RcppDist Introduction

JB Duck-Mayr

2018-10-01

The Rcpp package provides a C++ library to make it easier to use C++ with R. R and Rcpp provide functions for a variety of statistical distributions. Several R packages make functions available to R for additional statistical distributions. However, to access these functions from C++ code, a costly call to the R functions must be made.[1]

RcppDist provides a C++ header-only library with functions for additional statistical distributions that can be called from C++ when writing code using Rcpp or RcppArmadillo. Functions are available that return NumericVectors as well as doubles, and for multivariate or matrix distributions, Armadillo vectors and matrices.

RcppDist provides functions for the following distributions:

- the four parameter beta distribution
- the location-scale t distribution
- the truncated normal distribution
- the truncated t distribution
- a truncated location-scale t distribution
- the triangle distribution
- the multivariate normal distribution*
- the multivariate t distribution*
- the Wishart distribution*
- the inverse Wishart distribution*

Distributions marked with an asterisk rely on RcppArmadillo.

Including RccpDist in Your Code

RcppDist provides several header files you can include in your code. `RcppDist.h` includes all of them, as well as `Rcpp.h` or `RcppArmadillo.h` as appropriate (see "Use Rpp or RcppArmadillo?" below). So, in any C++ file you need to use functions from RcppDist, you can simply use

```
#include <RcppDist.h>
```

which will also take care of Rcpp(Armadillo) headers for you. You can alternatively pull in only the header(s) you need; for example, if you only need functions for the four parameter beta distribution, you can just use

```
#include <4beta.h>
```

However, you'll then be responsible for pulling in Rcpp/RcppArmadillo headers as appropriate. The header names that correspond to the various distributions are as follows:

[1] Some of the R packages alluded to have written these functions in C++ (in some cases using Rcpp). However, these packages do not make such functions available as a header library for other package writers intending to make use of the functions in C++ code, thus the motivation for this package.

Distribution	Header
Four Parameter Beta	4beta.h
Location-Scale t	lst.h
Truncated Normal	truncnorm.h
Truncated t	trunct.h
Truncated Location-Scale t	trunclst.h
Triangle	triangular.h
Multivariate Normal	mvnorm.h
Multivariate t	mvt.h
Wishart and Inverse Wishart	wishart.h

Using Rcpp or RcppArmadillo?

Including `RcppDist.h` by default will pull in the RcppArmadillo headers (and therefore the Rcpp headers), as well as the RcppDist headers. If you would prefer to use Rcpp but *not* RcppArmadillo (i.e. include the Rcpp headers but not the RcppArmadillo headers), include the line

```
#define RCPPDIST_DONT_USE_ARMA
```

before any inclusion of `RcppDist.h`, though this will make the asterisked (multivariate and Wishart) distributions unavailable.

Using RcppDist in a Package

With Rcpp

To use RcppDist in a package that does not link to RcppArmadillo, you must

- Add Rcpp and RcppDist to the LinkingTo field of your DESCRIPTION file.
- Add Rcpp to the Imports field of your DESCRIPTION file.
- Remember to define `RCPPDIST_DONT_USE_ARMA` before any include of `RcppDist.h`.

With RcppArmadillo

To use RcppDist in a package that links to RcppArmadillo, you must

- Add Rcpp, RcppArmadillo, and RcppDist to the LinkingTo field of your DESCRIPTION file.
- Add Rcpp to the Imports field of your DESCRIPTION file.
- Remember to add the appropriate `PKG_LIBS` flags to `src/Makevars` and `sr/Makevars.win` (see the RcppArmadillo manual for more details).

Using RcppDist in a Standalone file

With Rcpp

If you are using RcppDist in a standalone file (i.e., not as part of a package), and you don't want to pull in the Armadillo headers, you'll need

```
#define RCPPDIST_DONT_USE_ARMA
#include <RcppDist.h>
// [[Rcpp::depends(RcppDist)]]
```

at the top of your file. If you want to pull in just one or more of the distribution specific headers, you won't need the define, but don't forget to also include `Rcpp.h`; as an example:

```
#include <Rcpp.h>
#include <triangular.h>
// [[Rcpp::depends(RcppDist)]]
```

With RcppArmadillo

If you are using RcppDist in a standalone file (i.e., not as part of a package), and you do want the Armadillo headers, you'll need

```
#include <RcppDist.h>
// [[Rcpp::depends(RcppArmadillo)]]
// [[Rcpp::depends(RcppDist)]]
```

at the top of your file. If you want to pull in just one or more of the distribution specific headers, don't forget to also include `RcppArmadillo.h`; as an example:

```
#include <RcppArmadillo.h>
#include <mvnorm.h>
// [[Rcpp::depends(RcppArmadillo)]]
// [[Rcpp::depends(RcppDist)]]
```

RcppDist Functions

Much like distributions in R, functions are prefixed by d, p, q, and r to mean density, distribution, quantile, and random number generating functions respectively. Functions that are return double rather than, say, a NumericVector are instead prefixed by d__, p__, q__, and r__. Below are more detailed descriptions of the functions provided by each header.

