, it does not specify any particular baseline hazard function. Nevertheless, we must assume some specific failure-time distribution to simulate experimental data. In this example, we use a Weibull model because it satisfies the proportional hazards assumption and is easy to implement. To some extent, the power of the test $\beta = 0$ will depend on which distribution we use, but we will assume that the answer to the question of which design is superior will be the same over any reasonable set of distributions we might choose. In addition to the question of what marginal distribution to use for failure times, we must also consider how the within-batch dependence should be modeled and how to incorporate censoring in our simulated data.

Simulation of data satisfying the proportional hazards assumption

Given a random uniform number u, one may generate a new random variable with distribution function F by the simple transformation $y = F^{-1}(u)$. In our case, F is the Weibull CDF, parameterized here as $F(y) = 1 - \exp(-cy^k)$. Therefore, given $U \sim \text{uniform}[0,1]$,

$$Y = \left(\frac{-\log U}{c}\right)^{1/k} \tag{2}$$

has the aforementioned Weibull distribution. (Note that we may use U here instead of $1-U$, since both have the same uniform distribution.) The hazard function corresponding to $F(y)$ is given by $h(y) = cky^{k-1}$. Under the Cox model, the parameter c would be equal to some base value c_0 when $X = 0$ and would become equal to $c_1 = c_0 \exp(\beta)$ when $X = 1$.

Simulation of intra-batch dependence

Suppose the correlation between log failure times of any two vessels manufactured from the same batch is $\rho > 0$. How can we incorporate this dependence into our simulated data while still maintaining the marginal Weibull distribution for each observation? One simple method (which we use here) is to make use of a *copula* constructed from a particular multivariate normal distribution. A copula is usually defined as a bivariate CDF whose marginals are uniform; see, for example, Genest and MacKay (1986). However, the concept can easily be extended to a multivariate distribution. The method follows three steps. First, for $\rho_0 > 0$, we use the random effects model $Z_{ij} = z_i + e_{ij}$ for $i = 1, \ldots, B$ and $j = 1, \ldots, n$, where $z_i \sim N(0, \rho_0)$, $e_{ij} \sim N(0, 1-\rho_0)$, and the z_i and e_{ij} are mutually independent. The Z_{ij} all have the same $N(0,1)$ marginal distribution, are independent between batches (different values of i) and have correlation ρ_0 within a batch (different values of j for the same i). (Note: The value of ρ_0 is chosen such that the correlation between the simulated log failure times within a batch is approximately ρ. How this is done will be explained in the next section.) Next, we transform the Z_{ij} into uniform random variables with a similar correlation structure (although the intra-batch correlation will no longer be equal to ρ_0). This is accomplished by the transformation $U_{ij} = \Phi(Z_{ij})$, where Φ is the standard normal CDF (`normprob` in Stata). Finally, we transform the U_{ij} into dependent Weibulls using Equation 2. Although not used explicitly here, the multivariate CDF of the U_{ij} is a copula. Oakes (1989) provides additional examples of simulated correlated survival data using copulas.

Obtaining the value of ρ_0

In terms of the original normally distributed Z_{ij}, the simulated log failure times may be expressed as $g_{ij} = G(Z_{ij})$, where

$$G(z) = \log\left[\left\{\frac{-\log\Phi(z)}{c}\right\}^{1/k}\right] \tag{3}$$

Using second-order expansions of g_{ij} and $g_{ij'}$ about $z = 0$, it can be shown (approximately) that

$$\mathrm{Cov}(g_{ij}, g_{ij'}) = \{G'(0)\}^2\rho_0 + \frac{\{G''(0)\}^2}{2}\rho_0^2 \tag{4}$$

and that

$$\mathrm{Var}(g_{ij}) = \mathrm{Var}(g_{ij'})\{G'(0)\}^2 + \frac{\{G''(0)\}^2}{2} \tag{5}$$

Combining (4) and (5), we find that $\mathrm{Corr}(g_{ij}, g_{ij'})$ is a weighted average of ρ_0 and ρ_0^2; namely,

$$\mathrm{Corr}(g_{ij}, g_{ij'}) = w\rho_0 + (1-w)\rho_0^2 \tag{6}$$

where

$$w = \frac{\{G'(0)\}^2}{\{G'(0)\}^2 + \{G''(0)\}^2/2} \tag{7}$$

It can be shown that w does not depend on the parameters k or c; specifically,

$$w = \frac{\pi(\log 2)^2}{\pi(\log 2)^2 + (1-\log 2)^2} \doteq 0.94128 \tag{8}$$

