RcppOctave: Octave along with R

by Dirk Eddelbuettel and Renaud Gaujoux

Abstract The **RcppOctave** package connects Octave to R. TODO: Expand

Introduction

Octave (Eaton et al., 2008) is a high-level interactive language which is primarily intended for numerical computations. It has found widespread adoption across different disciplines as it is mostly compatible with Matlab (MATLAB, 2010). While Matlab has historically been pre-eminent in domains such as applied mathematics, electrical engineering and signal processing, it is also widely used in other fields such as machine learning, bioinformatics, and finance. Consequently, a large corpus of application programs are available as `.m`-files (named after the commonly-chose file extension) for which Octave provides an open source engine.

R (R Core Team, 2012), a language and environment for statistical computing and graphics, has become the dominant language for statistical research, and a widely-used environment for empirical work in a variety of fields.

While both languages share commonalities, their respective focus is different making a combination of both environments an even more compelling choice. This short paper illustrates the **RcppOctave** package by Gaujoux (2012) which implements an interface between both these environments.

Octave

TODO: Two or three short paragraphs about Octave

Example: Matrix Operations

TODO: Show two or three simple call illustrating the variable arguments etc.

Example: RNG

```
library(RcppOctave)

# R runif
set.seed(1)
a <- runif(10)

# define octave function
o_source(text="
function [x] = orng(n)
 x = rand(1, n);
end
")
set.seed(1)
identical(.O$orng(10), a)

## [1] TRUE
```

Example: Kalman Filter

Eddelbuettel and Sanderson (2012) motivate the **RcppArmadillo** package with an example comparing a Kalman Filter implementation in both R and C++. As the code underlying this example was initially published for Matlab[1], it can of course also be used with RcppOctave.

TODO: Few words about the example

```
function Y = kalmanM(pos)
  dt=1;
  %% Initialize state transition matrix
  A=[ 1 0 dt 0 0 0;...     % [x  ]
     0 1 0 dt 0 0;...      % [y  ]
     0 0 1 0 dt 0;...      % [Vx]
     0 0 0 1 0 dt;...      % [Vy]
     0 0 0 0 1 0 ;...      % [Ax]
     0 0 0 0 0 1 ];        % [Ay]
  % Initialize measurement matrix
  H = [ 1 0 0 0 0 0; 0 1 0 0 0 0 ];
  Q = eye(6);
  R = 1000 * eye(2);
  x_est = zeros(6, 1);
  p_est = zeros(6, 6);

  numPts = size(pos,1);
  Y = zeros(numPts, 2);

  for idx = 1:numPts
    z = pos(idx, :)';

    %% Predicted state and covariance
    x_prd = A * x_est;
    p_prd = A * p_est * A' + Q;
    %% Estimation
    S = H * p_prd' * H' + R;
    B = H * p_prd';
```

[1] See http://www.mathworks.com/products/matlab-coder/demos.html?file=/products/demos/shipping/coder/coderdemo_kalman_filter.html.

```
    klm_gain = (S  B)';
    %% Estimated state and covariance
    x_est = x_prd + klm_gain * (z - H * x_prd);
    p_est = p_prd - klm_gain * H * p_prd;
    %% Compute the estimated measurements
    Y(idx, :) = H * x_est;
  end                % of the function
end   % of the function
```

This function, along with several R implementations, is provided in the **RcppOctave** package as `demo(Kalman)`. Table 1 summaries the performance.

Implementation	Time in sec.	Rel. to best
KalmanOctave	3.943	1.000
KalmanRC	5.987	1.518
KalmanR	6.047	1.534
KalmanRfunC	6.262	1.588
KalmanRfun	6.648	1.686
FirstKalmanRC	8.992	2.281
FirstKalmanR	9.405	2.385

Table 1: Performance comparison of various implementations of a Kalman filter. KalmanOctave is the **RcppOctave** based implementation shown above. KalmanR is the Rimplementation using an environment; KalmanRfun use a function; FirstKalmanR is a direct translation of the original Matlab implementation; versions ending in C are the byte-compiled variants of the respective version. Timings are averaged over 100 replications. The comparison was made using R version 2.15.1, Rcpp version 0.9.13 and RcppOctave version 0.8.5 on Ubuntu 12.04 running in 64-bit mode on a 2.67 GHz Intel i7 processor.

TODO: More about the example, performance comparison, knocking socks off R

Example: Gibbs Sampler

Wilkinson (TODO: Reference from his repeated blog posts) used a simple bivariate Gibbs Sampler as a basis for comparisons between different programming languages such as C, Java, Python and R. His example has been re-used in number of other presentations.

We can adapt this example here as it provides a suitable framework for showing how **RcppOctave** can interact with the random number generators in R.

Bibliography

J. W. Eaton, D. Bateman, and S. Hauberg. *GNU Octave Manual Version 3*. Network Theory Limited, 2008. ISBN 0-9546120-6-X.

D. Eddelbuettel and C. Sanderson. RcppArmadillo: Easily extending R with high-performance C++ code. Manuscript submitted to *Computational Statistics and Data Analysis*, 2012.

R. Gaujoux. *RcppOctave: Seamless Interface to Octave – and Matlab*, 2012. URL http://CRAN.R-Project.org/package=RcppOctave. R package version 0.8.5.

MATLAB. *Version 7.10.0 (R2010a)*. The MathWorks Inc., Natick, Massachusetts, 2010.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2012. URL http://www.R-project.org/. ISBN 3-900051-07-0.

Dirk Eddelbuettel
River Forest, IL
USA
edd@debian.org

Renaud Gaujoux
Computational Biology
University of Cape Town
Cape Town
South Africa
renaud@cbio.uct.ac.za