4beta.h (Four Parameter Beta Distribution)

The four parameter beta distribution is a beta distribution supported over an interval $[a, b]$ rather than only $[0, 1]$. The functions provided in this header are:

```
Rcpp::NumericVector d4beta(Rcpp::NumericVector x, double shape1,
    double shape2, double a, double b, bool log_p = false)

Rcpp::NumericVector p4beta(Rcpp::NumericVector q, double shape1,
    double shape2, double a, double b, bool lower_tail = true,
    bool log_p = false)

Rcpp::NumericVector q4beta(Rcpp::NumericVector p, double shape1,
    double shape2, double a, double b, bool lower_tail = true,
    bool log_p = false)

Rcpp::NumericVector r4beta(int n, double shape1, double shape2,
    double a, double b)

double d_4beta(double x, double shape1, double shape2, double a,
    double b, int log_p = 0)

double p_4beta(double q, double shape1, double shape2,
    double a, double b, int lower_tail = 1, int log_p = 0)
```

```
double q_4beta(double p, double shape1, double shape2,
    double a, double b, int lower_tail = 1, int log_p = 0)

double r_4beta(double shape1, double shape2, double a, double b)
```

Where

- `x` and `q` are quantiles (either a single value or a vector depending)
- `p` is a single probability or a vector of probabilities
- `n` is the numer of observations to draw
- `shape1` and `shape2` are the positive shape parameters of the Beta distribution
- `a` and `b` are the minimum and maximum values of the distribution respectively
- `log_p` is a `bool` or `int` (the default is `false`/0); if `true` (or > 0), the probabilities are given as $\log(p)$
- `lower_tail` is a `bool` or `int`; if `true` (or > 0), the probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.

lst.h (Location-Scale t Distribution)

The location-scale t distribution is a t distribution shifted by a location parameter μ and scaled by a scaling parameter σ. The functions provided in this header are:

```
Rcpp::NumericVector dlst(Rcpp::NumericVector x, double df, double mu,
    double sigma, bool log_p = false)

Rcpp::NumericVector plst(Rcpp::NumericVector q, double df, double mu,
    double sigma, bool lower_tail = true, bool log_p = false)

Rcpp::NumericVector qlst(Rcpp::NumericVector p, double df, double mu,
    double sigma, bool lower_tail = true, bool log_p = false)

Rcpp::NumericVector rlst(int n, double df, double mu, double sigma)

double d_lst(double x, double df, double mu, double sigma,
    int log_p = 0)

double p_lst(double q, double df, double mu, double sigma,
    int lower_tail = 1, int log_p = 0)

double q_lst(double p, double df, double mu, double sigma,
    int lower_tail = 1, int log_p = 0)

double r_lst(double df, double mu, double sigma)
```

Where

- `x` and `q` are quantiles (either a single value or a vector depending)
- `p` is a single probability or a vector of probabilities
- `n` is the numer of observations to draw
- `df` is the positive degrees of freedom
- `mu` is the location/shifting parameter
- `sigma` is the scaling parameter
- `log_p` is a `bool` or `int` (the default is `false`/0); if `true` (or > 0), the probabilities are given as $\log(p)$
- `lower_tail` is a `bool` or `int`; if `true` (or > 0), the probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.

truncnorm.h (Truncated Normal Distribution)

The truncated normal distribution is a normal distribution supported over an interval $[a, b]$ rather than $(-\infty, \infty)$. The functions provided in this header are:

```
Rcpp::NumericVector dtruncnorm(Rcpp::NumericVector x, double mu,
    double sigma, double a, double b, bool log_p = false)

Rcpp::NumericVector ptruncnorm(Rcpp::NumericVector x, double mu,
    double sigma, double a, double b, bool lower_tail = true,
    bool log_p = false)

Rcpp::NumericVector qtruncnorm(Rcpp::NumericVector p, double mu,
    double sigma, double a, double b, bool lower_tail = true,
    bool log_p = false)

Rcpp::NumericVector rtruncnorm(int n, double mu, double sigma,
    double a, double b)

double d_truncnorm(double x, double mu, double sigma, double a,
    double b, int log_p = 0)

double p_truncnorm(double x, double mu, double sigma, double a, double b,
    int lower_tail = 1, int log_p = 0)

double q_truncnorm(double p, double mu, double sigma, double a, double b,
    int lower_tail = 1, int log_p = 0)

double r_truncnorm(double mu, double sigma, double a, double b)
```