Thus, we can solve (6) for the value of ρ_0 such that $\mathrm{Corr}(g_{ij}, g_{ij'}) = \rho$. The solution is

$$\rho_0 = \frac{-w + \sqrt{w^2 + 4\rho(1-w)}}{2(1-w)} \tag{9}$$

Simulation program

Here is a program to estimate the power as a function of B, n, ρ, β, α, and the number of iterations (NIT). While we could assume that the user is to input specific values of the Weibull parameters k and c_0, we instead assume that the user provides the median, $y_{.50}$, and the 10% point, $y_{.10}$, of the failure time distribution for the standard vessels. The program calculates k and c_0 so that $y_{.50}$ and $y_{.10}$ match the user input values. This program also allows the user to specify a censoring time y_0. Simulated failure times exceeding y_0 are set equal to y_0, with an indicator of censoring accordingly set for use with `stcox`. When `stcox` is run, the cluster option with robust standard errors is used to account for the batch effect.

```
/* coxsim.do*/
args B n rho be pct50 pct10 y0 alpha NIT
/*
   B is the number of batches (blocks)
   n is the number of vessels per batch
   rho is the desired intra-batch correlation
   be is value of "beta" under the alternative hypothesis
   pct50 is desired median of failure time distribution, when x = 0
   pct10 is desired 10% point of failure time distribution, when x = 0
   y0 is censoring time
   alpha is level of test
   NIT is number of simulated experiments
*/
scalar lg2=log(2)
scalar w = _pi*lg2*lg2/(_pi*lg2*lg2+(1-lg2)^2)
/* w is weight that describes correlation of log failure times
   in terms of correlation between N(0,1) variates
   r0 is rho-zero - the original Normal correlation needed to make the final
   Weibull correlation equal to rho
*/
local disc=w*w+4*‘rho’*(1-scalar(w))
scalar r0 = 0.5*(-w+sqrt(‘disc’))/(1-scalar(w))
/* Solve for c0 and k such that F0(‘pct50’) = 0.5 and F0(‘pct10’)=0.10
   where F0(y) = 1 - exp(-c0*y^k). See test50 and test10 below for checkout.
*/
local k = log( log(.5)/log(.9) )/log(‘pct50’/‘pct10’)
local c0 = -log(.5)/(‘pct50’^‘k’)
scalar test50=exp(-(‘c0’*‘pct50’^‘k’) )   /* should be .50  */
scalar test10=1-exp(-(‘c0’*‘pct10’^‘k’) ) /* should be .10  */
```

```
/* y0 is the desired censoring time
   pcen = P(censoring) when x=0  =exp(-c0y0^k)
   k is shape parameter
   k < 1 => h(y) -> 0
   k > 1 => h(y) increasing to oo
*/
scalar pcen = exp(-(‘c0’)*‘y0’^‘k’)
local sige=sqrt(1-scalar(r0))
local sigu=sqrt(scalar(r0))
tempname buff
postfile ‘buff’ pval b1 v11 zrat using temppow,replace
forv it = 1/‘NIT’ {
        drop _all
        range block 1 ‘B’ ‘B’
        gen x=1-mod(_n,2)
        gen zu=‘sigu’*invnorm(uniform())
        expand ‘n’
        gen ze=‘sige’*invnorm(uniform())
        gen z=zu+ze         /* equi-correlated N(0,1)’s with correlation rho */
        gen u=normprob(z)   /* equi-correlated uniforms */
        sort block
        gen xbe=x*‘be’
        gen c = ‘c0’*exp(xbe)

        /* marginal distribution of y is Weibull
        F(y) = 1 - exp(-cy^k)
        f(y)=cky^(k-1)*exp-cy^k
        h(y)=cky^(k-1)
        prop hazards model: c = c0*exp(xbe)
        */

        cap gen y=.
        replace y=(-log(u)/c)^(1/‘k’)
        gen byte fail=0
        replace fail=1 if y <= ‘y0’
        replace y=‘y0’ if y > ‘y0’
        qui stset y,failure(fail)
        stcox x ,nohr nolog cluster(block) robust
        matrix b=e(b)
        matrix V=e(V)
        scalar zrat=b[1,1]/sqrt(V[1,1])  /* the "z"-ratio */
        scalar pval = 2*(1-normprob(abs(zrat)))  /* the "p"-value  */
        post ‘buff’ (pval) (b[1,1]) (V[1,1]) (zrat)
        if ‘it’==‘NIT’ {save temp,replace}
}
postclose ‘buff’
use temppow,clear
/*The dataset in memory now contains NIT simulated p-values. To get an
estimate of the power, say for alpha=.05, just count the proportion of pv’s
that are less than 0.05:  */
count if pval<‘alpha’
scalar power=r(N)/‘NIT’
noi di "Power is ",%8.4f scalar(power),/*
*/ " for ‘B’ blocks, ‘n’ reps per block, alpha = ‘alpha’"
exit
```

