Probability Distributions for the Insight Toolkit

James V. Miller

GE Research

Abstract. Probability distributions are a key component of clinical research. One only has to make a cursory review of clinical literature to realize that nearly all clinical publications reference some sort of statistical test; be it t-test, chi-squared test, or F-test. In this paper, we describe an architecture for providing the Insight Toolkit with access to probability distributions. The architecture can support parametric and nonparametric probability distributions. Each distribution provides access to its probability density function (PDF), cumulative distribution function (CDF), inverse cumulative distribution function (inverse CDF), mean, and variance. These methods form the basis of statistical tests.

This work is part of the National Alliance for Medical Image Computing (NAMIC), funded by the National Institutes of Health through the NIH Roadmap for Medical Research, Grant U54 EB005149. Information on the National Centers for Biomedical Computing can be obtained from http://nihroadmap.nih.gov/bioinformatics.

1 Introduction

Statistical analysis has application in both clinical research and medical image analysis. In clinical research, statistical analysis is used to determine the relative merit of various treatments and in stratifying collections of subjects. In medical image analysis, statistical analysis is used for signal decomposition, segmentation, classification, and detection. The basis of these statistical analyses are probability distributions. Probability distributions can be parametric, where the distribution is described by a small set of parameters, or nonparametric, where the distribution is a function of a set of samples.

A probability distribution is characterized by its probability density function, $f(x;\cdot)$. The probability density function can be integrated over a small interval to determine the probability of a random variable being in the specified interval. When the probability density function is integrated from $-\infty$ to a value x, the result is the cumulative distribution function

$$F(x;\cdot) = \int_{-\infty}^{x} f(u;\cdot)du. \tag{1}$$

The cumulative distribution function determines the probability of a random variable having a value less than or equal to x. The cumulative distribution function is the foundation of statistical inference. A typical statistical inference measures the probability of a random variable attaining a value *more* extreme than a given measurement x. If the probability of a more extreme value is low, we may choose to infer that measurement x is not a measurement from the given distribution and perhaps is from a different population. Arguments similar to this are used in constructing t-tests, chi-square tests, and F-tests.

In this paper, we present an architecture for providing the Insight Toolkit with access to probability distributions. These distributions can be used for a variety of clinical and medical image analysis applications. Each distribution in the *Probability Distributions* library provides access to the distribution's probability density function, cumulative distribution function, inverse cumulative distribution function, mean and standard deviation.

2 Architecture

2.1 Class hierarchy

The *Probability Distributions* class hierarchy is intended to be shallow and wide. There is a single base class `ProbabilityDistribution` from which all distributions inherit, see Figure 1. Gaussian, Chi-square, and

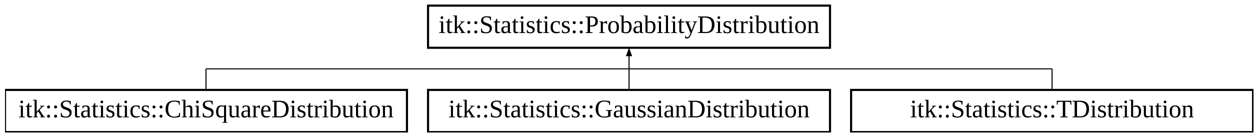
itk::Statistics::ProbabilityDistribution
itk::Statistics::ChiSquareDistribution
itk::Statistics::GaussianDistribution
itk::Statistics::TDistribution

Fig. 1. Class hierarchy for probability distributions.

Student-t distribution are currently implemented. In the future, F, exponential, and uniform distributions will be added.

