ASK THE EXPERT

10 Tips for Simulating Data With SAS®

Rick Wicklin
Distinguished Researcher in Computational Statistics

Rick Wicklin

Distinguished Researcher in Computational Statistics

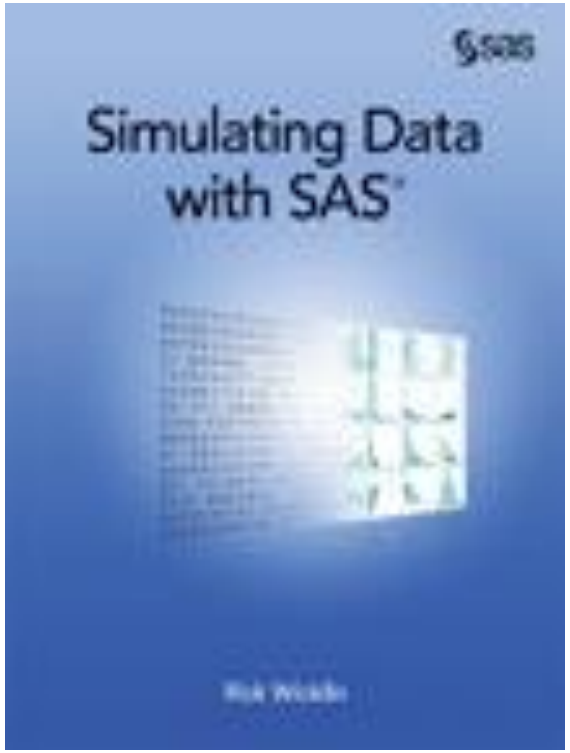
Rick is the principal developer of the SAS/IML® matrix language. His areas of expertise include simulation, computational statistics and statistical graphics. Rick is the author of two books, including Simulating Data With SAS. He writes a popular blog about statistical programming in SAS. Rick frequently presents talks, tutorials and workshops at conferences.

10 TIPS FOR SIMULATING DATA WITH SAS®

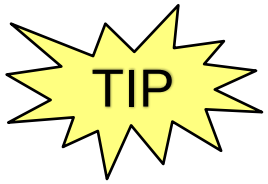
RICK WICKLIN, SAS
@RICKWICKLIN
blogs.sas.com/content/iml

OVERVIEW

- SAS software provides many tools for simulating data that have a variety of statistical properties
- The tools are spread throughout SAS software
  - The DATA step
  - SAS/STAT® and SAS/ETS® procedures
  - SAS/IML® functions
- This talk includes 10 tips for simulating data efficiently and effectively
- Tips are from *Simulating Data with SAS* (Wicklin 2013)
- To download this SAS Global Forum presentation, search the internet for
  Wicklin 2015 “ten tips”



TIP 1: USE THE RAND FUNCTION FOR CONTINUOUS DISTRIBUTIONS



- Use the RAND function instead of the older random number functions (RANUNI, RANNOR, RANBIN,…)
  - RAND uses the Mersenne-Twister algorithm, which has a long period and excellent statistical properties
  - Easy to specify seed values
  - Superior handling of certain regions in parameter space
- The older random number functions (RANUNI, RANNOR, RANBIN,…) use a linear congruential generator, which was popular in the 1970s
- The RANDGEN subroutine in SAS/IML software is a vectorized version of RAND

COMMON CONTINUOUS DISTRIBUTIONS

```
/* Four letters suffice */
x = rand('Beta',        a, b);
x = rand('ChiSquare', d);
x = rand('Exponential');
x = rand('Gamma', a);
x = rand('Lognormal');
x = rand('Normal', mu, sigma);    /*opt args */
x = rand('T', d);
x = rand('Uniform');
x = rand('Weibull', a, b);
```

Name of distribution

parameters

RAND FUNCTION EXAMPLE

DATA step:

- The **STREAMINT** call sets the random seed for a random number stream
- The **RAND** function generates a random value

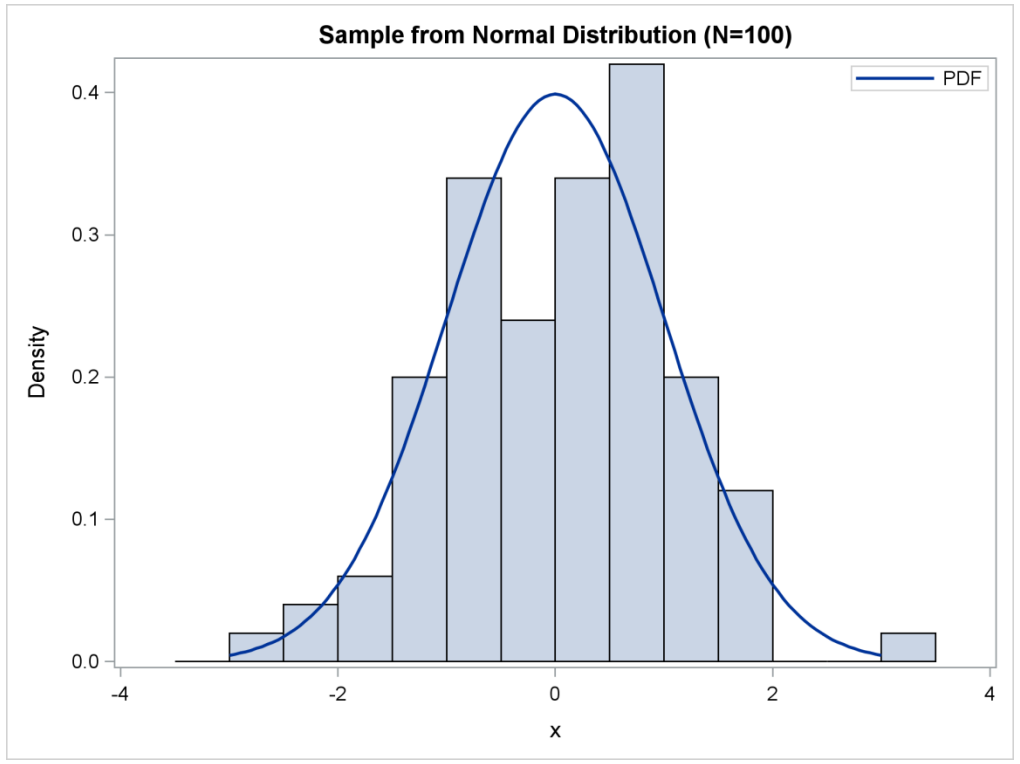
```
%let N = 100;
data Sample;
call streaminit(1234);          /* set seed for stream */
do i = 1 to &N;
   x = rand("Normal");          /* X~N(0,1) or N(μ,σ) */
   u = rand("Uniform");         /* U~U(0,1) */
   output;
end;
run;
```

