

Microarray Informatics at the EBI

Alvis Brazma Group

BIOCEP Project

Karim Chine

BBSRC (UK) grant BB/E001653/1

Grant's Principal Investigators: Wolfgang Huber, Misha Kapushesky

What do we need ?

- Provide tools and frameworks for Bioconductor packages automatic exposure as Web Services
- Provide a scalable, robust architecture for R integration within ArrayExpress Warehouse and Expression Profiler

State of the art

- **SJava and rJava/JRI**
 - Basic mapping via JNI of the R C API
- **TypeInfo**
 - Plug meta descriptions to R functions
- **RWebservices**
 - Generated Java Beans for basic R Types / S4 Classes
 - Axis Web Services based on SJava and ActiveMQ
- **JavaGD**
 - R devices connection to Java (JGR)
- **Rserve**
 - TCP/IP interface to R

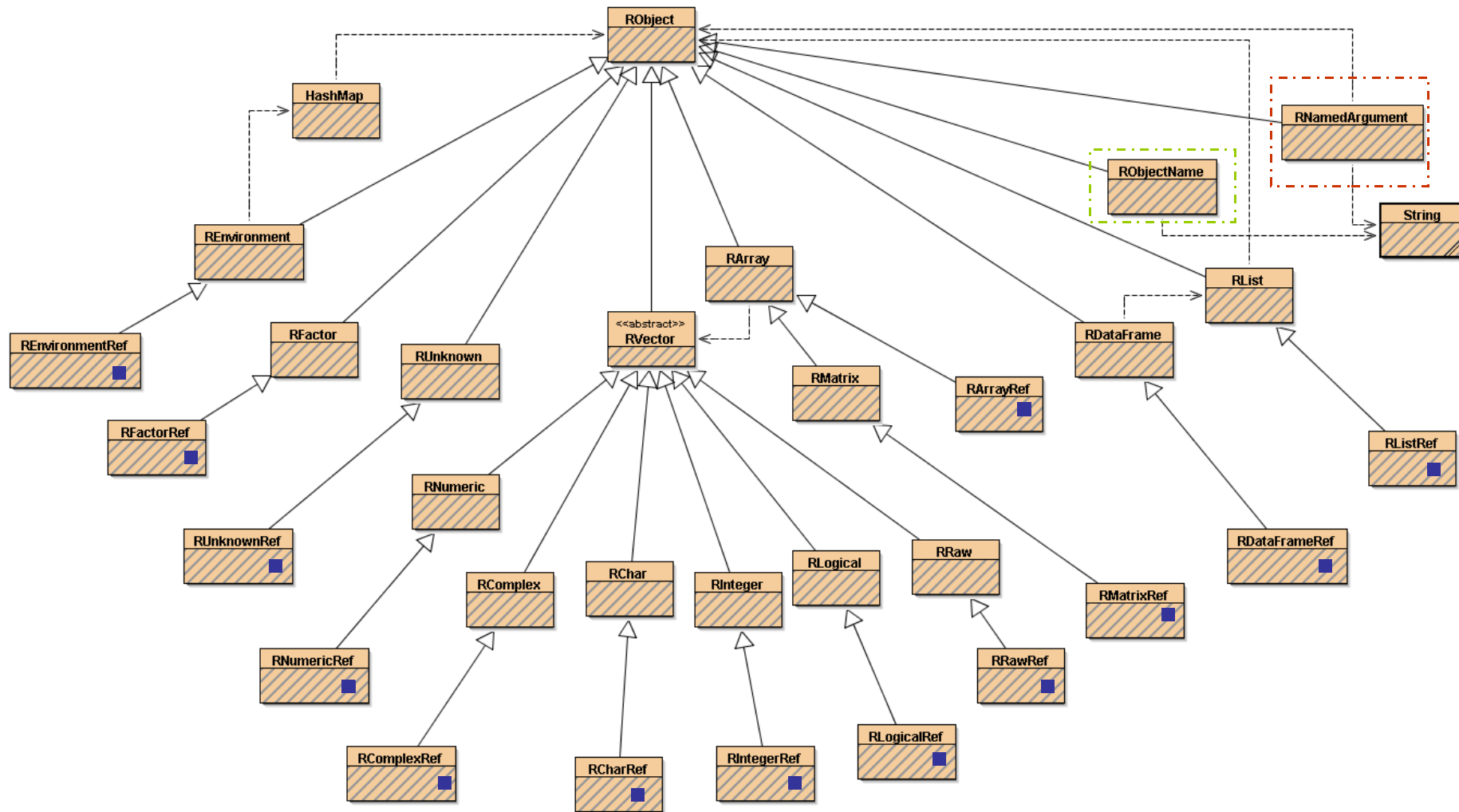
What is missing ?

- High Level Java API for Accessing R
- Stateful, Resuable, Remotable R Components
- Scalable, Distributed, R Based Infrastructure
- Safe multiple clients framework for components usage as a pool of indistinguishable Remote Resources
- User friendly Interface for the remote resources creation, tracking and debugging

