Using the DDG Explorer

Barbara Lerner, Emery Boose

December 2013

What is the DDG Explorer?

The DDG Explorer is a tool that allows the user to view and query the Data Derivation Graphs (DDGs) that are created by running programs that collect this data derivation information as they execute. The focus of this document is on its use with DDGs created from execution of instrumented R scripts. Currently, DDGs can also be created through execution of Little-JIL processes. The DDG notation is general enough to support many languages, but there are currently no other implementations.

DDG Explorer has the following functionality:

- Visualization of DDGs, with the ability to zoom in and out to selectively show or hide details.
- Ability to view the data or R functions referenced by pieces of the DDG
- Ability to query a DDG to discover how an input data value gets used, or what data and processing steps lead to the derivation of a particular output value
- Ability to compare R scripts used to generate different DDGs
- Ability to search for where a particular data file is used or generated.

An overview of the project that DDG Explorer was developed in is available at http://www.mtholyoke.edu/~blerner/DataProvenance/.

Downloading DDG Explorer

The DDG Explorer software can be downloaded from:

http://www.mtholyoke.edu/~blerner/DataProvenance/code/ddg-explorer.jar

Required software:

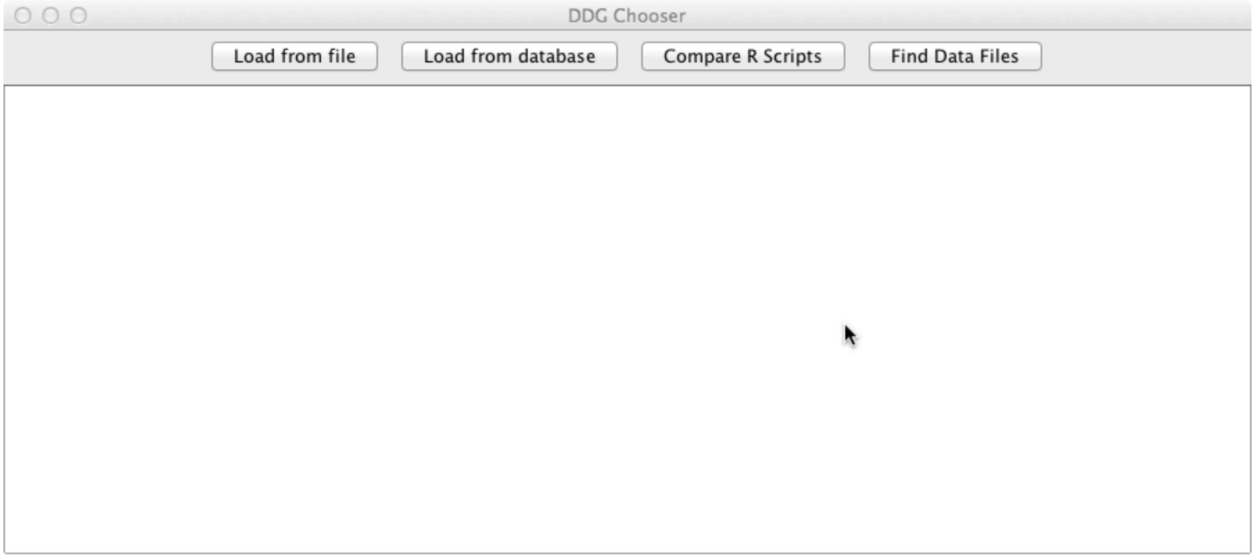
- DDG Explorer requires Java 1.7. Most computers come with Java installed, but if you do not have it, you can download it from http://www.oracle.com/technetwork/java/javase/downloads/index.html. Be sure to select **JRE** and then the version appropriate for your operating system.

Starting the DDG Explorer

After downloading DDGExplorer.jar, you should be able to start it by double-clicking on the icon.

Loading a DDG from a File

When DDG Explorer starts, you should see a window that looks like this:



Across the top of this window, you should see 4 buttons:

- Load from file
- Load from database
- Compare R scripts
- Find Data Files

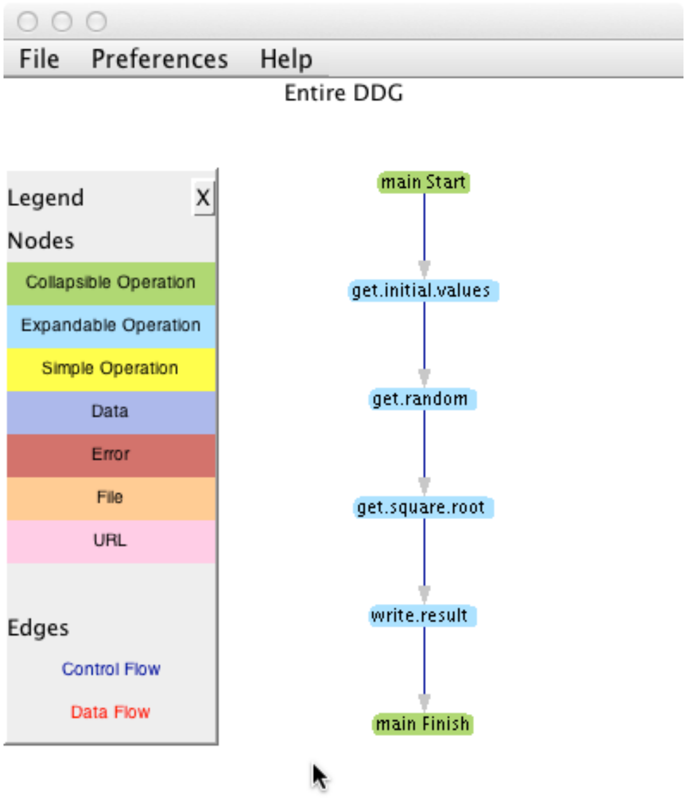
The large empty area is used to display error messages or other informational messages as necessary.

The DDG Explorer stores its data in a database (details in the Behind the scenes section below if you are interested). The Load from Database, Compare R Scripts, and Find Data Files buttons only do useful things once there are DDGs in the database. To get data into the database, you must first load it from a file and then save it to the database.

When you click on Load from File, you will be presented with a standard file browser. Exactly how this file browser looks will depend on the operating system that you are using. Using the file browser, you should navigate to a ddg.txt file, select that and click Open. The ddg.txt is located in the DDG directory used by the R script. The location of the directory is defined by the ddg.path variable defined at the top of the R script.

The DDG window

After opening a ddg.txt file, you will see a window that looks like this:



Along the top of the window are 3 menus: File, Preferences and Help. On the left side is a legend that explains the colors used in the DDG. On the right is the initial view of the DDG in a collapsed state.

A DDG is drawn as a number of nodes (the oval shapes) connected with edges (the arrows). The nodes represent either data or processing steps, while the edges show how execution goes from one processing step to the next, or how data is used or produced by a processing step.

