

# RemoteREngine

Keeping R at a distance with java rmi

**Romain François**

Ian Long

John James

*Professionnal R Enthusiast*

*Stoat Software*

*Mango Solutions*

LondonR, 2009-11-03

# Agenda

Why

What

How

REngine  
java rmi  
design

For example

hello world  
console

Future

Drinks

RemoteREngine

Romain François, John  
James, Ian Long

Why

What

How

REngine

java rmi

design

For example

hello world

console

Future

Drinks

# Why

Why do we want R remote

- ▶ Distribute R computing load to other available computers
- ▶ Off load heavy jobs to dedicated machine
- ▶ Access R without accessing the actual machine
- ▶ Leave the R process open and interact with it
- ▶ Call linux R from windows

RemoteREngine

Romain François, John  
James, Ian Long

Why

What

How

REngine

java rmi

design

For example

hello world

console

Future

Drinks

# What

## Basic needs and requirements for remotng R

- ▶ Client (java) and server (R) run on different machines
- ▶ Multiple client applications share the same R session
- ▶ A client may require many R sessions to work together
- ▶ Established REngine API
- ▶ R package (RemoteREngine) containing client and server side jar files

- ▶ server side : RemoteREngine-server.jar and startup script

```
Rscript -e "RemoteREngine::start.server()"
```

- ▶ client side: RemoteREngine-client.jar

```
$ java -cp  
"RemoteREngine-client.jar:example-helloworld.jar"  
RemoteHelloWorld
```

# REngine API

REngine abstraction

The `org.rosuda.REngine` java package defines :

- ▶ Java representation of R objects.

**REXP**, `REXPEnvironment`, `REXPReference`, `REXPDouble`,  
`REXPInteger`, `REXPList`, `REXPLogical`, `REXPRaw`, `REXPString`,  
`REXPSymbol`, `REXPExpressionVector`, `REXPFactor`,  
`REXPGenericVector`, `REXPLanguage`, `REXPNull`, `REXPS4`,  
`REXPUnknown`, `REXPVector`,

- ▶ How to access/modify R objects.

`parse`, `eval`, `assign`, `getReference`,  
`resolveReference`, `getParentEnvironment`, `newEnvironment`

Established API used for several years by projects through JRI or Rserve, i.e *JGR*

# rJava, JRI, REngine

## REngine abstraction

```
public abstract class REngine{

    public REXP parse(String text, boolean resolve)
    public REXP eval(REXP what, REXP where, boolean resolve)
    public void assign(String symbol, REXP value, REXP env)
    public REXP get(String symbol, REXP env, boolean resolve)

    public REXP resolveReference(REXP ref)
    public REXP createReference(REXP value)

    public REXP getParentEnvironment(REXP env, boolean resolve)
    public REXP newEnvironment(REXP parent, boolean resolve)

}
```

# JRI, REngine

currently available implementations

- ▶ **JRIEngine**: R is embedded as a thread within a JVM
  - ▶ local applications
  - ▶ If R crashes, the application crashes (same process)
- ▶ **RConnection** : Rserve implementation. R runs on a server machine and uses TCP/IP for data transport via the Rserve package
  - ▶ Low level data transport.
  - ▶ No support for environments or references
- ▶ **RemoteREngine**: best of both worlds ?

# java rmi

## Remote Method Invocation

- ▶ RMI is a technology, part of standard java, that allows to call a method of a java object that lives in a different JVM.
- ▶ Data transport can be configured. http, https through ssl, ...
- ▶ Classes can be dynamically loaded, at runtime

Details and tutorial available at

<http://java.sun.com/docs/books/tutorial/rmi/>



# Basic design

- ▶ On the **server** side, R is embedded in java via *JRIEngine*
- ▶ The **client** side gets a remote reference to this server
- ▶ Calls to methods of the REngine are sent to the server and data is serialized back to client
- ▶ The server has the ability to **call back** the client

