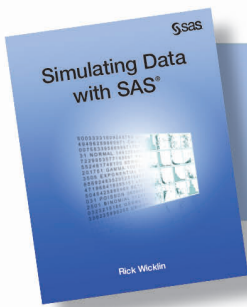


Simulating Data with SAS[®]



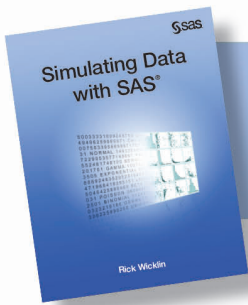
Rick Wicklin



From *Simulating Data with SAS*[®]. Full book available for purchase [here](#).

Contents

Acknowledgments	v
I Essentials of Simulating Data	1
Chapter 1. Introduction to Simulation	3
Chapter 2. Simulating Data from Common Univariate Distributions	11
Chapter 3. Preliminary and Background Information	29
II Basic Simulation Techniques	49
Chapter 4. Simulating Data to Estimate Sampling Distributions	51
Chapter 5. Using Simulation to Evaluate Statistical Techniques	73
Chapter 6. Strategies for Efficient and Effective Simulation	93
III Advanced Simulation Techniques	107
Chapter 7. Advanced Simulation of Univariate Data	109
Chapter 8. Simulating Data from Basic Multivariate Distributions	129
Chapter 9. Advanced Simulation of Multivariate Data	153
Chapter 10. Building Correlation and Covariance Matrices	175
IV Applications of Simulation in Statistical Modeling	195
Chapter 11. Simulating Data for Basic Regression Models	197
Chapter 12. Simulating Data for Advanced Regression Models	225
Chapter 13. Simulating Data from Times Series Models	251
Chapter 14. Simulating Data from Spatial Models	263
Chapter 15. Resampling and Bootstrap Methods	281
Chapter 16. Moment Matching and the Moment-Ratio Diagram	297
V Appendix	323
Appendix A. A SAS/IML Primer	325
Index	339



From *Simulating Data with SAS*[®]. Full book available for purchase [here](#).

Chapter 2

Simulating Data from Common Univariate Distributions

Contents

2.1	Introduction to Simulating Univariate Data	11
2.2	Getting Started: Simulate Data from the Standard Normal Distribution	12
2.3	Template for Simulating Univariate Data in the DATA Step	13
2.4	Simulating Data from Discrete Distributions	14
2.4.1	The Bernoulli Distribution	14
2.4.2	The Binomial Distribution	15
2.4.3	The Geometric Distribution	16
2.4.4	The Discrete Uniform Distribution	17
2.4.5	Tabulated Distributions	18
2.4.6	The Poisson Distribution	19
2.5	Simulating Data from Continuous Distributions	20
2.5.1	The Normal Distribution	21
2.5.2	The Uniform Distribution	22
2.5.3	The Exponential Distribution	22
2.6	Simulating Univariate Data in SAS/IML Software	24
2.6.1	Simulating Discrete Data	24
2.6.2	Sampling from Finite Sets	25
2.6.3	Simulating Continuous Data	26
2.7	Univariate Distributions Supported in SAS Software	27
2.8	References	28

2.1 Introduction to Simulating Univariate Data

There are three primary ways to simulate data in SAS software:

- Use the DATA step to simulate data from univariate and uncorrelated multivariate distributions. You can use the RAND function to generate random values from more than 20 standard univariate distributions. You can combine these elementary distributions to build more complicated distributions.

12 Chapter 2: Simulating Data from Common Univariate Distributions

- Use the SAS/IML language to simulate data from many distributions, including correlated multivariate distributions. You can use the RANDGEN subroutine to generate random values from standard univariate distributions, or you can use several predefined modules to generate data from multivariate distributions. You can extend the SAS/IML language by defining new functions that sample from distributions that are not built into SAS.
- Use specialized procedures in SAS/STAT software and SAS/ETS software to simulate data with special properties. Procedures that generate random samples include the SIMNORMAL, SIM2D, and COPULA procedures.

This chapter describes the two most important techniques that are used to simulate data in SAS software: the DATA step and the SAS/IML language. Although the DATA step is a useful tool for simulating univariate data, SAS/IML software is more powerful for simulating multivariate data. To learn how to use the SAS/IML language effectively, see Wicklin (2010).

Most of the terminology in this book is standard. However, a term that you might not be familiar with is the term *random variate*. A random variate is a particular outcome of a random variable (Devroye 1986). For example, let X be a Bernoulli random variable that takes on the value 1 with probability p and the value 0 with probability $1 - p$. If you draw five observations from the probability distribution, you might obtain the values 0, 1, 1, 0, 1. Those five numbers are random variates. This book also uses the terms “simulated values” and “simulated data.” Some authors refer to simulated data as “fake data.”

2.2 Getting Started: Simulate Data from the Standard Normal Distribution

To “simulate data” means to generate a random sample from a distribution with known properties. Because an example is often an effective way to convey main ideas, the following DATA step generates a random sample of 100 observations from the standard normal distribution. Figure 2.1 shows the first five observations.

```
data Normal (keep=x) ;
call streaminit (4321) ;           /* Step 1 */
do i = 1 to 100 ;                 /* Step 2 */
    x = rand ("Normal") ;         /* Step 3 */
    output ;
end ;
run ;

proc print data=Normal (obs=5) ;
run ;
```

Figure 2.1 A Few Observations from a Normal Distribution

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

The DATA step consists of three steps:

1. Set the seed value with the STREAMINIT function. Seeds for random number generation are discussed further in Section 3.3.
2. Use a DO loop to iterate 100 times.
3. For each iteration, call the RAND function to generate a random value from the standard normal distribution.

If you change the seed value, you will get a different random sample. If you change the number 100, you will get a sample with a different number of observations. To get a nonnormal distribution, change the name of the distribution from “Normal” to one of the families listed in Section 2.7. Some distributions, including the normal distribution, include parameters that you can specify after the name.

2.3 Template for Simulating Univariate Data in the DATA Step

It is easy to generalize the example in the previous section. The following SAS pseudocode shows a basic template that you can use to generate N observations with a specified distribution:

```
%let N = 100;                                /* size of sample */

data Sample(keep=x);
call streaminit(4321);                        /* or use a different seed */
do i = 1 to &N;                                /* &N is the value of the N macro var */
  /* specify distribution and parameters */
  x = rand("DistribName", param1, param2, ...);
  output;
end;
run;
```

The simulated data are written to the Sample data set. The macro variable **N** is defined in order to emphasize the role of that parameter. The expression **&N** is replaced by the value of the macro parameter (here, 100) before the DATA step is run.

The (pseudo) DATA step demonstrates the following steps for simulating data:

1. A call to the STREAMINIT subroutine, which specifies the seed that initializes the random number stream. When the argument is a positive integer, as in this example, the random sequence is reproducible. If you specify 0 as the argument, the random number sequence is initialized from your computer's internal system clock. This implies that the random sequence will be different each time that you run the program. Seeds for random number generation are discussed in Section 3.3.
2. A DO loop that iterates N times.
3. A call to the RAND function, which generates one random value each time that the function is called. The first argument is the name of a distribution. The supported distributions are enumerated in Section 2.7. Subsequent arguments are parameter values for the distribution.

2.4 Simulating Data from Discrete Distributions

When the set of possible outcomes is finite or countably infinite (like the integers), assigning a probability to each outcome creates a *discrete probability distribution*. Of course, the sum of the probabilities over all outcomes is unity.

The following sections generate a sample of size $N = 100$ from some well-known discrete distributions. The code is followed by a frequency plot of the sample, which is overlaid with the exact probabilities of obtaining each value. You can use PROC FREQ to compute the empirical distribution of the data; the exact probabilities are obtained from the probability mass function (PMF) of the distribution. Section 3.4.2 describes how to overlay a bar chart with a scatter plot that shows the theoretical probabilities.

2.4.1 The Bernoulli Distribution

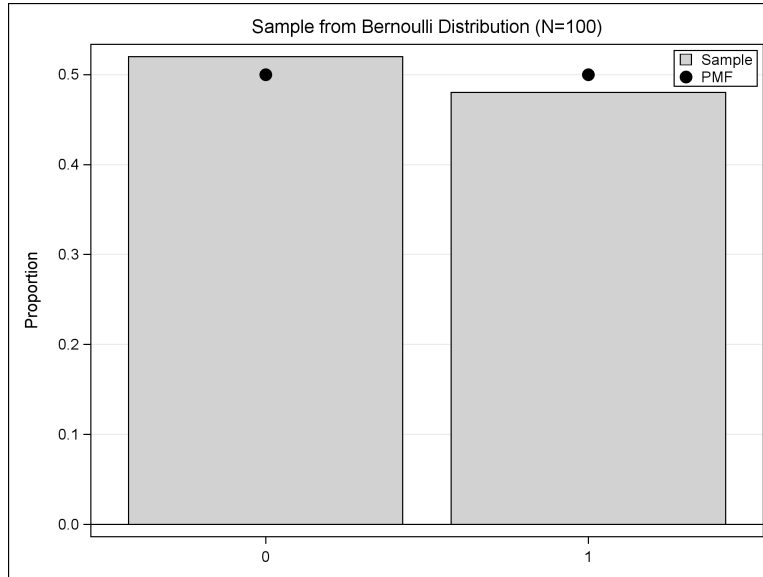
The Bernoulli distribution is a discrete probability distribution on the values 0 and 1. The probability that a Bernoulli random variable will be 1 is given by a parameter, p , $0 \leq p \leq 1$. Often a 1 is labeled a “success,” whereas a 0, which occurs with probability $1 - p$, is labeled a “failure.”

The following DATA step generates a random sample from the Bernoulli distribution with $p = 1/2$. If you identify $x = 1$ with “heads” and $x = 0$ with “tails,” then this DATA step simulates $N = 100$ tosses of a fair coin.

```
%let N = 100;
data Bernoulli(keep=x);
call streaminit(4321);
p = 1/2;
do i = 1 to &N;
    x = rand("Bernoulli", p);          /* coin toss */
    output;
end;
run;
```

You can use the FREQ procedure to count the outcomes in this simulated data. For this sample, the value 0 appeared 52 times, and the value 1 appeared 48 times. These frequencies are shown by the bar chart in Figure 2.2. The expected percentages for each result are shown by the round markers.

Figure 2.2 Sample from Bernoulli Distribution ($p = 1/2$) Overlaid with PMF



If X is a random variable from the Bernoulli distribution, then the expected value of X is p and the variance is $p(1 - p)$. In practice, this means that if you generate a large random sample from the Bernoulli distribution, you can expect the sample to have a sample mean that is close to p and a sample variance that is close to $p(1 - p)$.

2.4.2 The Binomial Distribution

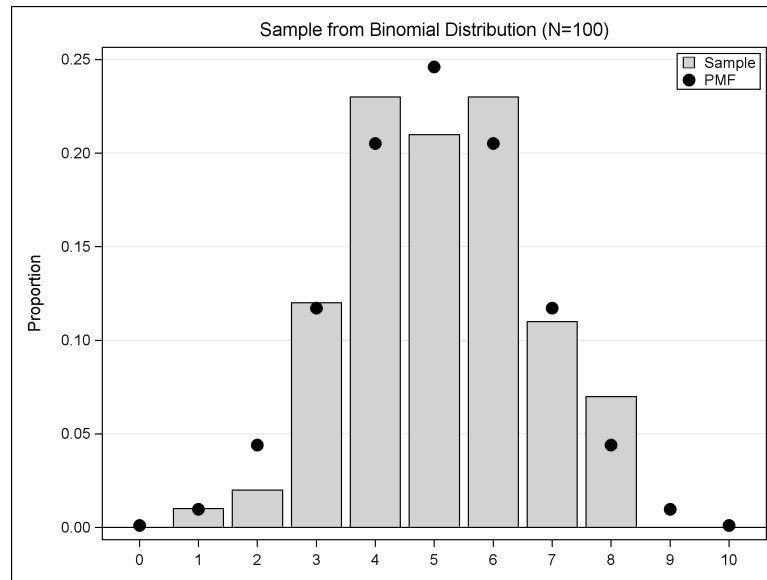
Imagine repeating a Bernoulli trial n times, where each trial has a probability of success equal to p . If p is large (near 1), you expect most of the Bernoulli trials to be successes and only a few of the trials to be failures. On the other hand, if p is near $1/2$, you expect to get about $n/2$ successes.