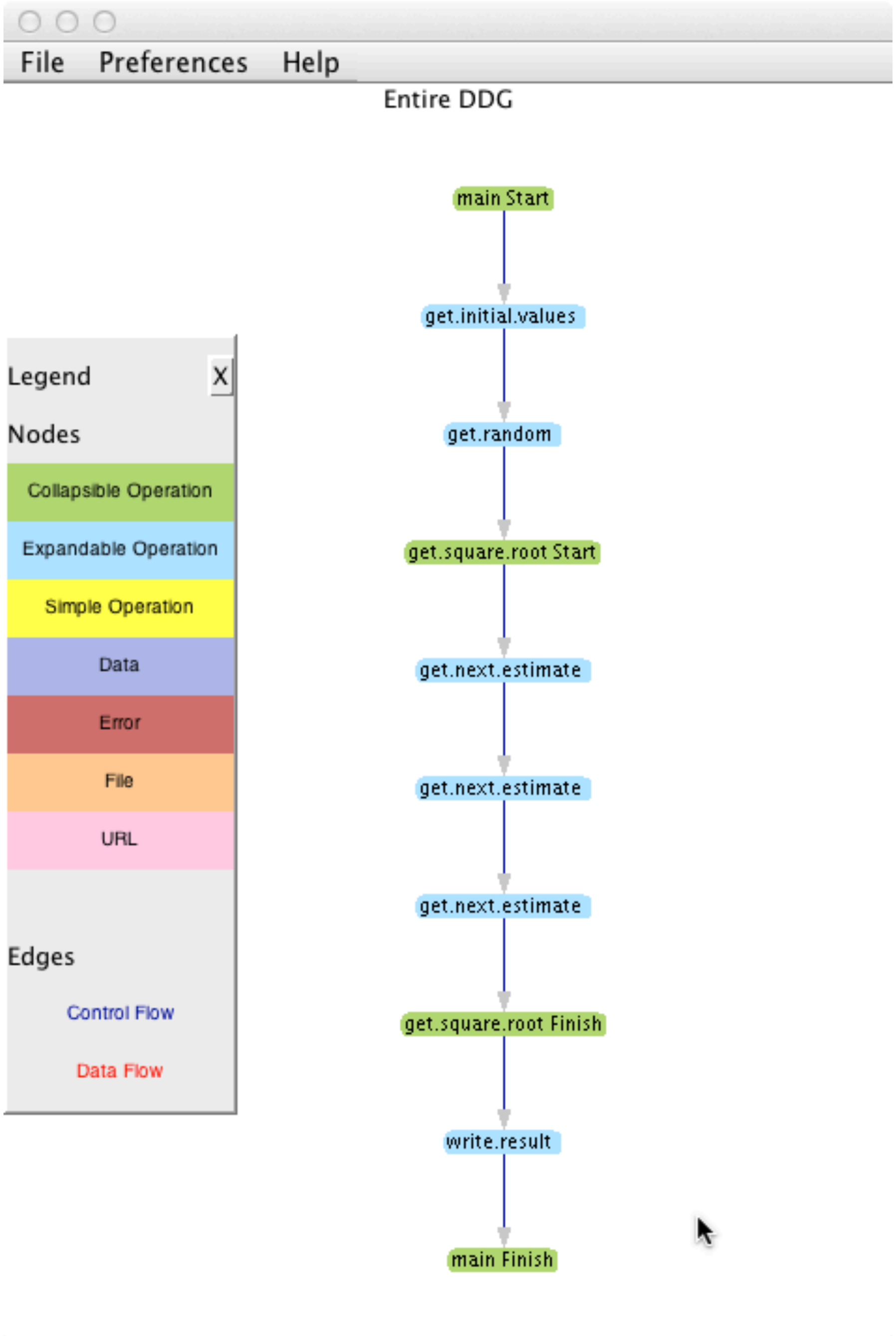
For example, the DDG as drawn above corresponds to the following pieces of R code:

```
get.initial.values()
estimate <- get.random(number)

# Begin get.square.root
check <- number
while (check > 0) {
  # repeat calculation until tests OK
  estimate <- calc.square.root(number,estimate)
  difference <- get.difference(number,estimate)
  check <- get.check.value(difference,tolerance)
}
sqr.root <- store.result(number,estimate)
# End get.square.root

write.result("sqr-root.csv",sqr.root)
```

The exact nodes that are drawn are based on the instrumentation that was placed in the code. (See the *Using the R DDG Library* document to learn how to do this best.) Here is the same code with the instrumentation added.

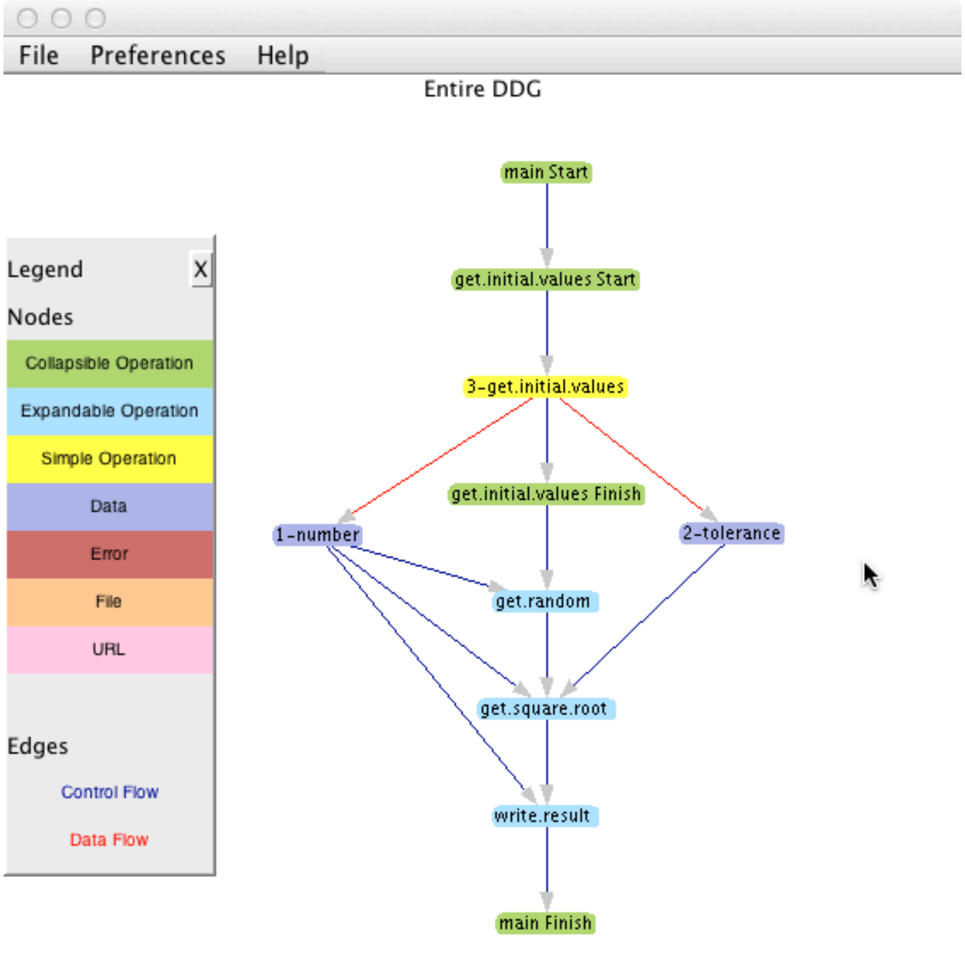
```
ddg.start("main")

ddg.start("get.initial.values")
get.initial.values()
ddg.finish("get.initial.values")

ddg.start("get.random")
estimate <- get.random(number)
ddg.finish("get.random")

ddg.start("get.square.root")

check <- number

while (check > 0) {
  ddg.start("get.next.estimate")

  # repeat calculation until tests OK
  estimate <- calc.square.root(number,estimate)
  difference <- get.difference(number,estimate)
  check <- get.check.value(difference,tolerance)

  ddg.finish("get.next.estimate")
}

ddg.finish("get.square.root")

ddg.start("write.result")
sqr.root <- store.result(number,estimate)
write.result("sqr-root.csv",sqr.root)
ddg.finish("write.result")

ddg.finish("main")
```