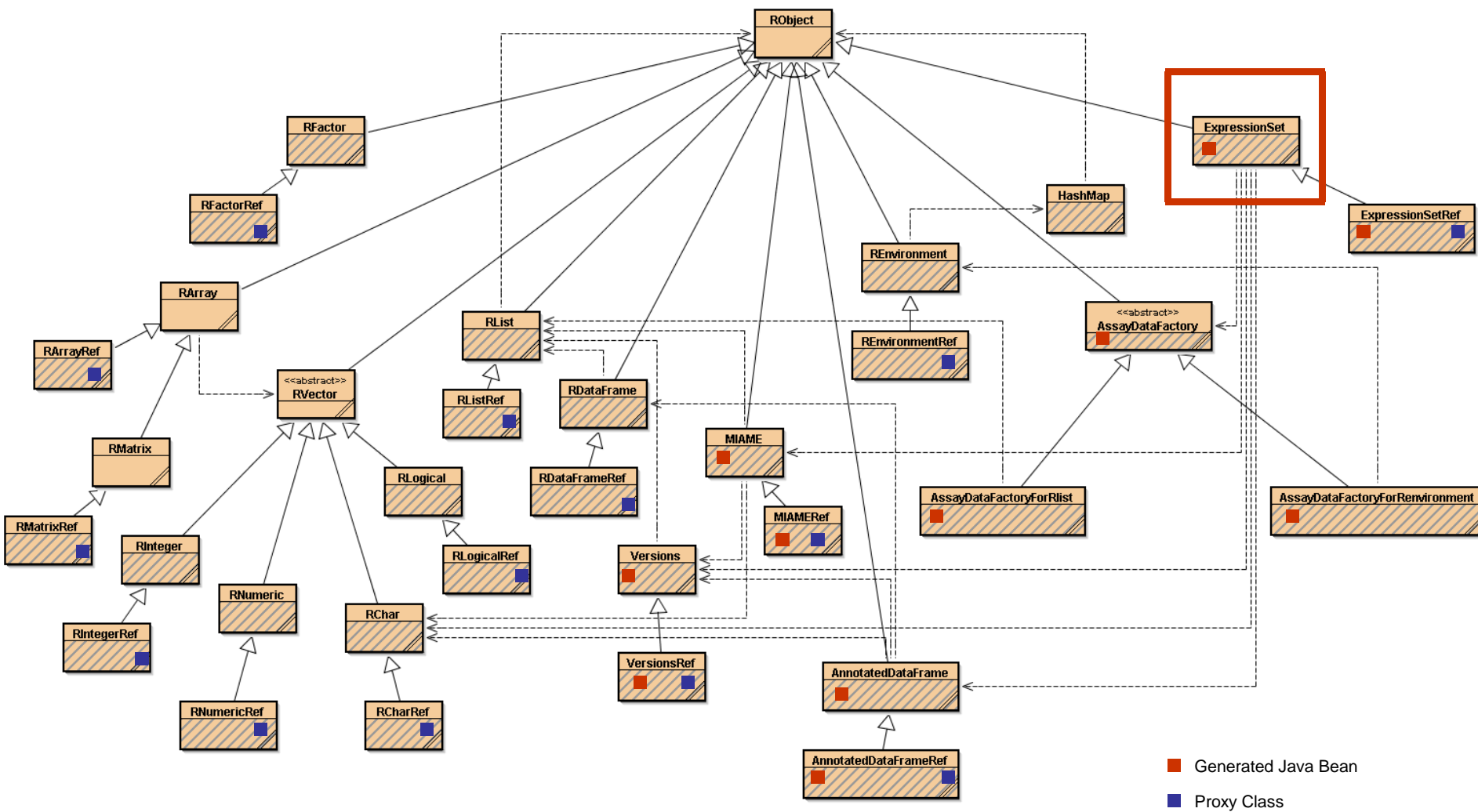
What is missing ?

- Generated light-weight Java proxies for R Types / S4 Classes
- On-demand mapping and deployment of R packages as RMI Components or as JAX-WS Web Services
- Remotable R Graphics / Swing Components for R
- Remote R components files exchange API
- Semi-thick client (applet) for web based tools using R

Standard R objects mapping to Java



Generated beans for ExpressionSet



High Level API for Accessing R

public interface RServices extends ManagedServant {

public String **evaluate**(String expression) **throws** RemoteException;

...

public RObject **call**(String methodName, RObject... args) **throws** RemoteException;

...

public RObject **callAsReference**(String methodName, RObject... args) **throws** RemoteException;

...

public RObject **evalAndGetObject**(String expression) **throws** RemoteException;

...

public RObject **evalAndGetObjectAsReference**(String expression) **throws** RemoteException;

...

public RObject **putObjectAndGetReference**(RObject obj) **throws** RemoteException;

...

public void **putObjectAndAssignName**(RObject obj, String name) **throws** RemoteException;

...

public RPackage **getPackage**(String packageName) **throws** RemoteException;

...

public void **setCallback**(RCallback callback) **throws** RemoteException;

...

public GDDevice **newDevice**(int w, int h) **throws** java.rmi.RemoteException;

...

public byte[] **readWorkingDirectoryFileBlock**(String fileName, long offset, int blocksize) ..

...

public String **getStatus**() **throws** RemoteException;

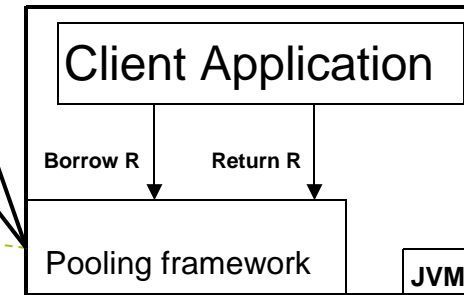
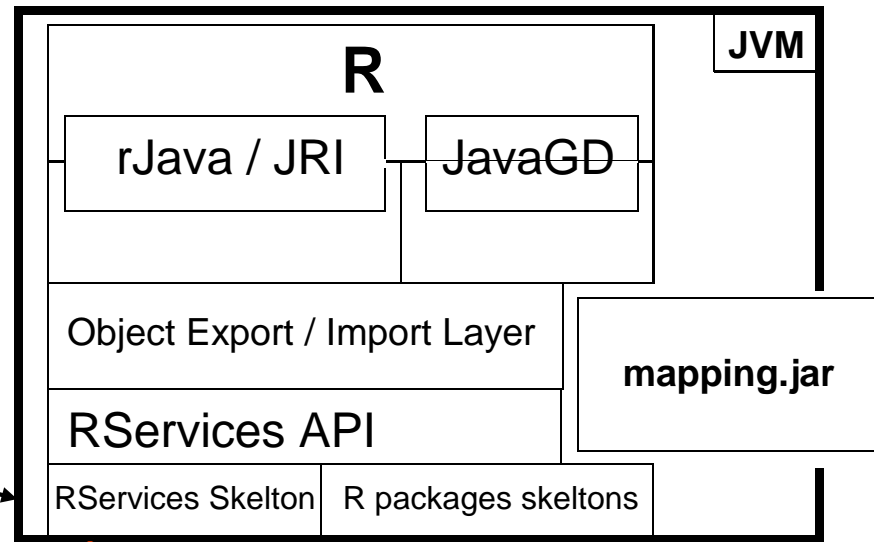
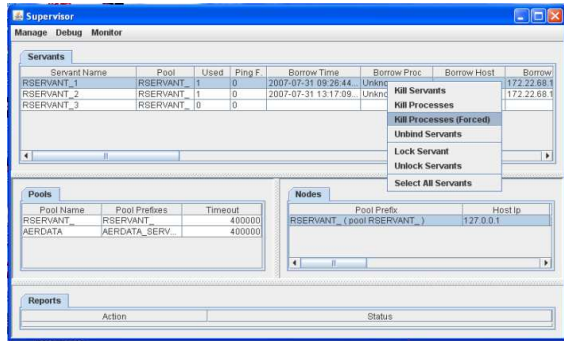
...

