Higher-Performance R via C++

Part 6: RInside

Dirk Eddelbuettel

UZH/ETH Zürich R Courses

June 24-25, 2015

Overview: Standard

First Example: `rinside_sample0.cpp`

The simplest example:

```
#include <RInside.h>                        // for the embedded R via RInside

int main(int argc, char *argv[]) {

    RInside R(argc, argv);                  // create an embedded R instance

    R["txt"] = "Hello, world!\n";           // assign a char* (string) to 'txt'

    R.parseEvalQ("cat(txt)");               // eval string, ignoring any returns

    exit(0);
}
```

RInside in a Nutshell

Key aspects:

- RInside uses the embedding API of R
- An instance of R is launched by the RInside constructor
- It behaves just like a regular R process
- We submit commands as C++ strings which are parsed and evaluated
- Rcpp used to easily get data in and out from the enclosing C++ program.

Overview: MPI

Parallel Computing via MPI

R is famously single-threaded.

High-performance Computing with R frequently resorts to fine-grained (multicore/parallel, doSMP) or coarse-grained (Rmpi, pvm, ...) parallelism. R spawns and controls other jobs.

Jianping Hua suggested to embed R via RInside in MPI applications.

Now we can use the standard and well understood MPI paradigm to launch multiple R instances, each of which is indepedent of the others.

Parallel Computing via MPI

```
#include <mpi.h>     // mpi header
#include <RInside.h> // for the embedded R via RInside

int main(int argc, char *argv[]) {
    MPI::Init(argc, argv);                          // mpi initialization
    int myrank = MPI::COMM_WORLD.Get_rank();        // obtain current node rank
    int nodesize = MPI::COMM_WORLD.Get_size();      // obtain total nodes running.

    RInside R(argc, argv);                          // create an emb. R instance

    std::stringstream txt;
    txt << "Hello from node " << myrank             // node information
    << " of " << nodesize << " nodes!" << std::endl;

    R["txt"] = txt.str();                           // assign string var to R var
    R.parseEvalQ("cat(txt)");                       // eval init string

    MPI::Finalize();                                // mpi finalization
    exit(0);
}
```

Parallel Computing via MPI

```
$ orterun -n 4 ./rinside_mpi_sample2
Hello from node 0 of 4 nodes!
Hello from node 3 of 4 nodes!
Hello from node 2 of 4 nodes!
Hello from node 1 of 4 nodes!
$
```

This uses Open MPI just locally, other hosts can be added via `-H node1,node2,node3`.

The other example(s) shows how to gather simulation results from MPI nodes.

Application Example: Qt

"How to embed R within a larger application" ?