The methods available to a `ProbabilityDistribution` are shown in Table 1. Access to the probability density function, cumulative distribution function, and inverse cumulative distribution function is made through the `EvaluatePDF()`, `EvaluateCDF()`, and `EvaluateInverseCDF()` methods. The argument to `EvaluatePDF()` and `EvaluateCDF()` can be any real number. The argument to `EvaluateInverseCDF()` must be between 0.0 and 1.0 inclusive. `ProbabilityDistribution` also provides access to the mean and variance of the distribution. Since not all distributions have a mean or variance, `ProbabilityDistribution` has methods `HasMean()` and `HasVariance()` to query whether the mean and variance exist. If `GetMean()` or `GetVariance()` is evaluated on a distribution that does not have a mean or variance, then (quiet) `NaN` is returned.

virtual double	**EvaluatePDF**(double x) const
virtual double	**EvaluateCDF**(double x) const
virtual double	**EvaluateInverseCDF**(double p) const
virtual bool	**HasMean**() const
virtual bool	**HasVariance**() const
virtual double	**GetMean**() const
virtual double	**GetVariance**() const

Table 1. Methods provided by ProbabilityDistribution

Concrete subclasses of `ProbabilityDistribution` provide additional methods to specify the parameters of the distribution. For instance, `GaussianDistribution` provides methods `SetMean()` and `SetVariance()` while `TDistribution` and `ChiSquareDistribution` provide a `SetDegreesOfFreedom()` method.

Subclasses of `ProbabilityDistribution` that represent parametric distributions can also provide access to the probability density function, cumulative distribution function, and inverse cumulative distribution function via static methods of the class. `GaussianDistribution` provides the static methods in Table 2. `TDistribution` provides the static methods in Table 3. `ChiSquareDistribution` provides the static methods in Table 4. The static methods allow an algorithm to access a distribution without having to create an instance of the distribution.

static double	**PDF**(double x)
static double	**PDF**(double x, double mean, double variance)
static double	**CDF**(double x)
static double	**CDF**(double x, double mean, double variance)
static double	**InverseCDF**(double p)
static double	**InverseCDF**(double p, double mean, double variance)

Table 2. Static methods provided by `GaussianDistribution`.

static double	**PDF**(double x, long degreesOfFreedom)
static double	**CDF**(double x, long degreesOfFreedom)
static double	**InverseCDF**(double p, long degreesOfFreedom)

Table 3. Static methods provided by `TDistribution`.

static double	**PDF**(double x, long degreesOfFreedom)
static double	**CDF**(double x, long degreesOfFreedom)
static double	**InverseCDF**(double p, long degreesOfFreedom)

Table 4. Static methods provided by `ChiSquareDistribution`.

2.2 Usage

There are two distint ways in which to use the distributions in the *Probability Distributions* library. The first is to create an instance of the class, call the methods to set the parameters of the distribution, and then call the methods like `EvaluateCDF()`.

```
double p;

// Create and configure a Gaussian distribution with mean 5.0 and variance 2.0
typedef itk::Statistics::GaussianDistribution GaussianDistributionType;
GaussianDistributionType::Pointer gaussian = GaussianDistributionType::New();
gaussian->SetMean(5.0);
gaussian->SetVariance(2.0);

// Evaluate the cumulative distribution function at 1.0
p = gaussian->EvaluateCDF(1.0);


// Create and configure a Student-t distribution with 50 degrees of freedom
typedef itk::Statistics::TDistribution TDistributionType;
TDistributionType::Pointer t = TDistributionType::New();
t->SetDegreesOfFreedom( 50 );

// Evaluate the cumulative distribution function at 30
p = t->EvaluateCDF(30.0);
```

Since all distributions provide uniform access to the probability density function, cumulative distribution function, and inverse cumulative distribution, it is possible to write algorithms that merely take a `ProbabilityDistribution` as input, allowing the algorithm to be reused for a variety of distributions.

A second way to use the distributions in the *Probability Distributions* library is to use the static methods provided by the specific subclasses of `ProbabilityDistribution`.

```
double p;

// Evaluate the cumulative distribution of a Gaussian with mean 5.0
// and variance 2.0 at 1.0
p = itk::Statistics::GaussianDistribution::CDF(1.0, 5.0, 2.0);

// Evaluate the cumulative distribution of a Student-t distribution
// with 50 degrees of freedom at 30
p = itk::Statistics::TDistribution::CDF(30.0, 50);
```

Accessing the distributions via the static methods is useful when an algorithm writer knows the distribution of the data at the time he/she is writing the algorithm. Using the static methods avoids having to create an instance of the distribution.