Remote Resource Pool Framework

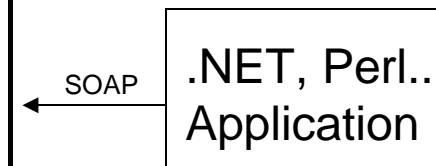
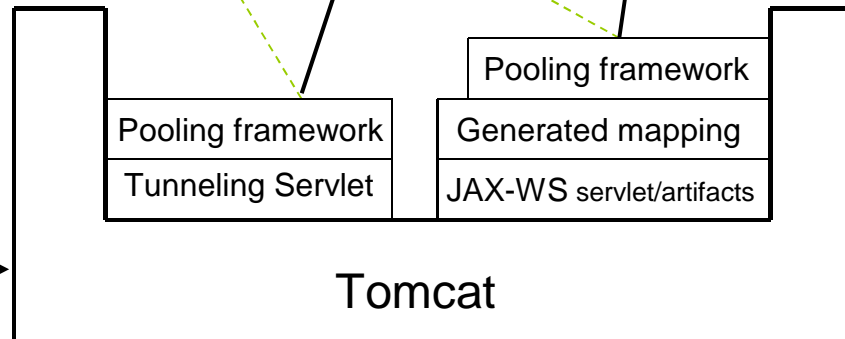
- Generic Standalone framework
- Pooling of any RMI components and if combined with JNI of any library / open architecture
- New Remote Object Registry based on Derby | Oracle | MySql
- Three implementations available
 - rmiregistry / mono-node / single client process
 - rmiregistry / multinodes / single client process
 - database ROR / multinodes / multiple client processes
- User friendly interface for the remote resources creation, tracking and debugging, nodes and pools management