The binomial distribution models the number of successes in a sequence of n independent Bernoulli trials. The following DATA step generates a random sample from the binomial distribution with $p = 1/2$ and $n = 10$. This DATA step simulates a series of coin tosses. For each trial, the coin is tossed 10 times and the number of heads is recorded. This experiment is repeated $N = 100$ times. Figure 2.3 shows a frequency plot of the results.

```

data Binomial(keep=x);
call streaminit(4321);
p = 1/2;
do i = 1 to &N;
    x = rand("Binomial", p, 10);    /* number of heads in 10 tosses */
    output;
end;
run;

```

Figure 2.3 Sample from Binomial Distribution ($p = 1/2, n = 10$) Overlaid with PMF

For this series of experiments, you expect to get five heads most frequently, followed closely by four and six heads. The expected percentages are indicated by the round markers. For this particular simulation, Figure 2.3 shows that four heads and six heads appeared more often than five heads appeared. The sample values are shown by the bars; the expected percentages are shown by round markers.

If X is a random variable from the binomial(p, n) distribution, then the expected value of X is np and the variance is $np(1 - p)$. In practice, this means that if you generate a large random sample from the binomial(p, n) distribution, then you can expect the sample to have a sample mean that is close to np .

Some readers might be concerned that the distribution of the sample shown in Figure 2.3 differs so much from the theoretical distribution of the binomial distribution. This deviation is not an indication that something is wrong. Rather, it demonstrates *sampling variation*. When you simulate data from a population model, the data will almost always look slightly different from the distribution of the population. Some values will occur more often than expected; some will occur less often than expected. This is especially apparent in small samples and for distributions with large variance. It is this sampling variation that makes simulation so valuable.

2.4.3 The Geometric Distribution

How many times do you need to toss a fair coin before you see heads? Half the time you will see heads on the first toss, one quarter of the time it requires two tosses, and so on. This is an example of a geometric distribution.

In general, the geometric distribution models the number of Bernoulli trials (with success probability p) that are required to obtain one success. An alternative definition, which is used by the MCMC procedure in SAS, is to define the geometric distribution to be the number of *failures* before the first success.

You can simulate a series of coin tosses in which the coin is tossed until a heads appears and the number of tosses is recorded. If p is the probability of tossing heads, then the following statement generates an observation from the $\text{Geometric}(p)$ distribution:

```
x = rand("Geometric", p);      /* number of trials until success */
```

Figure 3.6 shows a graph of simulated geometric data and an overlaid PMF.

If X is a random variable from the $\text{geometric}(p)$ distribution, then the expected value of X is $1/p$ and the variance is $(1 - p)/p^2$.

Exercise 2.1: Write a DATA step that simulates observations from a $\text{Geometric}(0.5)$ distribution.

2.4.4 The Discrete Uniform Distribution

A Bernoulli distribution models two outcomes. You can model situations in which there are multiple outcomes by using either the discrete uniform distribution or the “Table” distribution (see the next section).

When you toss a standard six-sided die, there is an equal probability of seeing any of the six faces. You can use the discrete uniform distribution to produce k integers in the range $[1, k]$. SAS does not have a built-in discrete uniform distribution. Instead, you can use the continuous uniform distribution to produce a random number u in the interval $(0, 1)$, and you can use the CEIL function to produce the smallest integer that is greater than or equal to ku .

The following DATA step generates a random sample from the discrete uniform distribution with $k = 6$. This DATA step simulates $N = 100$ rolls of a fair six-sided die.

```
data Uniform(keep=x);
call streaminit(4321);
k = 6;                               /* a six-sided die          */
do i = 1 to &N;
    x = ceil(k * rand("Uniform"));    /* roll 1 die with k sides */
    output;
end;
run;
```

You can also simulate data with uniform probability by using the “Table” distribution, which is described in the next section.

To check the empirical distribution of the simulated data, you can use PROC FREQ to show the distribution of the x variable. The results are shown in Figure 2.4. As expected, each number 1, 2, . . . , 6 is generated about 16% of the time.

```
proc freq data=Uniform;
    tables x / nocum;
run;
```

Figure 2.4 Sample from Uniform Distribution ($k = 6$)

The FREQ Procedure

x	Frequency	Percent
1	15	15.00
2	18	18.00
3	15	15.00
4	12	12.00
5	22	22.00
6	18	18.00

2.4.5 Tabulated Distributions

In some situations there are multiple outcomes, but the probabilities of the outcomes are not equal. For example, suppose that there are 10 socks in a drawer: five are black, two are brown, and three are white. If you close your eyes and draw a sock at random, the probability of that sock being black is 0.5, the probability of that sock being brown is 0.2, and the probability of that sock being white is 0.3. After you record the color of the sock, you can replace the sock, mix up the drawer, close your eyes, and draw again.

The RAND function supports a “Table” distribution that enables you to specify a table of probabilities for each of k outcomes. You can use the “Table” distribution to sample with replacement from a finite set of outcomes where you specify the probability for each outcome. In SAS/IML software, you can use the RANDGEN or SAMPLE routines.

The following DATA step generates a random sample of size $N = 100$ from the “Table” distribution with probabilities $p = \{0.5, 0.3, 0.2\}$. You can use PROC FREQ to display the observed frequencies, which are shown in Figure 2.5.

```

data Table(keep=x);
call streaminit(4321);
p1 = 0.5; p2 = 0.2; p3 = 0.3;
do i = 1 to &N;
    x = rand("Table", p1, p2, p3);          /* sample with replacement */
    output;
end;
run;

proc freq data=Table;
    tables x / nocum;
run;

```

Figure 2.5 Sample from “Table” Distribution ($p = \{0.5, 0.2, 0.3\}$)**The FREQ Procedure**

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

For the simulated sock experiment with the given probabilities, a black sock (category 1) was drawn 48 times, a brown sock (category 2) was drawn 21 times, and a white sock was drawn 31 times.

If you have many potential outcomes, it would be tedious to specify the probabilities of each outcome by using a comma-separated list. Instead, it is more convenient to specify an array in the DATA step to hold the probabilities, and to use the OF operator to list the values of the array as shown in the following example:

```
data Table(keep=x);
call streaminit(4321);
array p[3] _temporary_ (0.5 0.2 0.3);
do i = 1 to &N;
    x = rand("Table", of p[*]);          /* sample with replacement */
    output;
end;
run;
```

The `_TEMPORARY_` keyword makes `p` a temporary array that holds the parameter values. The elements of a temporary array do not have names and are not written to the output data set, which means that you do not need to use a `DROP` or `KEEP` option to omit them from the data set.

The “Table” distribution is related to the multinomial distribution, which is discussed in Section 8.2. If you generate N observations from the “Table” distribution and tabulate the frequencies for each category, then the frequency vector is a single observation from the multinomial distribution. Consequently, the “Table” and multinomial distributions are related in the same way that the Bernoulli and binomial distributions are related.

Exercise 2.2: Use the “Table” distribution to simulate rolls from a six-sided die.

2.4.6 The Poisson Distribution

Suppose that during the work day a worker receives email at an average rate of four messages per hour. What is the probability that she might get seven messages in an hour? Or that she might get only one message? The Poisson distribution models the counts of an event during a given time period, assuming that the event happens at a constant average rate.

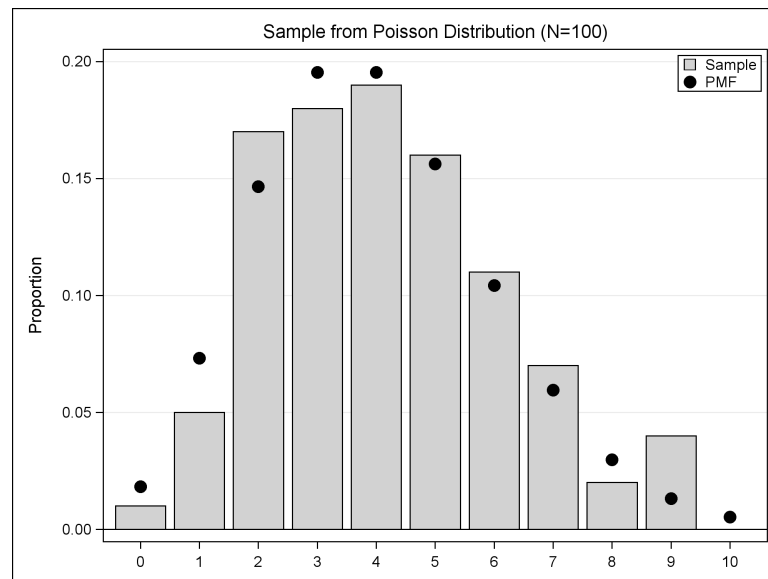
For an average rate of λ events per time period, the expected value of a random variable from the Poisson distribution is λ , and the variance is also λ .

The following DATA step generates a random sample from the Poisson distribution with $\lambda = 4$. This DATA step simulates the number of emails that a worker receives each hour, under the assumption

that the number of emails arrive at a constant average rate of four emails per hour. This experiment simulates $N = 100$ hours at work. The results are shown in Figure 2.6.

```
data Poisson(keep=x);
call streaminit(4321);
lambda = 4;
do i = 1 to &N;
    x = rand("Poisson", lambda);      /* num events per unit time */
    output;
end;
run;
```

Figure 2.6 Sample from Poisson Distribution ($\lambda = 4$) Overlaid with PMF



For the Poisson model, the worker can expect to receive four emails during a one-hour period about 20% of the time. The same is true for receiving three emails in an hour. She can expect to receive six emails during an hour slightly more than 10% of the time. These expected percentages are shown by the round markers. The “actual” number of emails received during each hour is shown by the bar chart for the 100 simulated hours. There were 18 one-hour periods during which the worker received three emails. There were 11 one-hour periods during which the worker received six emails.

Exercise 2.3: A negative binomial variable is defined as the number of failures before k successes in a series of independent Bernoulli trials with probability of success p . Define a trial as rolling a six-sided die until a specified face appears $k = 3$ times. Simulate 1,000 trials and plot the distribution of the number of failures.

2.5 Simulating Data from Continuous Distributions

When the set of possible outcomes is uncountably infinite (like an interval or the set of real numbers), assigning a probability to each outcome creates a *continuous probability distribution*. Of course, the integral of the probabilities over the set is unity.

The following sections generate a sample of size $N = 100$ from some well-known continuous distributions. Most sections also show a histogram of the sample that is overlaid with the probability density curve for the population. The probability density function (PDF) is described in Section 3.2. Section 3.4.3 describes how to create the graphs.

See Table 2.3 for a list of common distributions that SAS supports.

2.5.1 The Normal Distribution

The normal distribution with mean μ and standard deviation σ is denoted by $N(\mu, \sigma)$. Its density is given by the following:

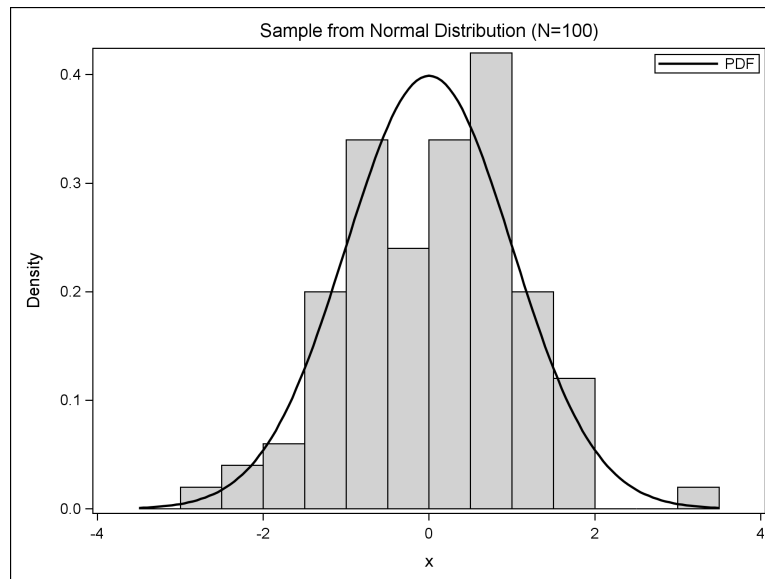
$$f(x; \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right)$$

The *standard* normal distribution sets $\mu = 0$ and $\sigma = 1$.