2.3 Netlib

Much of the numerical processing in the *Probability Distributions* library is provided by Netlib routines (http://www.netlib.org) [1]. Originally crafted in the 1970's, the Netlib routines were implemented in Fortran with great attention paid to numerical precision and accuracy. From Netlib, the *Probability Distributions* library uses implementations of the

1. Gamma function
2. Incomplete Gamma function
3. Logarithm of the Gamma function
4. Incomplete Beta function
5. Logarithm of the Beta function

The *Probability Distributions* library uses C versions of these Netlib routines, converted from the original Fortran using `f2c`:

```
f2c -a -p *.f
```

One issue with using `f2c` converted code within ITK, is that the resulting C code is rarely thread safe. The root cause of the thread un-safeness is the preponderance of `static` variables in the generated C code. The `-a` and `-p` options to `f2c` attempt to convert the original Fortran common blocks to local variables instead of naively translating Fortran common blocks to C static variables. But even using the `-a` and `-p` options to `f2c`, the resulting C code can still have static variables.

One pattern used in the Netlib code is that the first time a routine is invoked, some pre-calculations are performed and results cached for future invocations of the routine. These pre-calculations are usually to determine machine precision. Determining machine precision in C is a lot simpler than it was when Fortran ruled the world. The Fortran code had to determine the machine architecture (Vax, IBM Mainframe, Convex) and lookup the precision hardcoded for that architecture. C code, on the other hand, only has to include `<float.h>` and access symbols like `DBL_MIN` or `DBL_MAX` or `DBL_EPSILON` to determine machine precision. To take advantage of this, the C code emitted by `f2c` was hand editted to remove the pattern of pre-calculating machine precision on the first invocation of the routine, resulting in code that is thread safe.

2.4 Inverse CDF calculation

The cumulative distribution function is a key function in statistical analysis. The cumulative distribution function $F(x;\cdot)$ is

$$p = F(x;\cdot) = \int_{-\infty}^{x} f(u;\cdot)dx \tag{2}$$

where $f(x;\cdot)$ is the probability density function. The cumulative distribution function evaluates the probability of having a random variable less than or equal to x. In a one-side statistical test, the cumulative distribution function can be evaluated at a measurement or sample x to determine the probability of a having a measurement less than or equal to the current measurement. If the probability is high — greater than a threshold P^* (typical values of P^* are 0.95, 0.99 or 0.999) — then the current measurement x is in the upper tail of the distribution and perhaps the measurement is not actually from this distribution at all. Similar evaluations can be performed on the lower tail of the distribution by comparing against small values of P^*, identifying measurements that are less than P^* and therefore in the lower tail.

Evaluating the cumulative distribution function can be an expensive calculation, one which you may want to avoid if you need to analyze a large number of measurements or samples. An alternative construction of this statistical test is to use the inverse of the cumulative distribution function. Here, the probability threshold P^* is prescribed, for instance $P^* = 0.99$, and the inverse cumulative distribution function evaluated at P^* to determine the value x^* such that the probability of a measurement being less than or equal to x^* is P^*. Once x^* is determined, the statistical test is merely a comparison of each measurement x against x^*. In you have a lot of measurements to evaluate, using the inverse cumulative distribution function approach requires only a single evaluation of the inverse cumulative distribution function instead of multiple evaluations of the (forward) cumulative distribution function.

The inverse cumulative distribution function is

$$x = F^{-1}(p; \cdot) = \arg_x p - \int_{-\infty}^{x} f(u; \cdot)du = 0 \tag{3}$$

For most distributions, a closed form solution for $F^{-1}(p; \cdot)$ is not available. Therefore, the inverse cumulative distribution function is solved numerically. There are approximations to the inverse cumulative distribution function for many distributions, see [2] for specific approximations. Some of these approximations are polynomial and some are based on other inverse functions. For example, the approximation for the inverse CDF of the Student-t distribution is a polynomial of order 9. The approximation for the inverse CDF of the chi-square distribution uses the inverse cumulative distribution function of the Gaussian under a change of variables. These approximations are only asymptotically accurate but can serve as reasonable initial guesses for a numerical optimization process. In many cases, just a few Newton iterations (3 to 10) is all that is needed to find the inverse of the cumulative distribution function to roughly 10^{-10} accuracy.