RAND FUNCTION OUTPUT

For the simulated normal data

- Histogram shows distribution of simulated data
- Expected density shown by curve (PDF)

Obs	x
1	1.24067
2	-0.53532
3	-1.01394
4	0.68965
5	-0.32458
6	-1.71788



RANDGEN SUBROUTINE EXAMPLE IN SAS/IML

In SAS/IML, use the **RANDGEN** routine or the **RANDFUN** function

```
proc iml;
call randseed(1234);              /* set seed for stream */
y = j(100, 20);                   /* allocate 100 x 20 matrix */
call randgen(y, "Gamma", 4);      /* each col y[i] ~ Gamma(4)  */
u = randfun(100, "Uniform");      /* 100 x 1 vector, u ~ U(0,1) */
```

TIP 2: USE THE "TABLE" DISTRIBUTION FOR CATEGORIES

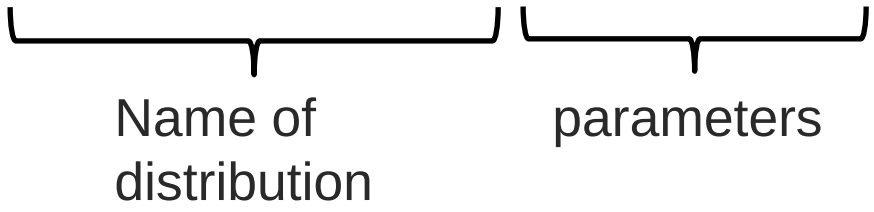
TIP

```
data Categories(keep=x);
call streaminit(4321);
array p[3] (0.5 0.3 0.2);          /* probabilities */
do i = 1 to 100;
   Type = rand("Table", of p[*]); /* use OF keyword operator */
   output;
end;
run;

/* -or- */

proc iml;
call randseed(4321);
p = {0.5 0.3 0.2};
T = j(100, 1);                     /* allocate vector */
call randgen(T, "Table", p);       /* fill with 1,2,3 */
```

Type	Frequency	Percent
1	48	48.00
2	31	31.00
3	21	21.00

OTHER BUILT-IN DISCRETE DISTRIBUTIONS

```
/* Four letters suffice */
x = rand('Bernoulli', p);
x = rand('Binomial',  p, n);
x = rand('Geometric', p);
x = rand('Hypergeom', N, R, n);
x = rand('NegBinom',  p, k);
x = rand('Poisson',   lambda);
x = rand('Table',     of p[*]); /* p is array */
```

