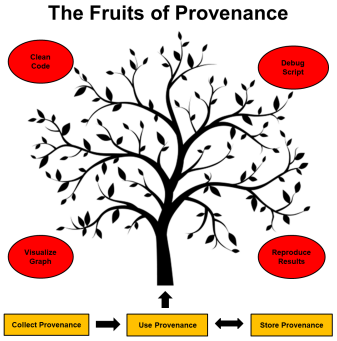
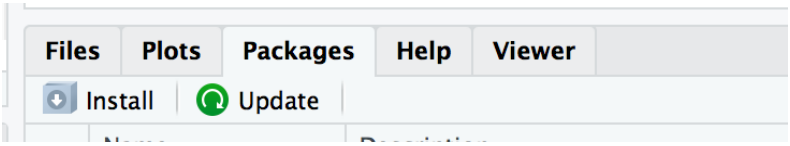
end-to-end-provenance.github.io



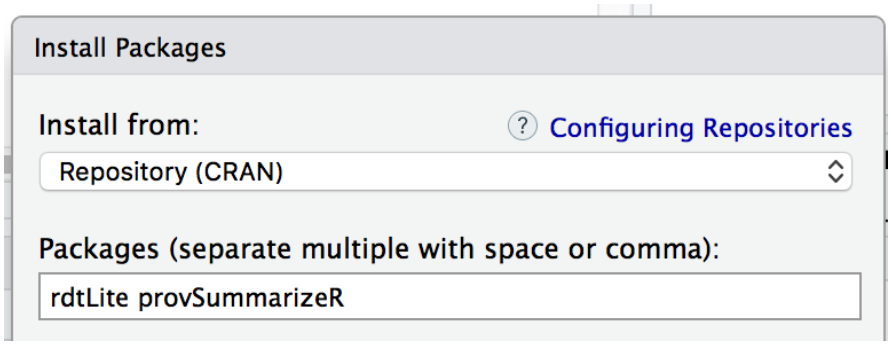
Collecting and Using Provenance from R
February 14, 2019

1. Start up RStudio.

2. Download the provenance libraries. In RStudio, go to the Packages tab.



   Click Install



   Be sure Repository (CRAN) is selected and type in rdtLite `provSummarizeR` for the package. This should install the rdtLite and provSummarizeR packages.

   Then, in the Console enter the following to get two more packages:

```
install.packages("devtools")
devtools::install_github("End-to-end-provenance/provGraphR")
devtools::install_github("End-to-end-provenance/provDebugR",ref="debugging")
```

3. Open a Web browser and go to
   https://www.mtholyoke.edu/~blerner/ProvenanceDemo.zip
   This should download ProvenanceDemo.zip. Unzip the file.

4. In RStudio, go to the folder containing the scripts downloaded (Session menu, Set working directory, Choose directory…)

5. Go to the BuggyCars folder and open BuggyCars.R. This script loads the built-in cars dataset and then plots the miles-per-gallon (mpg) of a car against its number of cylinders (or at least it is supposed to). Run the script and look at the plot. It should contain a datapoint for 4, 6, and 8 cylinders. You should notice that it does not seem complete as there is no datapoint for 8 cylinders. We will now use provDebugR to find the problem. In RStudio's console, enter these commands:

`> library (provDebugR)`	Load the provenance debugger.
`> debug.init ("BuggyCars.R")`	This runs the script, collecting provenance, and initializes the debugger
`> debug.browser()`	This starts the interactive debugging session.
`Debug> l 18`	The plot doesn't look right, so this jumps to line 18 of the script, where the plot is created.
`Debug> p cylinders`	This prints the value of the cylinders variable at line 18. It should have 3 values, 4, 6, and 8 as we expect.
`Debug> p mpg`	This prints the value of the mpg variable at line 18. This looks funny, You should see 2 numbers and then NaN, which means Not A Number. Something is wrong here.
`Debug> p cars8Cyl.df`	Looking at the script, I see that mpg gets set on line 14 by creating a vector. I am suspicious about the value inserted in the 8 cylinder case. This command shows me that value.
`Debug> p allCars.df`	cars8Cyl.df got its value by extracting some information from allCars.df. So, let's print that out. Unfortunately, the debugger can only print the first line of the data frame and it looks ok. Let's look at the value a different way. In the top right of RStudio is an Environment tab. Click on allCars.df to see its value. This still seems ok. In particular, I can see that there are cars there with 8 cylinders. Now, since I know allCars.df looks fine, but cars8Cyl.df does not, I examine line 10 more closely. Do you see the bug?
`Debug> Quit`	Let's quit the debugger.

6. Go to the BuggyCars2 folder and run the BuggyCars2.R script. This one also contains a bug. This time it results in an error message:

```
> source("BuggyCars2.R")
Error in data.frame(cylinders, mpg) :
  arguments imply differing number of rows: 4, 3
```

The debugger can be very helpful in understanding what causes an error. Try this:

`> debug.init ("BuggyCars2.R")`	This runs the script, collecting provenance, and initializes the debugger
`> debug.error.trace()`	This will show you which lines of the script lead to the error. In particular, we can see the error occurred on line 15.
`> debug.error.trace(stack.overflow=TRUE)`	This looks up your error message in stack overflow and suggests some pages that might help you understand the problem! Select an choice and it will open a browser on that page!

7. Go to the DailySolarRadiation folder and use provSummarizeR to run DailySolarRadiation.R:

```
> library (provSummarizeR)
> prov.summarize.run("DailySolarRadiation.R", prov.dir=".", overwrite=FALSE)
```

This script reads some csv files, creates some plots and saves them to files.

In addition, provSummarizeR produces output on the screen that includes information about the computing environment used to run the script, when the script was executed, the libraries used along with their versions, and the timestamps and hash values of the files input and output.

provSummarizeR also saves copies of the script and all the input and output files. In this way, if your run provSummarizeR repeatedly, if you get different results, you will be able to compare your files to see what has changed. Did you change the script? Is the input different? Has one of the libraries or R been updated?

You can see these files in the scripts and data subfolders of the prov folder:

```
> dir()
 [1] "calibrated-plot.jpeg"
 [2] "DailySolarRadiation.R"
 [3] "ddg"
 [4] "gap-filled-plot.jpeg"
 [5] "met-daily.csv"
 [6] "par-cal.csv"
 [7] "par-gf.csv"
 [8] "par-qc.csv"
 [9] "processed-data.csv"
[10] "prov_DailySolarRadiation_2019-02-10T16.17.34EST"
[11] "quality-controlled-plot.jpeg"
[12] "raw-plot.jpeg"
```

```
> dir("prov_DailySolarRadiation_2019-02-10T16.17.34EST")
[1] "data"       "debug"     "prov.json" "scripts"
> dir("prov_DailySolarRadiation_2019-02-10T16.17.34EST/data")
[1] "20-met-daily.csv"      "21-par-cal.csv"        "22-par-qc.csv"
[4] "23-par-gf.csv"         "32-processed-data.csv"
> dir("prov_DailySolarRadiation_2019-02-10T16.17.34EST/scripts")
[1] "DailySolarRadiation.R"
```

Conclusion

That's it! What have we seen?

provDebugR allowed us to look at the values of variables at different points in the execution of the script without actually re-running the script. Instead it looks up the saved values in the provenance.

provSummarizeR keeps a record of each execution of a script so that as the script or the data changes, we know which output was generated from each execution, and can compare different versions to understand what changed. This can be very valuable if working with a set of data over a period of time and trying to analyze it in different ways.