Architecture I

Supervisor



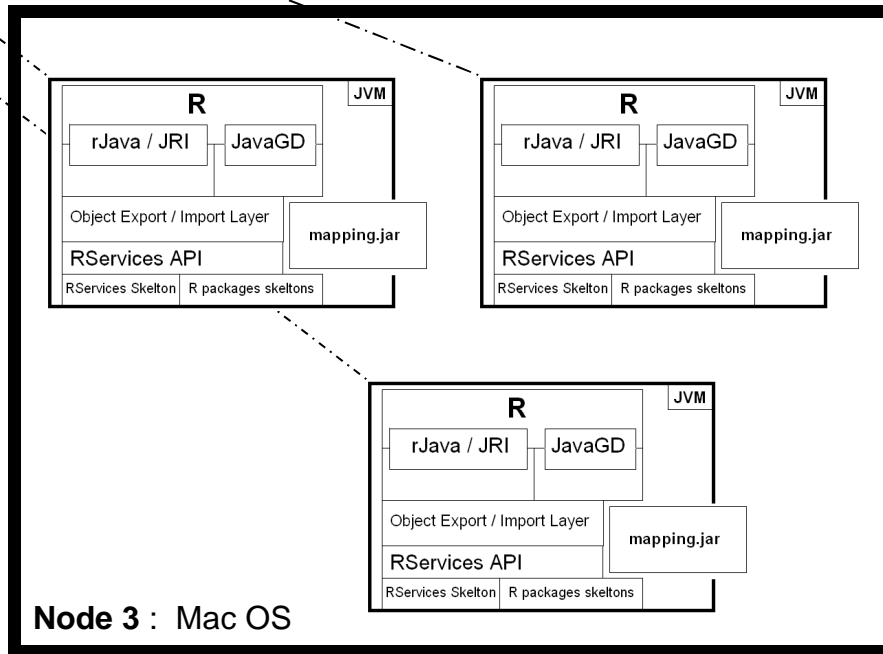
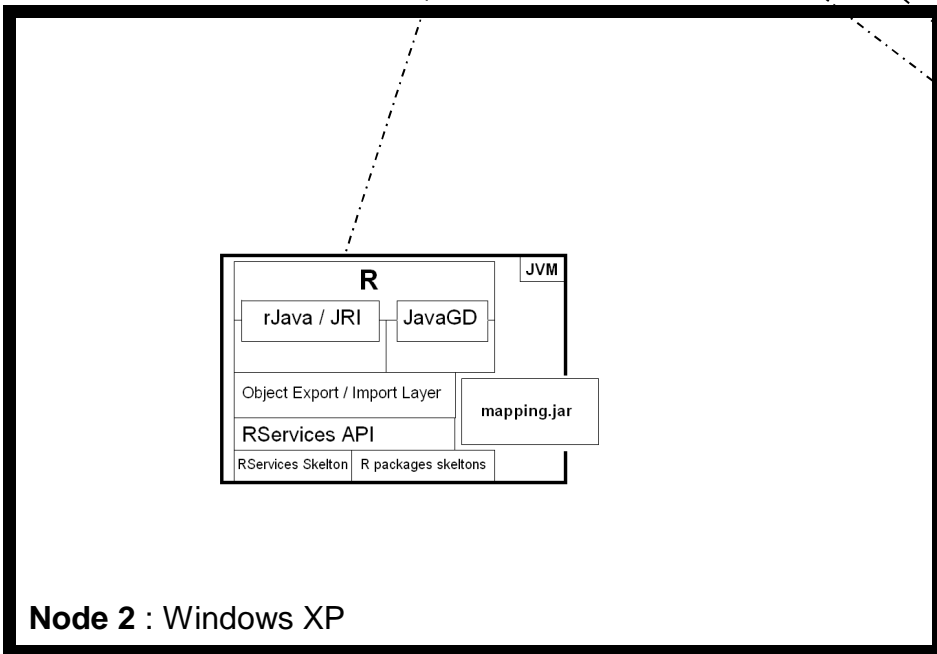
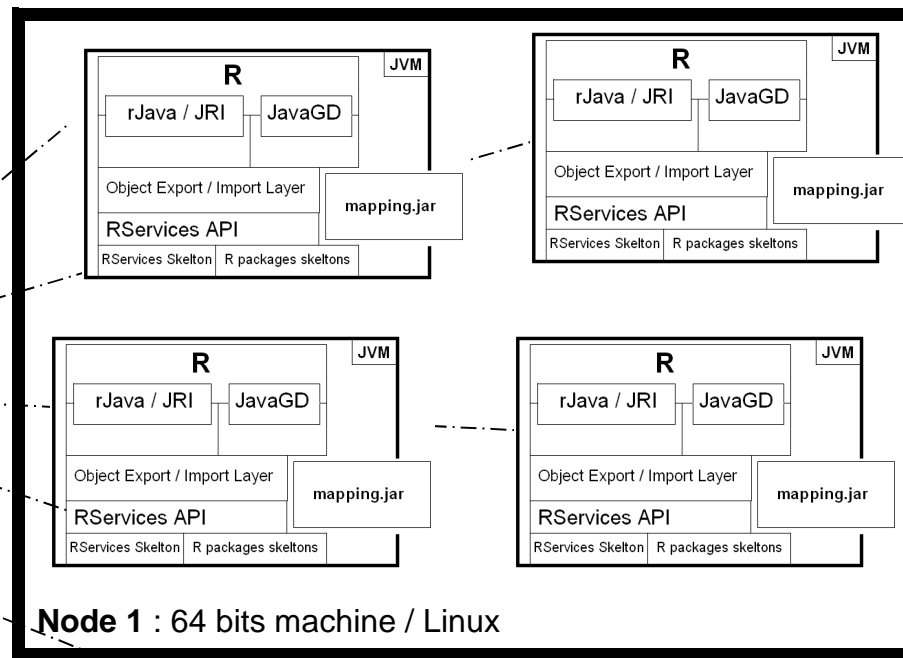
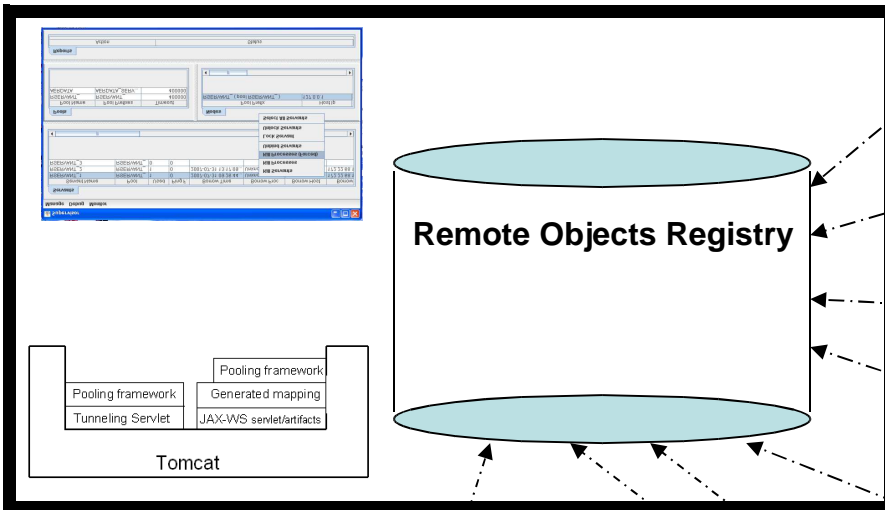
Web Browser (java plugins (applet))



Http Tunneling

SOAP

Architecture II



Parallel Computing

```
final double[][] m=..;
Future<Double>[] result=new Future[m.length];
ExecutorService exec = Executors.newFixedThreadPool(50);
for (int i=0; i<result.length; ++i) {
    final double[] v=m[i];
    result[i]= exec.submit(
new Callable<Double>() {
    public Double call() throws Exception {
        RServices r=null;
        try {
            r=(RServices)ServantProviderFactory.getFactory().getServantProvider().borrowServantProxy();
            Rnumeric mean=(RNumeric)r.call("mean", new RNumeric(v));
            return mean.getValue()[0];
        } finally {
            ServantProviderFactory.getFactory().getServantProvider().returnServantProxy(r);
        }
    }
});
}
while(true) {
    int count=0; for (int i=0; i<result.length; ++i) if (result[i].isDone()) ++count; if (count==result.length) break;
    Thread.sleep(100);
}
for (int i=0; i<result.length; ++i) System.out.println(result[i].get());
```

Web Services Generation

Script / globals.r

```
square ← function(x) {return(x^2) }  
typeInfo(square) ← SimultaneousTypeSpecification(  
TypedSignature(x = "numeric"), returnType = "numeric")
```