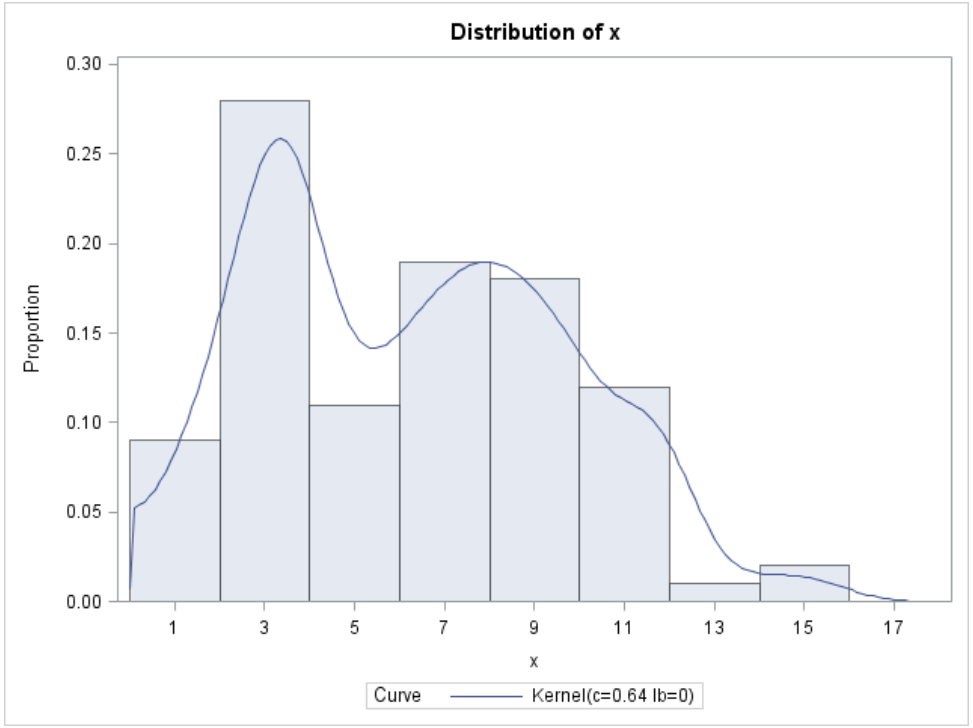


TIP 3. SIMULATE FROM FINITE MIXTURE DISTRIBUTIONS

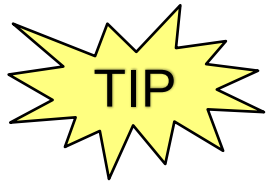
TIP

Use the “Table” distribution to simulate mixture distribution, outliers, or clusters

```
data Calls(keep=type x);
call streaminit(12345);
array prob [3] (0.5 0.3 0.2);
do i = 1 to &N;
   Type = rand("Table", of prob[*]);
   if      Type=1 then x = rand("Normal", 3, 1);
   else if Type=2 then x = rand("Normal", 8, 2);
   else                x = rand("Normal", 10, 3);
   output;
end;
run;

proc univariate data=Calls;
   ods select Histogram;
   histogram x / vscale=proportion
         kernel(lower=0 c=SJPI);
run;
```

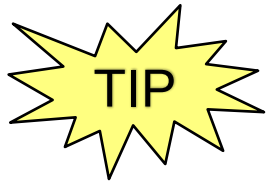


TIP 4: CONSTRUCT COMPLEX DISTRIBUTIONS FROM SIMPLE DISTRIBUTIONS



- Transform simple distributions to simulate data from distributions that SAS (seemingly) does not support
  - **Gumbel**: if $E \sim \mathrm{Exp}(1)$, then $\mu - \sigma \log E$ is Gumbel
  - **Pareto:** if $U \sim U(0,1)$, then $kU^{-1/a}$ is Pareto
  - **Power function**: if $E \sim \mathrm{Exp}(1)$, then $\left(1 - \exp(-\mathrm{E})\right)^{1/\mathrm{a}}$ follows the power function distribution
- Acceptance-rejection techniques for truncated distributions
- Inverse CDF transformations: If F is continuous CDF and U ~ U(0,1), then $X = F^{-1}(U)$ is distributed according to *F*
- Translate and scale: If X is random variable, then $Y = \theta + \sigma X$ is random variable that has new location and scale parameter.

TIP 5: USE SAS/IML SOFTWARE FOR MULTIVARIATE SIMULATION



- The SAS/IML language supports many multivariate distributions
  - Multinomial (discrete)
  - Multivariate normal
  - Multivariate *t*, Wishart, Dirichlet, and other distributions
  - Time series ARMA and VARMA models
- You can use matrix computations to sample from other models
  - Multivariate correlated binary
  - Multivariate correlated ordinal
  - Copulas

SIMULATE MULTINOMIAL COUNTS

- Just as a binomial distribution is N Bernoulli draws, a multinomial sample contains N "Table" draws
- Let p_i be probability of drawing the ith item, $i = 1, \ldots, k$
- Draw N items with replacement. The counts $(N_1, N_2, \ldots, N_k)$ follow the multinomial distribution where $N = \Sigma_i N_i$

```
%let NumSamples = 1000; /* = number of rows*/
proc iml;
prob = {0.5 0.2 0.3};
X = RandMultinomial(&NumSamples, 100, prob); /* N=100 */
c = {"N1", "N2", "N3"};
print (X[1:3,])[colname=c];
```

N1	N2	N3
54	17	29
57	15	28
46	23	31

SIMULATE MULTIVARIATE NORMAL DATA

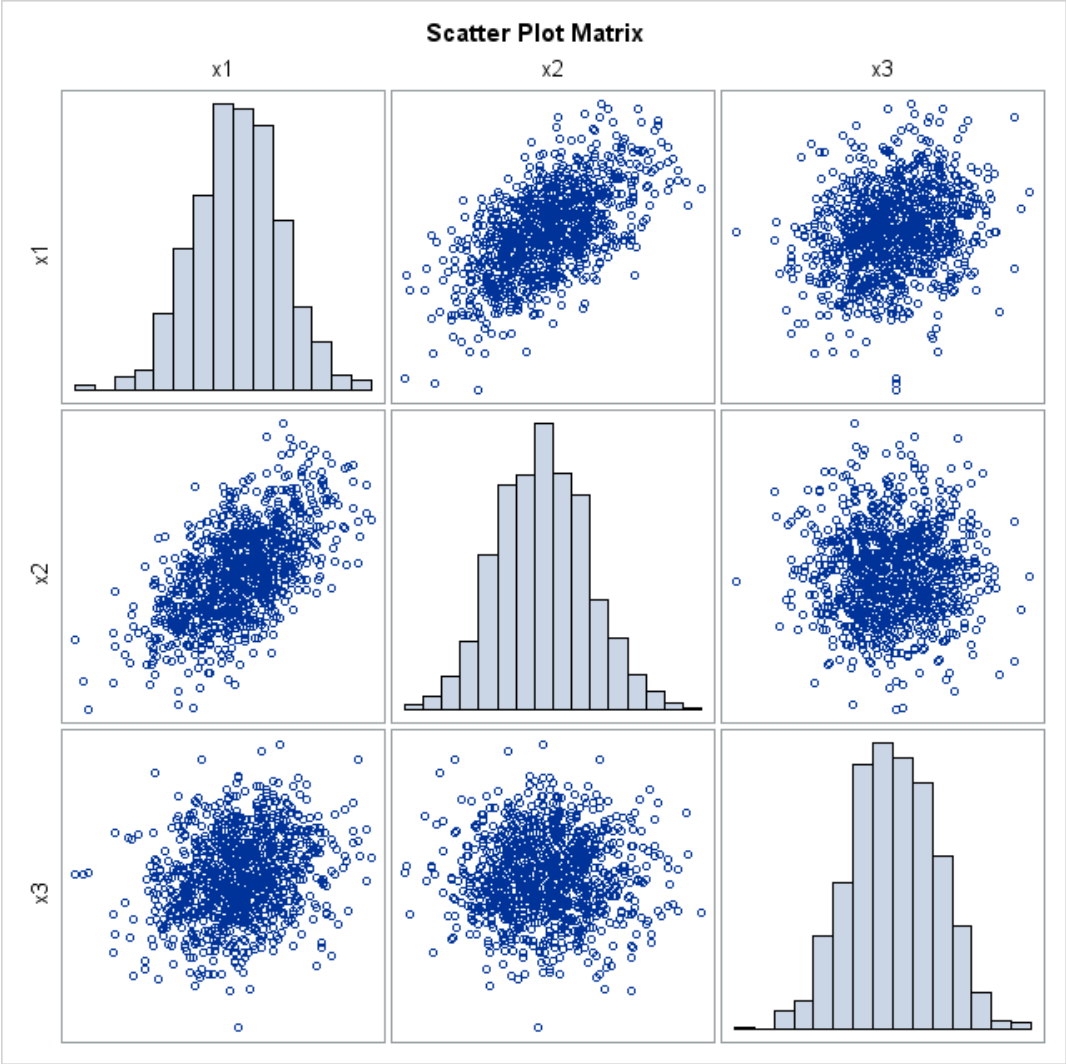
- The RANDNORMAL function in SAS/IML software
- PROC SIMNORMAL in SAS/STAT software
- Specify the mean vector and the covariance matrix

```
proc iml;
/* specify the mean and covariance of the population */
Mean = {1, 2, 3};
Cov = {3 2 1,
       2 4 0,
       1 0 5};
N = 1000;
call randseed(4321);              /* seed for RandNormal */
X = RandNormal(N, Mean, Cov);      /* 1000 x 3 matrix */
```