Many physical quantities are modeled by the normal distribution. Perhaps more importantly, the sampling distribution of many statistics are approximately normally distributed.

Section 2.2 generated 100 observations from the standard normal distribution. Figure 2.7 shows a histogram of the simulated data along with the graph of the probability density function. For this sample, the histogram bars are below the PDF curve for some intervals and are greater than the curve for other intervals. A second sample of 100 observations is likely to produce a different histogram.

Figure 2.7 Sample from a Normal Distribution ($\mu = 0, \sigma = 1$) Overlaid with PDF



To explicitly specify values of the location and scale parameters, define **mu** and **sigma** outside of the DO loop, and then use the following statement inside the DO loop:

```
x = rand("Normal", mu, sigma);      /* X ~ N(mu, sigma) */
```

If X is a random variable from the $N(\mu, \sigma)$ distribution, then the expected value of X is μ and the variance is σ^2 . Be aware that some authors denote the normal distribution by $N(\mu, \sigma^2)$, where

the second parameter indicates the *variance*. This book uses $N(\mu, \sigma)$ instead, which matches the meaning of the parameters in the RAND function.

2.5.2 The Uniform Distribution

The uniform distribution is one of the most useful distributions in statistical simulation. One reason is that you can use the uniform distribution to sample from a finite set. Another reason is that “random variates with various distributions can be obtained by cleverly manipulating” independent, identically distributed uniform variates (Devroye 1986, p. 206).

The uniform distribution on the interval (a, b) is denoted by $U(a, b)$. Its density is given by $f(x) = (b - a)^{-1}$ for x in (a, b) . The standardized uniform distribution on $[0, 1]$ (often called *the* uniform distribution) is denoted $U(0, 1)$.

You can use the following statement in the DATA step to generate a random observation from the standard uniform distribution:

```
x = rand("Uniform");          /* X ~ U(0, 1) */
```

The uniform random number generator never generates the number 0 nor the number 1. Therefore, all values are in the open interval $(0, 1)$.

You can also use the uniform distribution to sample random values from $U(a, b)$. To do this, define **a** and **b** outside of the DO loop, and then use the following statement inside the DO loop:

```
y = a + (b-a)*rand("Uniform"); /* Y ~ U(a, b) */
```

If X is a random variable from the standard uniform distribution, then the expected value of X is $1/2$ and the variance is $1/12$. In general, the uniform distribution on (a, b) has a uniform density of $1/(b - a)$. If Y is a random variable from the $U(a, b)$, the expected value of Y is $(a + b)/2$ and the variance is $(b - a)^2/12$.

Exercise 2.4: Generate 100 observations from a uniform distribution on the interval $(-1, 1)$.

2.5.3 The Exponential Distribution

The exponential distribution models the time between events that occur at a constant average rate. The exponential distribution is a continuous analog of the geometric distribution. The classic usage of the exponential distribution is to model the time between detecting particles emitted during radioactive decay.

The exponential distribution with scale parameter σ is denoted $\text{Exp}(\sigma)$. Its density is given by $f(x) = (1/\sigma) \exp(-x/\sigma)$ for $x > 0$. Alternatively, you can use $\lambda = 1/\sigma$, which is called the *rate parameter*. The rate parameter describes the rate at which an event occurs.

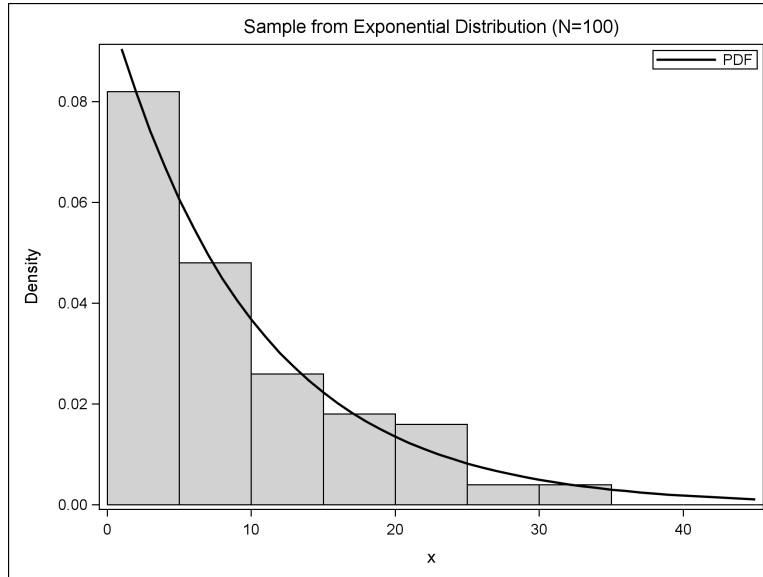
The following DATA step generates a random sample from the exponential distribution with scale parameter $\sigma = 10$. A histogram of the sample is shown in Figure 2.8.

```

data Exponential(keep=x);
call streaminit(4321);
sigma = 10;
do i = 1 to &N;
    x = sigma * rand("Exponential");    /* X ~ Expo(10) */
    output;
end;
run;

```

Figure 2.8 Sample from the Exponential Distribution ($\sigma = 10$) Overlaid with PDF



Notice that the scale parameter for the exponential distribution is not supported by the RAND function as of SAS 9.3. However, you can show that if X is distributed according to an exponential distribution with unit scale parameter, then $Y = \sigma X$ is distributed exponentially with scale parameter σ . The expected value of X is σ ; the variance is σ^2 . For example, the data shown in Figure 2.8 have a mean close to $\sigma = 10$.

If you use the exponential distribution with a scale parameter frequently, you might want to define and use the following SAS macro, which is used in Chapter 7 and in Chapter 12:

```

%macro RandExp(sigma);
    ((&sigma) * rand("Exponential"))
%mend;

```

The following statement shows how to call the macro from the DATA step:

```
x = %RandExp(sigma);
```

Exercise 2.5: Some distributions include the exponential distribution for particular values of the distribution parameters. For example, a Weibull(1, b) distribution is an exponential distribution with scale parameter b . Modify the program in this section to simulate data as follows:

```
x = rand("Weibull", 1, sigma);
```

Do you obtain a similar distribution of values? Use PROC UNIVARIATE to fit the exponential model to the simulated data.

2.6 Simulating Univariate Data in SAS/IML Software

You can also generate random samples by using the RANDGEN subroutine in SAS/IML software. The RANDGEN subroutine uses the same algorithms as the RAND function, but it fills an entire matrix at once, which means that you do not need a DO loop.

The following SAS/IML pseudocode simulates N observations from a named distribution:

```
%let N = 100;
proc iml;
call randseed(4321);          /* or use a different seed */
x = j(1, &N);                /* allocate vector or matrix */
call randgen(x, "DistribName", param1, param2, ...); /* fill x */
```

The PROC IML program contains the following function calls:

1. A call to the RANDSEED subroutine, which specifies the seed that initializes the random number stream. If the argument is a positive integer, then the sequence is reproducible. Otherwise, the system time is used to initialize the random number stream, and the sequence will be different each time that you run the program.
2. A call to the J function, which allocates a matrix of a certain size. The syntax $J(r, c)$ creates an $r \times c$ matrix. For this example, \mathbf{x} is a vector that has one row and N columns.
3. A call to the RANDGEN subroutine, which fills the elements of \mathbf{x} with random values from a named distribution. The supported distributions are listed in Section 2.7.

When you use the J function to allocate a SAS/IML matrix, the matrix is filled with 1s by default. However, you can use an optional third argument to fill the matrix with another value. For example $\mathbf{y}=j(1, 5, 0)$ allocates a 1×5 vector where each element has the value 0, and $\mathbf{y}=j(4, 3, .)$ allocates a 4×3 matrix where each element is a SAS missing value.

Notice that the SAS/IML implementation is more compact than the DATA step implementation. It does not create a SAS data set, but instead holds the simulated data in memory in the \mathbf{x} vector. By not writing a data set, the SAS/IML program is more efficient. However, both programs are blazingly fast. On the author's PC, generating a million observations with the DATA step takes about 0.2 seconds. Simulating the same data in PROC IML takes about 0.04 seconds.

2.6.1 Simulating Discrete Data

The RANDGEN subroutine in SAS/IML software supports the same distributions as the RAND function. Because the IML procedure does not need to create a data set that contains the simulated data, well-written simulations in the SAS/IML language have good performance characteristics.

The previous sections showed how to use the DATA step to generate random data from various distributions. The following SAS/IML program generates samples of size $N = 100$ from the same set of distributions:

```
proc iml;
  /* define parameters */
  p = 1/2;  lambda = 4;  k = 6;  prob = {0.5 0.2 0.3};

  /* allocate vectors */
  N = 100;
  Bern = j(1, N);  Bino = j(1, N);  Geom = j(1, N);
  Pois = j(1, N);  Unif = j(1, N);  Tabl = j(1, N);

  /* fill vectors with random values */
  call randseed(4321);
  call randgen(Bern, "Bernoulli", p);      /* coin toss */
  call randgen(Bino, "Binomial", p, 10);  /* num heads in 10 tosses */
  call randgen(Geom, "Geometric", p);     /* num trials until success */
  call randgen(Pois, "Poisson", lambda);  /* num events per unit time */
  call randgen(Unif, "Uniform");          /* uniform in (0,1) */
  Unif = ceil(k * Unif);                  /* roll die with k sides */
  call randgen(Tabl, "Table", prob);      /* sample with replacement */
```

Notice that in the SAS/IML language, which supports vectors in a natural way, the syntax for the “Table” distribution is simpler than in the DATA step. You simply define a vector of parameters and pass the vector to the RANDGEN subroutine. For example, you can use the following SAS/IML program to simulate data from a discrete uniform distribution as described in Section 2.4.4. The program simulates the roll of a six-sided die by using the RANDGEN subroutine to sample from six outcomes with equal probability:

```
proc iml;
  call randseed(4321);
  prob = j(6, 1, 1)/6;      /* equal prob. for six outcomes */
  d = j(1, &N);             /* allocate 1 x N vector */
  call randgen(d, "Table", prob); /* fill with integers in 1-6 */
```

2.6.2 Sampling from Finite Sets

It can be useful to sample from a finite set of values. The SAS/IML language provides three functions that you can use to sample from finite sets:

- The RANPERM function generates random permutations of a set with n elements. Use this function to sample without replacement from a finite set with equal probability of selecting any item.
- The RANPERK function (introduced in SAS/IML 12.1) generates random permutations of k items that are chosen from a set with n elements. Use this function to sample k items without replacement from a finite set with equal probability of selecting any item.
- The SAMPLE function (introduced in SAS/IML 12.1) generates a random sample from a finite set. Use this function to sample with replacement or without replacement. This function can sample with equal probability or with unequal probability.

Each of these functions uses the same random number stream that is set by the RANDSEED routine. DATA step versions of the RANPERM and RANPERK functions are also supported.

These functions are similar to the “Table” distribution in that you can specify the probability of sampling each element in a finite set. However, the “Table” distribution only supports sampling with replacement, whereas these functions are suitable for sampling without replacement.

