SSJ User's Guide

Package **`randvarmulti`**

Generating Multi-dimensional Non-Uniform Random Numbers

Version: April 25, 2007

This package implements random number generators for some multi-dimensional distributions.

Contents

Overview

This package provides a collection of classes for non-uniform random variate generation for multi-dimensional distributions.

À compléter.

RandomMultiVariateGen

This is the base class for all random variate generators over $\mathbb{R}^d$, the d-dimensional space over the reals. It specifies the signature of the `nextPoint` method, which is normally called to generate a real-valued random vector following a given distribution. A random multi-variate generator object can be created simply by invoking the constructors of this class with some previously created `RandomVariateGen` or `RandomStream` objects. The multi-dimensional generator normally uses one or more one-dimensional generators or a primitive stream to generate the components of the points one at a time.

```
package umontreal.iro.lecuyer.randvarmulti;

public abstract class RandomMultiVariateGen
```

Constructors

```
public RandomMultiVariateGen (RandomVariateGen gen1)
```
Creates a new multi-variate random generator using the one-dimensional generator `gen1`.

```
public RandomMultiVariateGen (RandomStream s)
```
Creates a new multi-variate random generator using stream `s`.

Methods

```
abstract public void nextPoint(double[] p);
```
Generates a random point p using the one-dimensional generator or the stream contained in this object.

```
public void nextArrayOfPoints (double[][] v, int start, int n)
```
Generates n random points. These points are stored in the array `v`, starting at index `start`. Thus `v[start][i]` contains coordinate i of the first generated point. By default, this method calls `nextPoint` n times, but one can override it in subclasses for better efficiency.

```
public int getDimension()
```
Returns the dimension of this multi-variate generator (the dimension of the random points).

```
public RandomVariateGen getGen1()
```
Returns the one-dimensional `RandomVariateGen` used by this object.

```
public void setGen1 (RandomVariateGen gen1)
```
Sets the `RandomVariateGen` used by this object to `gen1`.

```
public RandomStream getStream()
```
Returns the `RandomStream` used by this object.

```
public void setStream (RandomStream stream)
```
Sets the `RandomStream` used by this object to `stream`.

MultiNormalGen

Extends `RandomMultiVariateGen` for a *multivariate normal* distribution [1]. For a mean vector $\boldsymbol{\mu} \in \mathbb{R}^d$ and a symmetric positive-definite covariance matrix $\mathbf{\Sigma}$, $\mathbf{X} \in \mathbb{R}^d$ has the d-dimensional multivariate normal distribution $N(\boldsymbol{\mu}, \mathbf{\Sigma})$ with density

$$f(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^d \det(\mathbf{\Sigma})}} \exp\left(-(x-\boldsymbol{\mu})^T \mathbf{\Sigma}^{-1}(x-\boldsymbol{\mu})/2\right),$$

where $\mathbf{x} \in \mathbb{R}^d$.

In the two-dimensionnal case, assuming mean $\boldsymbol{\mu} = [\mu_1, \mu_2]$, variances $\mathrm{var}[X] = \sigma_1^2$, $\mathrm{var}[Y] = \sigma_2^2$ such that $\sigma_1 > 0$, $\sigma_2 > 0$ and correlation ρ, we have

$$\mathbf{\Sigma} = \left[\begin{array}{cc} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{array}\right].$$

```
package umontreal.iro.lecuyer.randvarmulti;

public class MultiNormalGen extends RandomMultiVariateGen
```

Constructors

`public MultiNormalGen (NormalGen gen1, int d)`

Constructs a standard d-dimensional multinormal generator, using the one-dimensional generator `gen1`.

`public MultiNormalGen (NormalGen gen1, double[] mu, DoubleMatrix2D sigma)`

Constructs a multinormal generator with mean vector `mu` and covariance matrix `sigma`, using the one-dimensional normal generator `gen1`. The mean vector must have the same length as the dimensions of the covariance matrix, which must be symmetric and positive-definite. If any of the above conditions is violated, an exception is thrown.

`public MultiNormalGen (NormalGen gen1, double[] mu, double[][] sigma)`

Equivalent to `MultiNormalGen (gen1, mu, new DenseDoubleMatrix2D (sigma))`.

Methods

`public double[] getMu()`

Returns the mean vector used by this generator.

`public double getMu (int i)`

Returns the i-th component of the mean vector for this generator.

```
public void setMu (double[] mu)
```

Sets the mean vector to `mu`.

```
public void setMu (int i, double mui)
```

Sets the i-th component of the mean vector to `mui`.

```
public DoubleMatrix2D getSigma()
```

Returns the covariance matrix $\mathbf{\Sigma}$ used by this generator.

```
public DoubleMatrix2D getCholeskyDecompSigma()
```

Returns the Cholesky decomposition of the covariance matrix used for generating the vectors. The returned matrix $\mathbf{A}$ is defined such that $\mathbf{\Sigma} = \mathbf{A}^{\mathbf{T}}\mathbf{A}$.

```
public void setSigma (DoubleMatrix2D sigma)
```

Sets the covariance matrix of this multinormal generator to `sigma`.

```
public static void nextPoint (NormalGen gen1, double[] mu,
                              DoubleMatrix2D sigma, double[] p)
```

Generates a point from the multinormal distribution with mean vector `mu`, and covariance matrix `sigma`, using the one-dimensional normal generator `gen1`. The resulting vector is put into `p`. Note that this static method may be slow for large dimension, because it needs to compute the Cholesky decomposition at every call. It is therefore recommended to use a `MultiNormalGen` object instead.

```
public static void nextPoint (NormalGen gen1, double[] mu,
                              double[][] sigma, double[] p)
```

Equivalent to `nextPoint (gen1, mu, new DenseDoubleMatrix2D (sigma), p)`.

```
public void nextPoint (double[] p)
```

Generates a point from the multinormal distribution.

DirichletGen

Extends `RandomMultiVariateGen` for a *Dirichlet* [1] distribution. This distribution uses the parameters α_i, $i = 1, \ldots, k$, and has density

$$f(x_1, \ldots, x_k) = \frac{\Gamma(\alpha_0) \prod_{i=1}^{k} x_i^{\alpha_i - 1}}{\prod_{i=1}^{k} \Gamma(\alpha_i)}$$

where $\alpha_0 = \sum_{i=1}^{k} \alpha_i$.

```
package umontreal.iro.lecuyer.randvarmulti;

public class DirichletGen extends RandomMultiVariateGen
```

Constructor

```
public DirichletGen (RandomStream stream, double[] alphas)
```

Constructs a new Dirichlet generator with parameters $\alpha_{i+1} =$ `alphas[i]`, for $i = 0, \ldots, k-1$, and the stream `stream`.

Methods

```
public double getAlpha (int i)
```

Returns the α_{i+1} parameter for this Dirichlet generator.

```
public static void nextPoint (RandomStream stream,
                              double[] alphas, double[] p)
```

Generates a new point from the Dirichlet distribution with parameters `alphas`, using the stream `stream`. The generated values are copied into `p`.

```
public void nextPoint (double[] p)
```

Generates a point from the Dirichlet distribution.

References

[1] N. L. Johnson and S. Kotz. *Distributions in Statistics: Continuous Multivariate Distributions.* John Wiley, New York, 1972.