EVALUATE MULTIVARIATE NORMAL SAMPLES

Variable	N	Mean	Std Dev
x1	1000	0.98208	1.75963
x2	1000	1.96218	1.96464
x3	1000	3.03620	2.18670

Covariance Matrix, DF = 999			
	x1	x2	x3
x1	3.096309864	2.076631494	0.969236830
x2	2.076631494	3.859822736	0.024983749
x3	0.969236830	0.024983749	4.781649321

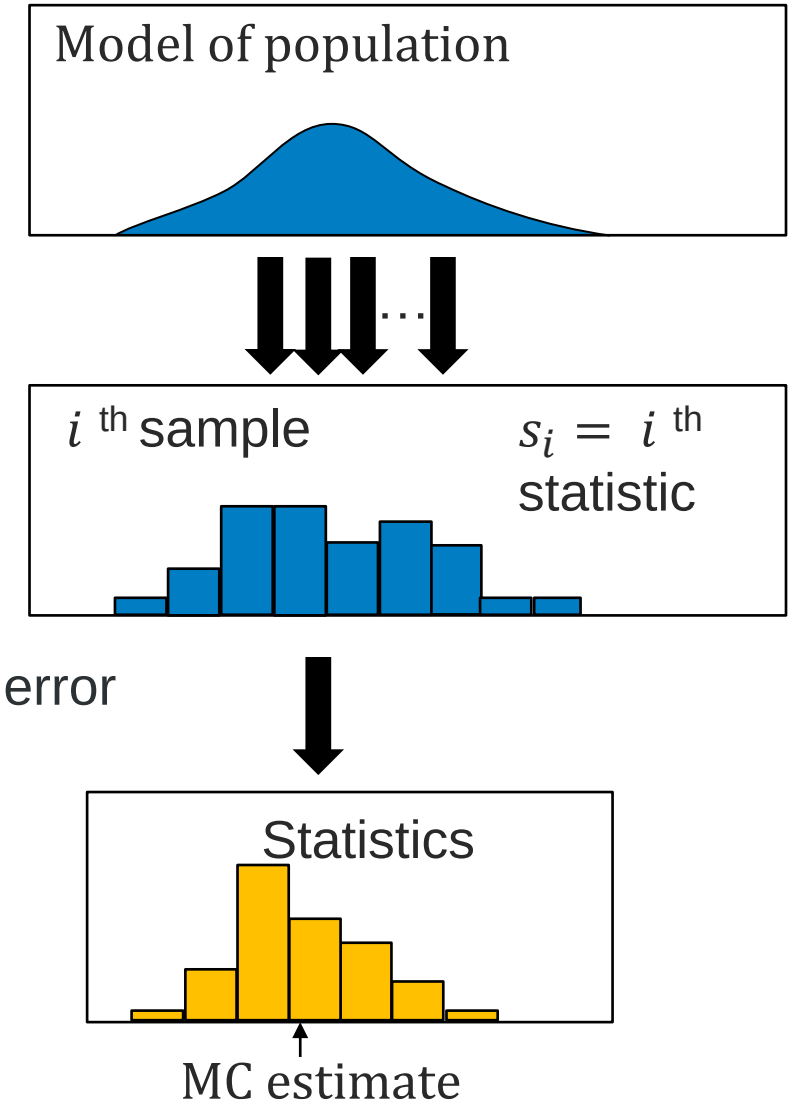


TIP 6: USE BY-GROUP PROCESSING FOR MONTE CARLO SIMULATIONS

TIP

Monte Carlo Simulation

- Repeat B times:
  - Simulate N observations
  - Compute statistic
- Union of statistics is approximate sampling distribution (ASD)
  - Mean is the Monte Carlo estimate
  - Standard deviation is estimate of standard error
  - Percentiles are estimates for CI
- In general, more simulations (larger B) ➔ better estimates



SIMULATE THE DISTRIBUTION OF THE SAMPLE MEAN

A sample of size $N = 10$ is drawn from $U(0,1)$. What is the distribution of the sample mean?