As an example, suppose that you have 10 socks in a drawer as in Section 2.4.5. Five socks are black, two socks are brown, and three socks are white. The following SAS/IML statements simulate three possible draws, without replacement, of five socks. The results are shown in Figure 2.9.

```
proc iml;
call randseed(4321);
socks = {"Black" "Black" "Black" "Black" "Black"
         "Brown" "Brown" "White" "White" "White"};
params = { 5,                               /* sample size           */
          3 };                               /* number of samples     */
s = sample(socks, params, "WOR");           /* sample without replacement */
print s;
```

Figure 2.9 Random Sample without Replacement

s				
White	Black	White	Black	Brown
Brown	Brown	Black	Black	Black
White	Black	White	Black	Brown

The SAMPLE function returns a 3×5 matrix, **s**. Each row of **s** is an independent draw of five socks (because `param[1] = 5`). After each draw, the socks are returned to the drawer and mixed well. The experiment is repeated three times (because `param[2] = 3`). Because each draw is without replacement, no row can have more than two brown socks or more than three white socks.

2.6.3 Simulating Continuous Data

Section 2.6.1 shows how to simulate data from discrete distributions in SAS/IML software. In the same way, you can simulate data from continuous distributions by calling the RANDGEN subroutine. As before, if you allocate a vector or matrix, then a single call of the RANDGEN subroutine fills the entire matrix with random values.

The following SAS/IML program generates samples of size $N = 100$ from the normal, uniform, and exponential distributions:

```
proc iml;
/* define parameters */
mu = 3; sigma = 2;

/* allocate vectors */
N = 100;
StdNor = j(1, N); Normal = j(1, N);
Unif = j(1, N); Expo = j(1, N);
```

```

/* fill vectors with random values */
call randseed(4321);
call randgen(StdNor, "Normal");           /* N(0,1)      */
call randgen(Normal, "Normal", mu, sigma); /* N(mu,sigma) */
call randgen(Unif, "Uniform");           /* U(0,1)      */
call randgen(Expo, "Exponential");       /* Exp(1)      */

```

2.7 Univariate Distributions Supported in SAS Software

In SAS software, the RAND function in Base SAS software and the RANDGEN subroutine in SAS/IML software are the main tools for simulating data from “named” distributions. These two functions call the same underlying numerical routines for computing random variates. However, there are some differences, as shown in Table 2.1:

Table 2.1 Differences Between RAND and RANDGEN Functions

	RAND Function	RANDGEN Subroutine
Called from:	DATA step	PROC IML
Seed set by:	CALL STREAMINT	CALL RANDSEED
Returns:	Scalar value	Vector or matrix of values

Because SAS/IML software can call Base SAS functions, it is possible to call the RAND function from a SAS/IML program. However, this is rarely done because it is more efficient to use the RANDGEN subroutine to generate many random variates with a single call.

Table 2.2 and Table 2.3 list the discrete and continuous distributions that are built into SAS software. Except for the t , F , and “NormalMix” distributions, you can identify a distribution by its first four letters. Parameters for each distribution are listed after the distribution name. Parameters in angled brackets are optional. If an optional parameter is omitted, then the default value is used.

The functions marked with an asterisk are supported by the RANDGEN function in SAS/IML 12.1. In general, parameters named μ and θ are location parameters, whereas σ denotes a scale parameter.

Table 2.2 Parameters for Discrete Distributions

Distribution	<i>distname</i>	<i>parm1</i>	<i>parm2</i>	<i>parm3</i>
Bernoulli	‘BERNOULLI’	p		
Binomial	‘BINOMIAL’	p	n	
Geometric	‘GEOMETRIC’	p		
Hypergeometric	‘HYPERGEOMETRIC’	N	R	n
Negative Binomial	‘NEGBINOMIAL’	p	k	
Poisson	‘POISSON’	m		
Table	‘TABLE’	p		

Table 2.3 Parameters for Continuous Distributions

Distribution	<i>distname</i>	<i>parm1</i>	<i>parm2</i>	<i>parm3</i>
Beta	'BETA'	a	b	
Cauchy	'CAUCHY'			
Chi-Square	'CHISQUARE'	d		
Erlang	'ERLANG'	a	$\langle \sigma = 1 \rangle$	
Exponential	'EXPONENTIAL'	$\langle \sigma = 1 \rangle$		
F	'F'	n	d	
Gamma	'GAMMA'	a	$\langle \sigma = 1 \rangle$	
Laplace*	'LAPLACE'	$\langle \theta = 0 \rangle$	$\langle \sigma = 1 \rangle$	
Logistic*	'LOGISTIC'	$\langle \theta = 0 \rangle$	$\langle \sigma = 1 \rangle$	
Lognormal	'LOGNORMAL'	$\langle \mu = 0 \rangle$	$\langle \sigma = 1 \rangle$	
Normal	'NORMAL'	$\langle \mu = 0 \rangle$	$\langle \sigma = 1 \rangle$	
Normal Mixture*	'NORMALMIX'	p	μ	σ
Pareto*	'PARETO'	a	$\langle k = 1 \rangle$	
t	'T'	d		
Triangle	'TRIANGLE'	h		
Uniform	'UNIFORM'	$\langle a = 0 \rangle$	$\langle b = 1 \rangle$	
Wald*	'WALD' or 'IGAUSS'	λ	$\langle \mu = 1 \rangle$	
Weibull	'WEIBULL'	a	b	

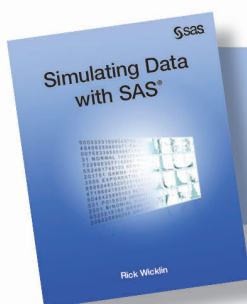
Densities for all supported distributions are included in the documentation for the RAND function in *SAS Functions and CALL Routines: Reference*.

2.8 References

Devroye, L. (1986), *Non-uniform Random Variate Generation*, New York: Springer-Verlag.

URL <http://luc.devroye.org/rnbookindex.html>

Wicklin, R. (2010), *Statistical Programming with SAS/IML Software*, Cary, NC: SAS Institute Inc.



From *Simulating Data with SAS*®. Full book available for purchase [here](#).

Index

A

acceptance-rejection technique 126–128
addition (+) operator 329
Akaike information criterion 249
ALL function 326
alternative parameterizations 216, 218
annotation facility, SGPLOT procedure 301
ANOVA procedure 199, 208–210
ANY function 326
APPEND statement 330–331
approximate sampling distribution
 See ASD (approximate sampling distribution)
AR model 252, 256–258
AR(1) model
 about 185–186, 252
 approximating sampling distributions for
 parameters 254–256
 generating covariance matrix with known structure
 183
 generating multivariate binary variates 155
 simulating data in DATA step 252–254
 simulating data in SAS/IML software 258–260
ARIMA procedure
 about 252
 BY statement 254
 ESTIMATE statement 254
 estimating AR and MA model parameters 258
 IDENTIFY statement 253–254
 simulating AR(1) data 252–253
ARMA models
 about 252
 approximating sampling distributions for AR(1)
 parameters 254–256
 multivariate 260–261
 simple AR model 252
 simulating AR and MA data in DATA step
 256–258
 simulating AR(1) data in DATA step 252–254
 simulating AR(1) data in SAS/IML software
 258–260
ARMACOV function 258
ARMALIK function 258
ARMASIM function 251, 258–259
ARRAY statement 284
arrays, holding explanatory variables 201
ASD (approximate sampling distribution)
 about 52–55
 number of samples and 96
 sampling distribution of Pearson correlations
 70–71
 sampling distribution of statistics for normal data
 60–61
 sampling distribution of the mean 57–59, 68
 simple regression model example 202–203

at (@) symbol 233
autocorrelated data 251
autoregressive and moving average models
 See ARMA models
autoregressive model 252, 256–258

B

BARChart statement, TEMPLATE procedure
 38–39
BarPMF template 39
baseline hazard function 242–243
BC (bias-corrected) confidence intervals 295
Bernoulli distribution
 about 14–15
 logistic regression and 226–227
 parameters for 27
 simulating data from inhomogeneous Poisson
 process 275
beta distribution 28, 301
bias-corrected (BC) confidence intervals 295
binary variates 154–157
binomial distribution
 about 15–16
 negative 27, 39
 parameters for 27
BINOMIAL option, TABLES statement (FREQ) 81,
 86
BISECTION module 118, 156, 334–335
block-diagonal matrices 232–233
BLOCK function 233, 326
%BOOT macro 295
%BOOTCI macro 295
bootstrap confidence intervals 283, 295
bootstrap distribution
 about 283, 285
 computing standard deviation of 294–295
 for skewness and kurtosis 285–289
bootstrap methods
 about 281
 computing bootstrap confidence intervals 283,
 295
 computing bootstrap standard error 294–295
 parametric 281, 291
 plotting estimates of standard errors 305–306
 resampling with DATA step 262–266
 resampling with SAS/IML software 282, 288–291
 resampling with SURVEYSELECT procedure
 282, 286–288
 smooth 281, 292–294
bootstrap standard error 283, 294–295, 305–306
BY-group technique
 about 55
 computing *p*-values 90
 macro usage considerations 101
 resampling example 285
 suppressing output and graphics 97–100
 writing efficient simulations 96–97, 99

- BY statement
 - ARIMA procedure 254
 - BY-group technique and 55
 - MEANS procedure 64
 - simulating data with DATA step and procedures 56
 - TTEST procedure 80, 85
 - writing efficient simulations 97
- C**
- CALIS procedure 180
- case resampling 284
- case sensitivity 6
- Cauchy distribution 28
- CDF function
 - about 116–117, 326
 - checking correctness of simulated data 35
 - parameter considerations 110
 - QUANTILE function and 30, 32
 - working with statistical distributions 30–32
- CEIL function 17, 326
- censored observations 124–125, 244
- central limit theorem (CLT) 57–58
- chi-square distribution 28, 62–63
- chi-square statistic 89
- chi-square test 88–90
- CHISQ option, TABLES statement (FREQ) 90
- Cholesky transformation 146–150
- CHOOSE function 326
- CL option, MIXED procedure 235
- CLASS statement
 - LOGISTIC procedure 218
 - MEANS procedure 64
- classification variables
 - explanatory variables and 199
 - linear regression models with 208–211
- CLB option, MODEL statement (REG) 221
- CLOSE statement 331
- CLT (central limit theorem) 57–58
- coefficient of excess 299
- COLNAME= option
 - FROM clause, APPEND statement 331
 - FROM clause, CREATE statement 331
 - PRINT statement 328
- colon (:) operator
 - See* mean (:) operator
- COLVEC function 326, 331
- comparison (<=) operator 6
- complete spatial randomness 273
- components (subpopulations) 119–121
- compound symmetry model 184
- conditional distribution technique 145
- conditional distributions 142–144
- conditional simulations
 - about 264
 - of one-dimensional data 270–271
 - of two-dimensional data 272–273
- CONDMV function 270
- CONDMVN function 270
- CONDMVNMEANCOV function 144–145
- confidence intervals
 - about 74
 - bias-corrected 295
 - bootstrap 283, 295
 - computing coverage in SAS/IML language 77–78
 - coverage for nonnormal data 76–77
 - coverage for normal data 74–76
 - MEANS procedure computing 315
- CONSTANT function 178
- contaminated normal distribution
 - multivariate 138–140
 - univariate 121–122
- CONTENTS procedure 46
- continuous distributions
 - about 20–21
 - CDF function and 31
 - exponential distribution 22–24, 28
 - moment-ratio diagram for 300–301
 - normal distribution 21–22, 28
 - parameters for 28
 - PDF function and 30
 - simulating in SAS/IML software 26–27
 - skewness and kurtosis for 300–301
 - uniform distribution 22, 28
- continuous mixture distribution 122
- continuous variables
 - explanatory variables and 199
 - linear regression models with 200–203, 210–211
- contour plots 268–269
- control statements 6, 326
- COORDINATES statement, SIM2D procedure 272
- COPULA procedure
 - about 12, 169–172
 - SIMULATE statement 170
- copula technique
 - about 153, 164, 168–169
 - fitting and simulating data 169–173
 - usage example 164–168
- CORR function 290, 326
- CORR procedure
 - computing sample moments 319
 - COV option 180
 - estimating covariance matrix from data 180
 - FISHER option 168, 173
 - fitting and simulating data from copula model 171, 173
 - generating data from copulas 168
 - NOMISS option 180, 190
 - OUTP= option 97, 180
 - POLYCHORIC option 190
 - simple linear regression models 203
 - simulating data from multinomial distributions 135–140
- correlated random errors 232–236

correlation matrices
 about 175–176
 converting between covariance and 176–177
 finance example 189–190
 finding nearest 191–193
 generating random 187–189
 problems faced with 189–191