Each ddg.start()...ddg.finish() pair identifies a number of nodes that can be collapsed into a single node. This allows the user to zoom in and out on the detail. In the initial drawing, the “main” node is expanded (resulting in a pair of green nodes labeled “main Start” and “main Finish”), while the remaining nodes are “collapsed” (drawn in blue). The user can single-click on a blue node to expand it. For example, if the user clicks on the get.square.root node, the DDG is redrawn as:



Single-clicking on either the get.square.root Start node or the get.square.root Finish node would return to the original view of the DDG. To see the entire DDG, click on the top Start node to collapse the DDG to a single node. Then right-click and select "Expand All".

A Simple Operation (drawn as a yellow node) represents a processing step that the user cannot zoom in on. It represents the lowest level of processing captured by the DDG. Below is a view of the DDG where we have expanded the get.initial.values step. Here you can see a simple operation, named "3-get.initial.values". We also see 2 lilac-colored nodes. These nodes represent data. In this case, the get.initial.values operation is setting a variable named "number" and another variable named "tolerance". We can tell that get.initial.values is setting these values since the arrow points from the Operation node to the Data nodes. The number variable is used in the operations get.random, get.square.root and write.result. Similarly, tolerance is used by get.square.root. Here, we can tell that these operations are using the data because the arrows point from the data to the operations, signifying that the data are inputs to those operations.



There are three additional node types: File, URL and Error. The lilac color indicates a data value stored in memory, either a simple value like a number or character, or a more complex value like a vector or data frame. A File node signifies that the data are stored in a file, while a URL node indicates that the data came from a website. An Error node (in red) represents an error during execution of the R script that caused the script to fail.

Scrolling, magnifying, clicking, right-clicking

As mentioned above, the user can click on green nodes to collapse them and on blue nodes to expand them. There are some other simple operations that can be done using the mouse.

Left mouse click	On a green node, collapse. On a blue node, expand.
Left mouse down and Drag	Drag a node to move it. Drag on the background to pan.
Control + Right mouse down + drag (Windows) Command-Drag (Mac)	Magnify or shrink the entire DDG
Right-click (Windows) Control-click (Mac)	Pulls up a menu whose contents depend on the type of node. • On a blue node, the user can expand the current node (same as a click), expand that node to complete detail, or view the R function if this node corresponds to a function. • On a green node, the user can collapse (same as a click) or see the corresponding R function if the node maps to a function. • On a yellow node, the user can see the function definition if the node corresponds to an R function. • On a lilac, beige or pink node, the user can see the data value, file contents or URL contents. • On a red node, the user can see the error message.

The Help menu contains a Command Overview command that provides the same information as the table above.

After gaining a little familiarity with DDGs, you may find it convenient to remove the Legend, which you can do by clicking on the X in its upper-right corner. You can display the legend again, if you like, by using the "Show Legend" command in the Preferences menu.

Menu commands

File menu

The File menu contains 3 commands:

- Show attributes: The Show Attributes command displays a window that contains basic metadata about the R script that was executed. The attributes shown are:
  - Architecture - this identifies the type of processor that the script was executed on, such as x86_64.
  - Operating System - the operating system that the script was executed on. Note that this will report "Unix" when run on a Mac.
  - Language - the language that the script was written in. Currently, DDG Explorer supports R and Little-JIL.
  - Script - the full path to the file containing the script
  - Script Timestamp - the date and time that the script was last modified

    - Working Directory - the directory in which the script was executed
    - DDG Directory - the directory in which the ddg was stored
    - DDG Timestamp - the time at which the script was executed to create the current DDG.
- Show R script - this will bring up a window that will display the entire R script that was executed
- Save to database - this will save the current DDG along with the datafiles cached in the ddg directory to a database. Saving to the database allows other features, like searching. Please see the section below titled "Using the DDG database" for more details of this functionality.

Preferences menu

The Preferences menu has two options:

- Draw arrows from inputs to outputs - This option allows the user to control the direction of arrow heads. When drawn from inputs to outputs, arrows are generally downward-pointing and go in the order of execution. When drawn from outputs to inputs, the arrowheads denote what output data was derived from and are generally drawn upward. The default is to draw arrows from input to output.
- Show legend - Initially, the legend is drawn. The user can remove the legend either by clicking the X in the top right of the legend, or by deselecting "Show legend" in the preferences menu. If the legend is not showing and the user would like to see it, selecting "Show legend" will cause it to reappear.

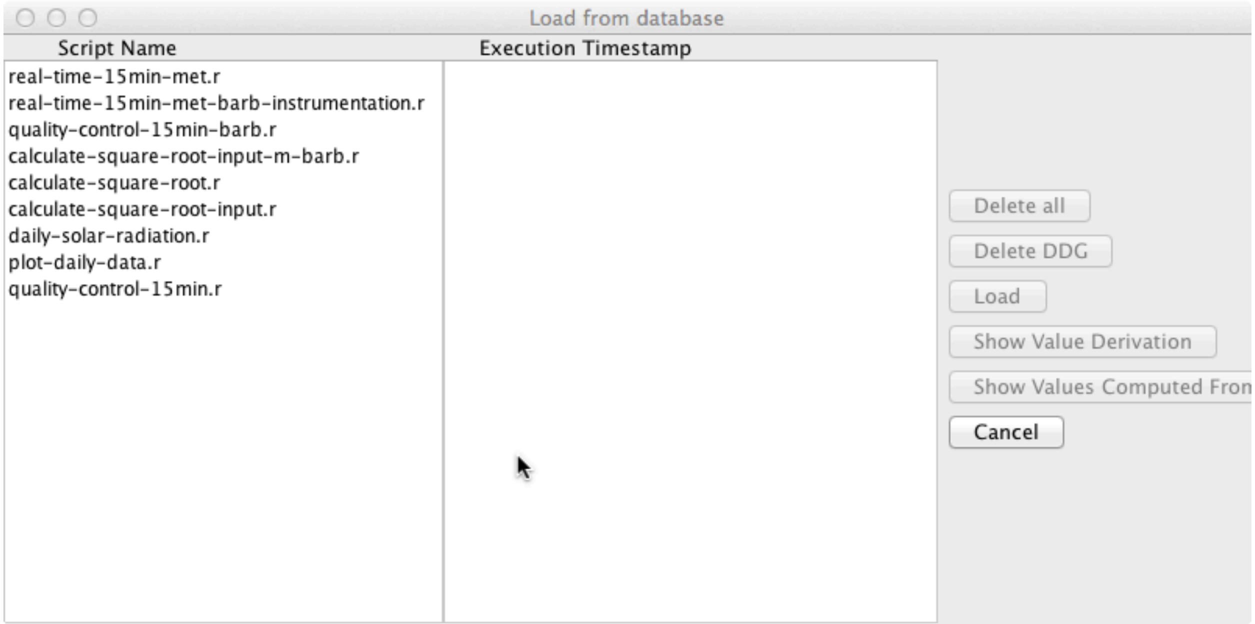
Help menu

The help menu contains a single command: Command Overview. This gives a brief description of the commands involving use of the mouse and trackpad to control the display of the DDG.