In the Newton iterations, we are trying to find the roots of

$$g(x) = p - F(x; \cdot) \tag{4}$$

where p is given. Let x_0 be an initial guess to $F^{-1}(p)$ provided by an approximation as discussed above. The Newton iterations are

$$\begin{aligned} x_{i+1} &= x_i - \frac{g(x_i)}{g'(x_i)} & (5) \\ &= x_i - \frac{p - F(x_i; \cdot)}{-f(x_i; \cdot)} & (6) \\ &= x_i + \frac{p - F(x_i; \cdot)}{f(x_i; \cdot)} & (7) \end{aligned}$$

where $F(x; \cdot)$ is the cumulative distribution function of x and $f(x; \cdot)$ is the probability density function of x.

2.5 Numerical precision

One goal of the *Probability Distributions* library is to maximize numerical precision while limiting computational complexity. Since the values computed by the methods of the *Probability Distributions* library are typically used in other calculations to form statistical tests — with each of these calculations loosing numerical precision — maximizing numerical precision at the level of the *Probability Distributions* library is paramount. Numerical precision is one of the reasons the *Probability Distributions* library utilizes routines from Netlib. The precision of the methods in the *Probability Distributions* library are typically between 10^{-10} and 10^{-15} (though some calculations drop down to 10^{-8}).

The numerical precision attained by these methods does come at the cost of computational complexity. Where possible, iterative techniques have been limited in duration to yield only the precision needed. One option to mitigate computational complexity when many statistical tests need to be perfomed is to use the methods in the *Probability Distributions* library to build tables of thresholds and refer to these thresholds instead of evaluating the distributions for every sample being tested.

3 Statistical inference

Statistical inference is the process of inferring information about a process or population based on a set of samples. The main activity in statistical inference is hypothesis testing [4]. Hypothesis testing involves:

- constructing a null hypothesis, H_0,
- modelling the null hypothesis via a test statistic,
- calculating the probability of the test statistic, assuming the null hypothesis is true.

The null hypothesis is the research question being considered. An example of a null hypothesis is ”there is no difference between the response of two groups of subjects receiving different treatments”. The null hypothesis is transformed into a concrete concept by modelling the hypothesis with a test statistic. For the previous example, the test statistic may be based on the difference between the two (unknown) population means of a response variable. Finally, the test statistic is measured and the probability of this measurement is determined assuming the null hypothesis is true. For the previous example, the probability of the difference between the group means being different from zero is calculated. This probability is referred to as the *P value*, with values near 0 indicating an unlikely occurence if the null hypothesis is true. For the previous example, a group mean difference with a low P value would indicate that the two groups are unlikely to be from the same population.

The key to statistical inference is in modelling the null hypothesis with a test statistic that is descriptive, sufficient, and whose probabilities can be evaluated.

There are many statistical tests that can be used in statistical inference. Several of these statistical tests are used as motivating examples in the next section.

4 Distributions

4.1 Gaussian distribution

Many natural processes have been found to approximately follow a Gaussian or normal distribution. Whenever a process is the result of many independent and additive events, we can infer from the central limit thereom the resulting distribution will be approximately Gaussian [5]. The Gaussian distribution has a probability density function

$$f(x;\mu,\sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{8}$$

where μ and σ^2 are the mean and variance of the distribution. The cumulative distribution function is

$$F(x;\mu,\sigma^2) = \frac{1}{2}\left(1 + \operatorname{erf}\frac{x-\mu}{\sigma\sqrt{2}}\right) \tag{9}$$

where $\operatorname{erf}(x)$ is the error function.