COUNTN function 326

COV function 180, 326

COV option, CORR procedure 180

covariance matrices
 about 175–176
 building 179–186
 converting between correlation and 176–177
 estimating from data 180–181
 generating diagonally dominant 181–182
 generating from Wishart distribution 186–187
 generating with known structure 183–186
 testing for 177–179
 with AR(1) structure 185–186
 with compound symmetry 184
 with diagonal structure 183–184
 with Toeplitz structure 185

Cox models 242–243

CREATE statement
 about 330–331
 FROM clause 331
 row-major order for matrices 260
 VAR clause 331

cumulative distributions
See CDF function

CUPROD function 326

CUSUM function 326

CUTVAL. format 60

D

data sets
 creating from ODS tables 45–46, 57
 creating matrices from 330
 macro usage considerations 101
 reading data from 329–330
 writing data to 330–331

data simulation
See simulating data

DATA step
See also specific techniques
 observations and 6
 resampling with 282–286
 SAS/IML language comparison 6
 simulating AR data in 256–258
 simulating AR(1) data in 252–254
 simulating data using 55–67
 simulating MA data in 256–258

densities, computing
See also PDF function
 finite mixture distribution and 120

overlying theoretical density on histograms
 40–41

overlying theoretical PMF on frequency plots 38

design matrices
 about 215–216
 creating for fixed and random effects 238–239
 for alternative parameterizations 218
 reading 239–240
 with GLM parameterization 216–218

design of simulation studies
 about 93–95
 disadvantages of simulation 105
 effect of number of samples 95–96
 moment-ratio diagram as tool for 315–317
 writing efficient simulations 96–105

DIAG function 326

DIAGONAL= option, MATRIX statement
 (SGSCATTER) 291

diagonally dominant covariance matrices 181–184

discrete distributions
 about 14
 Bernoulli distribution 14–15, 27
 binomial distribution 15–16, 27
 CDF function and 31
 discrete uniform distribution 17–18
 geometric distribution 16–17, 27
 parameters for 27
 PDF function and 30
 Poisson distribution 19–20, 27
 simulating in SAS/IML software 24–25
 tabulated distributions 18–19

discrete uniform distribution 17–18

dispersion constant 226

DISTANCE function
 about 326, 333–334
 simulating data from Gaussian random field 265
 simulating data from regular process 277

DO function 326

The DO Loop blog 5, 325

DO loops
 as simulation loops 55
 effect of sample size on sampling distribution 64
 matrix arithmetic versus 155
 sampling distribution of Pearson correlations 70
 simulating fixed effect by reversing order of
 202–203
 tips for shortening simulation times 104
 using multivariate data 55, 97
 using univariate data 13–14, 21–22, 24
 writing efficient simulations 97

DYNAMIC statement, SGRENDER procedure 39, 41

E

ECDF (empirical CDF) 119, 281

effect parameterization 218

effect size 87–88

EIGEN routine 193, 326

eigenvalue decomposition 150–151
 eigenvalues (spectrum) 187–189
 EIGVAL function 179, 182, 326
 elliptical distributions 169
 empirical CDF (ECDF) 119, 281
 Emrich-Piedmonte algorithm 154, 158
 equality (=) operator 6
 Erlang distribution 28
 ESTIMATE statement, ARIMA procedure
 NOPRINT option 254
 OUTEST= option 254
 WHERE clause 254
 EUCLIDEANDISTANCE module 265, 277, 333–334
 evaluating power of *t* test 84–86
 evaluating statistical techniques
 about 73–74
 assessing two-sample *t* test for equality of means
 78–84
 confidence interval for a mean 74–78
 effect of sample size on power of *t* test 87–88
 evaluating power of *t* test 84–86
 using simulation to compute *p*-values 88–90
 excess kurtosis
 about 299
 computing 336–337
 for continuous distributions 300–301
 EXP function 326
 EXPAND2GRID function 315
 explanatory variables
 about 198–199
 arrays holding 201
 classification variables and 199
 continuous variables and 199
 exponential distribution
 about 22–24
 confidence interval for a mean 76–77
 goodness-of-fit tests 117
 inverse transformation algorithm and 117–118
 parameters for 28, 110
 plotting PDF of 31
 proportional hazards model and 242
 shape parameters and 301

F

F distribution 28, 117–119
F test 212, 215
 factor pattern matrix 133, 150–151
 FACTOR procedure 133, 150
 feasible region 300
 final weighted least squares (FWLS) estimate 220–221
 FINISH statement 326
 finite mixture distributions
 about 119–120
 contaminated normal distribution 121–122
 simulating from 120–121
 FISHER option, CORR procedure 168, 173

Fisher's *z* transformation 290
 FITFLEISHMAN module 312
 fixed effects
 creating design matrices for 238–239
 generating variables for 201
 simulating by reversing order of DO loops
 202–203
 simulating random effects components 236–242
 simulating with arrays 201
 Fleishman's method 115, 298, 311–314
 FMM procedure 120, 297
 FORM= option, SIMULATE statement (SIM2D)
 267–268
 FORMAT= option, PRINT statement 328
 FORMAT procedure 60
 FREE statement 327
 FREQ procedure
 BY-group processing 90
 chi-square tests and 89
 confidence interval for a mean 75
 design of simulation studies and 315
 OUTPUT statement 97, 190
 simulating multivariate ordinal variates 161
 simulating univariate data 15, 17–18
 t tests and 80–81, 85–86
 TABLES statement 56, 81, 86, 90
 usage examples involving tables 44–45
 frequency plots 37–39
 frequency variables 287–288
 FROM clause
 APPEND statement 331
 CREATE statement 331
 FROOT function
 about 118–119, 327, 334–335
 finding intermediate correlations 156
 functions
 See also specific functions
 parameters and 110
 SAS/IML language supported 326–328
 SAS/IML modules replicating 333–336
 FWLS (final weighted least squares) estimate 220–221
 FWLS option, ROBUSTREG procedure 220–221

G

GAM procedure 247
 GAMINV function 32
 gamma distribution
 checking correctness of simulated data 35–37
 chi-square distribution and 62–63
 fitting to data 305–308
 parameters for 28, 110
 shape parameters and 301
 GAUSS functions 188
 Gaussian random field
 about 263–264
 conditional simulation of one-dimensional data
 270–271

- conditional simulation of two-dimensional data 272–273
- unconditional simulation of one-dimensional data 264–267
- unconditional simulation of two-dimensional data 267–269
- generalized Pareto distribution 113
- GENMOD procedure 229
- geometric distribution
 - about 16–17
 - drawing random sample from 37–39
 - parameters for 27
- Givens rotations 187
- GLIMMIX procedure
 - estimating covariance matrix 181
 - OUTDESIGN= option 238–239
 - reading design matrices 239
 - simulating random effects components 238
- GLM parameterization 216–218
- GLM procedure
 - design matrices with GLM parameterization 217
 - OUTSTAT= option 97
 - simple linear regression models and 199, 211
- GLMMOD procedure
 - creating design matrices for fixed and random effects 238–239
 - GLM parameterization and 216–217
 - simulating random effects components 238
- goodness-of-fit tests 35–37, 117
- Graph Template Language (GTL)
 - defining contour plots 268
 - overlying theoretical density on histograms 40–41
 - overlying theoretical PMF on frequency plots 37–39
- Grid Manager, SAS 102
- grid of values, creating 332–333
- GRID statement, SIM2D procedure 267
- GTL (Graph Template Language)
 - defining contour plots 268
 - overlying theoretical density on histograms 40–41
 - overlying theoretical PMF on frequency plots 37–39
- Gumbel distribution 111–112, 301, 305

H

- hard-core processes 276–278
- hazard function 242
- hazard rate 123
- high-leverage points 219, 221–224
- Higham's method 190–193
- HISTOGRAM statement, UNIVARIATE procedure
 - overlying theoretical density on histograms 40
 - plotting bootstrap estimates of standard errors 306
 - sampling distribution for AR(1) parameters 255
 - sampling distribution of the variance 62

- HistPDF template 41
- homogeneous Poisson process
 - about 273
 - regular process and 276–278
 - simulating data from 273–275
- HOMOGPOISSONPROCESS function 277
- hypergeometric distribution 27
- hypothesis testing, computing *p*-values for 32, 88–90

I

- I function 327
- ID vectors, creating 331–332
- IDENTIFY statement, ARIMA procedure
 - NOPRINT option 254
 - VAR= option 253
- IF-THEN/ELSE control statement 326
- Iman-Conover method 161–164, 176
- IML (interactive matrix language) 5
 - See also* SAS/IML language
- IML procedure
 - Cholesky transformation and 148
 - DATA step function support 227
 - design matrices and 215–216
 - estimating covariance matrix from data 180
 - license considerations 5–6
 - LOAD statement 159, 188, 327
 - matrix multiplication and 217
 - multivariate normal distributions and 133
 - sampling distribution of Pearson correlations 70
 - simulating ARMA samples 259
 - simulating Gaussian random fields 267
 - simulating univariate data 24
 - t* tests and 84
- in-memory technique 55, 97
- index of maximum (<:>) operator 329
- index of minimum (>:<) operator 329
- inequality (^=) operator 6
- inhomogeneous Poisson process 273, 275–276
- INSET statement
 - SGPLOT procedure 90
 - UNIVARIATE procedure 306
- instrumental distribution 126–128
- interactive matrix language (IML) 5
 - See also* SAS/IML language
- intermediate correlation 155–156, 167
- INTO clause, READ statement 330
- INV function 327
- inverse CDF function
 - See* QUANTILE function
- inverse Gaussian distribution 28, 112
- inverse transformation algorithm 117–119
- iterative DO statement 326

J

- J function
 - about 327
 - sampling distribution of the mean 68

J function (*continued*)

- simulating univariate data 24
- writing efficient simulations 97

jackknife methods 287

jitter technique 132

Johnson system of distributions 114–116, 302, 308–311

K

KDE (kernel density estimate) 120–121

KDE procedure 132, 293

KEEP statement 284

kernel density estimate (KDE) 120–121

Kronecker product matrix operator 233

kurtosis

- bootstrap resampling 285–289
- checking correctness of simulated data 36
- computing 336–337
- design of simulation studies and 315–316
- estimate bias in small samples 65–67
- Fleishman distribution and 115, 311
- for gamma distribution 306–308
- Johnson system of distributions 310–311
- moment matching and 303
- moments and 299–301
- plotting variations on moment-ratio diagram 303–306

KURTOSIS module 288

KURTOSIS= option, OUTPUT statement (MEANS) 66

L

LABEL= option, PRINT statement 328

Laplace distribution 28

LCLM= option, OUTPUT statement (MEANS) 74–75

least trimmed squares (LTS) estimate 220–223

LEVERAGE option, MODEL statement (ROBUSTREG) 222

LEVERAGE= option, OUTPUT statement (ROBUSTREG) 222

LIFETEST procedure 123–124, 245–246

linear mixed models

- about 230–231
- repeated measures model with random effect 231–232
- simulating correlated random errors 232–236
- with random effects 226, 230–232

linear predictor

- about 226
- in generalized linear models 226
- in proportional hazards model 243

linear regression models

- about 199
- based on real data 204–208
- generalized 226–230
- with classification and continuous variables 210–211

- with interaction and polynomial effects 215–218
- with single classification variable 208–210
- with single continuous variable 200–203

