Deducer

An R data analysis GUI for everyone (including you!)

Agenda

- Motivation
- GUI tour
- Backend and plug-ins

Motivation

Researchers = Medical Doctors, Sociologists, Nurses, Psychiatrists, Biologists, Epidemiologists, Market researchers, Educators, etc

What is Deducer

- GUI Dialogs
- Data viewer
- R functions (Formatting and functionality)

Motivation: Possible solutions

- Stand alone program
  - Advantage: Can be custom tailored for the consumer group
  - Disadvantage: More work to build
  - Disadvantage: No room to grow
- R package
  - Advantage: Less code replication
  - Advantage: User can grow beyond GUI
  - Disadvantage: Difficult to integrate with all platforms/ consoles

GUI Design principles

- GUI should be as simple as possible, but no simpler.
- Organize implemented procedures by task.
- Don't cover all tasks, but covered tasks should be comprehensively covered.
- Don't restrict the user to analyzing one variable at a time

GUI Design Principles (cont.)

- Don't make the user come to you
  - Cross-platform is good
  - All consoles should be supported
- Don't hide the console or otherwise get in the way
- Help should be easy to find.
- Let the user customize and extend the GUI

Analysis Design Decisions

- All analyses should have a visualization.
- mid p-values are better than standard p-value for exact and monte carlo tests
- There is no a priori reason to believe that a linear model is Homoskedastic, and it is VERY difficult to assess the validity of this assumption.
- Type II SSQ > Type III SSQ
- You probably don't want to use a hypothesis test to detect assumption violation

rJava: Rengine

- R within Java
  - With rJava, we can take a java String and execute it as a command.
  - We can then take the result of that command and manipulate it from within java, and convert it to a standard Java object.
- Java within R
  - rJava lets us instantiate Java objects and call their methods

Setting up your package to use rJava

```
#In R
.First.lib <- function(libname, pkgname) {
        .jpackage(pkgname)
        .jengine(TRUE)
}
```

R within Java

```
import org.rosuda.REngine.REXP;
import org.rosuda.REngine.REXPMismatchException;
import org.rosuda.REngine.REngineException;
import org.rosuda.REngine.JRI.JRIEngine;

public class Deducer {
	public static REXP eval(String cmd){
		if(engine==null){
			try {
				engine = new JRIEngine(org.rosuda.JRI.Rengine.getMainEngine());
			} catch (REngineException e) {
				e.printStackTrace();
			}
		}
		try {
			return engine.parseAndEval(cmd);
		} catch (REngineException e) {return null;} catch (REXPMismatchException e) {
			return null;
		}
	}
}
```

R within Java

```
//using Deducer wrapper
String[] prestigeNames ;
REXP rVariable = Deducer.eval("names(Prestige)");
try{
        prestigeNames = rVariable.asStrings()
} catch (REXPMismatchException e) {}


//just with rJava
JRIEngine en;
try{
        en = new JRIEngine(org.rosuda.JRI.Rengine.getMainEngine());
catch(Exception e){}

double[] oneToTen;
try{
        oneToTen = en.parseAndEval("1:10").asDoubles();
catch(Exception e){}
```

R within Java: Converting REXP

Function	Converts to	Note
REXP.asInteger()	int	factors/logicals* ok
REXP.asDouble()	double	factors/integers/logical** ok
REXP.asString()	String	factors ok
REXP.asStrings()	String[]	factors ok
REXP.asDoubles()	double[]	factors/integers/logical** ok
REXP.asBytes()	byte[]	
REXP.asIntegers()	int[]	factors/logicals* ok
REXP.asList().at(int index)	REXP	for R lists and data.frames

*NA = -2147483648

**NA=Double.*longBitsToDouble(0x7ff00000000007a2L);*

Executing a command as if the user typed it in

```
//in Java

//using Deducer
Deducer.execute("print('I like ponies')");

//If JGR is running, this is the same as
JGR.execute("print('I like ponies')");

//otherwise, Deducer fakes it using:
String cmd ="print('I like ponies')";
Deducer.eval(".deducerExecute(\""+Deducer.addSlashes(cmd)+"\\n\")");

#in R
.deducerExecute<-function(cmd){
   cmds<-parse(text=cmd)
   for(i in 1:length(cmds)){
      out<-eval(parse(text=paste("capture.output(",as.character(cmds[i]),")")),globalenv())
      for(line in out)
         cat(line,"\n")
   }
}
```