4.2 Student-t distribution

William Sealey Gosset published the derivation of the t-distribution in 1908 under the pseudonym *Student* [6,7]. Gosset needed statistical methods appropriate for small sample sizes for selecting the best yielding variants of barley for Guinness brewery. The Student-t statistic arises when comparing the sample mean to the population mean when the population variance is unknown. Given n independent samples, identically distributed, from a normal distribution, the statistic

$$t = \frac{\hat{\mu} - \mu}{\hat{\sigma}/\sqrt{n}} \tag{10}$$

where $\hat{\mu}$ is the sample mean, $\hat{\sigma}^2$ is the sample variance, and μ is the population mean, has a Student-t distribution with probability density function

$$f(t;v) = \frac{\Gamma((v+1)/2)}{\sqrt{v\pi}\,\Gamma(v/2)(1+t^2/v)^{(v+1)/2}} \tag{11}$$

with $v = n-1$ degrees of freedom and where $\Gamma(x)$ is the Gamma function. The cumulative distribution of the Student-t distribution [2, derived from 26.7.1] is

$$F(t;v) = \begin{cases} 1 - \frac{1}{2} I_x(v/2, 1/2) & t \geq 0, \\ \frac{1}{2} I_x(v/2, 1/2) & t < 0 \end{cases} \tag{12}$$

where $I_x(a,b)$ is the incomplete Beta function and

$$x = \frac{v}{v + t^2}. \tag{13}$$

The Student-t distribution is symmetric, has a mean of zero, and a variance of $v/(v-2)$ for values of $v > 2$. The square of a t-distributed random variable with v degrees of freedom is an F-distributed random variable with 1 and v degrees of freedom.

t Tests In practice, the population mean is rarely known, so Equation 10 is actually used to make inferences on the population mean. One such inference is to identify a likely lower (or upper) bound on the true value of the population mean given just n identically distributed samples from a normal distribution. We define a statistical bound on a random variable t from a Student-t distribution

$$Pr[t < t^*] < P^* \tag{14}$$

where P^* is a desired confidence on the bound. A value for P^* of 0.95 defines an upper bound on the t-statistic with confidence of 95%. Frequently, the confidence level is specified by α where $P^* = 1 - \alpha$. The inverse cumulative distribution function with $v = n - 1$ degrees of freedom can be used to determine t^*,

$$t^* = F^{-1}(P^*; v). \tag{15}$$

Comparing the critical value of t^* with the Student-t statistic in Equation 10

$$\frac{\hat{\mu} - \mu}{\hat{\sigma}/\sqrt{n}} < t^* \tag{16}$$

provides an equation that can be solved for the population mean μ,

$$\mu > \hat{\mu} - \frac{t^* \hat{\sigma}}{\sqrt{n}}. \tag{17}$$

Equation 17 defines a lower bound on the population mean μ. Note that the lower bound is a function of the number of samples n. As the number of samples increases, the confidence bound on the population mean converges to the sample mean $\hat{\mu}$ (as expected since the sample mean $\hat{\mu}$ converges to the population mean μ as the number of samples increases). A similar analysis can be performed to determine an upper bound on the population mean. `TDistribution::InverseCDF(p, n - 1)` can be used to determine t^* for any desired confidence level P^*.

An alternative to this test is to hypothesize a particular population mean and test the hypothesis that the sample mean models the population mean. Here, we evaluate Equation 10 using the hypothesized mean μ, and the estimated $\hat{\mu}$ and $\hat{\sigma}$. The probability of the resulting t value is calculated from the cumulative distribution function,

$$Pr[t] = F(t; v) - F(-t; v) \tag{18}$$

at $v = n - 1$ degrees of freedom. This is the probability the sample is from the population with the hypothesized mean. The *P value* of this test is

$$P = 1 - (F(t; v) - F(-t; v)) \tag{19}$$

which yields a value close to zero when the sample mean is "far" from the hypothesized mean (where "far" is a function of the sample variance and the number of samples). `TDistribution::CDF(t, n − 1)` can be used to evaluate $F(t; v)$ and $F(-t; v)$.