1. Generate **one** data set that contains **many** samples of size 10. Identify each sample by using a BY variable.
2. Compute the means of each sample by using the BY statement in the MEANS procedure.
3. Visualize the distribution for the many sample means. Compute summary statistics.

SIMULATE MULTIPLE SAMPLES WITH THE DATA STEP

```
%let N = 10;          /* num obs in each sample */
%let NumSamples = 1000;       /* num samples */
data Sim;
call streaminit(1);
do SampleID = 1 to &NumSamples;
   do i = 1 to &N;
      x = rand("Uniform");
      output;
   end;
end;
run;
```

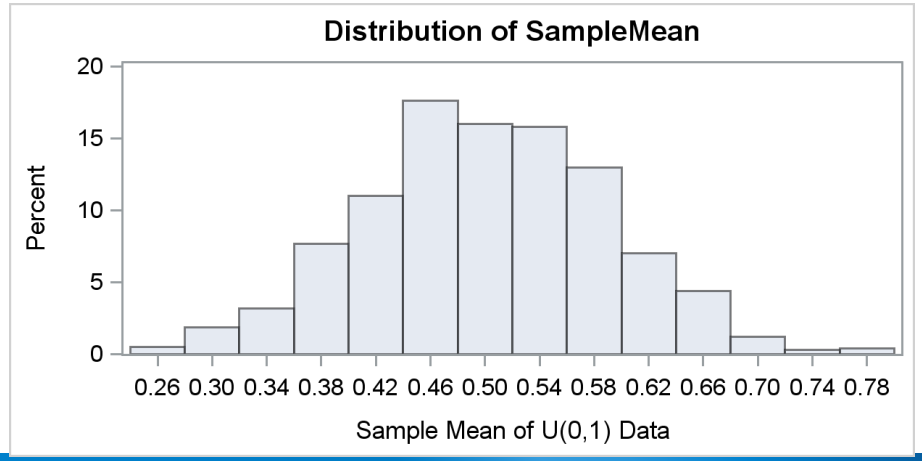
Obs	SampleID	i	x
1	1	1	0.88387
2	1	2	0.97382
3	1	3	0.50758
4	1	4	0.88694
⋮			
9995	1000	5	0.63604
9996	1000	6	0.20188
9997	1000	7	0.45520
9998	1000	8	0.64218
9999	1000	9	0.12999
10000	1000	10	0.99069

USE THE BY STATEMENT TO ANALYZE EACH SAMPLE

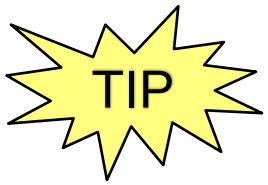
```
proc means data=Sim noprint;
  by SampleID;
  var x;
  output out=OutStats mean=SampleMean;
run;

proc univariate data=OutStats;
  var SampleMean;
  histogram SampleMean;
run;
```

Obs	SampleID	SampleMean
1	1	0.79583
2	2	0.65365
3	3	0.58957
4	4	0.55432
5	5	0.37845
6	6	0.45921

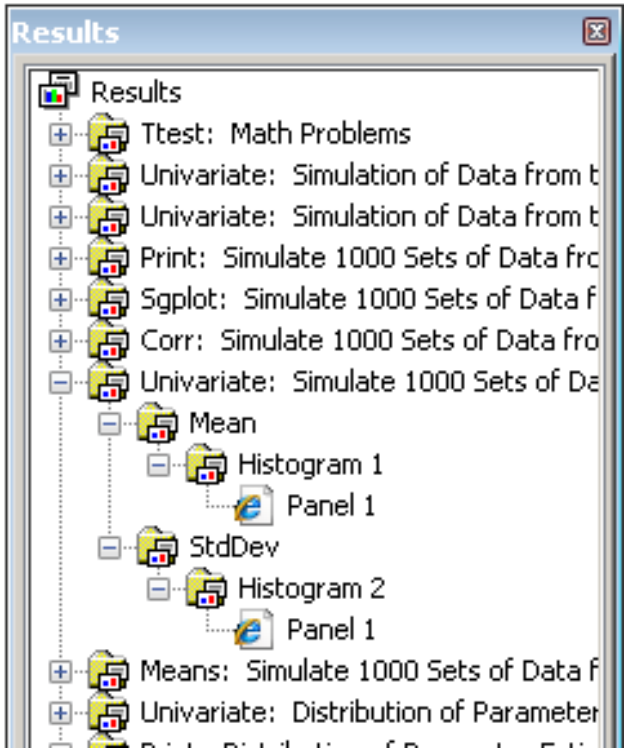
Analysis Variable : SampleMean				
N	Mean	Std Dev	5th Pctl	95th Pctl
1000	0.5017753	0.0901228	0.3518740	0.6526275



TIP 7: SUPPRESS OUTPUT DURING SIMULATION



- SAS procedures can create a LOT of output!
- For simple computations, it takes longer to print 1,000 tables than it does to compute them
- Suppress display of intermediate graphs and tables
  - About 50 procedures support the NOPRINT option
  - Use ODS to suppress output in other procedures
- Any unnecessary output will slow the simulation
  - NOTES can fill up the SAS Log
  - ODS Results window makes a node in a tree for every BY group



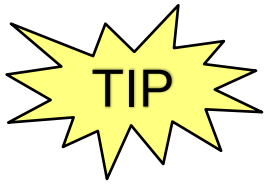
SAS MACROS FOR CONTROLLING OUTPUT

```
%macro ODSOff();     /* call prior to BY-group processing */
   ods graphics off;
   ods exclude all; /* all open destinations */
   ods noresults;   /* no updates to tree view */
   options nonotes; /* sometimes useful */
%mend;


%macro ODSOn();      /* call after BY-group processing */
   ods graphics on;
   ods exclude none;
   ods results;
   options notes;
%mend;
```