Java within R

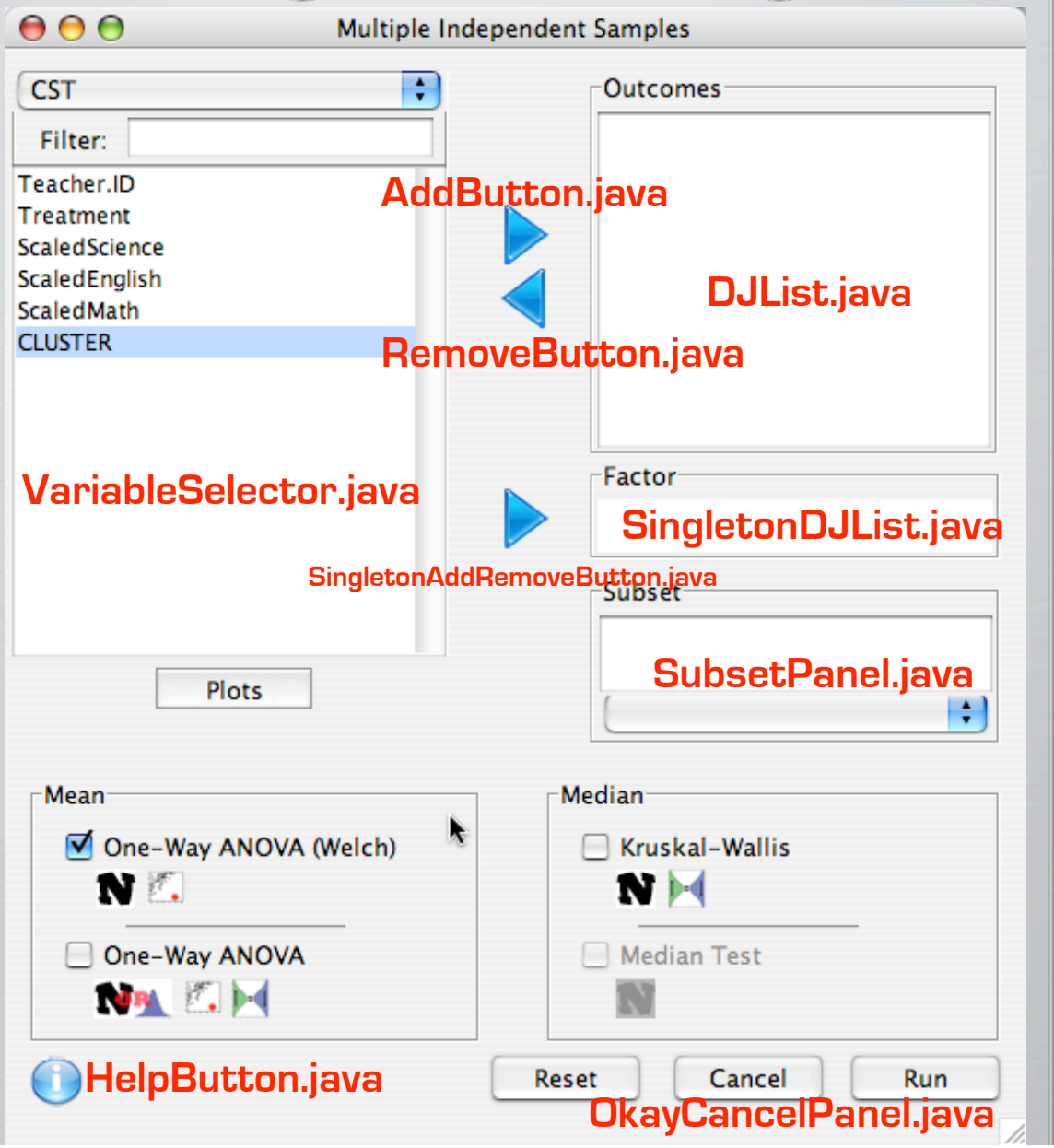
```
> #create a new Java object
> f <- .jnew("java/awt/Frame","Hello")
> f
[1] "Java-Object{java.awt.Frame[frame5,0,22,0x0,invalid,hidden,
layout=java.awt.BorderLayout,title=Hello,resizable,normal]}"

> #call the method setVisible(true)
> .jcall(f,"V","setVisible",TRUE)

> #Calling a static method
> os <- .jcall("java/lang/System","S","getProperty","os.name")
> paste(os,"is much better than the other operating systems")
[1] "Mac OS X is much better than the other operating systems"
```

Writing a plug-in

Steps:

1. Write Java dialog
2. Compile into a JAR file building off of deducer.jar, jri.jar, and jgr.jar
3. Create a menu item that launches the dialog

Step 1: Writing the dialog



Step 2: Compile into jar file

Compiling is simple and standard. use something like:

```
javac –classpath jri.jar:deducer.jar:jgr.jar plugin/NewDialog.java
jar fc $@ plugin
```

Then put your jar file into the java directory of your package

Step 3: Set up menu item

```
.jpackage(“plugin”)

menuCall <- “.jnew(‘plugin/NewDialog’)”
#add a menu
deducer.addMenu("TestMenu")
deducer.addMenuItem("test1",,menuCall,”TestMenu”)

#Add menu to gui if applicable
if(.windowsGUI){
        winMenuAdd("TestMenu")
        winMenuAddItem("TestMenu", menuCall)
}else if(.jgr){
        jgr.addMenu("TestMenu")
        jgr.addMenuItem("TestMenu", "test1", menuCall)
}
```

Future work

psst! This is where you come in

- Plots
- Multivariate/Longitudinal analysis
- Classroom dialogs

Why work on this:

1. Gain experience with R and Java
2. publications (JSS, R Journal)
3. Create tools that are (hopefully) widely used
4. Influence the way statistics is practiced and make it easier