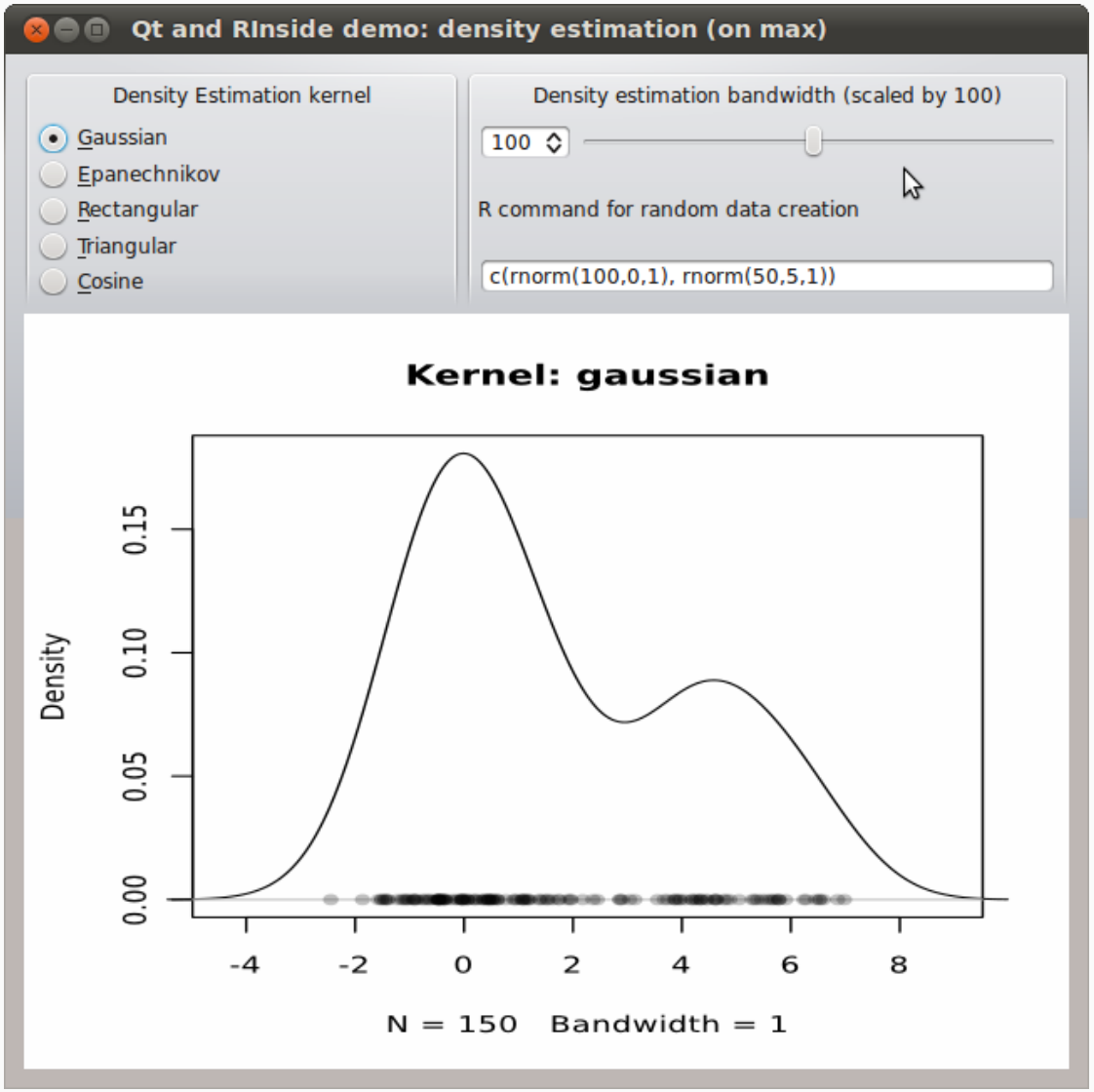
We have an example for Qt.

```
#include <QApplication>
#include "qtdensity.h"

int main(int argc, char *argv[]) {
    RInside R(argc, argv);              // create an emb. R inst

    QApplication app(argc, argv);
    QtDensity qtdensity(R);             // pass R inst. by ref.
    return app.exec();
}
```

Application Example: Qt



This uses standard Qt / GUI paradigms of

- radio buttons
- sliders
- textentry

all of which send values to the R process which provides an SVG (or PNG as fallback) image that is plotted.

Building with RInside

RInside needs headers and libraries from several projects as it

- *embeds R itself* so we need R headers and libraries
- *uses Rcpp* so we need Rcpp headers and libraries
- *used RInside itself* so we also need RInside headers and libraries

Building with RInside

The `GNUmakefile` is set-up to create a binary for each example example file supplied. It uses

- `R CMD config` to query all of `--cppflags`, `--ldflags`, `BLAS_LIBS` and `LAPACK\_LIBS`
- `Rscript` to query `Rcpp:::CxxFlags` and `Rcpp:::LdFlags`
- `Rscript` to query `RInside:::CxxFlags` and `RInside:::LdFlags`

The `qtdensity.pro` file does the equivalent for Qt.