Using the DDG database

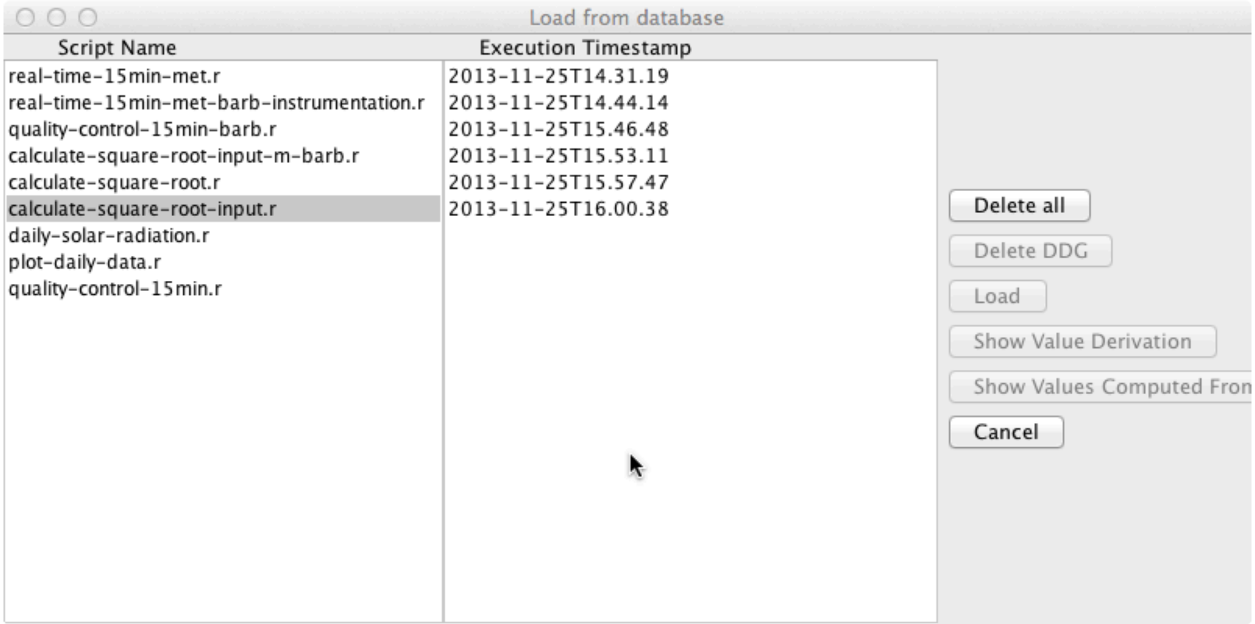
Initially, DDGs are stored in files. If you run an R script using the same DDG directory as a previous execution, the DDG will be overwritten. To save the DDG permanently and to enable querying functionality provided by the database, you must save the DDG to the database after loading it into DDG Explorer as a file. You can do this by selecting the "Save to database" command in the File menu when a DDG is being displayed. Alternatively, when you close a window displaying a DDG, it will prompt you as to whether you would like to save the DDG to the database.

Returning to the DDG Chooser (the main window of the DDG Explorer), we will now describe the functionality available for DDGs stored in the database. When you click on the "Load from database" button, you will see a new window that looks like this:



Script Name	Execution Timestamp
real-time-15min-met.r	
real-time-15min-met-barb-instrumentation.r	
quality-control-15min-barb.r	
calculate-square-root-input-m-barb.r	
calculate-square-root.r	
calculate-square-root-input.r	
daily-solar-radiation.r	
plot-daily-data.r	
quality-control-15min.r	

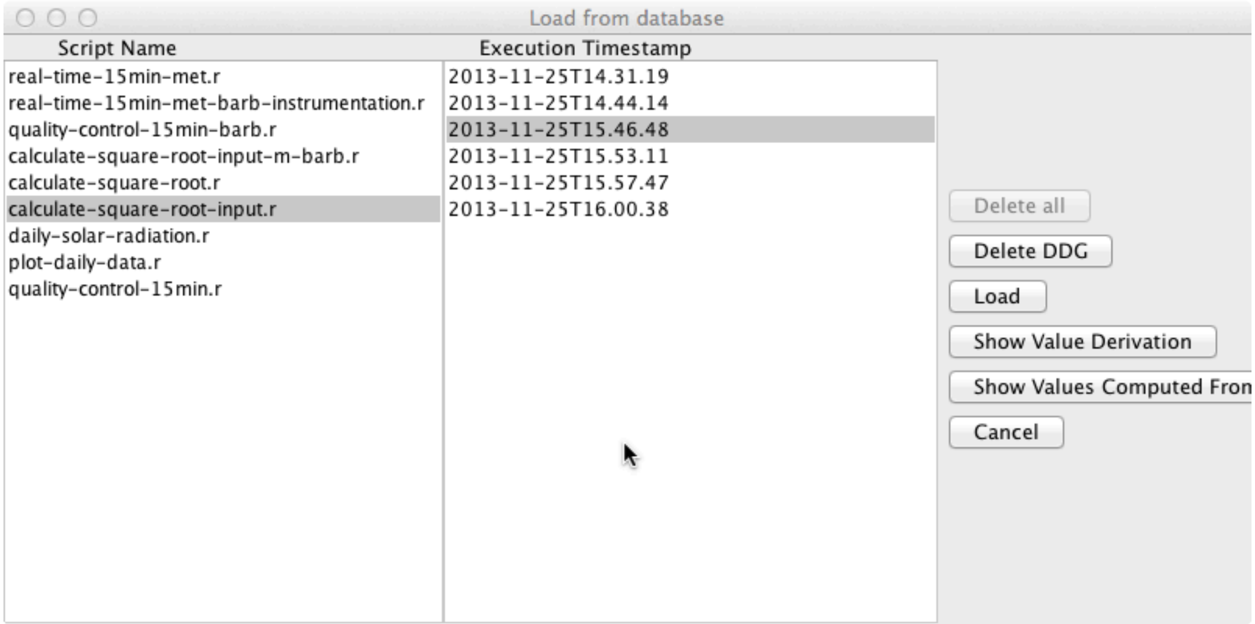
The left column lists the names of scripts that have DDGs associated with them in the database. If you select one of these, the window will change to look like this:



Script Name	Execution Timestamp
real-time-15min-met.r	2013-11-25T14.31.19
real-time-15min-met-barb-instrumentation.r	2013-11-25T14.44.14
quality-control-15min-barb.r	2013-11-25T15.46.48
calculate-square-root-input-m-barb.r	2013-11-25T15.53.11
calculate-square-root.r	2013-11-25T15.57.47
calculate-square-root-input.r	2013-11-25T16.00.38
daily-solar-radiation.r	
plot-daily-data.r	
quality-control-15min.r	

Now the second column shows the timestamps for all of the DDGs created from that script. In the rightmost column, the “Delete all” button is now enabled. If you click this button, it will delete all of the DDGs associated with that script from your database. You will be prompted to confirm that you want to delete them, but once deleted, they cannot be recovered, so be careful!