# Hello World example

Grab `norm(5)` from an engine running in another jvm of the same physical machine

```
if (System.getSecurityManager() == null) {
    System.setSecurityManager(new RMISecurityManager());
}

RemoteREngine r = new RemoteREngine( "RemoteREngine" ,
    "localhost", 1099 );

double[] d = { 1.0, 2.0, 3.0 } ;
r.assign( "xx", d ) ;
double[] result = r.parseAndEval( "xx^2" ).asDoubles() ;

for( int i=0; i<result.length; i++){
    System.out.println( " " + result[i]) ;
}
```

RemoteREngine

Romain François, John  
James, Ian Long

Why

What

How

REngine

java rmi

design

For example

hello world

console

Future

Drinks

# Remote R Console example

Client java application sending commands to R's REPL, use of the callback mechanism

```
public static void main( String[] args) {
    try{
        if (System.getSecurityManager() == null) {
            System.setSecurityManager(new RMISecurityManager());
        }
        RemoteREngine r = new RemoteREngine( "RemoteREngine",
            "localhost", 1099 );
        r.addCallbackListener( new ConsoleCallbackListener() );

        ConsoleThread console = new ConsoleThread( r ) ;
        console.start( ) ;
    } catch( Exception e){
        e.printStackTrace();
    }
}
```

RemoteREngine

Romain François, John  
James, Ian Long

Why

What

How

REngine

java rmi

design

For example

hello world

console

Future

Drinks

# Remote R Console example

callback listeners

```
private static class ConsoleCallbackListener implements  
CallbackListener {
```

```
public void handleCallback( RCallback callback ){  
    if( callback instanceof RWriteConsoleCallback ){  
        pr(((RWriteConsoleCallback)callback).getMessage() ) ;  
    } else if( callback instanceof RShowMessageCallback){  
        pr(((RShowMessageCallback)callback).getMessage() ) ;  
    } else if( callback instanceof ReadConsoleCallback){  
        pr(((ReadConsoleCallback)callback).getPrompt() ) ;  
    } else if( callback instanceof InputCallback ){  
        pr(((InputCallback)callback).getCommand() + "\n" ) ;  
    }  
}
```

```
public static void pr( String text){  
    System.out.print( text ) ;  
}
```

```
}
```

RemoteREngine

Romain François, John  
James, Ian Long

Why

What

How

REngine

java rmi

design

For example

hello world

console

Future

Drinks

# Remote R Console example

console thread

```
private static class ConsoleThread extends Thread {
    private DefaultConsoleReadLine readline ;
    private RemoteREngine engine;

    private ConsoleThread( RemoteREngine engine){
        super() ;
        this.engine = engine ;
        this.readline = new DefaultConsoleReadLine( );
    }

    public void run(){
        System.out.print( "> " ) ;
        while( true ){
            String line = readline.readLine();
            engine.sendToConsole( line ) ;
        }
    }
}
```

RemoteREngine

Romain François, John  
James, Ian Long

Why

What

How

REngine

java rmi

design

For example

hello world

console

Future

Drinks

# Future developments

- ▶ Generic callbacks holding arbitrary data
- ▶ Concurrency between multiple clients
- ▶ Transport of Graphics through rmi
- ▶ Activation, Daemon to create other engines
- ▶ help server

RemoteREngine

Romain François, John  
James, Ian Long

Why

What

How

REngine

java rmi

design

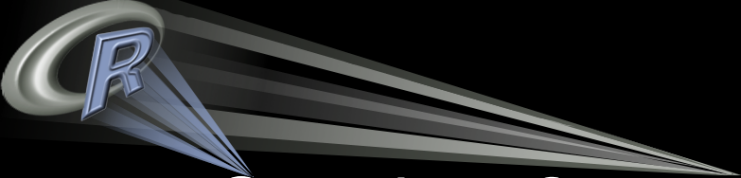
For example

hello world

console

**Future**

Drinks



# Questions ?

<http://r-forge.r-project.org/projects/remoterengine/>

**Romain François** <http://romainfrancois.blog.free.fr>

**Ian Long** <http://www.stoatsoftware.com>

**John James** <http://www.mango-solutions.com>