**RHook for MeV: documentation**

**@author: Raktim Sinha**

**@current R version supported: R-2.11**

**@current modules supported:**

**limma:**

**attract:**

**minet:**

**sam:**

**survival:**

**globalanc:**

**edger:**

**deseq:**

**goseq:**

**degseq**

**@date: Monday, June 27, 2011**

**INDEX**

1. [**Components of the RHook framework & how they interact**](#Compo_1)
   1. [***Dynamic Libs (dlls/libjri)***](#Compo_1_1)
   2. [***JRI.jar***](#Compo_1_2)
   3. [***RHook MeV library API***](#Compo_1_3)
   4. [***R install for Win & Mac***](#Compo_1_4)
   5. [***RHook FTP base URL***](#Compo_1_5)
   6. [***RHook property file on the ftp site***](#Compo_1_6)
   7. [***Ftp directory structure for OS/module/pkgs***](#Compo_1_7)
   8. [***RBinaries and their location on ftp***](#Compo_1_8)
   9. [***RHook URL for MeV & class location***](#Compo_1_9)
   10. [***RHook OS specific packaging support folders (part of SVN)***](#Compo_1_10)
   11. [***TMEV.bat/sh entries***](#Compo_1_11)
2. [**Adding a new RHook module**](#New_Mod_2)
3. [**Upgrading to a new R version**](#Uograde_R_3)
4. [**Setting up the Windows development environment for MeV RHook**](#Dev_Env_4)
5. [**How to check for R package dependencies**](#R_Pkg_Depnd_5)
6. [**Getting/Creating JRI libs & JRI.jar**](#Get_Libs_6)
   1. [***Building from source***](#Get_Libs_6_1)
   2. [***Extracting from pre-built rJava binary archives***](#Get_Libs_6_2)
7. [**Making release bundles for all OS**](#Release_7)
   1. [**Win**](#Release_7_1)
   2. [**Mac**](#Release_7_2)
8. **Components of the RHook framework & how they interact** 
   1. ***Dynamic Libs (dlls/libjri)***

The dynamic library is the C library that lets MeV (java) to hook in to R using JNI.

There should be a separate library for each of the different platform. The documentations and the details are at the rJava site: <http://www.rforge.net/rJava/files/>

The os specific libs exists in the **{OS}\_distribution\_support** directory of the SVN source and are copied to **lib** directory in windows & to the **MeV.app\Contents\Resources\Java\libjri.jnilib** in Mac, for release bundles.

Please refer to the “[**Getting/Creating JRI libs**](#Get_Libs_6)” sections for further details on how to generate them.

* 1. ***JRI.jar***

The java JRI Api provided by rJava/JRI group to make R command calls.

The JRI.jar is part of the SVN source & exists in the **lib** folder.

There should be a classpath entry for this jar in the ANT target for a R based module.

Please refer to the “[**Getting/Creating JRI libs**](#Get_Libs_6)” sections for further details on how to generate them.

* 1. ***RHook MeV library API***

MeV RHook API is a wrapper around JRI functionality that puts the components together, instantiates R and provides API functionality that allows algorithm module to pass data back and forth and execute R commands.

The API is in the MeV package ***org.tigr.rhook.\****.

* 1. ***R install for Win & Mac***

Get R from CRAN. The URLs for previous 3 versions or more of the R binaries, as of this date are as follows:

Win: <http://cran.r-project.org/bin/windows/base/old/>

Mac: <http://cran.r-project.org/bin/macosx/old>

* 1. ***RHook FTP base URL***

Ideally URL #1 should be used for development & URL #2 for production but any one of the 2 URLs can be used for the a special release as long as they in sync. Before a release happens both URL should be in sync.

From here on in the doc, the base URL will be referred to as **{RHook\_URL}**

1. <ftp://occams.dfci.harvard.edu/pub/bio/MeV_Etc/R_MeV_Support_devel/>
2. <ftp://occams.dfci.harvard.edu/pub/bio/MeV_Etc/R_MeV_Support/>
   1. ***RHook property file on the ftp site***

{RHook\_URL}/rhook.txt

MeV RHook client uses this property file to figure out among other things the R version currently being used, the supported OS architectures, the R modules, the dependencies of each R modules etc.

This file needs to be updated whenever any of the above mentioned properties change.

* 1. ***Ftp directory structure for OS/module/pkgs***

{RHook\_URL}/R{major-ver}/win/{module\_name}\*

{RHook\_URL}/R{major-ver}/mac/{module \_name}\*

**Naming Convention:**

R{major-ver} e.g.: R2.11

{module \_name} e.g.: folder(s) for each module goseq, deseq etc. **NOTE**: This folder name must match the module name used as key in the rhook.txt property file.

Once the MeV RHook client reads out the dependencies of a module it then uses this location(s) to download any package(s) that needs to be installed before executing an R algorithm.