Where

- `x` and `q` are quantiles (either a single value or a vector depending)
- `p` is a single probability or a vector of probabilities
- `n` is the numer of observations to draw
- `mu` is the mean of the distribution
- `sigma` is the standard deviation
- `a` and `b` are the minimum and maximum values of the distribution respectively
- `log_p` is a `bool` or `int` (the default is `false`/0); if `true` (or > 0), the probabilities are given as $\log(p)$
- `lower_tail` is a `bool` or `int`; if `true` (or > 0), the probabilities are $P[X \le x]$, otherwise, $P[X > x]$.

trunct.h (Truncated t Distribution)

The truncated t distribution is a t distribution supported over an interval $[a, b]$ rather than $(-\infty, \infty)$. The functions provided in this header are:

```
Rcpp::NumericVector dtrunct(Rcpp::NumericVector x, double df,
    double a, double b, bool log_p = false)

Rcpp::NumericVector ptrunct(Rcpp::NumericVector x, double df,
    double a, double b, bool lower_tail = true, bool log_p = false)

Rcpp::NumericVector qtrunct(Rcpp::NumericVector p, double df,
    double a, double b, bool lower_tail = true, bool log_p = false)
```

```
Rcpp::NumericVector rtrunct(int n, double df, double a, double b)

double d_trunct(double x, double df, double a, double b, int log_p = 0)

double p_trunct(double x, double df, double a, double b,
    int lower_tail = 1, int log_p = 0)

double q_trunct(double p, double df, double a, double b,
    int lower_tail = 1, int log_p = 0)

double r_trunct(double df, double a, double b)
```

Where

- `x` and `q` are quantiles (either a single value or a vector depending)
- `p` is a single probability or a vector of probabilities
- `n` is the numer of observations to draw
- `df` is the positive degrees of freedom
- `a` and `b` are the minimum and maximum values of the distribution respectively
- `log_p` is a `bool` or `int` (the default is `false`/0); if `true` (or > 0), the probabilities are given as $\log(p)$
- `lower_tail` is a `bool` or `int`; if `true` (or > 0), the probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.

trunclst.h (Truncated Location-Scale t Distribution)

The truncated location-scale t distribution is a location-scale t distribution supported over an interval $[a, b]$ rather than $(-\infty, \infty)$. The functions provided in this header are:

```
Rcpp::NumericVector dtrunclst(Rcpp::NumericVector x, double df,
    double mu, double sigma, double a, double b, bool log_p = false)

Rcpp::NumericVector ptrunclst(Rcpp::NumericVector x, double df,
    double mu, double sigma, double a, double b, bool lower_tail = true,
    bool log_p = false)

Rcpp::NumericVector qtrunclst(Rcpp::NumericVector p, double df,
    double mu, double sigma, double a, double b, bool lower_tail = true,
    bool log_p = false)

Rcpp::NumericVector rtrunclst(int n, double df, double mu, double sigma,
    double a, double b)

double d_trunclst(double x, double df, double mu, double sigma, double a,
    double b, int log_p = 0)

double p_trunclst(double x, double df, double mu, double sigma, double a,
    double b, int lower_tail = 1, int log_p = 0)

double q_trunclst(double p, double df, double mu, double sigma,
    double a, double b, int lower_tail = 1, int log_p = 0)

double r_trunclst(double df, double mu, double sigma,
    double a, double b)
```

Where

- `x` and `q` are quantiles (either a single value or a vector depending)
- `p` is a single probability or a vector of probabilities
- `n` is the numer of observations to draw
- `df` is the positive degrees of freedom
- `mu` is the location/shifting parameter
- `sigma` is the scaling parameter
- `a` and `b` are the minimum and maximum values of the distribution respectively
- `log_p` is a `bool` or `int` (the default is `false`/0); if `true` (or > 0), the probabilities are given as $\log(p)$
- `lower_tail` is a `bool` or `int`; if `true` (or > 0), the probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.