If you select one execution timestamp, you will now see more buttons enabled:



Script Name
real-time-15min-met.r
real-time-15min-met-barb-instrumentation.r
quality-control-15min-barb.r
calculate-square-root-input-m-barb.r
calculate-square-root.r
calculate-square-root-input.r
daily-solar-radiation.r
plot-daily-data.r
quality-control-15min.r

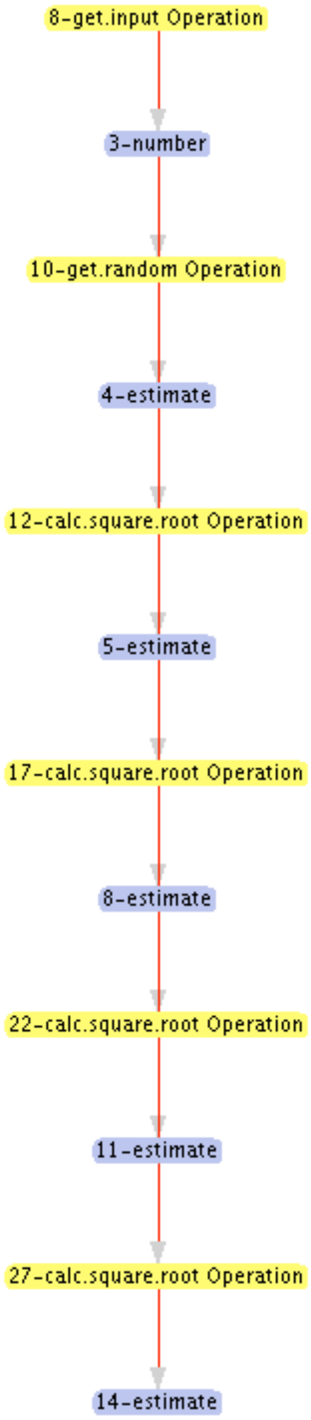
Execution Timestamp
2013-11-25T14.31.19
2013-11-25T14.44.14
2013-11-25T15.46.48
2013-11-25T15.53.11
2013-11-25T15.57.47
2013-11-25T16.00.38

Here is what each of these buttons does.

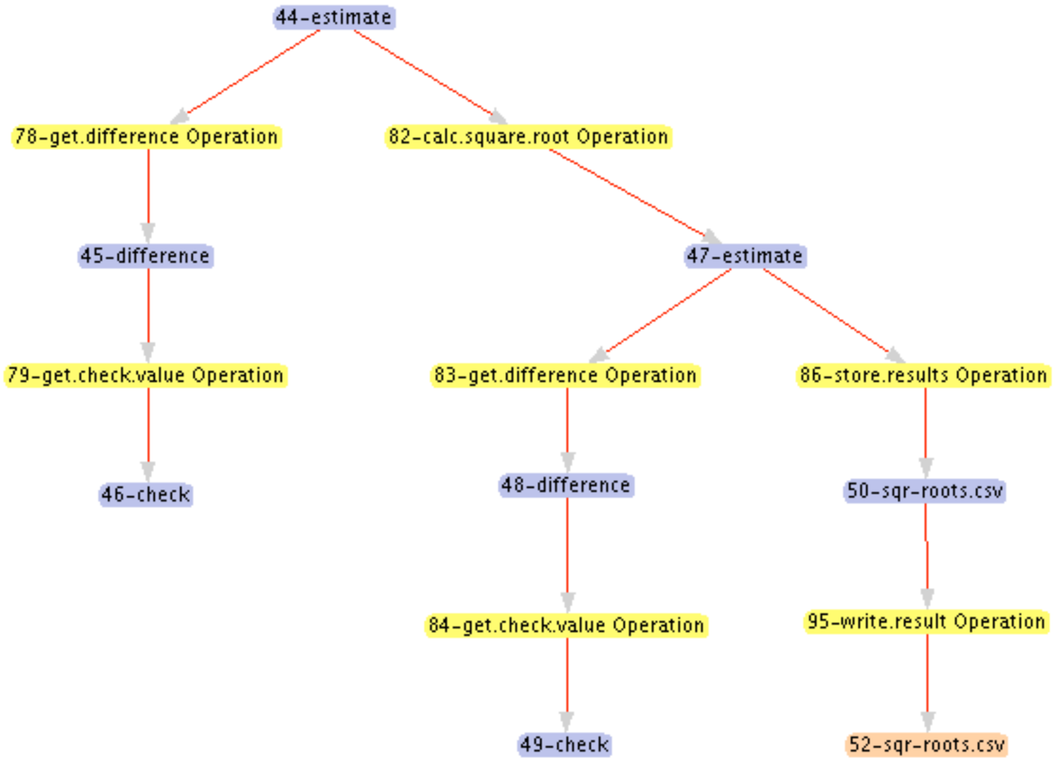
- Delete DDG - This will delete the one DDG that corresponds to the selected script and execution timestamp after confirming that you really want to delete it.
- Load - This will read the entire DDG from the database and create a window displaying the DDG, just as what happens when you read it from a file.

- Show value derivation - This query allows you to view just the portion of a DDG that explains how a particular output value was calculated. After clicking this button, a window will appear asking you to select a variable to display. For example, if the user asks to see the variable 14-estimate, the (partial) DDG shown is at the right. Here, we can see that get.input operation outputs a number, which is input to get.random that produces the first estimate value. This estimate is input to calc.square.root, which produces another estimate. This cycle repeats until we get to the desired estimate.

  We can get more insight by right-clicking on the data nodes to see what values they have. Here, we see that the 3-number has the value 3636363. 4-estimate has the value 3016450.69879702. Showing subsequent estimate values demonstrates how the algorithm narrows in on the square root value that is within the desired tolerance.


8-get.input Operation
3-number
10-get.random Operation
4-estimate
12-calc.square.root Operation
5-estimate
17-calc.square.root Operation
8-estimate
22-calc.square.root Operation
11-estimate
27-calc.square.root Operation
14-estimate