* 1. ***RBinaries and their location on ftp***

<ftp://occams.dfci.harvard.edu/pub/bio/MeV_Etc/RBinaries/>. This URL is used by MeV build script to download zipped binaries required for making Windows release bundles.

Win RBinaries are created simply by zipping up the R install directory, without installing additional libraries & packages other than what comes with the base installer. For example, if we want to create the RBinary for R-2.11.1, we first install R from CRAN, say in the folder **C:\Program Files\R\R-2.11.1**. Then we just zip –up the folder R-2.11.1 and upload it to the ftp location.

A lot of things can be excluded from the binary archive to make it smaller, but that is not in the scope of this discussion and is not recommended for users who are not familiar with the R components.

* 1. ***RHook URL for MeV & class location***

***org.tigr.rhook.RConstants.java*** uses a static variable - ***RHOOK\_BASE\_URL*** which points to **{RHook\_URL}**.

Switching between development & production URL is done by changing the variable in this file.

* 1. ***RHook OS specific packaging support folders (part of SVN)***

win\_distribution\_support\RPackages

win\_distribution\_support\lib\jri.dll

mac\_distribution\_support\RPackages

mac\_distribution\_support\MeV.app\Contents\Info.plist

mac\_distribution\_support\MeV.app\Contents\PkgInfo

mac\_distribution\_support\MeV.app\Contents\MacOS

mac\_distribution\_support\MeV.app\Contents\Resources\GenericJavaApp.icns

mac\_distribution\_support\MeV.app\Contents\Resources\Java\libjri.jnilib

* 1. ***TMEV.bat/sh entries***

The following are RHook specific entries in the tmev file:

1. ***JAVA\_HOME*** : Set locally only for 64Bit Windows
2. ***CurrDIR*** : MeV installed dir; required to set paths to R bin, lib dirs & R\_HOME
3. ***PATH*** : Modified locally to add JAVA\_HOME if required and to set R bin & lib
4. ***R\_HOME*** : Fully qualified path to R bundled with MeV
5. ***-Djava.library.path*** : java flag to location of JNI dynamic library, which is the lib folder in MeV root directory.

***Mac tmev:*** For Mac users a separate script file called ***tmev\_mac.sh*** is included in the MeV root folder to enable launching from a terminal.

1. **Adding a new module**
2. Get R pkg list using R(BioC)
3. Write R script to make module work in R with sample data.
4. Add/Edit the following in RHook property files
   1. ***r\_modules*** :Add the new module {name} in the end separated by “:”.

E.g.: ***r\_modules=limma:edger***

* 1. Add a section(s) for the new module that lists packages for os/R ver/arch in the form of: ***{name}\_{os}\_{R-version}\_{arch}*** initialized to pkg names separated by “:”. E.g.: ***edger\_win\_2.11\_32=limma\_3.4.5.zip:edgeR\_1.6.13.zip.*** There should be a separate line of entry for each unique module/os/R/arch combination.

1. Update Ftp repository with packages:
   1. Create ***{RHook\_URL}/{R-version}/{os}/{name}*** structure. Note, that web directory paths are based on the section naming convention used in (b) and should match exactly. E.g.: the ***edger*** module example in (b) should have the following directory structure under the ftp base directory: ***{RHook\_URL}/R2.11/win/edger***. There should be at least 2 separate path Win & Mac for each module.
   2. Upload R packages for the module in the win & mac directories created above. The packages should be exactly as listed in the Rhook file, as described in 2->3->b. The number of packages and their names should match exactly.
2. Add java code in MeV based on the R script
3. Add new ANT target(s) in build script, a template is provided here, where the module name is “**MINET**”. Please note the inclusion of JRI.jar which is specific to R module.

Algorithm Target:

<target name="MINET" depends="MINET-GUI" if="MINET">

<javac debug="${debug}" target="${java.target.version}" sourcepath="" srcdir="${alg.impl.dir}" destdir="${dest.dir}">

<include name="MINET.java"/>

<classpath>

<pathelement location="${lib.dir}/JSciCore.jar"/>

<!-- jars to support module compilation -->

<pathelement location="${lib.dir}/mev-util.jar"/>

<pathelement location="${lib.dir}/mev-gui-impl.jar"/>

<pathelement location="${lib.dir}/mev-gui-support.jar"/>

<pathelement location="${lib.dir}/mev-algorithm-impl.jar"/>

<pathelement location="${lib.dir}/mev-algorithm-support.jar"/>

<pathelement location="${lib.dir}/mev-base.jar"/>

**<pathelement location="${lib.dir}/JRI.jar"/>**

</classpath>

</javac>

<propertyfile file="${alg.properties.file}">

<entry key="MINET" value="org.tigr.microarray.mev.cluster.algorithm.impl.MINET"/>

</propertyfile>

</target>

GUI Target:

<target name="MINET-GUI">

<javac debug="${debug}" target="${java.target.version}" srcdir="${gui.impl.dir}/minet" destdir="${dest.dir}">