A second common inference is determining whether two groups of samples are from same population. This is a common test when analyzing two groups of subjects receiving different treatments. If we hypothesize the two groups of subjects are from the same population and hence have the same mean, then we can compare the confidence intervals on the population mean for the first group with the confidence intervals on the population mean for the second group. If the confidence levels do not overlap, then we conclude the two groups are from different populations at the 2α level. Again, `TDistribution::InverseCDF(p, n_i - 1)` can be used in determining the confidence levels for each population mean.

Alternatively, we can construct a single test statistic based on the difference between the two population means. The statistic

$$t = \frac{(\hat{\mu}_1 - \hat{\mu}_2) - (\mu_1 - \mu_2)}{\sqrt{\frac{\hat{\sigma}_1^2}{n_1} + \frac{\hat{\sigma}_2^2}{n_2}}} \tag{20}$$

where $\hat{\mu}_1$, $\hat{\sigma}_1^2$ and n_1 are the sample mean, sample variance, and number of samples in group one, and $\hat{\mu}_2$, $\hat{\sigma}_2^2$ and n_2 are the sample mean, sample variance, and number of samples in group two, is commonly approximated as a Student-t random variable with $n_1 + n_2 - 2$ degrees of freedom. For the two groups of subjects to be from the same population $\mu_1 - \mu_2$ must be zero. Therefore, Equation 20 reduces to

$$t = \frac{\hat{\mu}_1 - \hat{\mu}_2}{\sqrt{\frac{\hat{\sigma}_1^2}{n_1} + \frac{\hat{\sigma}_2^2}{n_2}}}. \tag{21}$$

A value of t can calculated from the data and the probability that the two group means are the same can be calculated using the cumulative distribution function

$$Pr[t] = F(t; v) - F(-t; v) \tag{22}$$

at $n_1 + n_2 - 2$ degrees of freedom. The *P value* of this test is same as the previous test

$$P = 1 - (F(t; v) - F(-t; v)) \tag{23}$$

which yields a value close to zero when the difference in the sample means is so large that it is unlikely to have occurred from random chance (where random chance is a function of the sample variances and sample sizes). `TDistribution::CDF(t,` $\mathtt{n_1 + n_2 - 2}$`)` can be used to evaluate $F(t; v)$ and $F(-t; v)$.

4.3 Chi-square distribution

Given n independent samples, normally distributed, with parameters (μ_i, σ_i^2), the statistic

$$\chi^2 = \sum_{i=1}^{n} \left(\frac{x_i - \mu_i}{\sigma_i} \right)^2 \tag{24}$$

is a chi-square random variable with $v = n$ degrees of freedom. The chi-square probability density function [2, 26.4.1] is

$$f(x; v) = \frac{1}{2^{v/2} \Gamma(v/2)} x^{v/2-1} e^{-x/2} \tag{25}$$

and the chi-square cumulative distribution function [2, 26.4.19] is

$$F(x; v) = \frac{\gamma(v/2, x/2)}{\Gamma(v/2)} \tag{26}$$

where $\gamma(a, x)$ is the incomplete gamma function and $\Gamma(x)$ is the gamma function. The chi-square distribution has mean v and variance $2v$.

Chi-square Tests The chi-square statistic in Equation 24 measures the total squared deviation of a set of samples from their respective population means and normalized by their respective population standard deviations. A simplified version of this statistic is used when the samples are all from the same distribution ($\mu_i = \mu$, $\sigma_i^2 = \sigma^2$). Under these conditions, chi-square statistics are frequently used for goodness of fit tests, for instance evaluating a fit calculated via regression.