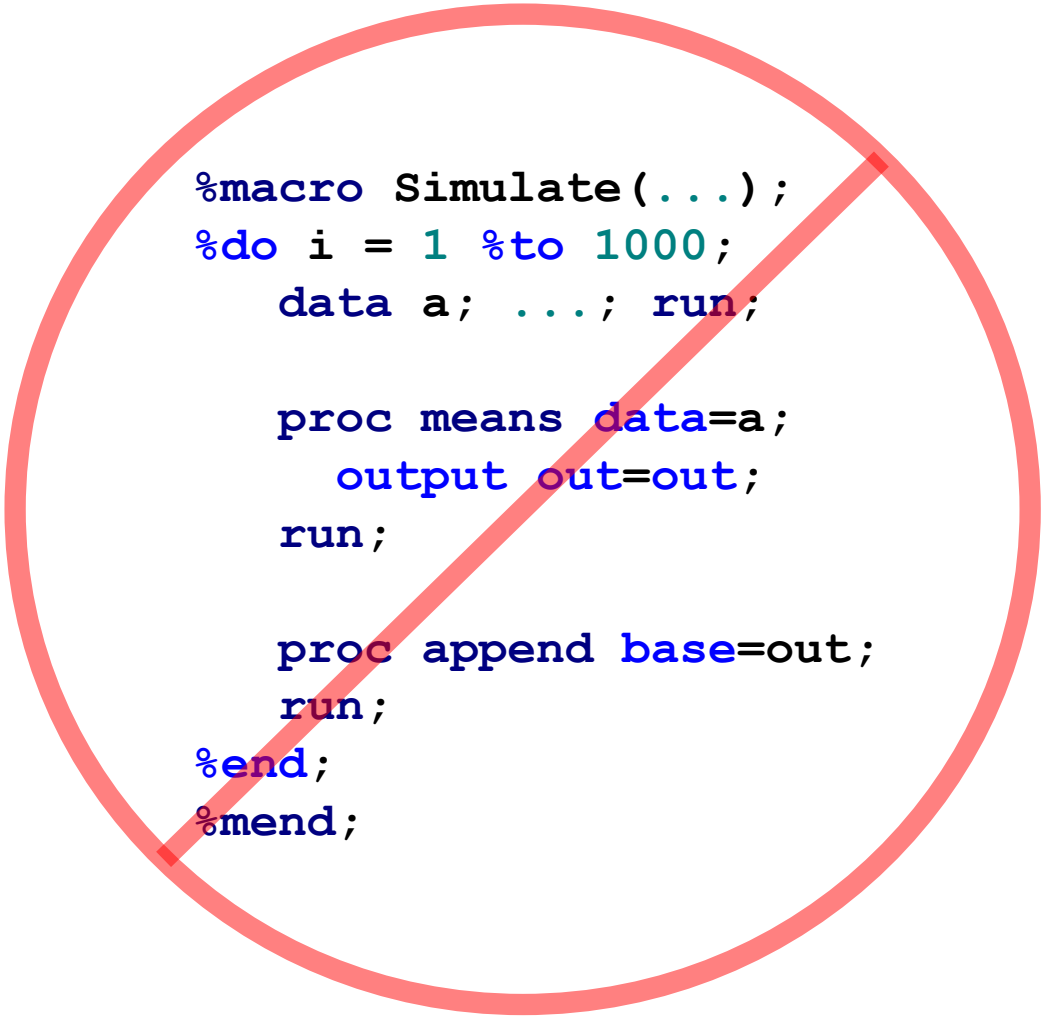
TEMPLATE FOR MONTE CARLO SIMULATION

```
data Sim;
do SampleID= 1 to &NumSamples;
   /* simulate data with known properties */
end;
run;

%ODSOff  /* or use NOPRINT option in procedure */
proc WhatEver data=Sim;
   by SampleID;
   /* generate statistic for simulated data */
   output out=OutStats ...;
run;
%ODSOn

/* use sampling distribution of statistic to answer
   questions about CIs, hypothesis tests, and more */
proc Univariate data=OutStats;
   var Stats;
run;
```

TIP 8: AVOID WRITING MACRO LOOPS



Do NOT write a SAS macro that creates and analyzes one sample at a time!



```
%macro Simulate(...);
%do i = 1 %to 1000;
   data a; ...; run;

   proc means data=a;
     output out=out;
   run;

   proc append base=out;
   run;
%end;
%mend;
```

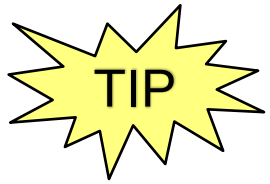
Novikov (2003) compares times for his application:
Macro > 7 minutes
BY group < 5 secs

➔ 100x faster!

WHY AVOID MACRO LOOPS?

- Each PROC and DATA step call requires overhead
  - Start procedure and parse statements
  - Open and load data set
  - [*Do computation*]
  - Close data set and exit procedure
- The macro code repeats these steps 1,000 times; the overhead costs **large** relative to the computation
- In contrast, BY group processing uses *one* DATA step and PROC call; overhead costs **small** relative to the computation
- If you omit the NONOTES option, the macro performance will be *even worse* and you might fill the SAS log with notes

TIP 9: USE SIMULATION TO ASSESS REGRESSION ESTIMATES



Components of regression models:

1. Explanatory variables: design matrix for fixed and random effects
2. Random error: distribution and correlation structures
3. Response variable:
   - Linear models: $\mathrm{Y} = X\beta + \epsilon$
   - Generalized models: $\eta = X\beta$, link function $g\big(E(Y)\big) = \eta$
     - » Logistic regression
     - » Poisson regression
   - Mixed models: $\mathrm{Y} = X\beta + Z\gamma + \epsilon$
   - Proportional hazards