Script / rjmap.xml

```
<rij>  
  <publish>  
    <functions> <function name="square" forWeb="true"/> </functions>  
  </publish>  
  <scripts> <initScript name="globals.r" embed="true"/> </scripts>  
</rij>
```

ant demogen

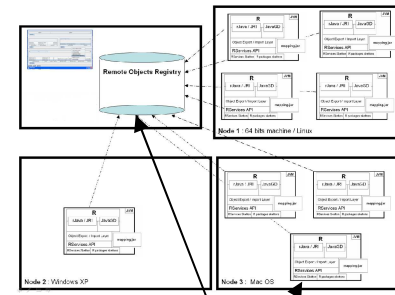
mapping.war

- + mapping.jar
- + pooling framework
- + R Java Bridge
- + JAX-WS
 - Servlets
 - Generated artifacts

Deploy

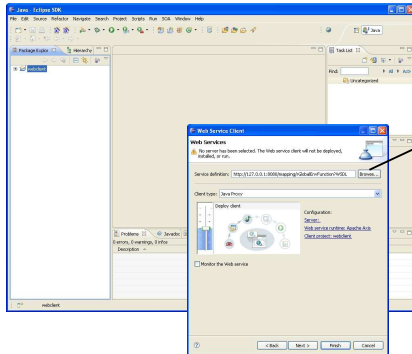
mapping.war

tomcat

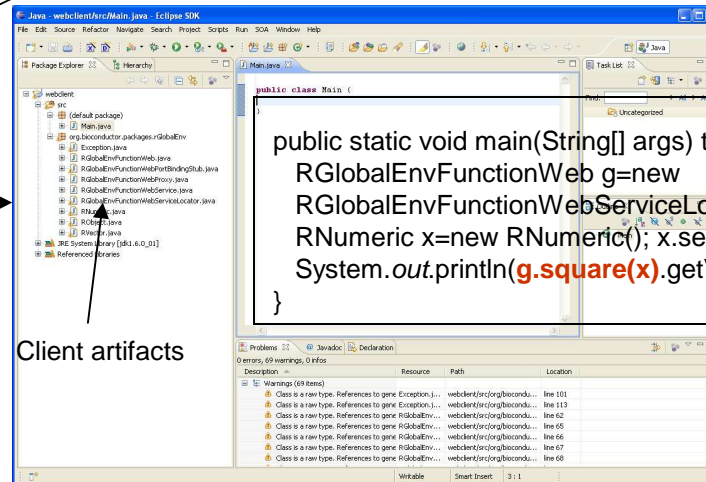


WSDL

<http://127.0.0.1:8080/mapping/rGlobalEnvFunction?WSDL>



Eclipse Web Service Client
Generator



Virtualization of R

Distributed Infrastructure Supervisor

R Rmi Servants exported log

Derby Based Naming Server

R Servants View

Used Servants

Pools View

Infrastructure Nodes View

Virtualized R Applet Running inside Firefox

Virtual R Console

Virtualized R Applet Running inside Internet Explorer

Resizable Virtual R Panel

Virtual file system

Tomcat

The screenshot displays a virtualized R environment. At the top, a 'Distributed Infrastructure Supervisor' window shows a table of servants and pools. To its right, a 'Derby Based Naming Server' window shows a command prompt running 'ant derbyrun'. Below the supervisor, a 'Virtual R Console' window shows R code and a histogram. To the right of the console, a 'Virtualized R Applet Running inside Internet Explorer' window shows a scatter plot. At the bottom left, a 'Virtualized R Applet Running inside Firefox' window shows a histogram. Arrows point from labels to these various windows and components.

Servant Name	Pool	Used	Ping F.	Borrow Time	Borrow Proc	Borrow Host	Borrow
RSERVANT_1	RSERVANT_1	1	0	2007-07-31 09:26:44...	Unknc	172.22.68.1	
RSERVANT_2	RSERVANT_1	0	0	2007-07-31 13:17:09...	Unknc	172.22.68.1	
RSERVANT_3	RSERVANT_1	0	0				

Pool Name	Pool Prefixes	Timeout
RSERVANT_1	RSERVANT_1	400000
AERDATA	AERDATA_SERV...	400000

Pool Prefix	Host Ip
RSERVANT_1 (pool RSERVANT_1)	127.0.0.1

```
done.
> kv
ExpressionSet (storageMode:
lockedEnvironment)
assayData: 8704 features, 2 samples
element names: exprs
phenoData
rowNames: green, red
varLabels and varMetadata:
channel: The scanner channel Cy3 or
Cy5
featureData
featureNames: 1, 2, ..., 8704 (8704
total)
varLabels and varMetadata: none
experimentData: use
'experimentData(object)'
Any other character(0)
> plot(exprs(kv))
> save(kv, file='kv')
```

Name	Size	Type	Last Modified
kv	148625		31 Jul 2007 13:26

Virtual R Workbench

R-Session File Tools Graphics Java Look & Feel Demos /Howtos Help

R Console

```
> my_frame <-  
data.frame(y1=rnorm(12),y2=rnorm(12),y3=rnorm(12),y4=rnorm(12)) #  
Creates data frame with vectors 1-12 and 12-1. rownames(my_frame) <-  
month.name[1:12] # Assigns row (index) names. These indices need to be  
unique. names(my_frame) <- c("y4", "y3", "y2", "y1") # Assigns new  
column titles.  
> my_frame  
      y1      y2      y3      y4  
1  0.3836018 0.19604532 1.77221268 0.7783875  
2 -0.2030605 -0.63840634 0.40200837 -1.4546609  
3 -0.1465145 0.89671370 1.01263803 -0.6763531  
4  1.8628089 1.21064146 1.54648559 -0.7255238  
5  0.5037848 0.86914381 -1.38158306 0.5583257  
6  0.2954943 -0.51745018 0.11047003 -0.2466242  
7 -0.7544302 2.33745900 0.52931483 -0.2168670  
8  1.2383185 0.85366212 0.76243685 -0.4031035  
9  0.3458652 -0.07091406 1.42797460 1.3075702  
10 -1.7789221 0.11382150 -1.51407213 0.4165589  
11  0.2027110 -0.28491206 0.85097868 -0.5768586  
12  1.1764290 -0.47201447 -0.04594193 0.7723748  
> sourcing demo 4 2D Histogram  
> save(my_frame,file='mf')
```