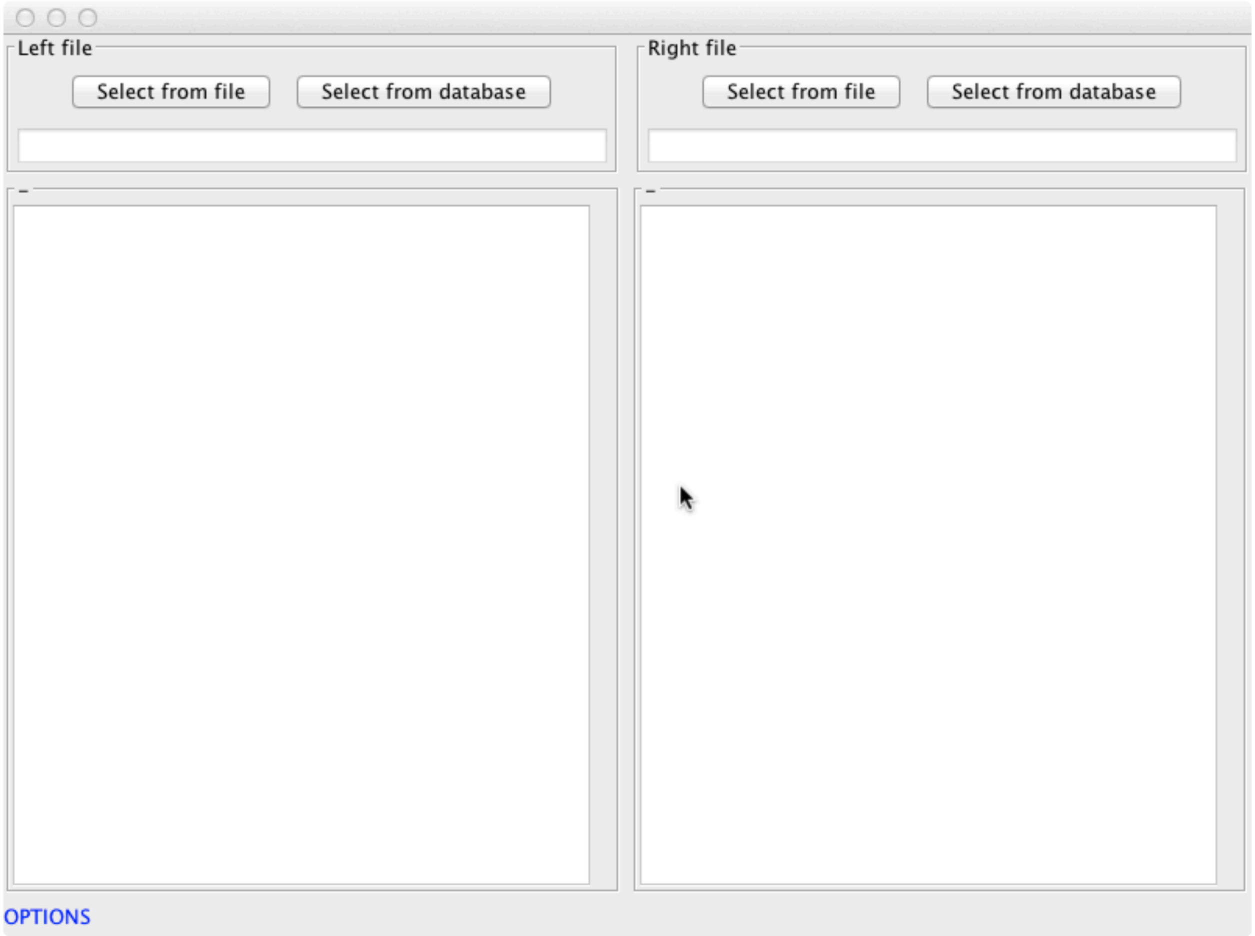
- Show values computed from: This query allows you to view the portion of the DDG that follows from a particular data value. After clicking this button, you will see a window like the one for the previous query where you can select a data value. This time, however, you will see the values that are computed from this data value as shown below.
  Here, we start with 44-estimate and can see that it is input to two operations. One operation determines if it is close enough to the actual square root (which it is not), while the other uses it to refine the estimate, producing 47-estimate. 47-estimate is within the desired tolerance, so that value is saved in the sqr-roots.csv output file.


44-estimate
78-get.difference Operation
82-calc.square.root Operation
45-difference
47-estimate
79-get.check.value Operation
83-get.difference Operation
86-store.results Operation
46-check
48-difference
50-sqr-roots.csv
84-get.check.value Operation
95-write.result Operation
49-check
52-sqr-roots.csv

Comparing R Scripts

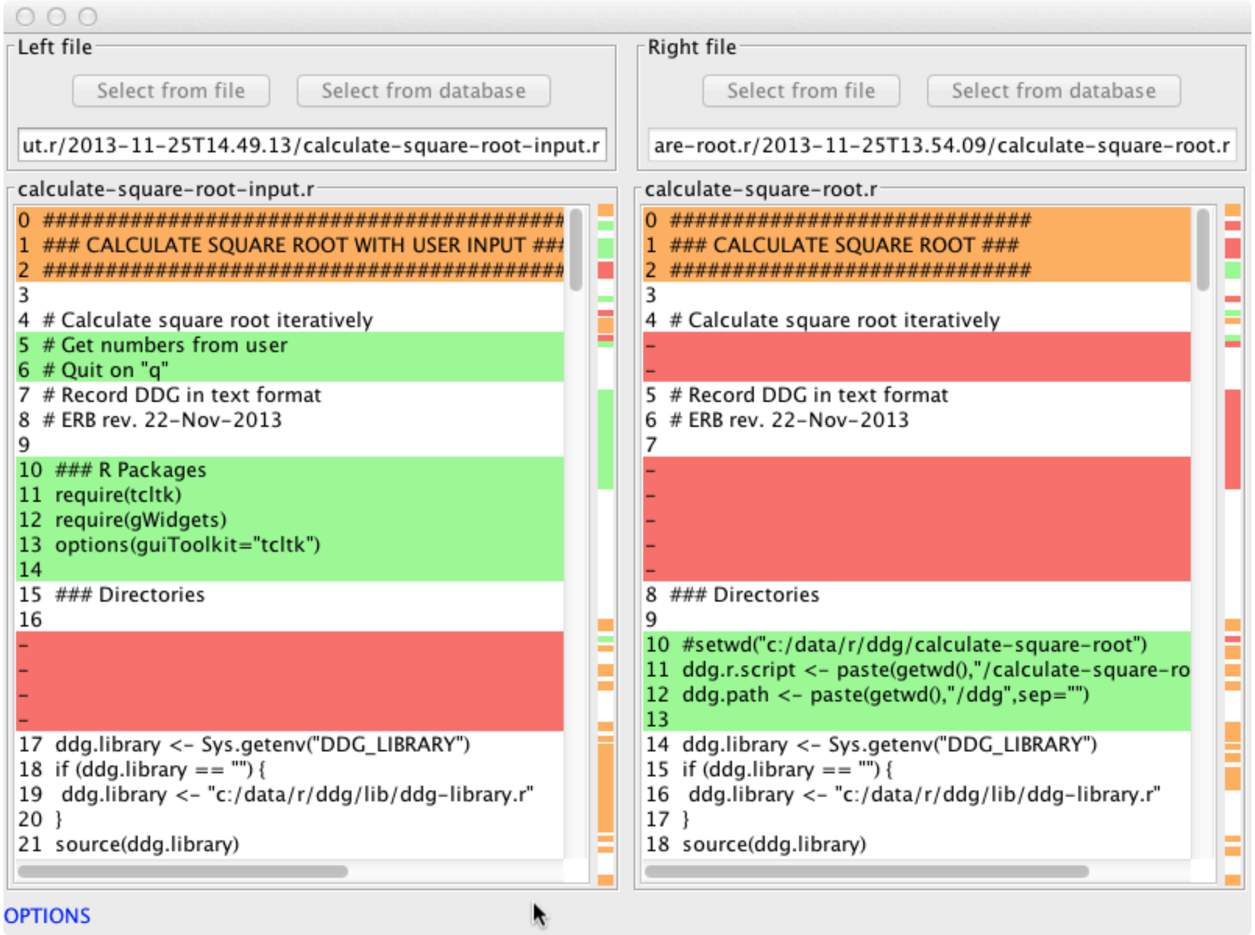
You will likely find that you change your R script over time. The R scripts are saved in the database along with the DDG, so the DDG Explorer can always show you the R script that corresponds to a DDG.