SIMULATE SIMPLE REGRESSION MODEL FROM DESIGN

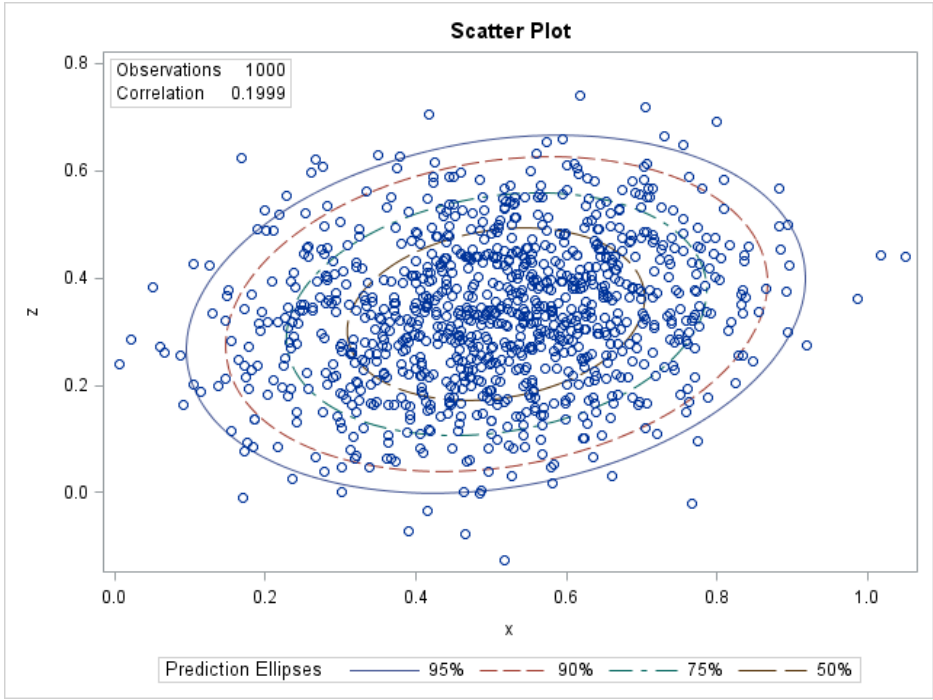
$$Y = 1 + X/2 + Z/3 + \epsilon, \quad \epsilon \sim N(0,1)$$

Assume the data set Explanatory contains variables X and Z

```
%let NumSamples = 1000;          /* number of samples */
data RegSim(drop=eta rmse);
   call streaminit(123);
   rmse = 1;                     /* scale of error term */
   set Explanatory;              /* implicit loop over obs*/
   ObsNum = _N_;                 /* in case obs order is important */
   eta = 1 + X/2 + Z/3;          /* linear predictor */
   do SampleID = 1 to &NumSamples;
      Y = eta + rand("Normal", 0, rmse); /* random error */
      output;
   end;
run;
/* sort for BY-group processing */
proc sort data=RegSim;  by SampleID ObsNum;  run;
```

SAMPLING DISTRIBUTION OF OLS PARAMETER ESTIMATES

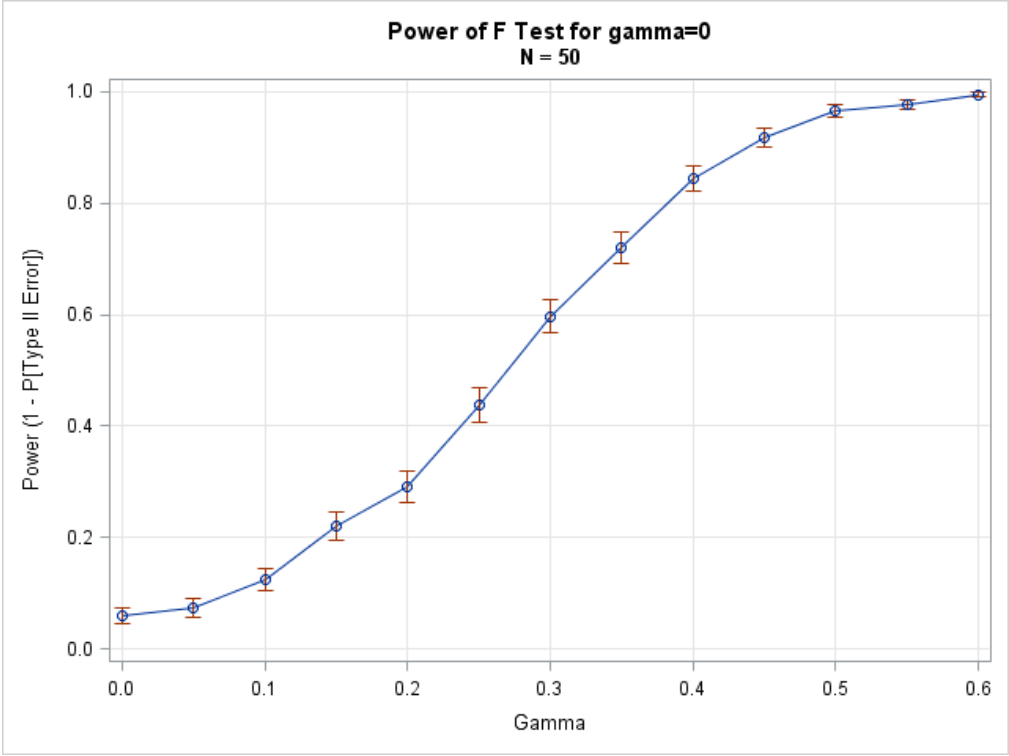
```
/* compute 1000 parameter estimates */
proc reg data=RegSim NOPRINT outest=OutEst;
  by SampleID;
  model y = x z;
run;

proc corr data=OutEst noprob
   plots=scatter(alpha=.05 .1 .25 .50);
   var Intercept x z;
run;
```