R Graphics

Help View

Package Index

[affy](#) Methods for Affymetrix Oligonucleotide Arrays

[affydata](#) Affymetrix Data for Demonstration Purpose

[affyio](#) Tools for parsing Affymetrix data files

[affyPLM](#) Methods for fitting probe-level models

[affyQCReport](#) QC Report Generation for affyBatch objects

[annaffy](#) Annotation tools for Affymetrix biological

Edit View Console Log Viewer

testr (J:\biocept)

```
# R Graph Gallery http://addictedtor.free.fr/graphiques/.  
layout(matrix(1:1)).  
#-----  
require(gplots) .  
# example data, bivariate normal, no correlation.  
x <- rnorm(2000, sd=4) .  
y <- rnorm(2000, sd=1) .  
# separate scales for each axis, this looks circular.  
hist2d(x,y, nbins=50, col = c("white",heat.colors(16))) .  
rug(x,side=1) .  
rug(y,side=2) .  
box() .  
#-----  
layout(matrix(1:1)) .
```

Spreadsheet View

	A	B	C	D	E	F
1			y1'	y2'	y3'	y4'
2			0.38360184	0.19604531	1.7722126	0.778
3		'1'	-0.20306051	-0.63840634	0.40200838	-1.454
4		'2'	-0.14651448	0.8967137	1.012638	-0.676
5		'3'	1.8628088	1.2106415	1.5464855	-0.7255
6		'4'	0.5037848	0.8691438	-1.3815831	0.5583
7		'5'	0.29549426	-0.51745015	0.11047003	-0.2466
8		'6'	-0.75443023	2.337459	0.5293148	-0.2168
9		'7'	1.2383184	0.85366213	0.76243687	-0.4031
10		'8'	0.34586525	-0.07091405	1.4279746	1.307
11		'9'	-1.7789221	0.1138215	-1.5140722	0.4165
12		'10'	0.20271096	-0.28491205	0.8509787	-0.5768
13		'11'	1.176429	-0.4720145	-0.0459419	0.772
14		'12'				
15						

Inspect View

my_frame

- outputMsg
- data
- rownames

File Space

Name	Size	Type	Last Mo...
mf	550		8 Nov 2...

20,1 All (rebol:none,Cp1252) - - - W 15/21 Mb

démarrer Java - bio... R & BioCo... J:\ The GIMP Layers, C... GIMP Tip o... *Untitled... FR 23:22

Acknowledgments

Misha Kapushesky

Wolfgang Huber

Alvis Brazma

Ugis Sarkans

Martin Morgan

Seth Falcon

Simon Urbanek

R O.O. Programming in a nutshell

- S4 supports : objects / classes / inheritance / polymorphism

Example

- `setClass("Point", representation(x="numeric", y="numeric"))`
- `setClass("StyledPoint", contains="Point", representation(style="integer"))`
- `setGeneric("distance", function(p1,p2) standardGeneric("distance"))`
- `setMethod("distance", signature("Point", "Point") ,
 function(p1, p2) { sqrt((p2@x-p1@x)^2 + (p2@y-p1@y)^2) })`
- `O<-new ("Point", x=12,y=2); P<-new("Point", x=52, y=90)`
- `distance(O,P)`

Introspection functions

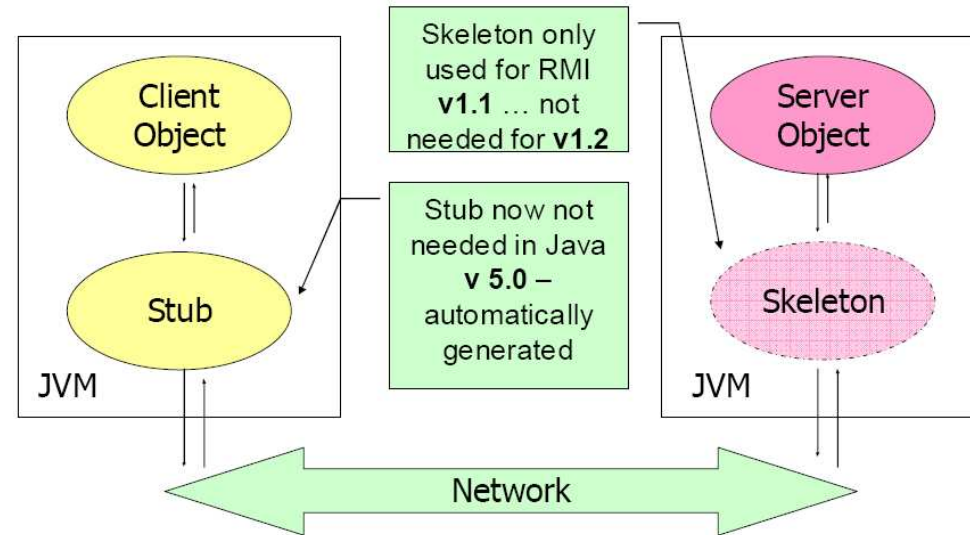
- `class(P) / getClass("Point") / getSlots("Point") / ..`
- `showMethods(class = "Point") / showMethods("distance") / isGeneric ("distance") / ..`
- `extends("StyledPoint") / ..`