Below, we run this program assuming intra-batch correlations of 0.60, 0.40, 0.20, and 0 for each of the values of B and n in Table 1. We will estimate the power of the test

$\beta = 0$, when in fact, $\beta = -1.25$ for $\alpha = 0.05$. We ask the program to calculate values of the Weibull parameters such that the median failure time for standard vessels is `pct50 = 10` months, and where 10% of these vessels fail within `pct10 = 2.5` months. We also impose an experiment observation limit of $y_0 = 20$ months (the censoring time). This value of y_0 corresponds to about 17% censoring. One thousand simulated Cox regressions were run for each set of inputs.

```
. run coxsim 24 2 .60 -1.250 10 2.5 20 .05 1000
Power is    0.7620  for 24 blocks, 2 reps per block, alpha = .05
. run coxsim 16 4 .60 -1.250 10 2.5 20 .05 1000
Power is    0.6940  for 16 blocks, 4 reps per block, alpha = .05
. run coxsim 12 6 .60 -1.250 10 2.5 20 .05 1000
Power is    0.5740  for 12 blocks, 6 reps per block, alpha = .05
. run coxsim 24 2 .40 -1.250 10 2.5 20 .05 1000
Power is    0.8230  for 24 blocks, 2 reps per block, alpha = .05
. run coxsim 16 4 .40 -1.250 10 2.5 20 .05 1000
Power is    0.7740  for 16 blocks, 4 reps per block, alpha = .05
. run coxsim 12 6 .40 -1.250 10 2.5 20 .05 1000
Power is    0.7170  for 12 blocks, 6 reps per block, alpha = .05
. run coxsim 24 2 .20 -1.250 10 2.5 20 .05 1000
Power is    0.8490  for 24 blocks, 2 reps per block, alpha = .05
. run coxsim 16 4 .20 -1.250 10 2.5 20 .05 1000
Power is    0.8700  for 16 blocks, 4 reps per block, alpha = .05
. run coxsim 12 6 .20 -1.250 10 2.5 20 .05 1000
Power is    0.8600  for 12 blocks, 6 reps per block, alpha = .05
. run coxsim 24 2 .000 -1.250 10 2.5 20 .05 1000
Power is    0.9260  for 24 blocks, 2 reps per block, alpha = .05
. run coxsim 16 4 .000 -1.250 10 2.5 20 .05 1000
Power is    0.9680  for 16 blocks, 4 reps per block, alpha = .05
. run coxsim 12 6 .000 -1.250 10 2.5 20 .05 1000
Power is    0.9820  for 12 blocks, 6 reps per block, alpha = .05
```

A plot of estimated power versus ρ for these 12 runs of `coxsim` is shown in Figure 2. Note that Design 3 is clearly best if ρ is moderate or high and that Design 1 is best only for very low values of ρ. The conclusion to be made is that Design 3 is the way to go.