triangular.h (Triangle Distribution)

The triangle (or triangular) distribution is supported over an interval $[a, b]$ with a mode c; as the name suggests, the density function is shaped like a triangle with vertices at a, b, and c. The functions provided in this header are:

```
Rcpp::NumericVector dtri(Rcpp::NumericVector x, double a, double b,
    double c, bool log_p = false)

Rcpp::NumericVector ptri(Rcpp::NumericVector x, double a, double b,
    double c, bool lower_tail = true, bool log_p = false)

Rcpp::NumericVector qtri(Rcpp::NumericVector p, double a, double b,
    double c, bool lower_tail = true, bool log_p = false)

Rcpp::NumericVector rtri(int n, double a, double b, double c)

double d_tri(double x, double a, double b, double c, int log_p = 0)

double p_tri(double x, double a, double b, double c,
    int lower_tail = 1, int log_p = 0)

double q_tri(double p, double a, double b, double c,
    int lower_tail = 1, int log_p = 0)

double r_tri(double a, double b, double c)
```

Where

- `x` and `q` are quantiles (either a single value or a vector depending)
- `p` is a single probability or a vector of probabilities
- `n` is the numer of observations to draw
- `a` and `b` are the minimum and maximum values of the distribution respectively
- `c` is the mode of the distribution
- `log_p` is a `bool` or `int` (the default is `false`/0); if `true` (or > 0), the probabilities are given as $\log(p)$
- `lower_tail` is a `bool` or `int`; if `true` (or > 0), the probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.

mvnorm.h (Multivariate Normal Distribution)

The multivariate normal distribution is a generalization of the normal distribution to multiple dimensions. Then each draw is a vector, the mean parameter μ is a vector, and rather than a scalar standard deviation parameter σ, we have a covariance matrix Σ (or here denoted S). The functions provided in this header are:

```
arma::vec dmvnorm(const arma::mat& x, const arma::vec& mu, arma::mat S,
    bool log_p = false)

arma::mat rmvnorm(const arma::uword n, const arma::vec& mu,
    const arma::mat& S)
```

Where

- `x` is a matrix of quantiles, such that each row is a quantile
- `n` is the numer of observations to draw
- `mu` is the mean vector
- `S` is the covariance matrix
- `log_p` is a `bool` (the default is `false`); if `true`, the probabilities are given as $\log(p)$

mvt.h (Multivariate t Distribution)

The multivariate t distribution is a generalization of the t distribution to multiple dimensions. Then each draw is a vector, and in addition to the degrees of freedom, we have a correlation matrix Σ (or here denoted S), and this implementation allows for a location vector μ. The functions provided in this header are:

```
arma::vec dmvt(const arma::mat& x, const arma::vec& mu, arma::mat S,
    const double df, const bool log_p = false)

arma::mat rmvt(const arma::uword n, const arma::vec& mu,
    const arma::mat& S, const double df)
```

Where

- `x` is a matrix of quantiles, such that each row is a quantile
- `n` is the numer of observations to draw
- `mu` is the location vector
- `S` is the correlation matrix
- `log_p` is a `bool` (the default is `false`); if `true`, the probabilities are given as $\log(p)$

wishart.h (Wishart and Inverse Wishart Distributions)

The Wishart distribution is a generalization of the gamma distribution to multiple dimensions defined over symmetric, nonnegative-definite random matrices. Its parameters are the degrees of freedom and a scale matrix S. If $X \sim \text{Wishart}(\text{df}, S)$, then $X^{-1} \sim \text{Inverse Wishart}(\text{df}, S^{-1})$. Due to their use in the density functions for these distributions, a multivariate gamma function and logged multivariate gamma function are also provided. Note that for now, al functions for this distribution are designed to deal with only one random matrix. The functions provided in this header are:

```
double mvgamma(const int p, const double x)

double lmvgamma(const int p, const double x)

double dwish(const arma::mat& X, const int df, const arma::mat& S,
    const bool log_p = false)

arma::mat rwish(const int df, const arma::mat& S)

double diwish(const arma::mat& X, const int df, const arma::mat& S,
    const bool log_p = false)
```

```
arma::mat riwish(const int df, const arma::mat& S)
```

Where

- `p` and `x` are the arguments to the multivariate gamma function
- `X` is a matrix, a draw from the Wishart or Inverse Wishart distribution
- `df` is the degrees of feedom
- `S` is the scale matrix
- `log_p` is a `bool` (the default is `false`); if `true`, the probabilities are given as $\log(p)$