LINEPARM statement, SGPLOT procedure 219–220

link functions 226

listwise deletion 190

LOAD statement, IML procedure 159, 188, 327

LOC function 327, 329

location parameter 109–111

LOESS procedure

- about 247
- MODEL statement 249

logistic distribution 28

LOGISTIC procedure

- alternative parameterizations 216, 218
- CLASS statement 218
- logistic regression example 228
- OUTDESIGN= option 218
- OUTDESIGNONLY option 218

logistic regression model 226–229

lognormal distribution

- parameters for 28, 111
- plotting bootstrap estimates of standard errors 305
- shape parameters and 301

LTS (least trimmed squares) estimate 220–223

M

MA model 256–258

machine epsilon 178

machine precision 178

macro-loop technique 100–101

macros

- macro-loop technique 100–101
- packaging commands into 98–99
- usage considerations 101–102

Matérn model II 277

MATLAB functions 188

matrices

- See also* correlation matrices
- See also* covariance matrices
- block-diagonal 232–233
- checking if PSD 178–179
- checking if symmetric 178
- constructing 240
- creating data sets from 331
- creating from data sets 330
- design 215–218
- efficiency of 6
- eigenvalues for 187–189
- Iman-Conover method 161–164
- reshaping 69
- row-major order for 260
- SAS/IML language and 6
- subscript reduction operators for 328–329
- tips for shortening simulation times 103

matrix arithmetic 155, 217

MATRIX statement, SGSCATTER procedure 291

- MAX function 327, 329
- maximum likelihood estimate
 - checking correctness of simulated data 36
 - fitting gamma distribution to data 306
 - suppressing notes to SAS log 99–100
- maximum (\heartsuit) operator 329
- MCD subroutine 140
- MCMC procedure
 - about 9
 - Gibbs sampling and 145
 - parameter considerations 110
- mean
 - assessing two-sample t test 78–84
 - computing variances of 61–62
 - confidence interval for 74–78, 295
 - sampling distribution of 57–59, 68–69
- MEAN function
 - about 68, 327
 - computing confidence interval for a mean 77
 - subscript reduction operator equivalent 329
 - writing efficient simulations 97
- mean mapping method 158–159
- mean (:) operator 68, 77, 329
- mean square error 54
- MEAN statement, SIM2D procedure 267–268
- MEANS procedure
 - approximating sampling distribution 55
 - bootstrap resampling 287–288
 - BY statement 64
 - CLASS statement 64
 - computing point estimates 282
 - computing sample kurtosis 66
 - computing sample moments 319
 - computing variances 61
 - design of simulation studies and 315
 - displaying descriptive statistics 255
 - OUTPUT statement 56, 66, 74–75
 - P5 option 58, 285
 - P95 option 58, 285
 - sampling distribution of the mean 57–59
 - unconditional simulation of one-dimensional data 266–267
 - VARDEF= option 295
- median, computing variances of 61–62
- MEDIAN function 68, 327
- Mersenne-Twister algorithm 32–33
- METHOD= option
 - ROBUSTREG procedure 220–221
 - SURVEYSELECT procedure 287
- MIN function 327, 329
- minimum (\heartsuit) operator 329
- mixed models
 - See* linear mixed models
- MIXED procedure
 - CL option 235
 - covariance structures supported 183
 - estimating covariance matrices 181
 - repeated measures model with random effect 231–232
- mixing probabilities 120
- mixture distributions
 - about 119–120
 - contaminated normal distribution 121–122
 - simulating from 120–121
- MODEL procedure
 - about 9, 252
 - parametric bootstrap method 291
 - simulating data from copula model 169
- MODEL statement
 - LOESS procedure 249
 - REG procedure 221
 - ROBUSTREG procedure 222
- moment matching
 - about 298, 303
 - as modeling tool 302–303
 - as tool for designing simulation studies 315–317
- moment-ratio diagram
 - about 298–302
 - as tool for designing simulation studies 315–317
 - comparing simulations and choosing models 314
 - extensions to multivariate data 318–331
 - fitting gamma distribution to data 306–308
 - Fleishman's method 311–314
 - for continuous distributions 300–301
 - Johnson system of distributions 308–311
 - plotting variation of skewness and kurtosis on 303–306
- MOMENTS module 312
- moments of a distribution 299–301
- Monte Carlo estimates
 - about 54
 - bias of kurtosis estimates in small samples 67
 - effect of sample size on sampling distribution 63–64
 - MCMC procedure and 9
 - number of samples and 96
 - sampling distribution of the mean 58
 - simple regression model example 202–203
- Monte Carlo standard error 96
- multinomial distribution
 - about 130–132
 - generating random samples from 89
 - simulating data from 130–132
 - tabulated distributions and 19, 130
- multiplication (#) operator 215, 329
- multivariate ARMA models 260–261
- multivariate contaminated normal distribution 138–140
- multivariate distributions
 - See also* multinomial distribution
 - See also* MVN (multivariate normal distributions)
 - advanced techniques for simulating data 153–174
 - basic technique for simulating data 129–152
 - Cholesky transformation and 146–150

multivariate distributions (*continued*)
 constructing with Fleishman distribution 115
 extensions to 318–331
 generating data from 137
 generating data from copulas 164–173
 generating multivariate binary variates 154–157
 generating multivariate ordinal variates 158–161
 methods for generating data from 144–146
 mixtures of 138–141
 reordering multivariate data 161–164
 resampling with SAS/IML software 289–291
 simulating data from 129, 153–154
 simulating data in time series 251
 simulating data with given moments 298
 spectral decomposition and 150–151
 using DO loop 55, 97

multivariate normal distributions (MVN)
 about 133
 conditional 142–144
 estimating covariance matrix from data 180
 mixtures of 140–141
 simulating in SAS/IML software 133–136
 simulating in SAS/STAT software 136

%MULTNORM macro 135–136, 140

MVN (multivariate normal distributions)
 about 133
 conditional 142–144
 estimating covariance matrix from data 180
 mixtures of 140–141
 simulating in SAS/IML software 133–136
 simulating in SAS/STAT software 136

MYSQRVECH function 335

N

naive bootstrap

See bootstrap methods

NARROW option, SIM2D procedure 267

NCOL function 327

nearest correlation matrix 191–193

negative binomial distribution 27, 39

NLIN procedure 291

NOMISS option, CORR procedure 180, 190

nonnormal distributions 81–82

NONOTES system option 101, 116

nonparametric models 247–249

nonsingular parameterizations 218

NOPRINT option

ESTIMATE statement, ARIMA procedure 254

IDENTIFY statement, ARIMA procedure 254

procedures and 97, 254

normal distribution

computing *p*-values 32

computing quantiles for 156

confidence interval for a mean 74–77

contaminated 121–122, 138–140

parameters for 28

shape parameters and 301

simulating data from 12–13, 21–22

normal mixture distribution 28

NORTA method 168–169

notes, suppressing to SAS log 99–100

NOTES system option 101

NROW function 327

number of samples (repetitions) 95–96

NUMREAL= option, SIMULATE statement (SIM2D)
267

O

observations

correlating 181

DATA step and 6

high-leverage points 219

ODS EXCLUDE ALL statement 97

ODS EXCLUDE statement 45–46

ODS GRAPHICS statement 46

ODS OUTPUT statement

creating data sets from tables 45–46, 57

suppressing output 97–98

usage example 80

ODS SELECT statement 45–46

ODS statements, controlling output with 44–46, 97–99

ODS TRACE statement 44

%ODSOFF macro 80, 98, 228, 236

%ODSON macro 99

OF operator 19

OLS (ordinary least squares) 200

one-dimensional data

conditional simulation of 270–271

unconditional simulation of 264–267

ordinal variates 158–161

ORDMEAN function 159–160

ORDVAR function 159–160

OUT= option

OUTPUT statement, FREQ procedure 97

TABLES statement, FREQ procedure 56

OUTDESIGN= option

GLIMMIX procedure 238–239

LOGISTIC procedure 218

OUTDESIGNONLY option, LOGISTIC procedure
218

OUTEST= option

ESTIMATE statement, ARIMA procedure 254

REG procedure 56, 97

OUTHITS option, SURVEYSELECT procedure
287–288

outliers 219–221

OUTP= option, CORR procedure 97, 180

output, controlling with ODS statements 44–46, 97–99

OUTPUT statement

FREQ procedure 97, 190

MEANS procedure 56, 66, 74–75

REG procedure 205

ROBUSTREG procedure 222

OUTSTAT= option, GLM procedure 97

P

- P= option, OUTPUT statement, REG procedure 205
- P5 option, MEANS procedure 58, 285
- P95 option, MEANS procedure 58, 285
- p*-values, computing for hypothesis testing 32, 88–90
- pairwise correlations 190
- PAIRWISEDIST module 333–334
- PARAM= option, CLASS statement (LOGISTIC) 218
- parameter estimates
 - reading 239–240
 - using as parameters 207–208
- parameters
 - for Bernoulli distribution 27
 - for binomial distribution 27
 - for continuous distributions 28
 - for discrete distributions 27
 - for Emrich-Piedmonte algorithm 155
 - for exponential distribution 28, 110
 - for gamma distribution 28, 110
 - for geometric distribution 27
 - for logistic distribution 28
 - for lognormal distribution 28, 111
 - for normal distribution 28
 - for Poisson distribution 27
 - for standard normal distribution 28
 - for tabulated distributions 27
 - for uniform distribution 28, 111
 - for univariate distributions 109–111
 - for Weibull distribution 28
 - location 109–111
 - rate 22
 - scale 109–111
 - shape 110, 299, 301
 - using parameter estimates as 207–208
- parametric bootstrap method 281, 291
- Pareto distribution 28, 112–113
- PD (positive definite)
 - about 177
 - generating covariance or correlation matrix 179–180
 - generating diagonally dominant covariance matrix 181–182
 - problems with covariance matrices 190
 - testing covariance matrices 177
- PDF function
 - about 326
 - checking correctness of simulated data 35
 - finite mixture distribution and 120
 - overlying theoretical density on histograms 40–41
 - overlying theoretical PMF on frequency plots 38
 - parameter considerations 110
 - simulating data from continuous distributions 21, 23
 - working with statistical distributions 30–31
- Pearson correlations
 - bootstrap resampling 290
 - copula technique and 169–170
 - correlation matrices and 176
 - sampling distribution of 69–71
 - simple regression model example 202–203
- Pearson system of distributions 302
- PHREG procedure 244
- PLCORR option, OUTPUT statement (FREQ) 190
- PLOTS= option, SIM2D procedure 268
- PMF function
 - checking correctness of simulated data 35
 - generating multivariate ordinal variates 158–161
 - overlying on frequency plot 37–39
 - working with statistical distributions 30–31
- POINT= option, SET statement 205, 282–284
- Poisson distribution 19–20, 27, 229
- Poisson process
 - about 273
 - homogeneous 273–275
 - inhomogeneous 273, 275–276
- Poisson regression model 226, 229–230
- %POLYCHOR macro 190
- POLYCHORIC option, CORR procedure 190
- polynomial effects, linear models 215–218
- POLYROOT function 327
- pooled variance *t* test
 - about 78
 - assessing in SAS/IML software 83–84
 - effect of sample size on power of 87–88
 - evaluating power of 84–86
 - robustness to nonnormal populations 81–82
 - robustness to unequal variances 78–81
- positive definite (PD)
 - about 177
 - generating diagonally dominant covariance matrix 181–182
 - problems with covariance matrices 190
 - testing covariance matrices 177
- positive semidefinite (PSD) 177–179
- power function distribution 113
- power of regression tests 211–215
- power of *t* test
 - effect of sample size on 87–88
 - evaluating 84–86
 - exact power analysis 84–85
 - simulated analysis 85–86
- POWER procedure
 - effect size and 87–88
 - evaluating power of *t* test 84–86
- PRINT statement
 - about 328
 - COLNAME= option 328
 - FORMAT= option 328
 - LABEL= option 328
 - ROWNAME= option 328
 - sampling distribution example 69
- PRINTTO procedure 99–100