Unions

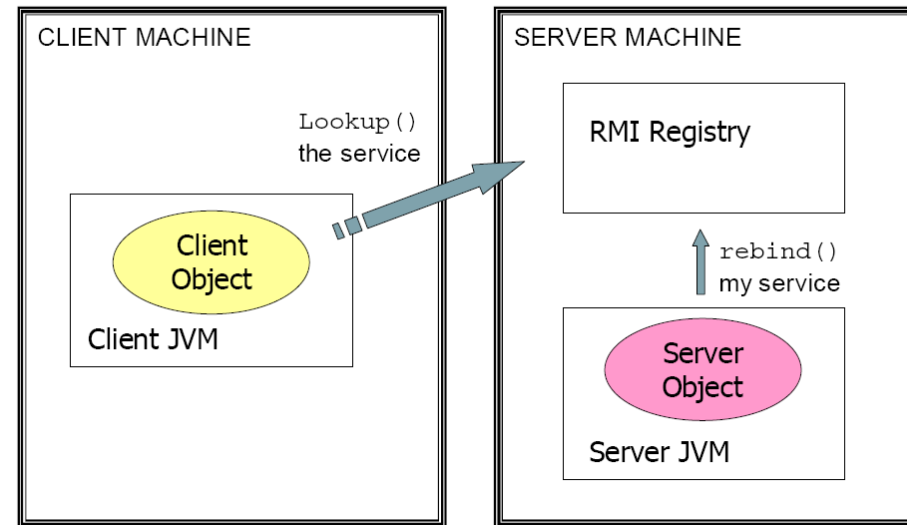
```
setClass("Egg", representation(color="integer")); setClassUnion("Thing", c("Point","Egg"));
```

RMI in a nutshell

How does RMI work ? Stubs and Skeletons



RMI Registry

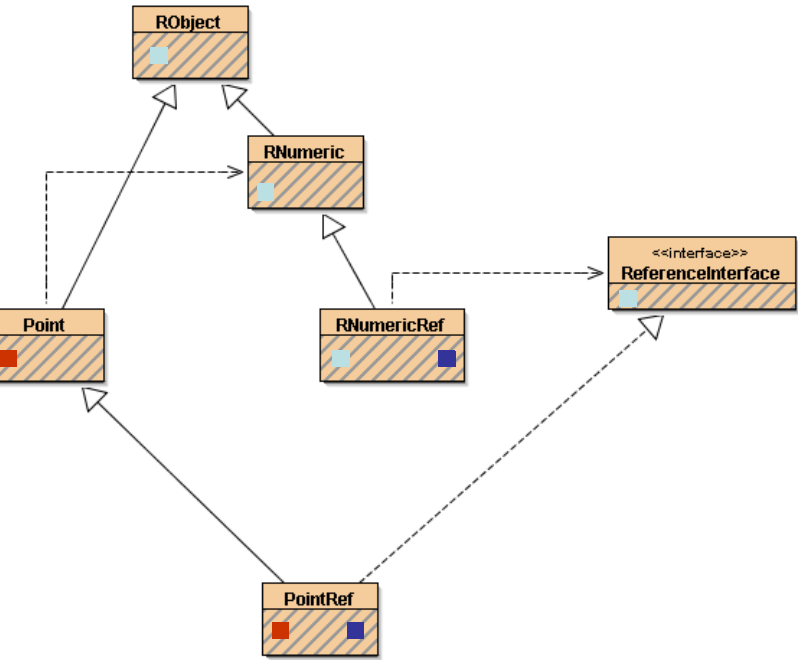
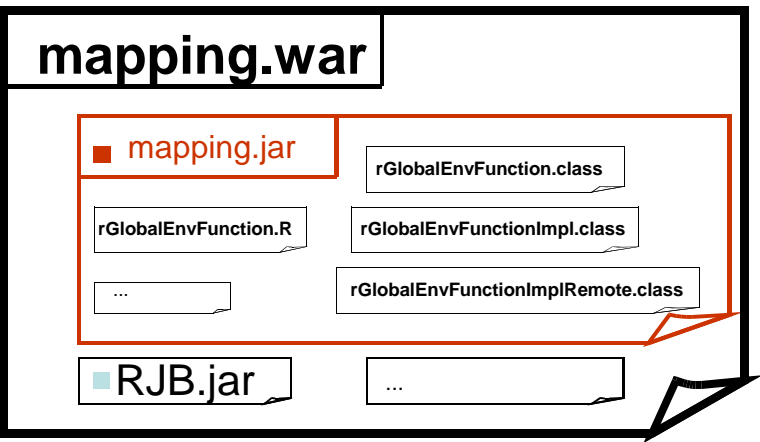


Functions Mapping

```
Script / globals.r
setClass("Point", representation(x="numeric", y="numeric"))
setGeneric("distance", function(p1,p2) standardGeneric("distance"))
setMethod("distance", signature("Point", "Point") , function(p1, p2) {
  sqrt( (p2@x-p1@x)^2 + (p2@y-p1@y)^2 ) })
```

```
Script / rjmap.xml
<rj>
<publish>
<functions><function name="distance" forWeb="true" returnType="numeric"/></functions>
</publish>
<scripts> <initScript name="globals.r" embed="true"/> </scripts>
</rj>
```