<classpath refid="module.build.class.path"/>

</javac>

<propertyfile file="${gui.properties.file}">

<entry key="gui.names" value="MINET:" operation="+"/>

<entry key="MINET.name" value="MINET"/>

<entry key="MINET.class" value="org.tigr.microarray.mev.cluster.gui.impl.minet.MINETGUI"/>

<entry key="MINET.category" value="${STATISTICS}"/>

<entry key="MINET.smallIcon" value="analysis16.gif"/>

<entry key="MINET.largeIcon" value="minet\_button.gif"/>

<entry key="MINET.tooltip" value="Mutual Information Network"/>

</propertyfile>

</target>

1. **Upgrading to a new R version**
2. Verify pkg lists and pkg versions for all modules in new R for all OS; See Special Note section below.
3. Creating new RHook property file (copy of the last version)
4. Add/Edit the following in the new property file:
   1. ***cur\_r\_ver*** & ***cur\_mac\_r\_ver*** to reflect the new R version
   2. Update ***r\_modules*** with new module names (if any)
   3. Add a new line to reflect package dependencies for each module/new R ver/os/arch combination in appropriate section.
5. Create test ***{RHook\_URL}***t URL for MeV (normaly would the last devel URL)
6. Populating test ***{RHook\_URL}*** with new property file and directory structure & packages.
7. Uploading new RBinaries to ftp location.
8. Get/Create new dynamic libraries & JRI.jar
9. Update the local MeV source folder “**lib**” with new ***JRI.jar*** in correct locations.
10. Update the local ***win\_distribution\_support\lib*** with new ***jri.dll***
11. Update the local ***mac\_distribution\_support\MeV.app\Contents\Resources\Java*** with new ***libjri.jnilib***
12. Modify the Rhook specific components in TMEV.bat/sh files
13. Test modules
14. Commit code with new dll & JRI.jar
15. **Setting up the Windows development environment for MeV-RHook**

For setting up the development environment some of the things need to be done ***by hand***. Once the source has been checked out, do the following:

1. Copy the Dynamic Lib (dll) in MeV ***lib*** dir from ***win\_distribution\_support\lib\jri.dll***
2. Create “***RPackages***” folder in MeV root directory
3. Unzip correct R from ***RBinaries location on ftp (see section)*** into a folder where the naming convention is **R-x.xx.x.** Basically the R folder name should match the name used in TMEV.bat file to initialize the variable **R\_HOME** to point to R’s location.
4. **How to check for R package dependencies**
5. Start with a blank install of R
6. Try to install a package & then load the package.
7. Note all the dependencies it installs
8. The package name & all its dependencies should go in the RHook.txt web property file under the key matching the module name.
9. **Getting/Creating JRI libs & JRI.jar**
   1. ***Building from source:***

The latest source files can be found here: <http://www.rforge.net/rJava/files/>

Instructions are here to build the library for each platform: <http://www.rforge.net/JRI/>

* 1. ***Extracting from pre-built rJava binary archives***:

JRI currently is not released separately anymore but bundled with rJava. The URL <http://www.rforge.net/rJava/files/> also has prebuilt archives of rJava(with JRI) for the last 2 versions of R. One can extract the platform specific dynamic libraries & JRI.jar from them. The “**rJava/jri**” folder in the archive will have both the components.

1. **Making release bundles for all OS**
   1. **Win:**

The ant target ***build-package-win***should produce the Windows **.zip** release bundle in the **{MeV\_dir}/deploy** directory.

Making exe ???

* 1. **Mac:**

**NOTE**: Should be run on a Mac or Linux box.

The ant target ***build-package-mac*** should produce the directory structure and MeV contents under **{MeV\_dir}/deploy** directory. The top level directory should have name of the form ***MeV\_x\_x\_x***. Do the following to produce the bundle:

* + - * 1. Modify **{MeV\_Dir}/source/org/tigr/microarray/build.properties**
        2. Run target ***build-package-mac***
        3. Open Terminal (Search for terminal in Finder)
        4. **cd** **{MeV\_dir}/deploy**
        5. **tar -czvf MeV\_whatever\_bundle\_name.tgz {*MeV\_x\_x\_x}/***
        6. Move the produced .**tgz** file to desired location
  1. **Linux:**

**NOTE**: Shoul;d be run on a Mac or Linux box.

The ant target ***build-package-linux*** should produce the directory structure and MeV contents under **{MeV\_dir}/deploy** directory. The top level directory should have name of the form ***MeV\_x\_x\_x***. Do the following to produce the bundle:

* + - * 1. Modify **{MeV\_Dir}/source/org/tigr/microarray/build.properties**
        2. Run target ***build-package-linux***
        3. Open Terminal
        4. **cd** **{MeV\_dir}/deploy**
        5. **tar -czvf MeV\_whatever\_bundle\_name.tar.gz {*MeV\_x\_x\_x}/***
        6. Move the produced .**tar.gz** file to desired location