- probability distributions
 - See* continuous distributions
 - See* discrete distributions
 - probability mass function
 - See* PMF function
 - PROBBNRM function 155
 - PROBGAM function 32
 - PROBIT function 32
 - PROBNORM function 32
 - procedures
 - BY statement in 55
 - data simulation using 55–67
 - NOPRINT option in 97, 254
 - suppressing notes to SAS log 99–100
 - PROD function 327, 329
 - profiling simulations 102–103
 - PROJS function 191
 - PROJU function 191
 - proportional hazards model 242–245
 - PSD (positive semidefinite) 177–179
 - pseudorandom numbers 33
 - %PUT statement 35
- Q**
- Q-Q (quantile-quantile) plot 41–44
 - QNTL subroutine 68, 289, 327
 - QQPLOT statement, UNIVARIATE procedure 41–44
 - QUANTILE (inverse CDF) function
 - about 326
 - acceptance-rejection technique and 126–127
 - computing confidence interval for a mean 77
 - computing quantile of normal distribution 156
 - creating Q-Q plots 42–43
 - fitting and simulating data from copula model 170–171
 - generating data from copulas 165
 - parameter considerations 110
 - sampling method 116–122
 - univariate distribution support 112–113
 - working with statistical distributions 30, 32
 - quantile-quantile (Q-Q) plot 41–44
 - quantiles
 - See also* QUANTILE function
 - about 32
 - checking correctness of simulated data 35
 - computing for normal distributions 156
- R**
- RAND function
 - about 11, 33–34
 - finite mixture distribution and 120
 - linear regression model and 227
 - logistic regression model and 227
 - overlaying theoretical PMF on frequency plots 38
 - parameter considerations 110–111
 - Poisson regression model and 229
 - simulating data from inhomogeneous Poisson process 275
 - simulating univariate data 13–14, 18, 23–24, 27
 - univariate distribution support 111–112, 114
 - working with statistical distributions 30
 - RANDDIRICHLET function 137
 - %RandExp macro 23, 123
 - RANDFLEISHMAN module 312
 - RANDGEN subroutine
 - about 33, 227, 327
 - computing confidence interval for a mean 77
 - distributions supported by 112
 - J function and 97
 - overlaying theoretical PMF on frequency plots 38
 - sampling distribution of the mean 68
 - simulating data from homogeneous Poisson process 274
 - simulating univariate data 12, 18, 24–27
 - two-sample pooled variance *t* test 83
 - working with statistical distributions 30
 - RANDMULTINOMIAL function 89, 130, 327
 - RANDMVBINARY function 157
 - RANDMVORDINAL function 159–160
 - RANDMVT function 137, 327
 - RANDNORMAL function
 - about 133, 327
 - Cholesky transformation 146
 - conditional simulations 143
 - simulating data from multinomial distributions 70, 133, 138, 145
 - unconditional simulation of one-dimensional data 265
 - random correlation matrices 187–189
 - random effects
 - about 226
 - creating design matrices for 238–239
 - generating variables for 201
 - linear mixed models with 226, 230–232
 - repeated measures model with 231–232
 - simulating components 236–242
 - random error term 198–199
 - random number generation
 - about 33–35
 - ARMASIM function and 259
 - Mersenne-Twister algorithm 32–33
 - RANDSEED subroutine and 259
 - setting seed value for 33–35
 - random values for distributions
 - See* RAND function
 - random variates 12
 - RANDSEED subroutine
 - about 33, 327
 - random number generation and 259
 - sampling distribution of the mean 68
 - simulating univariate data 24, 26
 - RANDVALEMAURELLI function 318–319
 - RANDWISHART function 137, 186, 327

- RANGAM function 32
 - RANGE= option, SIMULATE statement (SIM2D) 267
 - rank (Spearman) correlations 169, 176
 - RANK function 327
 - RANNOR function 32
 - RANPERK function 25–26
 - RANPERM function 25–26
 - rate parameter 22
 - Rayleigh distribution 114
 - READ statement
 - about 329
 - INTO clause 330
 - WHERE clause 55
 - reading data from data sets 329–330
 - reference parameterization 218
 - REFLINE statement, SGPLOT procedure 90
 - REG procedure
 - MODEL statement 221
 - OUTEST= option 56, 97
 - OUTPUT statement 205
 - simple linear regression models and 199–200, 204–205
 - TEST statement 211–213
 - regression models
 - about 197
 - components of 198–199
 - linear 199–211, 215–218, 226–230
 - linear mixed models 226, 230–242
 - logistic regression model 226–229
 - nonparametric models 247–249
 - outliers and 219–224
 - Poisson regression model 226, 229–230
 - power of regression tests 211–215
 - survival analysis models 123–125, 242–247
 - regular processes 276–278
 - rejection method 126–128
 - REPEAT function 69, 327, 331
 - repeated measures model with random effect 231–232
 - repetitions (number of samples) 95–96
 - REPS= option, SURVEYSELECT procedure 287
 - resampling
 - case 284
 - with DATA step 282–286
 - with SURVEYSELECT procedure 282, 286–288
 - reshaping matrices 69
 - response variables
 - about 197–198
 - in logistic regression 226
 - outliers for 219–221
 - simulating 240–242
 - RETURN statement 327
 - ridge factor 190–191
 - RMSE (root mean square error)
 - about 199
 - linear model based on real data 204
 - linear model with continuous variable 201
 - nonparametric models 248
 - RANDMVT function 327
 - ROBUSTREG procedure
 - FWLS option 220–221
 - METHOD= option 220–221
 - MODEL statement 222
 - OUTPUT statement 222
 - ROBUSTREG routine 140
 - ROOT function
 - about 327
 - checking if matrix is PD 182
 - checking if matrix is PSD 179
 - Cholesky transformation and 147
 - root mean square error (RMSE)
 - about 199
 - linear model based on real data 204
 - linear model with continuous variable 201
 - nonparametric models 248
 - row-major order for matrices 260
 - ROWNAME= option, PRINT statement 328
 - ROWVEC function 327
 - RSREG procedure 316
- ## S
- SAMPLE function
 - about 327, 336
 - simulating univariate data 18, 25–26
 - sample moments
 - checking correctness of simulated data 35–37
 - computing 336–337
 - sample size
 - bias of kurtosis estimates and 65–67
 - effect of on power of *t* test 87–88
 - effect of on sampling distribution 63–65
 - number of samples and 95–96
 - standard error and 96
 - SAMPLEREPLACE module 288, 336
 - sampling distribution
 - approximating 52–55
 - approximating for AR(1) parameters 254–256
 - bias of kurtosis estimates 65–67
 - effect of sample size on 63–65
 - estimating probability with 59–60
 - evaluating statistical techniques for 73–91
 - Monte Carlo estimates 54
 - of a statistic 51–53
 - of Pearson correlations 69–71
 - of statistics for normal data 60–63
 - of the mean 57–59, 68–69
 - simulating data using SAS/IML language 67–71
 - simulating data with DATA step and procedures 55–67
 - sampling variation 16
 - SAMPRATE= option, SURVEYSELECT procedure 287
 - SAS Grid Manager 102

- SAS/IML language
 - about 5–6, 12, 325–326
 - additional resources 325–326
 - assessing *t* test in 83–84
 - computing confidence interval for a mean 77–78
 - constructing block-diagonal matrix 232–233
 - creating grid of values 332–333
 - creating ID vectors 331–332
 - DATA step comparison 6
 - design of simulation studies and 315
 - Fleishman's method and 312–313
 - functions supported 326–328
 - generating symmetric matrices 181
 - Iman-Conover method 162
 - license considerations 5–6
 - matrices and 6
 - modules for sample moments 336–337
 - modules replicating functions 333–336
 - obtaining programs used in book 8
 - PRINT statement 328
 - reading data from data sets 329–330
 - reading design matrices into 239
 - resampling support 282, 288–291
 - simulating AR(1) data 258–260
 - simulating data from regression models 206–207
 - simulating data using 67–71
 - simulating multivariate normal data 133–136, 140
 - simulating responses 240–241
 - subscript reduction operators 328–329
 - writing data to data sets 330–331
- SAS log, suppressing notes to 99–100
- SAS Simulation Studio 9
- SAS/STAT software 133, 136, 140
- SASFILE statement 283
- SCALE= option, SIMULATE statement (SIM2D) 267
- scale parameter 109–111
- scatter matrix 186
- SCATTER statement, SGPLOT procedure
 - YERRORLOWER= option 86
 - YERRORUPPER= option 86
- SDF (survival distribution function) 245
- SEED= option, SURVEYSELECT procedure 287–288
- seed value
 - for random number generation 33–35
 - for sampling distribution examples 55
- semicolon (;) 6
- SET statement 205, 282–284
- SETDIF function 327
- SGPLOT procedure
 - annotation facility 301
 - bias of kurtosis estimates in small samples 66
 - conditional distributions 144
 - conditional simulations 271
 - creating Q-Q plots 42–43
 - generating data from copulas 168
 - INSET statement 90
 - jitter technique and 132
 - LINEPARM statement 219–220
 - nonparametric models example 248–249
 - plotting PDF 31
 - profiling simulations 102–103
 - REFLINE statement 90
 - SCATTER statement 86
 - visualizing stationary time series 257–258
- SGRENDER procedure
 - conditional simulations 272
 - DYNAMIC statement 39, 41
 - overlying theoretical density on histograms 40–41
 - overlying theoretical PMF on frequency plots 38
- SGSCATTER procedure 135, 141, 291
- SHAPE function
 - about 327
 - generating ID variables 331
 - generating matrices from Wishart distribution 187
 - reshaping matrices 69
- shape parameters 110, 299, 301
- shrinkage methods 190–191
- SIM2D procedure
 - about 12
 - COORDINATES statement 272
 - GRID statement 267
 - MEAN statement 267–268
 - NARROW option 267
 - PLOTS= option 268
 - producing contour plots 269
 - SIMULATE statement 267–268, 272
 - simulating data from Gaussian field 263–264, 267, 272
- SIMNORMAL procedure
 - about 12, 133
 - conditional simulations 142
 - simulating MVN distributions 136
- simple bootstrap
 - See* bootstrap methods
- SIMULATE statement
 - COPULA procedure 170
 - SIM2D procedure 267–268, 272
- simulating data
 - See also under specific techniques*
 - about 3–4
 - advanced techniques for multivariate data 153–174
 - advanced techniques for univariate data 109–128
 - building correlation and covariance matrices 175–194
 - checking correctness of 35–44
 - disadvantages of 105
 - for advanced regression models 225–249
 - for basic regression models 197–224
 - from basic multivariate distributions 129–152
 - from common univariate distributions 11–28