TIP 10: USE SIMULATION TO ESTIMATE POWER AND SAMPLE SIZE

TIP

- PROC POWER can provide a power curve for many situations, but theoretical results are not always available
- Simulation provides a way to obtain a power curve as a function of the sample size or effect size



ESTIMATE POWER STEP 1

SIMULATE THE REGRESSION DATA

```
%let NumSamples = 1000;                    /* number of samples     */

data PowerSim(drop=eta);
call streaminit(1);
set Explanatory;                           /* fixed vals for x & z */
do gamma = 0 to 0.6 by 0.05;               /* vary size of gamma    */
   eta = 1 + x/2 + gamma*z;                /* linear predictor      */
   do SampleID = 1 to &NumSamples;
      y = eta + rand("Normal");
      output;
   end;
end;                                       /* end gamma loop        */
run;

proc sort data=PowerSim out=Sim;           /* SORT before running   */
   by gamma SampleID;                      /* BY-group analysis     */
run;
```

ESTIMATE POWER
STEP 2

RUN HYPOTHESIS TESTS

```
/* turn off output when calling PROC for simulation */
%ODSOff
proc reg data=PowerSim;
   by gamma SampleID;    /* 1000 samples for each value of gamma */
   model y = x z;
   test z=0;             /* creates TestANOVA table for each BY group */
   ods output TestANOVA=TestAnova(where=(Source="Numerator"));
quit;
%ODSOn
```

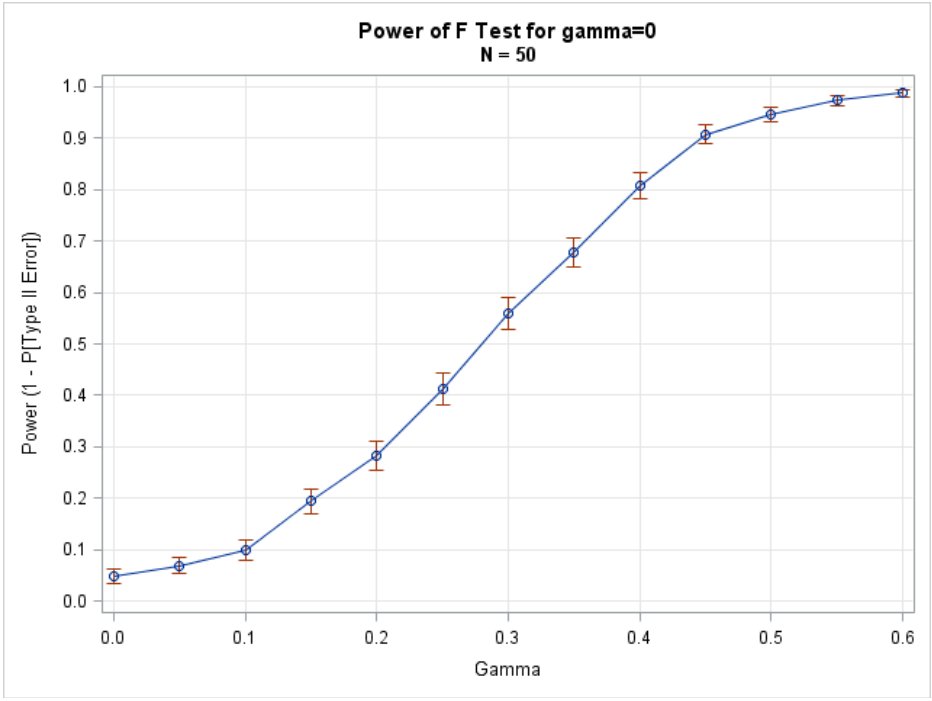
Obs	gamma	SampleID	FValue	ProbF
218	0	218	3.11	0.0841
219	0	219	4.88	0.0321
220	0	220	0.67	0.4175
221	0	221	1.35	0.2505
222	0	222	0.22	0.6418
223	0	223	0.13	0.7223
224	0	224	0.58	0.4483
225	0	225	0.25	0.6163
226	0	226	4.10	0.0485
227	0	227		

Reject H0
at $\alpha = 0.05$

ESTIMATE POWER STEP 3

ESTIMATE POWER OF REGRESSION TEST

```
/* create indicator variable for samples for which H0 is rejected */
data Results;
   set TestANOVA;
   Reject = (ProbF <= 0.05); /* indicator var is 1 if H0 rejected */
run;

/* count number of times H0 is rejected (and CI for proportion) */
proc freq data=Results noprint;
   by gamma;
   tables Reject / binomial(level='1') nocum;
   output out=Signif binomial;
run;

proc sgplot data=Signif noautolegend;
   series  x=gamma y=_BIN_;
   scatter x=gamma y=_BIN_ /
         yerrorlower = L_Bin
         yerrorupper = U_Bin;
run;
```



SUMMARY

1. Use the RAND function
2. Use the “Table” distribution
3. Use mixtures of distributions
4. Construct complex distributions from simple ones
5. Use the SAS/IML language for multivariate simulation
6. Use BY-group processing for efficient Monte Carlo simulations
7. Suppress output
8. Avoid macro loops
9. Use simulation to assess regression estimates
10. Use simulation for power and sample size estimates

All these tips (and much MORE!) in *Simulating Data with SAS* (Wicklin 2013)

FURTHER READING

- *Simulating Data with SAS*
  - Hundreds of programs and examples
  - Free download of programs: http://support.sas.com/wicklin
- *The DO Loop*: A statistical programming blog
  - http://blogs.sas.com/content/iml
  - Click on **Simulation** in the word cloud to see simulation articles

SUPPORT.SAS.COM/STATISTICS