In addition, you may wonder how a script has changed over time, or how the script that generated one DDG differs from another script. To help you understand that history of your scripts, the main window of the DDG Explorer contains a button labeled “Compare R scripts”. When you click that, you will see this window:



In this window, you can select two R scripts to compare. One will be displayed on the left side of the window. The other will be displayed on the right side of the window. These scripts can either come from the file system or from the database.

Here is what the window will look like after selecting two related but different scripts:


Left file
Select from file
Select from database
ut.r/2013-11-25T14.49.13/calculate-square-root-input.r
Right file
Select from file
Select from database
are-root.r/2013-11-25T13.54.09/calculate-square-root.r
OPTIONS

calculate-square-root-input.r

```
#############################################
### CALCULATE SQUARE ROOT WITH USER INPUT ###
#############################################

# Calculate square root iteratively
# Get numbers from user
# Quit on "q"
# Record DDG in text format
# ERB rev. 22-Nov-2013

### R Packages
require(tcltk)
require(gWidgets)
options(guiToolkit="tcltk")

### Directories

-
-
-
-
ddg.library <- Sys.getenv("DDG_LIBRARY")
if (ddg.library == "") {
 ddg.library <- "c:/data/r/ddg/lib/ddg-library.r"
}
source(ddg.library)
```

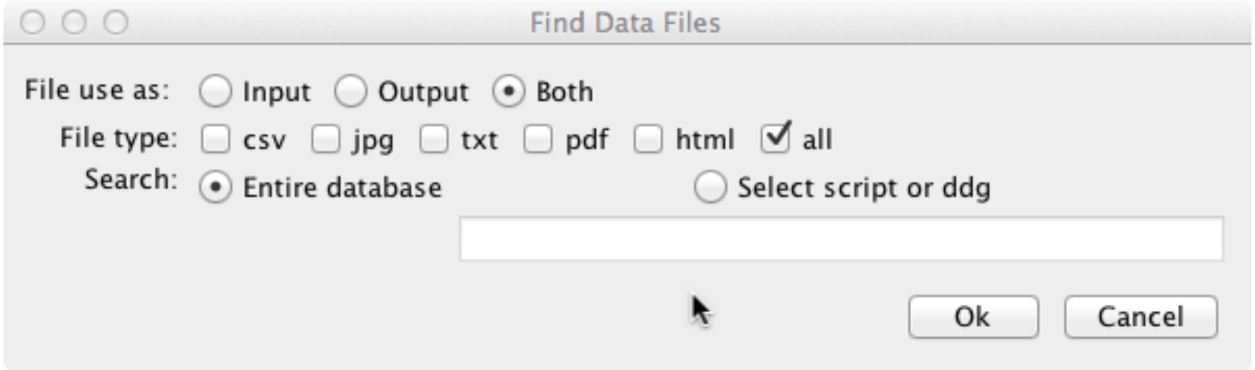
calculate-square-root.r

```
##############################
### CALCULATE SQUARE ROOT ###
##############################

# Calculate square root iteratively
-
-
# Record DDG in text format
# ERB rev. 22-Nov-2013

-
-
-
-
-
### Directories

#setwd("c:/data/r/ddg/calculate-square-root")
ddg.r.script <- paste(getwd(),"/calculate-square-ro
ddg.path <- paste(getwd(),"/ddg",sep="")

ddg.library <- Sys.getenv("DDG_LIBRARY")
if (ddg.library == "") {
 ddg.library <- "c:/data/r/ddg/lib/ddg-library.r"
}
source(ddg.library)
```

The brown color identifies lines that are similar but modified between the two versions. Green identifies lines that have no corresponding lines in the other file. Red identifies places where the comparison tool has inserted blank lines just to help align the files better. Lines with a white background are identical between the two files.

Finding where Data Files are Used

Another useful feature is to allow the user to find where a particular data file has been used as input or produced as output. To do that, go to the main window and click on the “Find Data Files” button. This will bring up the following window where you can limit the types of files searched for:

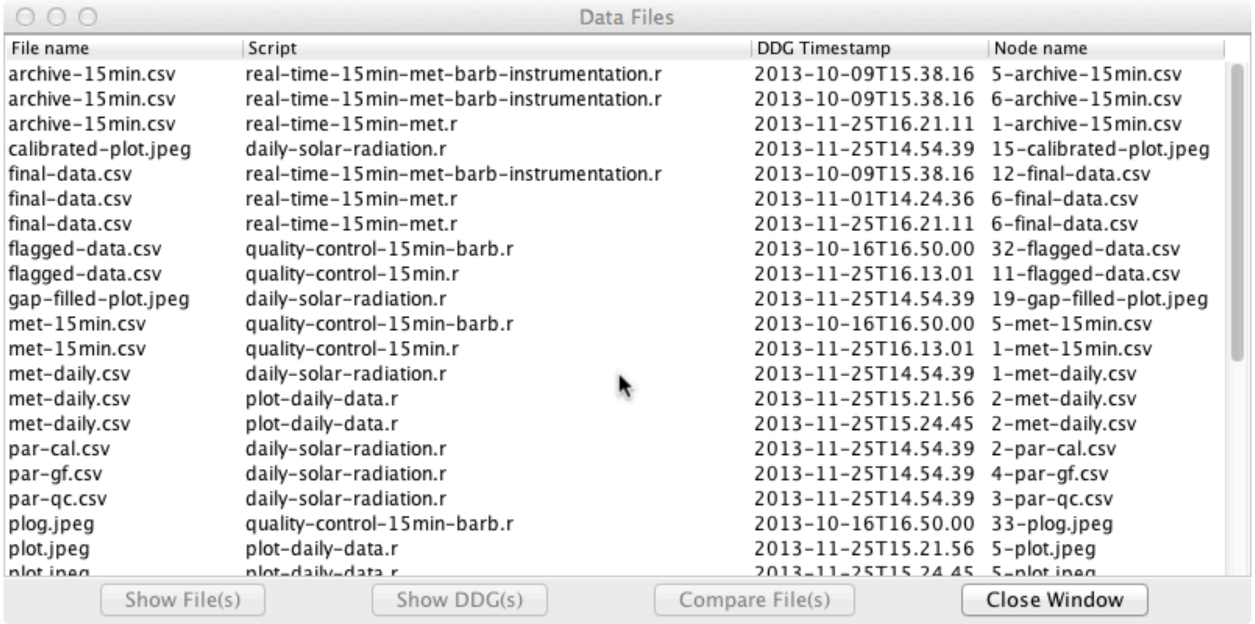


File use as: ○ Input ○ Output ● Both

File type: □ csv □ jpg □ txt □ pdf □ html ☑ all

Search: ● Entire database ○ Select script or ddg

Here, you can limit to the search to just input files, just output files, or both. You can select one or more file extensions. If you select all, it will include files with any extension, including extensions not appearing in the list. The files are searched for in the database. You can either select for files appearing anywhere in the database or limit the search to a particular script or a particular ddg. After making your selections, another window will appear listing the matching files as follows.