(*Continued on next page*)

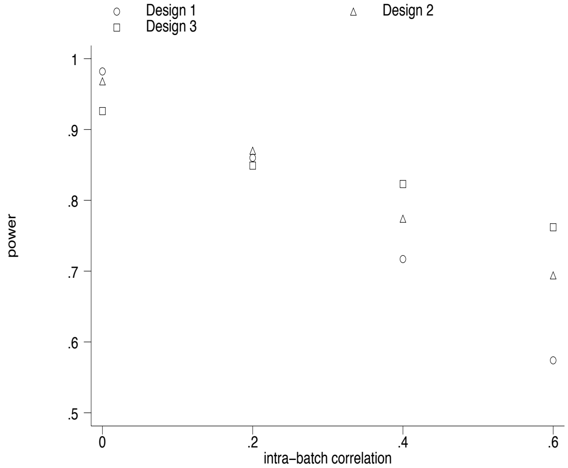

Figure 2: Power as a function of intra-batch correlation for three competing experimental designs.

We conclude this example with a case where H_0 is true and $\rho = 0.6$. The theoretical level of the test (0.05) is within the 95% confidence interval of (0.046, 0.077), formed from 60 rejections in 1,000 tries.

```
. run coxsim 24 2 .600 0.0 10 2.5 20 .05 1000
Power is    0.0600  for 24 blocks, 2 reps per block, alpha = .05
. cii 1000 60
                                                        -- Binomial Exact --
    Variable |        Obs        Mean    Std. Err.       [95% Conf. Interval]
-------------+---------------------------------------------------------------
             |       1000         .06      .00751         .046095    .0765593
```

3.4 Example 4: [rank sum test - one-sided]

You wish to design a study to compare men's and women's salaries for a given occupation class in some region. From past studies, you know that the logarithms of these salaries are approximately normally distributed with the same variance, but differing means. Being conservative, you do not want to make such distributional assumptions in your analysis, but instead wish to compare medians using the nonparametric `ranksum` command. However, you are willing to use the lognormal assumption to size your study. Suppose you want to sample men and women proportionately for your study and that approximately twice as many men as women in the region have the occupation of interest. Then you will be sampling n women and $2n$ men. What should n be to achieve a power of 0.80 for a one-sided test of equal medians against the alternative that the median salary of men exceeds that of women by a factor of 1.15? Assume the value 0.5 for the standard deviation of log salaries for each group, and that the level of the test is 0.05.

Simulation program

Below is a program to estimate the power. Note that the calculation of the *p*-value reflects the one-sided test. Under the alternative hypothesis, Group 2 (men) would have a higher median salary, hence in order for the null to be rejected, the standardized difference between the two groups in rank sums (`zrat` in the program) must be negative. Therefore, in calculating the *p*-value, we only consider the lower tail.

```
/* rksumsim.do
 Rank sum 2-sample rank test power simulation  */
version 7.0
args medrat sig n alpha NIT
/*
   medrat is the assumed ration of medians (men to women)
   sig is the standard deviation of log salaries
   n is sample size for women
   alpha is the level of the test
   NIT is number of simulations
*/
local dif=log(‘medrat’) /* difference in means of log salaries */
tempname buff
postfile ‘buff’ pv1 zrat using temppow,replace
forv it = 1/‘NIT’ {
        drop _all
        set obs ‘n’
        gen byte group=1
        local N=3*‘n’
        set obs ‘N’
        replace group=2 if group==.
/* without loss of generality - take mean log salary for women to be zero */
        gen y= 0 + ‘sig’*invnorm(uniform()) if group==1
/* mean log salary for men is ‘dif’ */
        replace y= ‘dif’+‘sig’*invnorm(uniform()) if group==2
        replace y=exp(y)
        ranksum y,by(group) /* test medians */
        local zrat=r(z)
        scalar pv1=normprob(‘zrat’)    /* 1-sided pvalue, H1: Group2 > Group1 */
        post ‘buff’ (pv1)  (zrat)
        if ‘it’==‘NIT’ {
                save temp,replace
        }
}
postclose ‘buff’
use temppow,clear
/*The dataset in memory now contains NIT simulated p-values. To get an
estimate of the power, say for alpha=.05, just count the proportion of pv’s
that are less than 0.05:  */
local n2 = 2*‘n’
count if pv1<‘alpha’
scalar power=r(N)/‘NIT’
noi di "Power is ",%8.4f scalar(power),/*
*/ " for ‘n’ women, ‘n2’ men, ratio = ‘medrat’, alpha = ‘alpha’"
```