- from spatial models 263–279
- from time series models 251–261
- moment matching and moment-ratio diagram 297–322
- preliminary and background information 29–47
- resampling and bootstrap methods 281–295
- shortening simulation times 103–105
- specialized tools for 8–9
- strategies for 93–106
- to estimate sampling distributions 51–71
- to evaluate statistical techniques 73–91
- using DATA step and procedures 55–67
- using SAS/IML language 67–71
- simulation loop 55
- Simulation Studio 9
- singular parameterization 216
- skewness
 - bootstrap resampling 285–289
 - checking correctness of simulated data 36
 - computing 336–337
 - design of simulation studies and 315–316
 - Fleishman distribution and 115, 311
 - for gamma distribution 306–308
 - Johnson system of distributions 310–311
 - moment matching and 303
 - moments and 299–301
 - plotting variations on moment-ratio diagram 303–306
 - sampling distribution example 65–67
- SKEWNESS module 288
- Sklar's theorem 169
- smooth bootstrap method 281, 292–294
- SMOOTH= option, MODEL statement (LOESS) 249
- SMOOTHBOOTSTRAP module 294
- SOLVE function 327
- SORT call 327
- SORT procedure 56
- spatial functions 273–274
- spatial models
 - about 263
 - simulating data from a regular process 276–278
 - simulating data from Gaussian random field 263–273
 - simulating data from homogeneous Poisson process 274–275
 - simulating data from inhomogeneous Poisson process 275–276
 - simulating data from spatial point process 273–274
 - simulating data using other techniques 278–279
- spatial point processes 273–274
- Spearman (rank) correlations 169, 176
- spectral decomposition 150–151
- spectrum (eigenvalues) 187–189
- SQRT function 326
- SQRVECH function
 - about 327, 335
 - generating symmetric matrices 181
 - multivariate normal distributions and 140–141
- SSQ function 327, 329
- standard errors
 - about 53
 - bootstrap 283, 294–295, 305–306
 - Monte Carlo 96
 - plotting bootstrap estimates of 305–306
 - sample size and 96
- standard normal distribution
 - computing p -values 32
 - computing quantiles for 156
 - contaminated 121–122, 138–140
 - parameters for 28
 - simulating data from 12–13, 21–22
- standardized uniform distribution 22
- START statement 327
- STAT= option, BARCHART statement (TEMPLATE) 38–39
- STATESPACE procedure 261
- statistic
 - sampling distribution of 51–53
 - standard error of 53
- statistical distributions
 - checking correctness of simulated data 35–44
 - essential functions for working with 30–33
 - random number streams 33–35
- STD function 68, 77, 328
- STOP statement 205, 328
- STORE statement 328
- STREAMINIT function
 - about 33–34
 - linear regression example 227
 - macro-loop technique and 101
 - simulating univariate data 13–14
- Student's t distribution 137
- subpopulations (components) 119–121
- subscript reduction operators
 - about 328–329
 - assessing t test 84
 - sampling distribution of the mean 68
 - writing efficient simulations 97
- SUM function 328–329
- sum of squares (##) operator 329
- SURVEYSELECT procedure
 - about 282
 - METHOD= option 287
 - OUTHITS option 287–288
 - REPS= option 287
 - resampling with 282, 286–288
 - SAMPRATE= option 287
 - SEED= option 287–288
- survival analysis models
 - about 125
 - proportional hazards model 242–245
 - simulating data from multiple survivor functions 245–247

survival analysis models (*continued*)
 simulating data in 123–125
 survival distribution function (SDF) 245
 survivor function 245–247
 SYMCHECK function 178
 symmetric matrices 181
 SYMPUTX subroutine 90
 %SYSEVALF macro 172
 SYSRANDOM macro variable 34–35
 system time, setting seed value from 34–35

T

t distribution 28, 301
 T function 328
t test
 assessing for equality of means 78–84
 effect of sample size on 87–88
 evaluating power of 84–88
 tables
 creating data sets from 45–46, 57
 excluding 45
 finding names of 44–45
 selecting 45
 TABLES statement, FREQ procedure
 BINOMIAL option 81, 86
 CHISQ option 90
 OUT= option 56
 TABULATE call 328
 tabulated distributions
 about 18–19
 finite mixture distribution and 120
 multinomial distribution and 19, 130
 parameters for 27
 sampling from finite sets and 26
 TEMPLATE procedure
 BARCHART statement 38–39
 overlaying theoretical density on histograms 40
 templates for simulating data
 defining contour plots 268
 macro-loop technique and 100–101
 overlaying theoretical densities on histograms 40–41
 overlaying theoretical PMF on frequency plots 37–38
 univariate distributions 13–14
 with DATA step and procedures 55–57
 TEMPORARY keyword 19
 TEST statement, REG procedure 211–213
 testing for covariance matrices 177–179
 thinning algorithms 275, 278–279
 TIME function 102, 161
 time series models
 about 251
 simulating data from ARMA models 252–261
 using arrays to hold explanatory variables 201
 visualizing stationary time series with SGPLOT 257–258

time-to-event data 123–127
 TOEPLITZ function 185, 328
 Toeplitz matrix 185
 TPSPLINE procedure 247
 transformation technique 146
 TRANSPOSE procedure 66
 triangle distribution 28
 TRISOLV function 149–150, 328
 truncated distribution 121, 126
 TTEST procedure
 BY statement 80, 85
 simulated power analysis 85
 two-sample pooled variance *t* test 80, 83, 85
 two-dimensional data
 conditional simulation of 272–273
 unconditional simulation of 267–269
 two-sample *t* test
 about 78
 assessing in SAS/IML software 83–84
 effect of sample size on power of 87–88
 evaluating power of 84–86
 robustness to nonnormal populations 81–82
 robustness to unequal variances 78–81
 Type I extreme value distribution 111–112

U

UCLM= option, OUTPUT statement (MEANS) 74–75
 unconditional simulations
 about 264
 of one-dimensional data 264–267
 of two-dimensional data 267–269
 UNCONDSIMGRF function 269
 unequal variances, robustness of *t* test to 78–81
 uniform correlation structure 184
 uniform distribution
 continuous 22
 discrete 17–18
 linear regression model example 210–211
 parameters for 28, 111
 UNION function 328
 UNIQUE function 328
 univariate distributions
 acceptance-rejection technique 126–128
 adding location and scale parameters 109–111
 finite mixture distributions 119–122
 inverse CDF sampling 116–119
 resampling with SAS/IML software 288–289
 SAS software support for 27–28
 simulating data 11–12
 simulating data from continuous distributions 20–24, 28
 simulating data from discrete distributions 14–20, 27
 simulating data from standard normal distribution 12–13
 simulating data in DATA step 11–14
 simulating data in SAS/IML software 24–27

- simulating data in time series 251
 - simulating data with given moments 297
 - simulating from less common 111–116
 - simulating survival data 123–125
 - UNIVARIATE procedure
 - approximating sampling distribution 55
 - bootstrap resampling 285
 - checking correctness of simulated data 35–37
 - distributions supported by 111–112, 114–116
 - fitting gamma distribution to data 306
 - HISTOGRAM statement 40, 62, 255, 306
 - INSET statement 306
 - inverse transformation algorithm 117
 - Johnson system of distributions and 309–310
 - parametric bootstrap method 291
 - QQPLOT statement 41–44
 - sampling distribution of Pearson correlations 70
 - sampling distribution of the mean 57–59
 - sampling distribution of the variance 62
 - VARDEF= option 295
 - USE statement 329
- V**
- Vale-Maurelli algorithm 318–321
 - VAR clause, CREATE statement 331
 - VAR function 68, 328
 - VAR= option, IDENTIFY statement (ARIMA) 253
 - VARDEF= option
 - MEANS procedure 295
 - UNIVARIATE procedure 295
 - variance components model 183–184
 - variance function 226
 - variance reduction techniques 96
 - variances
 - computing for mean 61–62
 - computing for median 61–62
 - of random error term 198–199
 - robustness of *t* test to unequal 78–81
 - sampling distribution of 62–63
 - VARIOGRAM procedure 264, 268, 272
 - VARMA model 260–261
 - VARMASIM subroutine 251
 - VECDIAG function 328
 - VECH function 181
 - vectors
 - creating data sets from 330–331
 - creating grid of values 332–333
 - creating ID vectors 331–332
 - efficiency of 6
 - reading data into 329
 - tips for shortening simulation times 103
 - VMTARGETCORR function 318–319
- W**
- Wald distribution 28, 112
 - Weibull distribution
 - about 23–24
 - parameters for 28
 - proportional hazards model and 242
 - Rayleigh distribution and 114
 - WHERE clause
 - ESTIMATE statement, ARIMA procedure 254
 - READ statement 55
 - Wishart distribution 176, 186–187
 - writing data to data sets 330–331
 - writing efficient simulations
 - avoiding macro loops 100–101
 - basic structure of efficient simulations 96–97
 - disadvantages of simulations 105
 - macro usage considerations 101–102
 - profiling SAS/IML simulation 102–103
 - shorting simulation times 103–105
 - suppressing notes to SAS log 99–100
 - suppressing ODS output and graphics 97–99

X

XSECT function 328

Y

- YERRORLOWER= option, SCATTER statement (SGPLOT) 86
- YERRORUPPER= option, SCATTER statement (SGPLOT) 86

Symbols

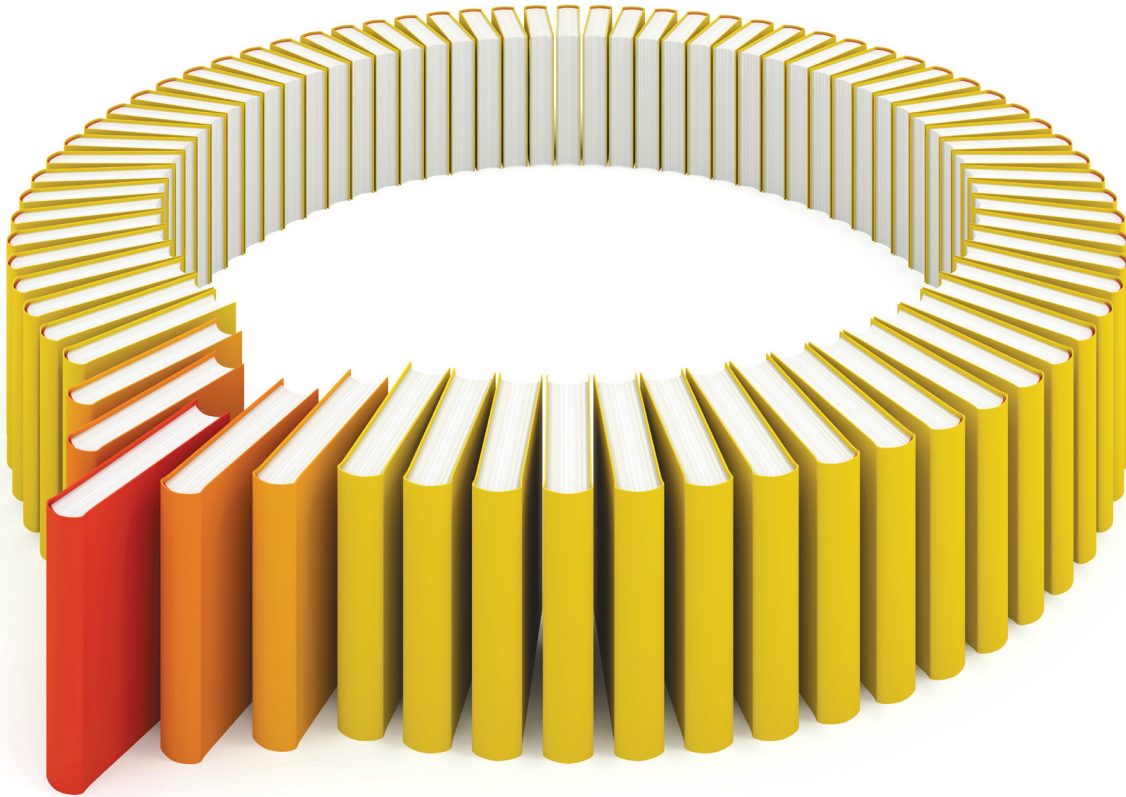
- + (addition) operator 329
- @ (at) symbol 233
- <= (comparison) operator 6
- = (equality) operator 6
- <:> (index of maximum) operator 329
- >:< (index of minimum) operator 329
- ^= (inequality) operator 6
- <> (maximum) operator 329
- : (mean) operator 68, 77, 329
- >< (minimum) operator 329
- # (multiplication) operator 215, 329
- ;(semicolon) 6
- ## (sum of squares) operator 329

About The Author



Rick Wicklin is a principal researcher in computational statistics at SAS, where he develops and supports the IML procedure and the SAS/IML Studio application. He received a PhD from Cornell University and has been a SAS user since 1997. Rick has presented numerous tutorials and papers at statistical and SAS users group conferences and is active in the American Statistical Association. Rick maintains a blog for statistical programmers at blogs.sas.com/content/iml/.

Learn more about this author by visiting his author page at <http://support.sas.com/wicklin>. There you can download free chapters, access example code and data, read the latest reviews, get updates, and more.



Gain Greater Insight into Your SAS[®] Software with SAS Books.

Discover all that you need on your journey to knowledge and empowerment.

 support.sas.com/bookstore
for additional books and resources.


THE POWER TO KNOW[®]