File name	Script	DDG Timestamp	Node name
archive-15min.csv	real-time-15min-met-barb-instrumentation.r	2013-10-09T15.38.16	5-archive-15min.csv
archive-15min.csv	real-time-15min-met-barb-instrumentation.r	2013-10-09T15.38.16	6-archive-15min.csv
archive-15min.csv	real-time-15min-met.r	2013-11-25T16.21.11	1-archive-15min.csv
calibrated-plot.jpeg	daily-solar-radiation.r	2013-11-25T14.54.39	15-calibrated-plot.jpeg
final-data.csv	real-time-15min-met-barb-instrumentation.r	2013-10-09T15.38.16	12-final-data.csv
final-data.csv	real-time-15min-met.r	2013-11-01T14.24.36	6-final-data.csv
final-data.csv	real-time-15min-met.r	2013-11-25T16.21.11	6-final-data.csv
flagged-data.csv	quality-control-15min-barb.r	2013-10-16T16.50.00	32-flagged-data.csv
flagged-data.csv	quality-control-15min.r	2013-11-25T16.13.01	11-flagged-data.csv
gap-filled-plot.jpeg	daily-solar-radiation.r	2013-11-25T14.54.39	19-gap-filled-plot.jpeg
met-15min.csv	quality-control-15min-barb.r	2013-10-16T16.50.00	5-met-15min.csv
met-15min.csv	quality-control-15min.r	2013-11-25T16.13.01	1-met-15min.csv
met-daily.csv	daily-solar-radiation.r	2013-11-25T14.54.39	1-met-daily.csv
met-daily.csv	plot-daily-data.r	2013-11-25T15.21.56	2-met-daily.csv
met-daily.csv	plot-daily-data.r	2013-11-25T15.24.45	2-met-daily.csv
par-cal.csv	daily-solar-radiation.r	2013-11-25T14.54.39	2-par-cal.csv
par-gf.csv	daily-solar-radiation.r	2013-11-25T14.54.39	4-par-gf.csv
par-qc.csv	daily-solar-radiation.r	2013-11-25T14.54.39	3-par-qc.csv
plog.jpeg	quality-control-15min-barb.r	2013-10-16T16.50.00	33-plog.jpeg
plot.jpeg	plot-daily-data.r	2013-11-25T15.21.56	5-plot.jpeg
plot.jpeg	plot-daily-data.r	2013-11-25T15.24.45	5-plot.jpeg

The first column identifies the file name. The second column identifies the script that used or created it. The third column shows the execution time of the DDG that references it. The last column shows the name of the node within the DDG. You can sort the list by clicking on the header of the column you want to sort by.

You can now select one or more files from the list. After selecting files, the buttons work as follows:

- Show Files will display the files themselves, each in a separate window.
- Show DDGs will display the DDGS, scrolling to the position in the DDG where the file is used.
- Compare Files will load the files using the text comparison tool. This will only work if exactly two files are selected. Currently, this button does not do anything.

Behind the scenes

This section provides some additional details on the DDG files and database.

ddg directories

All of the information that is collected during execution of your R scripts is saved in a directory. It is important that you know where this directory is in order to find your DDG files and load them into the DDG Explorer.

The directory used is the one identified by the ddg.path variable in R. This directory will contain a text file named ddg.txt along with files that are created by calling the instrumentation functions ddg.file, ddg.file.out, ddg.file.copy.out, ddg.snapshot, and ddg.snapshot.out. For details on these functions, please see the documentation on how to instrument your R code.

ddg.txt

The ddg.txt file contains a textual definition of the DDG. It is not necessary to understand the contents of this file, but the interested reader can learn more by reading the *Using the R DDG Library* document.

DDG database

The DDG database is stored in a directory called .ddg in your home directory. For example, on Windows 7, this would be something like C:\Users\emeryboose\.ddg. On the Mac, this would be something like /Users/barbaralerner/.ddg. You should not interact with the database through the file system, but only through the DDG Explorer.

.RProfile

The instrumentation needs to know where to find the R library. You can do this as follows:

```
ddg.library <- "c:/data/r/ddg/lib/ddg-library.r"
source(ddg.library)
```

Alternatively, you could use an environment variable to define where your library is stored. Doing this will make it easier to share your script with others. In that case, you would put this at the top of your R script.

```
ddg.library <- Sys.getenv("DDG_LIBRARY")
source(ddg.library)
```

To set an environment variable, create a file called .RProfile in your home directory. This file should contain the following, again using the location you have chosen:

```
# Tells R where to find the DDG library.
.First <- function() {
   Sys.setenv(DDG_LIBRARY = "c:/data/r/ddg/lib/ddg-library.r")
}
```

This function will be automatically executed whenever you start R. You may also find it convenient to put other commands inside the .First function that you find yourself using whenever you start R, such as a setwd call to get to your favorite working directory.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Awards No. CCR-0205575, CCR-0427071, and IIS-0705772, the National Science Foundation REU grants DBI-0452254 and DBI-1003938, the Mount Holyoke Center for the Environment Summer Leadership Fellowships, and the Charles Bullard Fellowship in Forest Research at the Harvard Forest. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation, Mount Holyoke College or Harvard University.

Numerous students have been involved in the research and tool development through the REU program at Harvard Forest, thesis research at Mount Holyoke and graduate students at the University of Massachusetts, Amherst. They are Cory Teshera-Sterne (Mount Holyoke and 2009 REU), Morgan Vigil (2010 REU), Sofiya Taskova (2010 REU, 2011 REU, Mount Holyoke honors thesis), Andy Galdunski (2011 REU), Garrett Rosenblatt (2011 REU), Miruna Oprescu (2012 REU), Yujia Zhou (2012 REU), Shay Addams (2013 REU), Vasco Carinhas (2013 REU), Xiang Zhao (University of Massachusetts, Amherst).

Lee Osterweil from the University of Massachusetts, Amherst, Aaron Ellison and David Foster from Harvard Forest have been involved in this data provenance work for a long time and got this project started many years ago with a collaboration between UMass and Harvard Forest.

The DDG Explorer builds on numerous packages that were developed elsewhere:

- Little-JIL, developed at the University of Massachusetts, Amherst, under the guidance of Lee Osterweil
- Jena, the database technology
- Prefuse, the library used to help draw the DDGs
- jdiff, the library that allows the comparison of R scripts. This is licensed by QArks.com under an LGPL license. The license description is available at http://www.gnu.org/licenses/lgpl.html.

Below is the detailed licensing information that allows us to use and distribute the DDG Explorer software.

Jena license

Unless otherwise noted, the following copyright statement applies:

Jena includes software developed by the Apache Software Foundation (http://www.apache.org/).

Jena includes RDF schemes from DCMI:

Portions of this software may use RDF schemas Copyright (c) 2006 DCMI, the Dublin Core Metadata Initiative. These are licensed under the Creative Commons 3.0 Attribution license.

Jena is built on top of other sub-systems which we gratefully acknowledge: details of these systems and their version numbers.

YourKit is kindly supporting open source projects with its full-featured Java Profiler. YourKit, LLC is the creator of innovative and intelligent tools for profiling Java and .NET applications. Take a look at YourKit's leading software products: YourKit Java Profiler and YourKit .NET Profiler.

Prefuse License