ant demogen

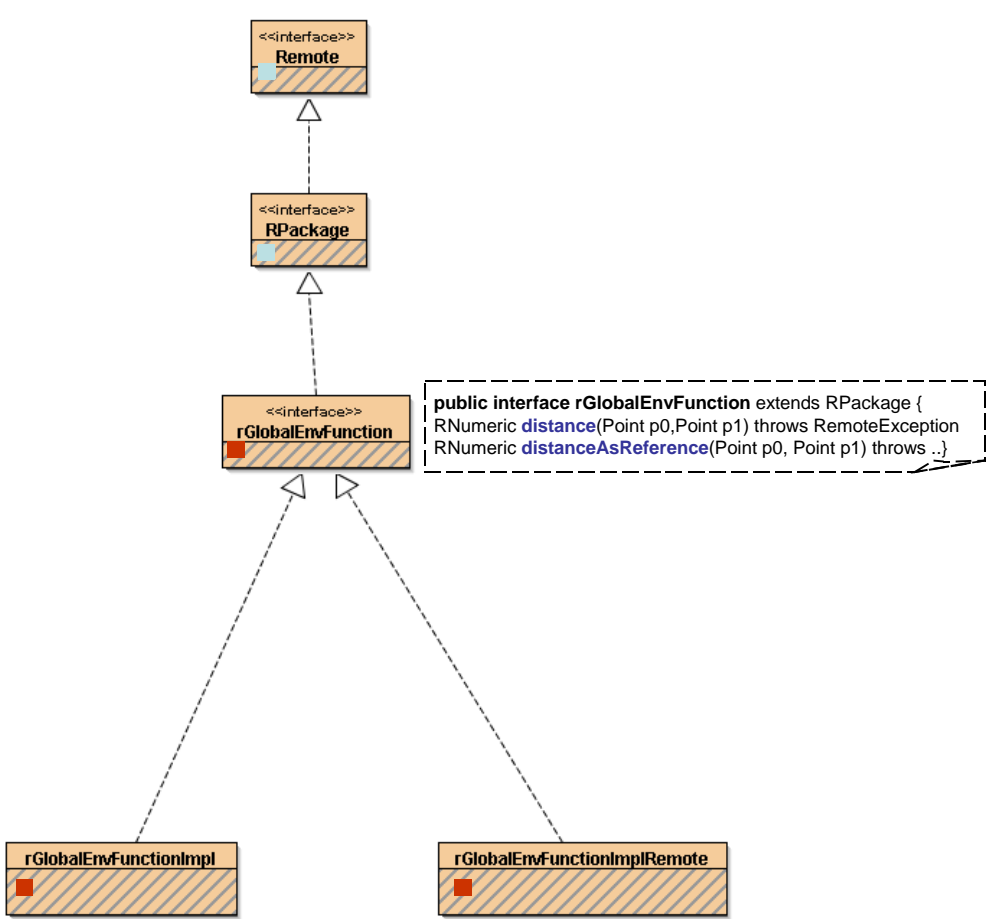


Framework class (RJB.jar)
Generated class (mapping.jar)
Proxy class

```
@WebService
public class rGlobalEnvFunctionWeb { ..
```



R Global Environment « Package » ▲ Web Implementation



▲ Local Implementation ▲ RMI Object Implementation

-

Prerequisites

1. Checkout the **EBI-Biocep** project
public svn URL : <https://svn.ebi.ac.uk/ma/branches/biocep/>
login : ma-guest
password : CN85JIZ5
(README.txt contains the instructions to run the demos)
2. **JDK 5 or 6** [Java SE 6 is required for JAX-WS Web Services]
 - JAVA_HOME environment variable set to the installation root folder of the jdk
 - PATH environment variable must include the jdk bin directory
3. **ant >= 1.7.x** installed
 - ANT_HOME environment variable set to the installation root folder of ant
 - PATH environment variable must include the ant bin directory
4. **R >= 2.5.x** installed
 - R_HOME environment variable set to the installation root folder of R
 - Bioconductor packages installed (only vsn & its dependencies for the demos)
 - **TypeInfo** package installed
 - **rJava** package installed
 - **JavaGD** package installed

if you encounter problems installing rJava, check the following :

on Unix like systems : grep JAVA \$R_HOME/Makeconf

- if R is not using the right jdk set JAVA_HOME to the path of the right one
- "R CMD javareconf"
- install rJava

5. **apache-tomcat-5.5.x** installed

- TOMCAT_HOME environment variable set to the installation root folder of tomcat

6. **apache derby 10.x** installed

- DERBY_HOME environment variable set to the installation root folder of derby

Useful links :

Java SE: <http://java.sun.com/javase/downloads/index.jsp>

ant: <http://ant.apache.org/bindownload.cgi>

R: <http://cran.r-project.org/>

BioConductor : `source("http://bioconductor.org/biocLite.R");biocLite()`

R packages : `biocLite(c('TypeInfo', 'rJava', 'JavaGD'))`

tomcat : <http://tomcat.apache.org/download-55.cgi>

derby : http://db.apache.org/derby/derby_downloads.html

Deployment of a Virtualized R infrastructure

cd biocep/VirtualRWorkbench

ant clean

ant compile

(on windows only) : **ant download.pstools**

ant demogen

ant webcompile

ant demodeploy

ant -Dspawn=true derbyrun

ant demodb

ant tomcat.startup

new terminal (cd biocep/VirtualRWorkbench) **ant demonode**

new terminal (cd biocep/VirtualRWorkbench) **ant demotop**

new browser → type the URL : <http://127.0.0.1/frontendapp/> (applet)

or <http://127.0.0.1/frontendapp/jaws> (Java Web Start)

Logon (click ok, Login Dialog)

In the console part of the R applet :

Create a normal distribution **x<-rnorm(1000)**

Plot it **plot(x)**

Draw the histogram of x **hist(x)**

Save x : **save(x,file='x_data')**

Right-click on the file and save it to your local disk

Use an editor to create an R script on your local disk : 'script.r'

Add any R commands to your script

Right click on the working directory area of the R applet and copy the script you've created there

Source your script : **source('script.r')**

Bring the supervisor window up front to have it simultaneously visible with the applet

Type **logoff**, notice the changes on the supervisor

Type **logon**, (ok) notice the changes on the supervisor

type **data(kidney)**

type **jk<-justvsn(kidney)**

plot the expression of jk

right-click on the console log area and copy the Java Serialization of jk to your disk
(jk.ser) [can be used directly by java code]

right-click on the console area and push the Java Serialization (jk.ser) to R, name the
target variable jk2

plot the expression of jk2 **plot(exprs(jk2))**

Project Home:

<http://www.ebi.ac.uk/microarray-srv/frontendapp/>

R Workbench installation link

<http://www.ebi.ac.uk/microarray-srv/frontendapp/rworkbench.jnlp>

Contacts:

Karim Chine

Email : kchine@ebi.ac.uk

Phone : + 44 (0) 1223 492 597

Misha Kapushesky

Email : ostolop@ebi.ac.uk

Phone : +44 (0)1223 494 647