Results

Now, we run the program for various values of n:

```
  . for n in num 50(10)160: run rksumsim 1.15 0.50 n .05 1000
->  run rksumsim 1.15 0.50 50 .05 1000
Power is    0.4820  for 50 women, 100 men, ratio = 1.15, alpha = .05
->  run rksumsim 1.15 0.50 60 .05 1000
Power is    0.5530  for 60 women, 120 men, ratio = 1.15, alpha = .05
->  run rksumsim 1.15 0.50 70 .05 1000
Power is    0.6490  for 70 women, 140 men, ratio = 1.15, alpha = .05
->  run rksumsim 1.15 0.50 80 .05 1000
Power is    0.6190  for 80 women, 160 men, ratio = 1.15, alpha = .05
->  run rksumsim 1.15 0.50 90 .05 1000
Power is    0.6840  for 90 women, 180 men, ratio = 1.15, alpha = .05
->  run rksumsim 1.15 0.50 100 .05 1000
Power is    0.6920  for 100 women, 200 men, ratio = 1.15, alpha = .05
->  run rksumsim 1.15 0.50 110 .05 1000
Power is    0.7340  for 110 women, 220 men, ratio = 1.15, alpha = .05
->  run rksumsim 1.15 0.50 120 .05 1000
Power is    0.7840  for 120 women, 240 men, ratio = 1.15, alpha = .05
->  run rksumsim 1.15 0.50 130 .05 1000
Power is    0.8130  for 130 women, 260 men, ratio = 1.15, alpha = .05
->  run rksumsim 1.15 0.50 140 .05 1000
Power is    0.8330  for 140 women, 280 men, ratio = 1.15, alpha = .05
->  run rksumsim 1.15 0.50 150 .05 1000
Power is    0.8560  for 150 women, 300 men, ratio = 1.15, alpha = .05
->  run rksumsim 1.15 0.50 160 .05 1000
Power is    0.8720  for 160 women, 320 men, ratio = 1.15, alpha = .05
```

From these results, it appears that a survey with about 125 women and 250 men should do the job. Let's check the true level of the test for a study of this scope:

```
  . run rksumsim 1.00 0.50 125 .05 1000
Power is    0.0460  for 125 women, 250 men, ratio = 1.00, alpha = .05
```

The observed proportion of rejections (46/1000) is well within the bounds of random variation for an unbiased test.

4 Concluding remarks

This paper has attempted to illustrate the potential of Stata for estimating the power for a variety of statistical tests based on classical inference methods. In the examples shown, heavy use is made of Stata's ability to organize data and to generate pseudorandom numbers. It is assumed throughout these examples that successive calls to `uniform()` actually produce a sequence of "independent" uniformly distributed numbers. It is important to realize that unless Stata's random number seed is set to the same number, the results of running any of the above simulation programs will not be perfectly repeatable. One may produce repeatable results by using Stata's `set seed` command. For example,

```
. set seed 23456789
. run t2sim 8 12 1.2   1 .05 1000
est power =  .684  for al = .05
. set seed 23456789
. run t2sim 8 12 1.2   1 .05 1000
est power =  .684  for al = .05
```

In our example programs, we made use of the `postfile` commands to conveniently compile the simulation results in a special file. Alternatively, this could be done with Stata's `simul` command. It should be remembered that in any calculation of power, whether by mathematical derivation or by simulation, the distribution of the data under the alternative hypothesis must be completely specified even for "distribution-free" test procedures. At first glance, this requirement appears formidable, but it should be remembered that in virtually all applications, power calculations, although somewhat dependent on the choice of distributional assumptions, are meant only to serve as a guide for experiment planning. Hopefully the decision of which design to use should be relatively robust to differences between reasonable distributional models. This robustness can, of course, be checked by further simulation with a variety of models.

5 References

Berger, J. and T. Sellke. 1987. The irreconcilability of P values and evidence (with discussion). *Journal of the American Statistical Association* 82: 112–122.

Genest, C. and J. MacKay. 1986. The joy of copulas: bivariate distributions with uniform marginals. *The American Statistician* 40: 280–283.

Loftus, G. 1993. A picture is worth a thousand p values: On the irrelevance of hypothesis testing in the microcomputer age. *Behavior Research Methods, Instruments, and Computers* 25: 250–256.

Oakes, D. 1989. Bivariate survival models induced by frailties. *Journal of the American Statistical Association* 84: 487–493.

Schervish, M. 1996. P values: What they are and what they are not. *The American Statistician* 50: 203–206.