Another goodness of fit application is testing whether a set of samples is from a particular hypothesized distribution. Here, the chi-square statistic takes the form

$$\chi^2 = \sum_{i} \frac{(O_i - E_i)^2}{E_i} \tag{27}$$

where O_i is the i^{th} observed frequency and E_i is the i^{th} expected frequency. Given a set of samples, histogram methods can be used to determine the frequencies of measurements. These frequencies are used as the O_i values in the test. E_i is determined by integrating the probability density function of the hypothesized distribution over the i^{th} histogram bin,

$$E_i = F(x_{\texttt{upper}}; \cdot) - F(x_{\texttt{lower}}; \cdot), \tag{28}$$

where $x_{\texttt{upper}}$ and $x_{\texttt{lower}}$ are the upper and lower bounds on the i^{th} histogram bin. `ChiSquareDistribution::CDF(x, v)` can be used to evaluate the test statistic in Equation 27. The number of degrees of freedom in the chi-square distribution is

$$v = n_k - n_p \tag{29}$$

where n_k is the number of nonempty histogram bins and n_p is the number of parameters in the hypothesized distribution. Any distribution in the *Probability Distributions* libary can be used for the hypothesized distribution in Equation 28.

4.4 Other distributions

At the time of this writing, the Gaussian, Student-t, and chi-square distributions have been implemented in the *Probability Distributions* library. We plan to extend this library with at least the F-distribution, uniform distribution, and exponential distribution. Other distributions may follow (Poisson, Binomial, Cauchy, lognormal, etc.).

5 Sample based distributions

The distributions presented in Section 4 are all parametric distributions. These are idealized distributions that can be modelled using a small number of parameters. The drawback to parametric distributions lies in the relevancy to the processes being analyzed. If the process under consideration can only be modelled *approximately* by a parametric distribution, then any resulting inference can be in error simply because the parameteric distribution did not model the process accurately.

Alternatives to parametric distributions are non-parametric distributions. These distributions are based on a collection of samples. Examples include histogram methods and kernel function methods [8]. The *Probability Distributions* library can be extended to include non-parametric distributions. Such distributions only need to respond to `EvaluatePDF()`, `EvaluateCDF()`, `EvaluateInverseCDF()`, `GetMean()`, and `GetVariance()`. Supporting non-parametric distributions would allow for for distributions to be learned from the data being analyzed. Non-parametric distributions can be constructed from a set of training data and used in evaluating test data.

6 Random variates

When validating a statistical model, it is common to draw samples from the hypothesized underlying distribution to compare against the actual measurements. The methods in the *Probability Distributions* library can be used to draw samples from any of the probability distributions modeled by using the *inverse method* [2]. First, draw a sample from a uniform distribution in the range of 0.0 to 1.0. ITK provides a Mersenne Twister [9] algorithm that can be used for produce uniform variates. Then evaluate the inverse cumulative distribution function of the distribution in question at this random value to map it to a random variable from the hypothesized distribution. This method of generating random variates can be used on any distribution that is at least numerically invertible.

7 Conclusion

In this paper, we present an architecture for providing the Insight Toolkit with access to standard parametric probability distributions. These distributions form the basis of the statistical tests used in the majority of the clinical research publications. We describe the architecture, two different ways to utilize the API, and described the underlying mathematics and algorithms. The architecture can be extended to include non-parametric statistical distributions as well as multivariate distributions. The architecture leverages methods from Netlib for much of the numerical processing. The Netlib routines were converted to C using `f2c` and then editted by hand to ensure thread safety.

References

1. Dongarra, J., Grosse, E., eds.: Netlib. (http://www.netlib.org)
2. Abramowitz, M., Stegun, I.: Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables. Volume 55 of Applied Mathematics Series. (1966)
3. Silvey, S.: Statistical Inference. Chapman and Hall, London (1975)
4. Whitley, E., Ball, J.: Statistics review 3: Hypothesis testing and p values. Critical Care **6** (2002)
5. Leon-Garcia, A.: Probability and Random Processes for Electrical Engineering. Addison-Wesley, Reading, Ma. (1989)
6. "Student" (W.S. Gosset): The probable error of a mean. Biometrika **6** (1908) 1–25
7. Wikipedia contributors: Wikipedia: The Free Encyclopedia. http://en.wikipedia.org/wiki/Main_Page (2005)
8. Silverman, B.: Density estimation for statistics and data analysis. Chapman and Hall (1992)
9. Matsumoto, M., Nishimura, T.: Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. ACM Transactions on Modeling and Computer Simulation **8** (1998) 3–30